PRACTICE WITH

ESP32 PROJECT

ESP8266, IoT Project, Crypto Project, Arduino Coding, Actuator node, PIR Sensor Projects and more

PRACTICE WITH ESP32 PROJECT

ESP8266, IoT Project, Crypto Project, Arduino Coding, Actuator node, PIR Sensor Projects and more

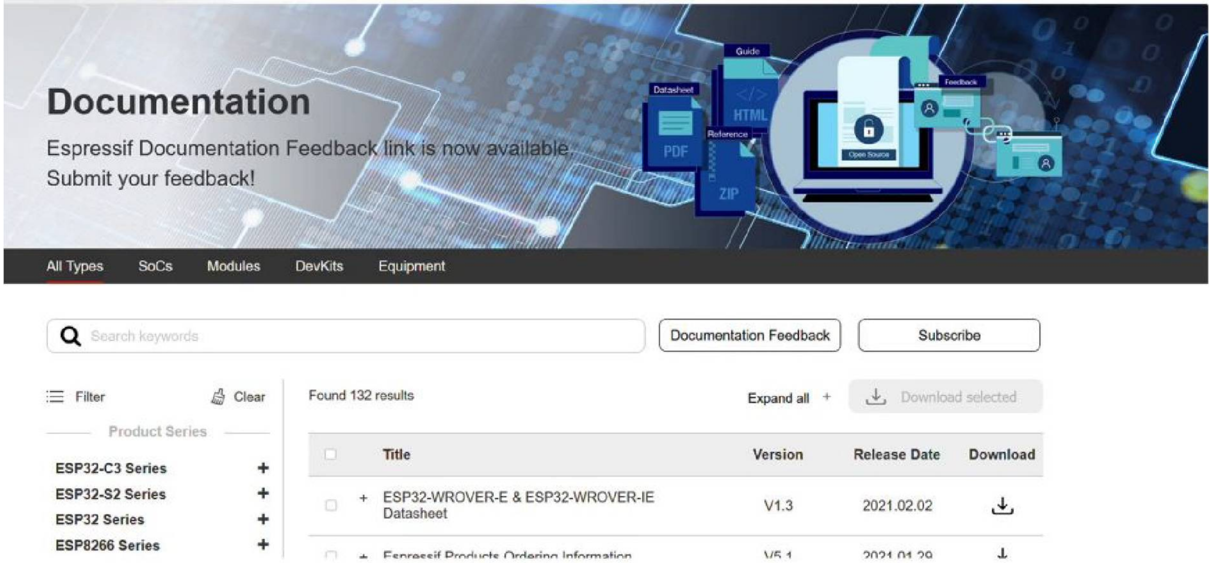
By

Jansa Selvam

TABLE OF CONTENTS

GET STARTED WITH THE ESP32

How to get started with the ESP32 microcontroller from Shanghai-based company Espressif. Over the last few years, the ESP32 has become a very popular microcontroller amongst makers and electronics hobbyists, but also in the industry.



Documentation

Espressif Documentation Feedback link is now available.
Submit your feedback!

Found 132 results

	Title	Version	Release Date	Download
□ +	ESP32-WROVER-E & ESP32-WROVER-IE Datasheet	V1.3	2021.02.02	⤓
□ +	Espressif Products Ordering Information	V5.1	2021.01.29	⤓

There are several reasons for this. It has built-in Wi-Fi and Bluetooth. It is incredibly powerful. It is cheap as unit prices for a single-core version start at around 1.5 euros and dual-core versions cost about 3 euros. There are many low-cost development boards available, and there is plenty of good documentation. Good documentation is important and Espressif has made a huge effort to make the ESP32 accessible to makers as well as professional users. Besides good documentation Espressif also provides development tools like the compiler and libraries for free.

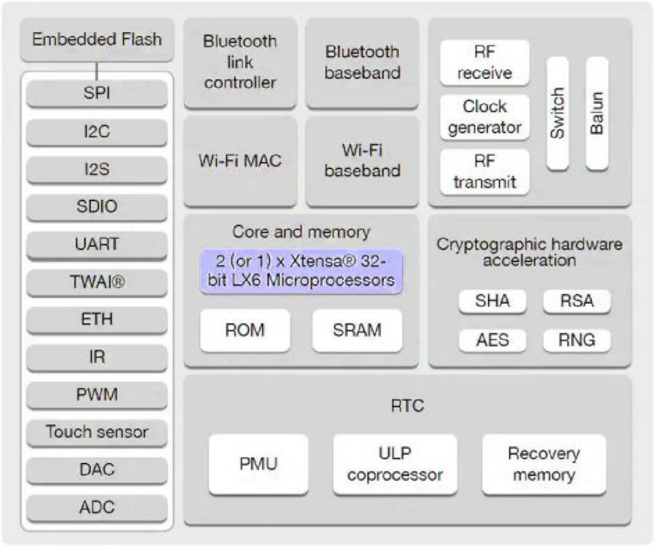
Arduino core for the ESP32

Need help or have a question? Join the chat at

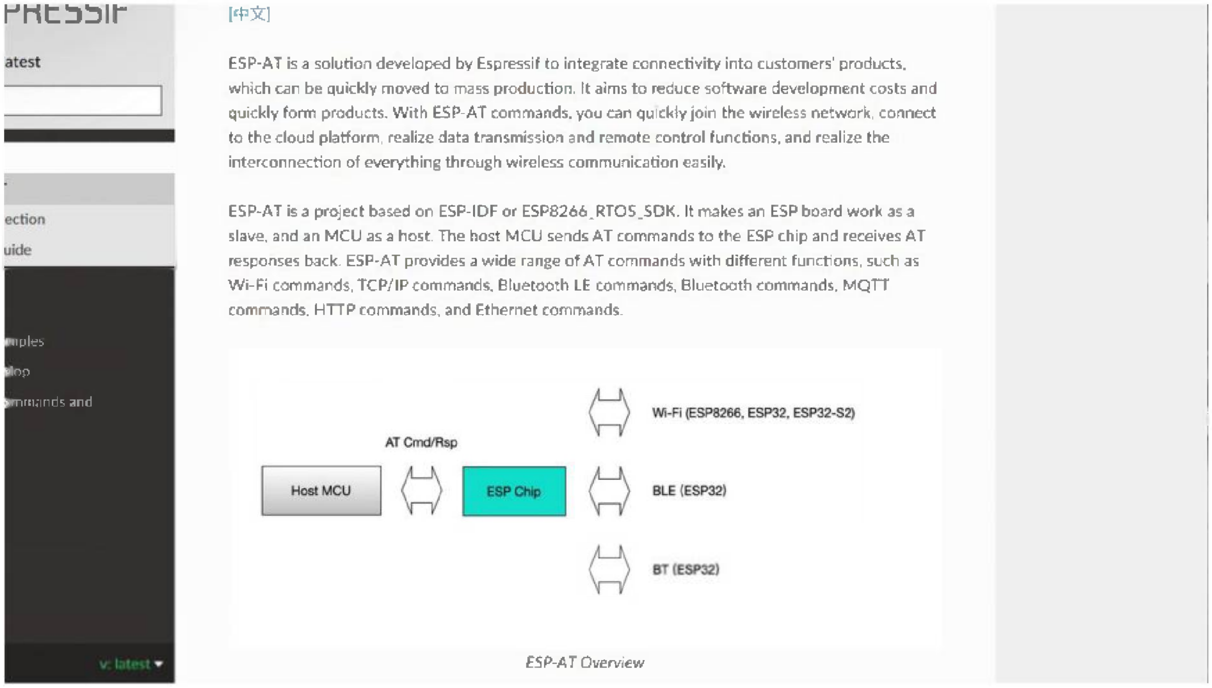
Contents

- Development Status
- Installation Instructions
- Decoding Exceptions
- Issue/Bug report template
- ESP32Dev Board PINMAP

The ESP32 is probably the first and maybe the only microcontroller that was launched with support for Arduino. The wireless capabilities of the ESP32 are what its users will mention first, but it has much more than that. The device comes with many peripherals like SPI, I2C and UART, but also CAN, which is called TWAI, which stands for Two-Wire Automotive Interface, it has LED and motor control support, PWM, I2S for audio, analogue in- and outputs, infrared remote control, timers and counters, Ethernet, SDIO for memory cards, touch sensing support and it even has a built-in a Hall magnetic field sensor. Even though this long list is incomplete, there are two things the ESP32 does not have. First of all, it does not have USB, and secondly, it does not have Flash memory. Well, this is not entirely true, there are versions that have built-in Flash but most don’t. Most ESP32-based boards and modules have external Flash memory even though it may be built into the chip. Confusing? Just don’t worry about it and remember that most ESP32 boards have 4 MB of Flash memory.



At the heart of the ESP32 are one or two Xtensa LX6 32-bit processor cores clocked at up to 240 MHz. The ESP32 started out as a dual-core device, but today single-core versions exist too. Actually, the ESP32 has one more processor core which is intended for low-power operation. It will be clear by now that the ESP32 is a versatile device that can be used in a wide variety of applications. The internet of things is, of course, one of them. And this is accentuated by support for popular cloud service providers. Apart from cloud support, Espressif has tons of repositories on GitHub with example code and demo applications. Besides the libraries from Espressif, ESP32 users have published lots of code too, making it quick and easy to get an application up and running. So, this is great to get started. But before you can start creating your own applications, you must install a development system or toolchain. There are several options, but here we will limit ourselves to MicroPython, AT Commands, Arduino, and ESP-IDF, and in that order. MicroPython is a stripped-down version of Python that runs on microcontrollers. With MicroPython you can program the ESP32 in Python. As many people already know Python, this is quite practical. The ESP32 does all the Python stuff itself, so no other tools or programs are needed. The only thing you must do is program the ESP32 with the MicroPython firmware and this can be done with Python. For this you need, of course, Python but you probably already have that installed since you are interested in Python. If not, install it first. Then you must install esptool, the tool to load a program into the flash memory of the ESP32 board. Download the MicroPython version suitable for your ESP32 board from the MicroPython website. To be sure, first erase the ESP32. Then program the firmware. After programming launch a serial terminal to bring up the REPL prompt. Type help() to see if things work. If they do, you are all set for ESP32 application development with MicroPython. AT Commands date back to the nineteen eighties. They consist of short text-based commands beginning with "AT" sent over a serial link to a target where they are executed. The target used to be a modem and the commands let you control the modem and send and receive data.

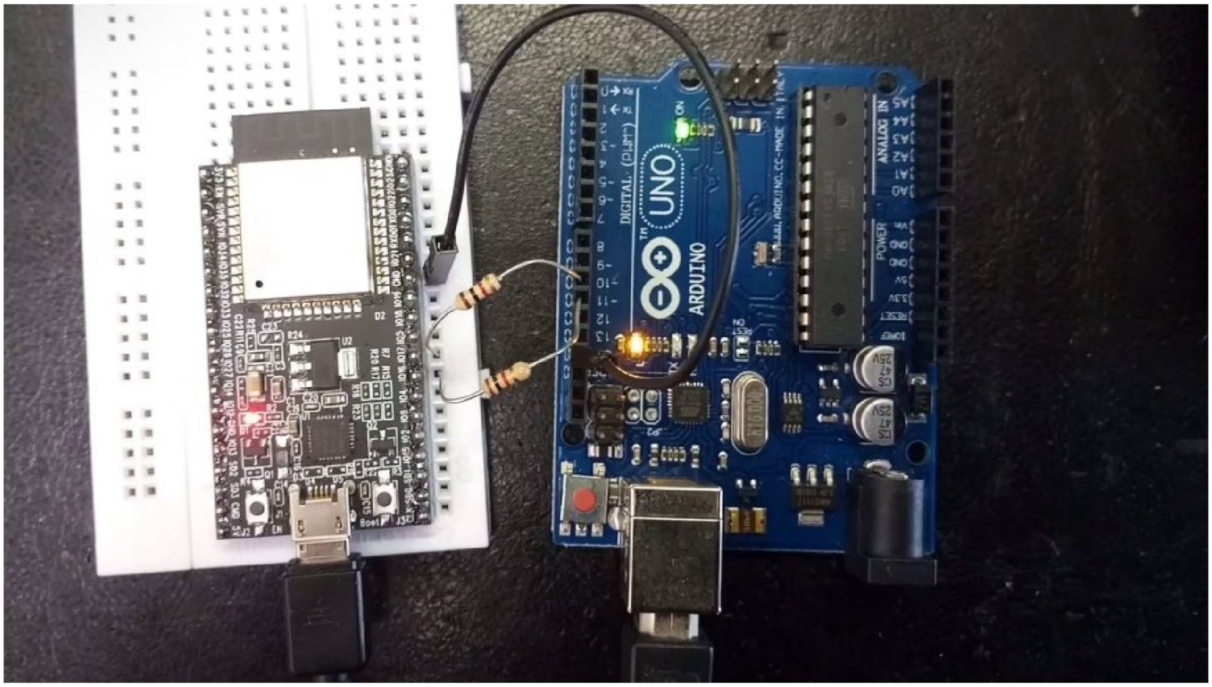


ESP-AT is a solution developed by Espressif to integrate connectivity into customers' products, which can be quickly moved to mass production. It aims to reduce software development costs and quickly form products. With ESP-AT commands, you can quickly join the wireless network, connect to the cloud platform, realize data transmission and remote control functions, and realize the interconnection of everything through wireless communication easily.

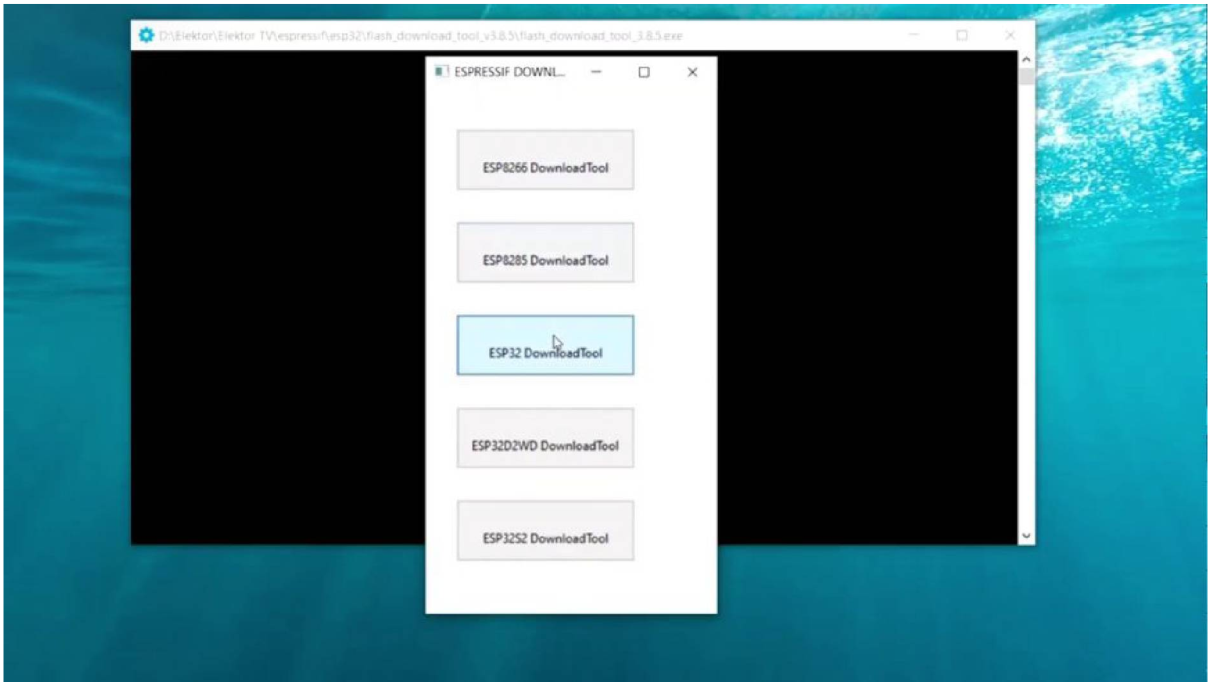
ESP-AT is a project based on ESP-IDF or ESP8266_RTOS_SDK. It makes an ESP board work as a slave, and an MCU as a host. The host MCU sends AT commands to the ESP chip and receives AT responses back. ESP-AT provides a wide range of AT commands with different functions, such as Wi-Fi commands, TCP/IP commands, Bluetooth LE commands, Bluetooth commands, MQTT commands, HTTP commands, and Ethernet commands.

ESP-AT Overview

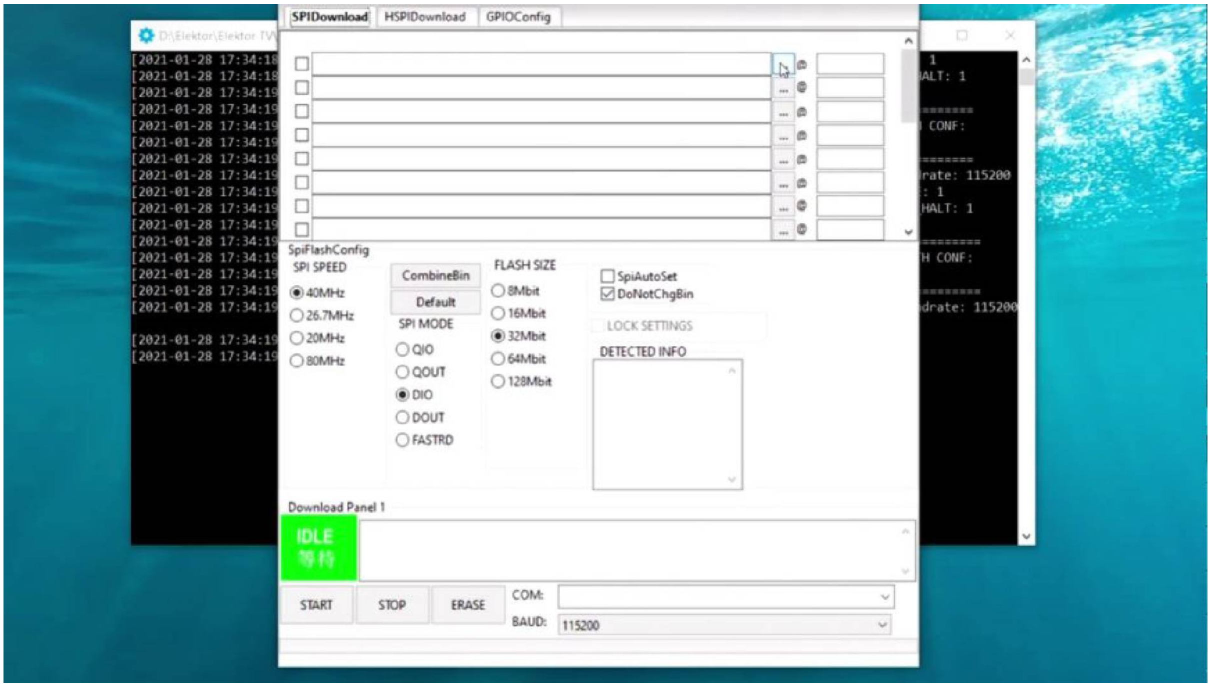
Espressif has developed an extended AT command interpreter called ESP-AT that allows controlling the ESP32 over a serial port by for instance an Arduino or another microcontroller that you already master.



This is nice, as it lets you use the ESP32 without having to program it at all. Setting up ESP-AT can be done with esptool in the same way as MicroPython, except that you must specify the ESP-AT BIN file instead that you downloaded from the Espressif website. However, for those who do not know anything about Python, we will now use the ESP Flash Download Tool instead of esptool. This tool exists only for Windows and can be found at the Espressif website. As a matter of fact, you can use this tool also for programming the MicroPython firmware.


D:\Elektor\Elektor TV\espressif\esp32\flash_download_tool_v3.8.5\flash_download_tool_3.8.5.exe

Launch the program, click Developer Mode and then ESP32 DownloadTool. Locate the ESP-AT BIN file, which is inside the factory folder of the download.


```
[2021-01-28 17:34:18
[2021-01-28 17:34:18
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19
[2021-01-28 17:34:19

[2021-01-28 17:34:19
[2021-01-28 17:34:19
```

```
 1
ALT: 1
=======
 CONF:
=======
rate: 115200
: 1
HALT: 1
========
H CONF:
========
drate: 115200
```

SpiFlashConfig

SPI SPEED: ● 40MHz ○ 26.7MHz ○ 20MHz ○ 80MHz

SPI MODE: ○ QIO ○ QOUT ● DIO ○ DOUT ○ FASTRD

FLASH SIZE: ○ 8Mbit ○ 16Mbit ● 32Mbit ○ 64Mbit ○ 128Mbit

□ SpiAutoSet
☑ DoNotChgBin

Download Panel 1: IDLE 等待

COM:
BAUD: 115200

Enter ‘0’ in the address field and do not forget to check the box in front. Make sure that DoNotChgBin is checked too. Select the COM port and set the speed to something fast. Finally, click the Start button and wait until programming is finished. ESP-AT uses a second serial port for the AT commands. To try it with a computer you need a serial-to-USB converter and open a second serial terminal to enter the AT commands and see the responses. Make sure to terminate commands with a return AND a line feed: CR+LF. Now you can use the ESP32 as a Wi-Fi or Bluetooth add-on for another microcontroller. A third way to work with the ESP32 is with the Arduino IDE.

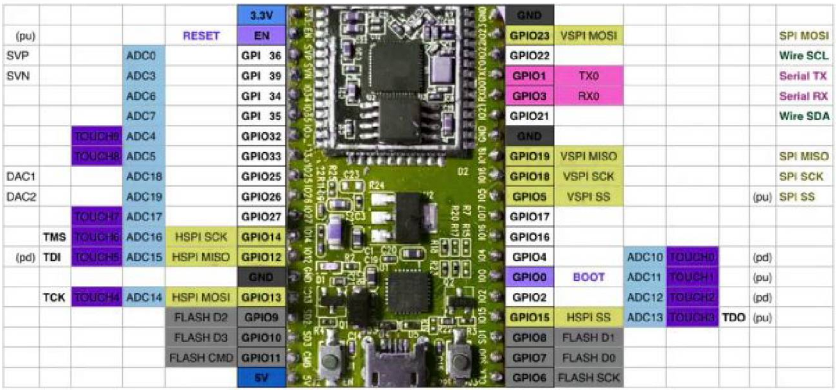
Arduino core for the ESP32

Need help or have a question? Join the chat at

Contents

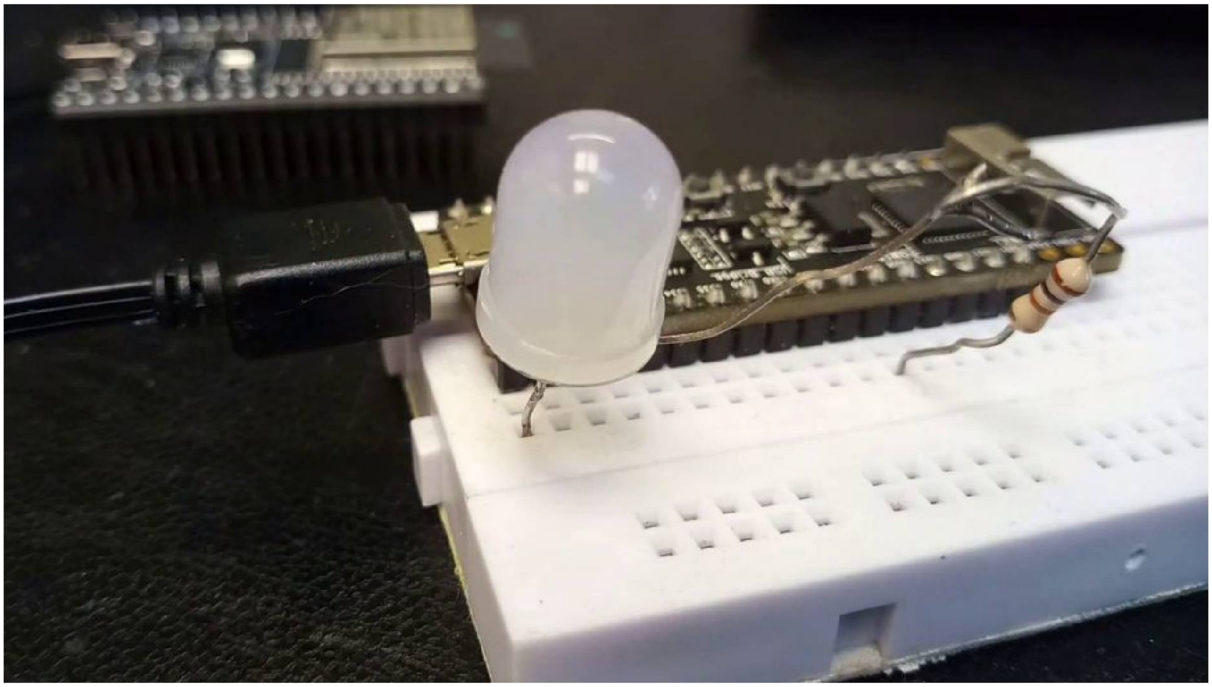
- Development Status
- Installation Instructions
- Decoding Exceptions
- Issue/Bug report template
- ESP32Dev Board PINMAP

This gives you access to the huge collection of libraries and applications developed for Arduino. Basically, it turns the ESP32 into an Arduino board with Wi-Fi and Bluetooth, simplifying application development quite a lot.



					3.3V	GND						
(pu)				RESET	EN	GPIO23	VSPI MOSI					SPI MOSI
SVP			ADC0		GPI 36	GPIO22						Wire SCL
SVN			ADC3		GPI 39	GPIO1	TX0					Serial TX
			ADC6		GPI 34	GPIO3	RX0					Serial RX
			ADC7		GPI 35	GPIO21						Wire SDA
		TOUCH9	ADC4		GPIO32	GND						
		TOUCH8	ADC5		GPIO33	GPIO19	VSPI MISO					SPI MISO
DAC1			ADC18		GPIO25	GPIO18	VSPI SCK					SPI SCK
DAC2			ADC19		GPIO26	GPIO5	VSPI SS				(pu)	SPI SS
		TOUCH7	ADC17		GPIO27	GPIO17						
	TMS	TOUCH6	ADC16	HSPI SCK	GPIO14	GPIO16						
(pd)	TDI	TOUCH5	ADC15	HSPI MISO	GPIO12	GPIO4		ADC10	TOUCH0		(pd)	
					GND	GPIO0	BOOT	ADC11	TOUCH1		(pu)	
	TCK	TOUCH4	ADC14	HSPI MOSI	GPIO13	GPIO2		ADC12	TOUCH2		(pd)	
				FLASH D2	GPIO9	GPIO15	HSPI SS	ADC13	TOUCH3	TDO	(pu)	
				FLASH D3	GPIO10	GPIO8	FLASH D1					
				FLASH CMD	GPIO11	GPIO7	FLASH D0					
					5V	GPIO6	FLASH SCK					

Note that the ESP32 Arduino Core can also be used with for instance Platform IO. Setting up the Arduino IDE for ESP32 is easy. It starts by adding a repository to the Boards Manager of the Arduino IDE. You do that in the IDE’s Preferences window. Then open the Boards Manager, search for the ESP32 package and install it. You can now choose an ESP32 board from the Boards menu. As there are many possibilities, chances are that your board is listed here. If it isn’t, don’t worry, just choose the first entry and then customize it from the Tools menu. The default values will work in most cases. You can now try out the examples or write your own program like I did to blink an LED with an ESP32 Pico Kit board.



The most advanced way for ESP32 application development is ESP-IDF. This gives you full control over everything, but it is also the most complicated. Luckily, installing it is easy, especially on Windows as a nice installer package is available. Application development is done in a Bash terminal or command prompt window, but an Eclipse plugin is available too. A project is built in a few steps. After creating a project folder, you must set the target chip and configure the project with the 'menuconfig' tool. Then build it, which takes a while the first time, and upload the executable to the board. If you specify the 'monitor' option, a serial terminal will open that displays program status. We will stop here. Summarizing, I have shown you four ways to get started with the ESP32 microcontroller from Espressif, from simple AT commands and MicroPython to Arduino and C/C++ programming with the full-fledged ESP-IDF environment.

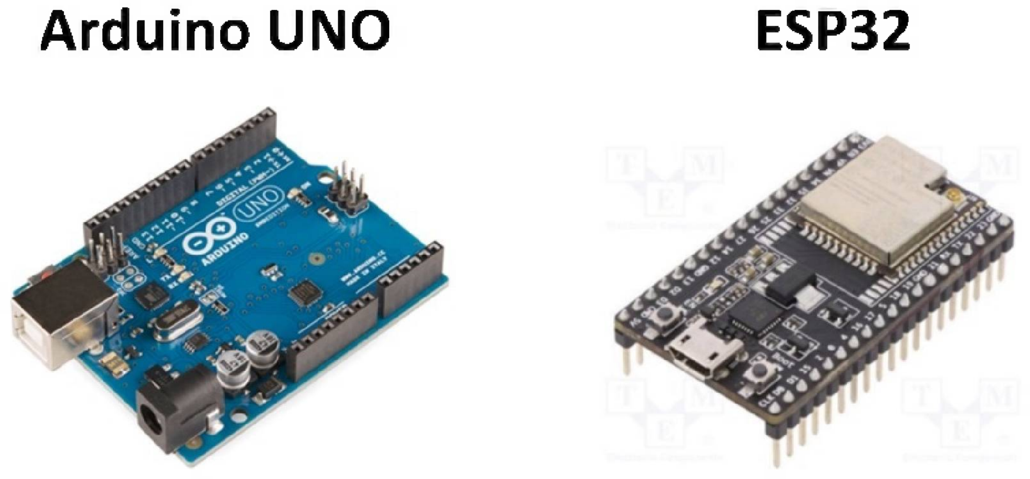
WHAT IS ESP32

What is ESP32? Features of ESP32, How to choose the right ESP32 board Why choose this board for IoT applications you should have heard enough about the ESP32 chip and from its specification it looks like a chip of the future which will help you build anything. Espressif has always been at the peak of the IoT market after releasing its game changing chip, the ESP8266.

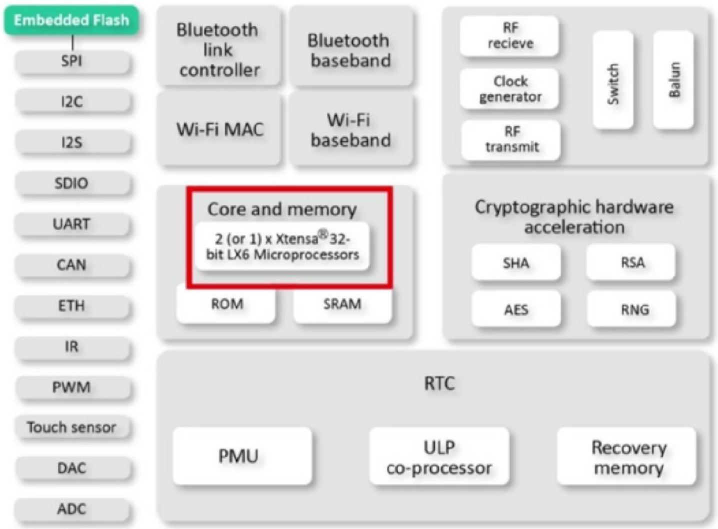
ESP8266

ESP WROOM-32

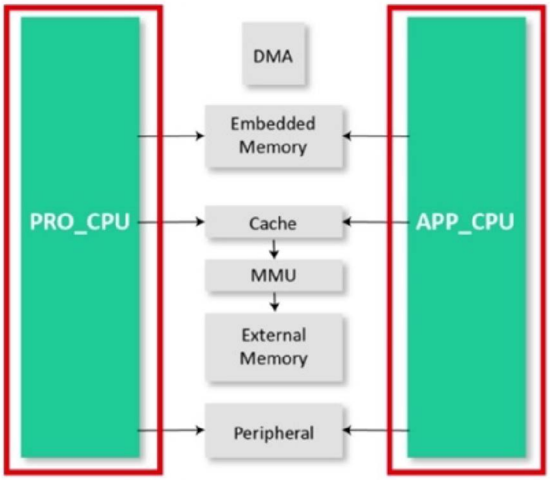
I would not think twice to mention that the ESP32 is an ESP8266 on steroids. The ESP32, designed by espressif systems, is a low cost, low power system on chip well suited for different IoT applications.



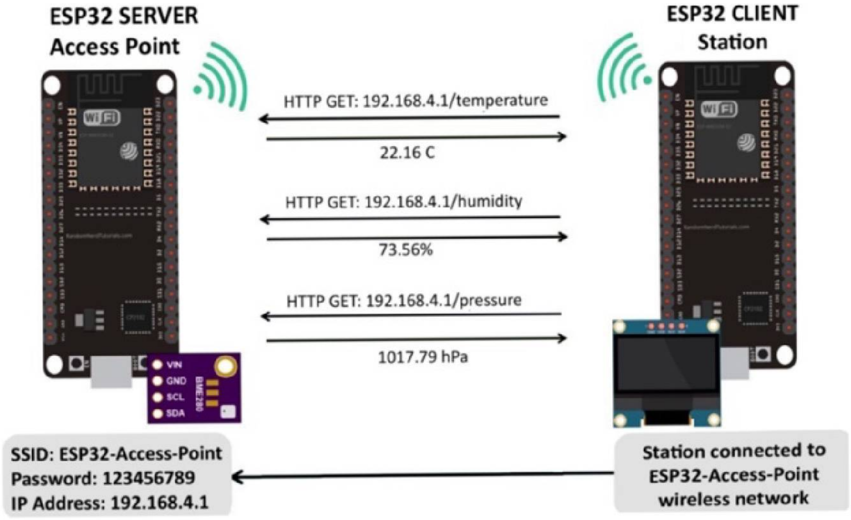
It is one of the best solutions for projects including wearables, home automation and weather stations with a lower price. But more power than an Arduino UNO. The ESP32 can be deployed in many fields, giving you a supercharged faster and better performance than other chips in many aspects.



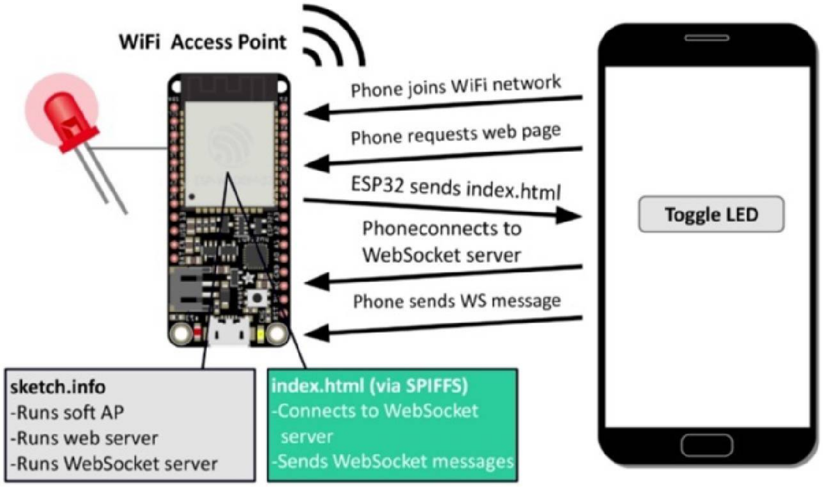
The ESP32 has two Tensilica Xtensa 32-bit LX6 microprocessors, which can run ten times faster than the Arduino board at 160 to 240 megahertz. The core feature that made the ESP32 a widely popular board is its wireless capabilities, specifically the inclusion of Wi-Fi and Bluetooth at a very low cost. The second important feature that made ESP32 what it is today is its dual core processors.



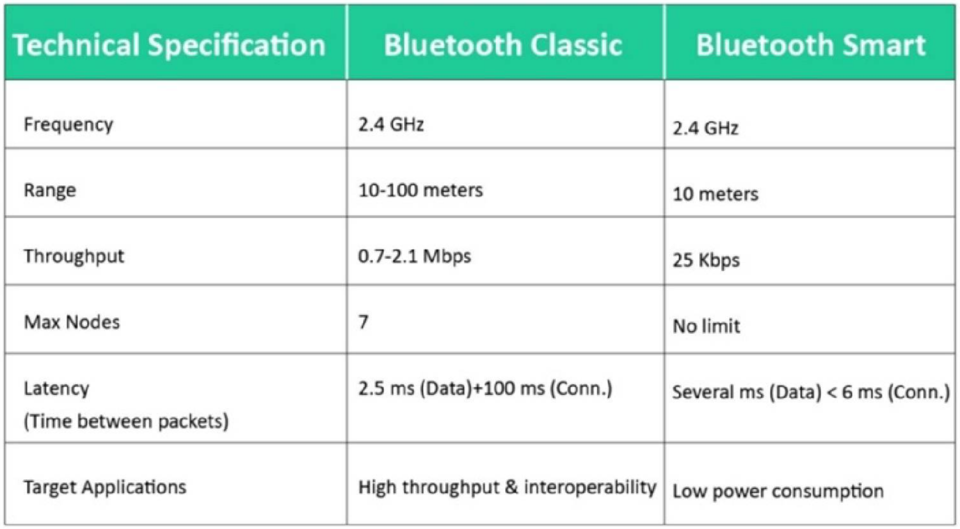
The two cores are named Protocol CPU or Pro CPU and Application CPU or APP CPU. The PRO CPU is responsible for handling Wi-Fi, Bluetooth and other internal peripherals like SPI, I2C, ADC and more. The APP CPU handles the application code. The differentiation is done using Espressif's internal development framework, commonly known as the ESP IDF The ESP IDF is an official software development framework for the chip. Other implementations such as the Arduino IDE for ESP32. are also based on the ESP IDF. The ESP32 uses TCP/IP, 802.11 WLAN MAC protocol This means that the ESP32 can communicate with most of the WiFi routers out there when used in client mode.



ESP32 also supports WiFi direct. WiFi direct is a good option for peer-to-peer communication without the need for an access point.



WiFi direct is easier to set up and configure and generally has higher data transfer speeds as compared to Bluetooth. This feature potentially helps to develop ESP32 projects from a phone or tablet that supports Wi-Fi direct.



Technical Specification	Bluetooth Classic	Bluetooth Smart
Frequency	2.4 GHz	2.4 GHz
Range	10-100 meters	10 meters
Throughput	0.7-2.1 Mbps	25 Kbps
Max Nodes	7	No limit
Latency (Time between packets)	2.5 ms (Data)+100 ms (Conn.)	Several ms (Data) < 6 ms (Conn.)
Target Applications	High throughput & interoperability	Low power consumption

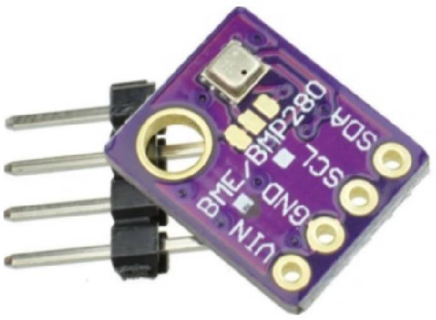
Along with its WiFi capabilities, The ESP32 also supports the latest BLE Bluetooth 4.0 and the classic Bluetooth. This makes it a great fit for designing devices that work with both existing and new phones in the market. The BLE makes it possible to consider the ESP32 as a right candidate in the making of smartwatches or wearables, health based sensors, beacons and more. If you want to learn more about BLE, Security is a big issue, hindering the progress of IoT. The ESP32 is built, keeping this in mind, it consists of a cryptographic hardware accelerator. The cryptographic hardware accelerator is a core processor designed specifically to perform computationally intensive cryptographic operations far more efficiently than the general-purpose CPU. This ensures that the security implementations are efficient.

The ESP32 can be used as a standalone module or as a development kit.

Some of the development kits available in the market are ESP Wroom 32 DevKit, Adafruit Huzzah 32, SparkFun ESP32 Thing. Lolin 32. So how do you know which ESP32 board is suitable for you? While selecting an

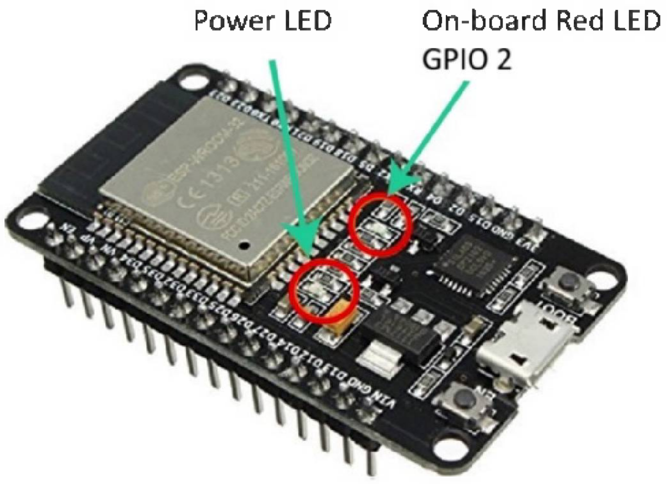
ESP32 board there are various aspects to consider. The most important thing is to check out the PIN configuration. You need to have proper access to the board pin out or else you will use the board incorrectly. The second important thing is to check if the ESP32 board has a proper USB to UART interface and a voltage regulator circuit. Most of the development boards will have these features in built in them. It will make your project development smoother. A boot and reset button on the board with a couple of LEDs to indicate particulars always help in debugging battery connectors and battery management is a very useful feature in the IoT domain. And some ESP32 development boards also come up with battery connectors for Lipo batteries. Finally, you should always keep in mind the application you want to use the ESP32 board. Some boards have extra features such as an OLED display, a built-in LoRa module or a camera.

BME 280 Sensor

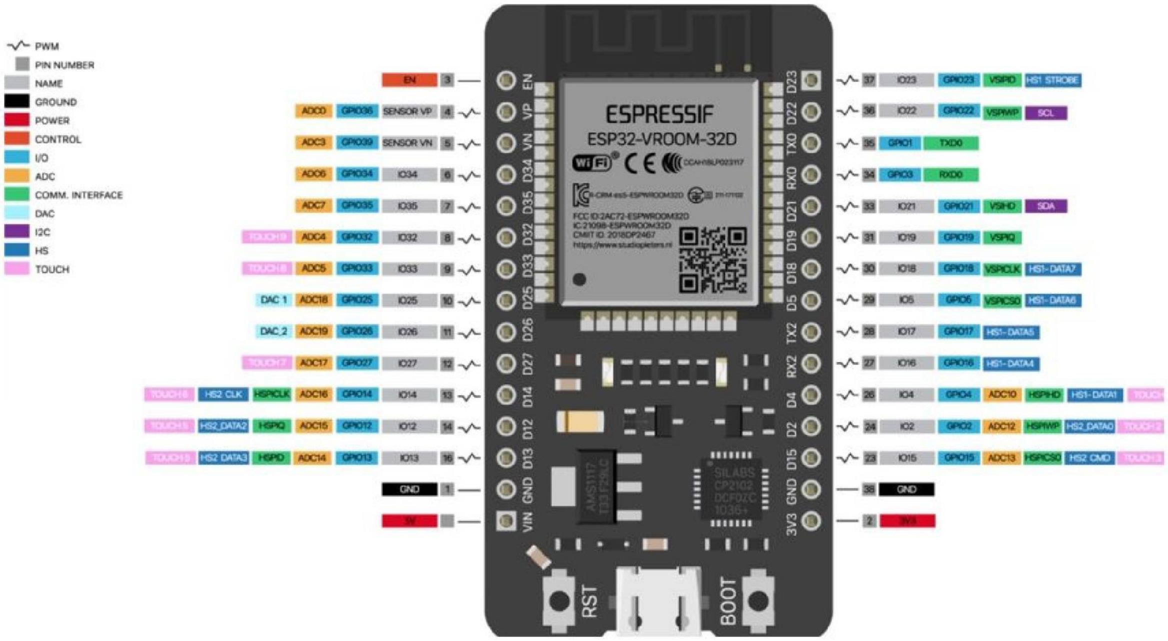


LCD Display

With the ESP32, you can interface a wide range of sensors and actuators to make home automation projects like a door sensor or a smart water pump. You can also manage devices securely and remotely due to its WiFi and Bluetooth connection capabilities. Furthermore, using cloud services like Amazon AWS in conjunction with the ESP32 will give you an edge on making solutions for complex problems.



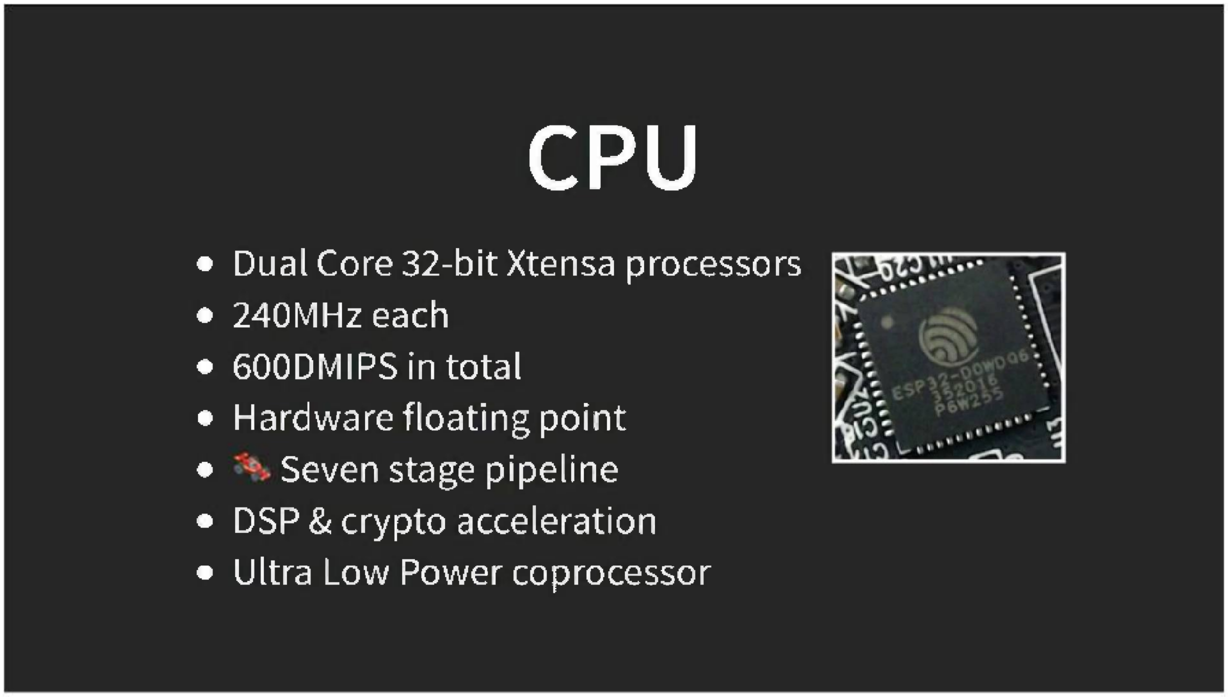
The board has access to 30 GPIO pins and it has on board debugging buttons and on board SMD LEDs. It also has a proper USB to UART port which can flash the code and can be used for power.


PWM
PIN NUMBER
NAME
GROUND
POWER
CONTROL
I/O
ADC
COMM. INTERFACE
DAC
I2C
HS
TOUCH
EN 3
ADC0 GPIO36 SENSOR VP 4
ADC3 GPIO39 SENSOR VN 5
ADC6 GPIO34 IO34 6
ADC7 GPIO35 IO35 7
TOUCH9 ADC4 GPIO32 IO32 8
TOUCH8 ADC5 GPIO33 IO33 9
DAC 1 ADC18 GPIO25 IO25 10
DAC_2 ADC19 GPIO26 IO26 11
TOUCH7 ADC17 GPIO27 IO27 12
TOUCH6 HS2 CLK HSPICLK ADC16 GPIO14 IO14 13
TOUCH5 HS2_DATA2 HSPIQ ADC15 GPIO12 IO12 14
TOUCH4 HS2 DATA3 HSPID ADC14 GPIO13 IO13 16
GND 1
5V
EN
VP
VN
D34
D35
D32
D33
D25
D26
D27
D14
D12
D13
GND
VIN
ESPRESSIF
ESP32-VROOM-32D
D23
D22
TX0
RX0
D21
D19
D18
D5
TX2
RX2
D4
D2
D15
GND
3V3
RST
BOOT
37 IO23 GPIO23 VSPID HS1 STROBE
36 IO22 GPIO22 VSPIWP SCL
35 GPIO1 TXD0
34 GPIO3 RXD0
33 IO21 GPIO21 VSIHD SDA
31 IO19 GPIO19 VSPIQ
30 IO18 GPIO18 VSPICLK HS1-DATA7
29 IO5 GPIO5 VSPICS0 HS1-DATA6
28 IO17 GPIO17 HS1-DATA5
27 IO16 GPIO16 HS1-DATA4
26 IO4 GPIO4 ADC10 HSPIHD HS1-DATA1 TOUCH
24 IO2 GPIO2 ADC12 HSPIWP HS2_DATA0 TOUCH 2
23 IO15 GPIO15 ADC13 HSPICS0 HS2 CMD TOUCH3
38 GND
2 3V3

The project is designed in such a way that irrespective of the type of ESP32 board used, the projects and the code will work seamlessly.

ESP32 - MICROCONTROLLER HARDWARE AND SOFTWARE

So the ESP 32 is the next generation chip from the ASPA. 266 was announced about a year and a half ago and it's been available. Patchouli for about the last three months. this picture is a module that incorporates the ASP 32 with some of the supporting components to make it go. So you can solve to that down onto a PCB. That's the ESP w room 32. And that's what we used on Iotas. actually, I should also mention here that, especially if we're kind enough to donate the SPW rooms that we have on those boards, which was great because they kind of hard to get hold of at the moment.



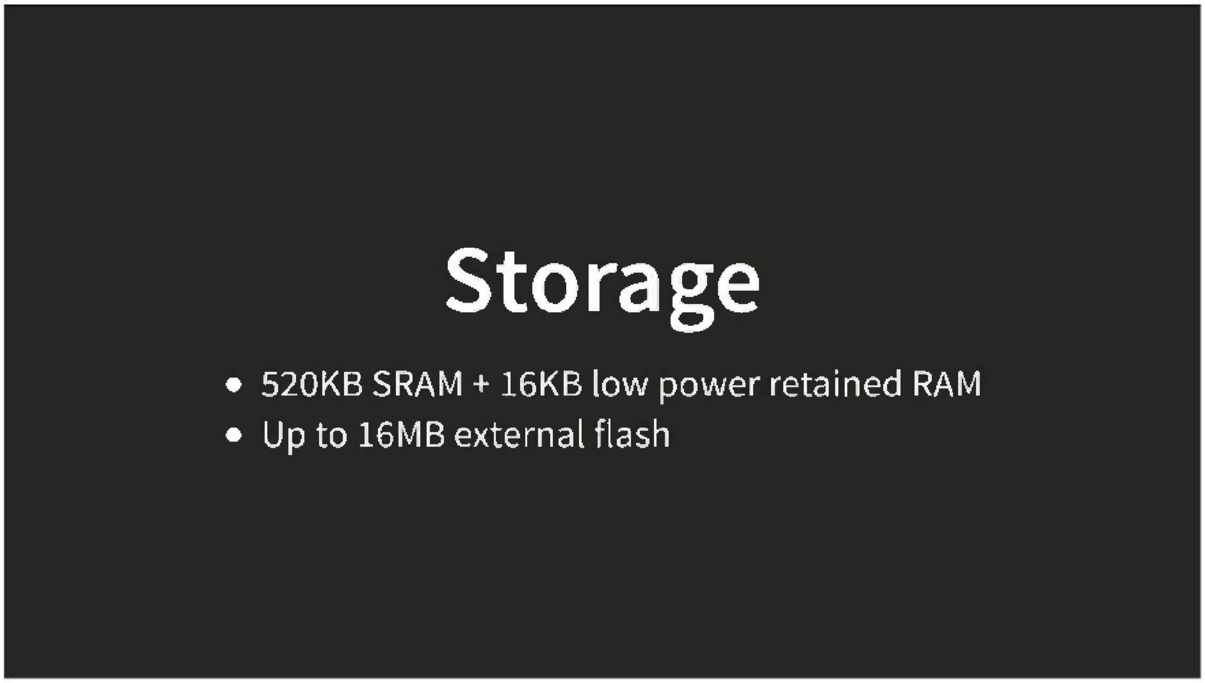
CPU

- Dual Core 32-bit Xtensa processors
- 240MHz each
- 600DMIPS in total
- Hardware floating point
- 🏎 Seven stage pipeline
- DSP & crypto acceleration
- Ultra Low Power coprocessor

ESP32-D0WDQ6
322016
P6W255

Similar CPU to the ASPA 2 66. the main processor in the ESP 32 is also an extensive 32 bit processor. It's a little bit beefed up. It's a jewel core. The cause can run it up to 240 megahertz. H I think the calculation is 600 drystone rips of processing power as hardware flooding point. It has a fairly large CPU fetch pipeline. So it's got a seven stage pipeline, which is unusual in microcontrollers, but it's usually microcontrollers that have to run at 240 megahertz that's to do with how many instructions can be queued up in the pipeline to run on the CPU and it's necessary to get good performance at high clock rate. If your memory is not particularly fast. So to compare this to sort of previous generation chips, the NIPT IVR has a two-stage pipeline. 4 86 desktop CPU has a five stage. there are some extra instructions for doing some DSP, like operations you can do in multiplication and division in a single cycle. and there's some, built-in, it's actually not in the CPU, but there's some support for accelerating, cryptographic operations, because that was one of the things on ESPN 266. If anybody tried to bring up a TLS connection or something, it was pretty slow. So you can save power by doing that. The ultra low power one is a hundred design from expressive, so it doesn't use an established instruction set. there's a very simple assembler for it available to them. And there's going to be a bin new tales, like normal JCCA assembler available soon. So it'd be able to program with that. One of the things to mention about the CPU's is that, there are actually quite a lot of these kinds of wifi modules or wifi devices, or particularly network devices that use a Juul CPU in a little package for an embedded device. Usually one of the CPU's is reserved for. The network stack and the wifi and you can't usually program it.

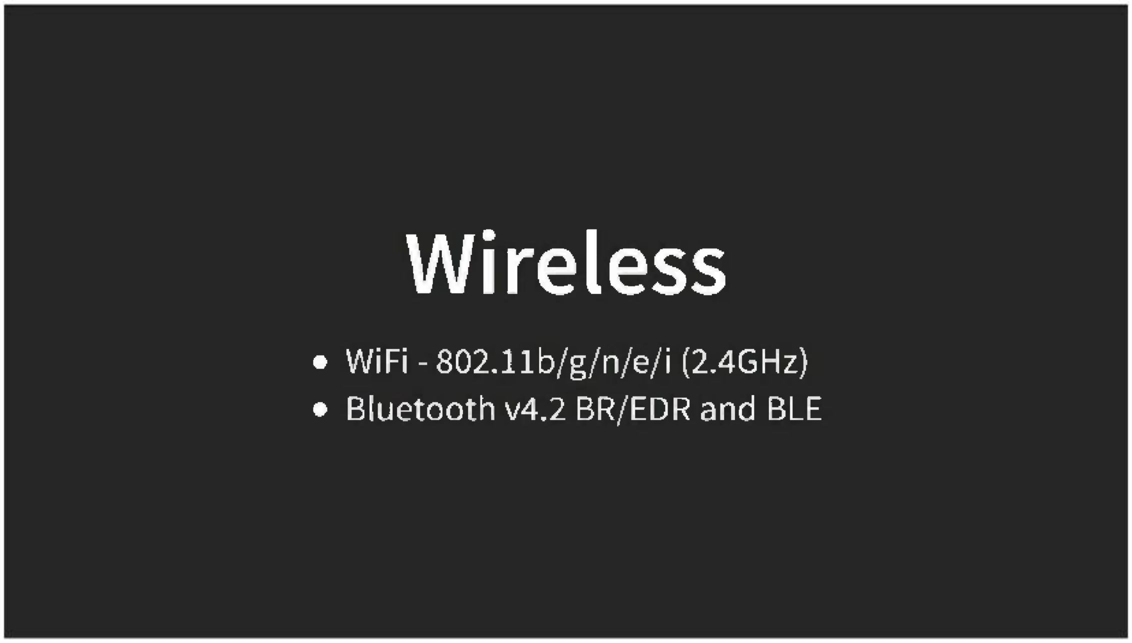
It's a closed black box and the other CPU is available. One of the unusual things about ASP 32 is that you can put your code to run on both processes as you want. So you can choose to have wifi on one core and your stuff on the other core, or you can choose to put everything on one core. And if you've got one particular bit of a central code, you can put that on the other core, or you can have a task scheduled on both of the calls as you sort of want.



Storage

- 520KB SRAM + 16KB low power retained RAM
- Up to 16MB external flash

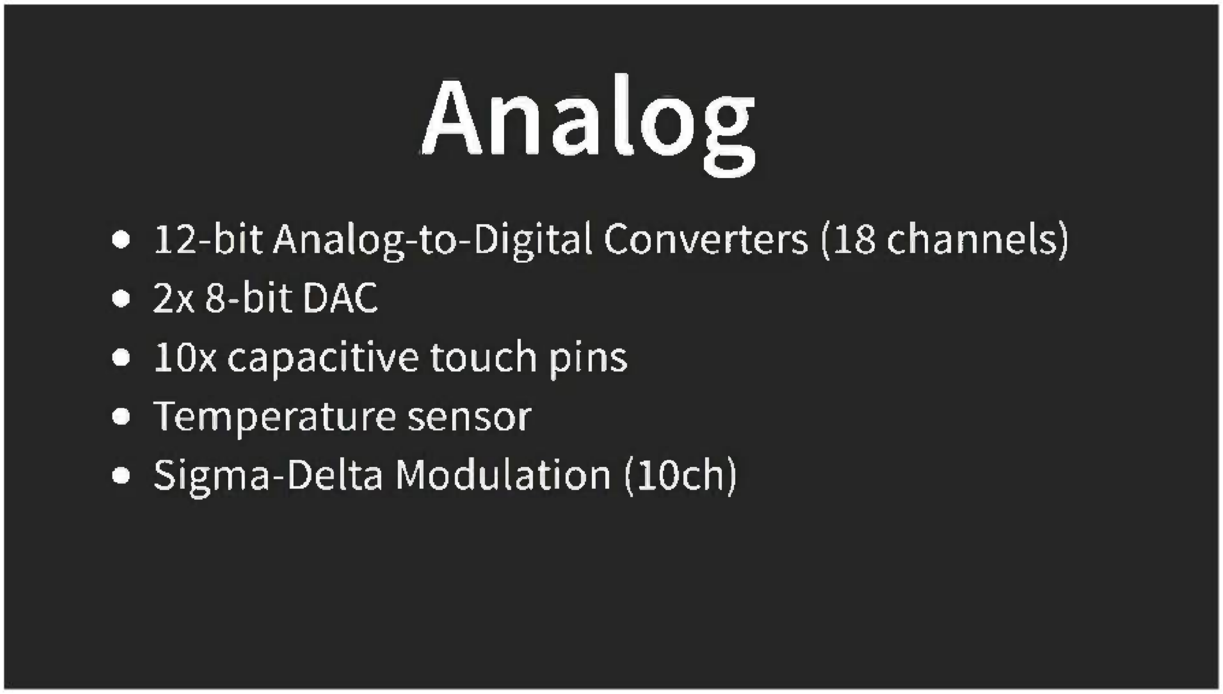
So there's a lot of flexibility there in how you decide to use. Mina storage has gone up. Anybody who worked with ASPA 266, especially if you use like a language like Lula or JavaScript, or even Python, somebody used to running out of memory, you kind of write the code you wanted and you'd run out of memory and then you'd spend the next day shrinking it so that it had run without using too much memory.



Wireless

- WiFi - 802.11b/g/n/e/i (2.4GHz)
- Bluetooth v4.2 BR/EDR and BLE

So that limit has gone up quite a lot. You have a half a megabyte of memory. You can have quite a lot of external flash and. Say from experience. That's pretty nice. You don't have to kind of worry about your overheads quite as much as you did before the chip still does wifi built in, has the Adelaide two 11 BJ and N that the ADA ESPN 266 had, so it was 82 11 E, which is a real-time media streaming extension for doing real-time signaling the wifi also now supports, HT 40. So you can push more, throughput over the network. but the really big improvement in wireless is that there's also Bluetooth. So please just 4.2 does classic Bluetooth and Bluetooth, low energy at the moment. I think there's only software support for a BLE, but there is going to be classic support for a classic Bluetooth.

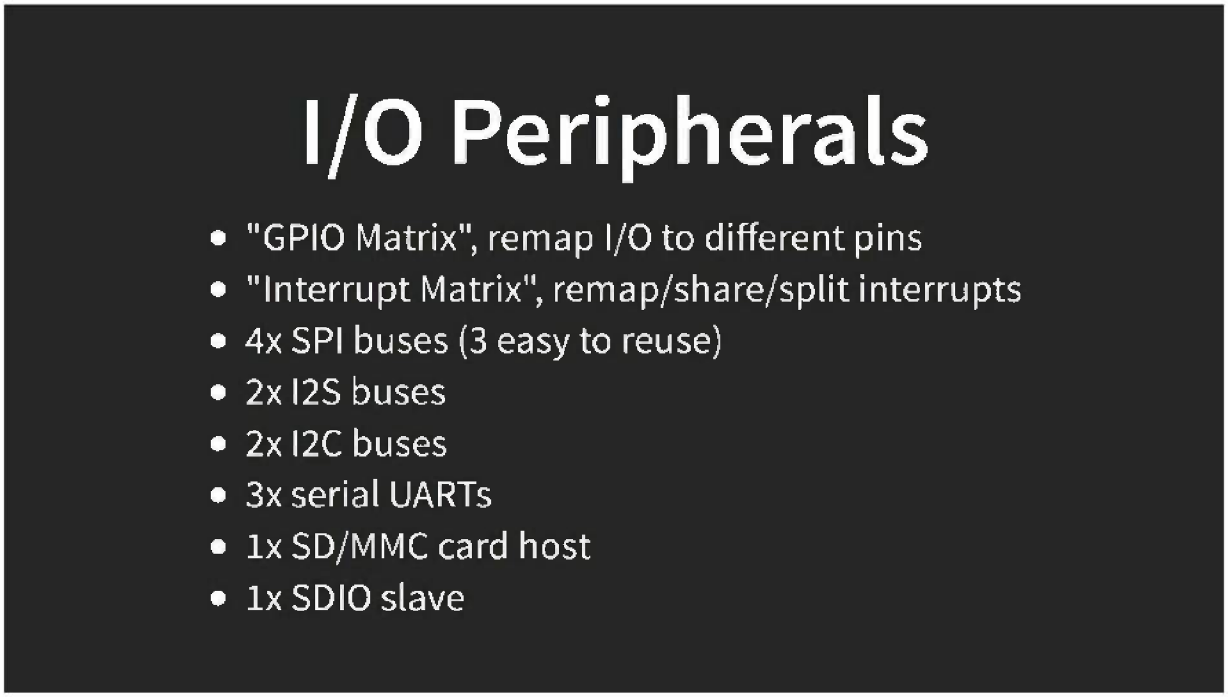


Analog

- 12-bit Analog-to-Digital Converters (18 channels)
- 2x 8-bit DAC
- 10x capacitive touch pins
- Temperature sensor
- Sigma-Delta Modulation (10ch)

I believe like audio networking, profiles, all that kind of thing. And you can run the wifi stack on the Bluetooth stack at the same time. Another thing that kind of probably, if anybody used an ASPA 266, it didn't have a lot of analog features for like reading voltages and stuff like that. They had 180 say that could rate up to 1.1 volt.

And even then only in certain circumstances, the SB 32 has an 18 Juul 80 say with up to 18 channels, 12 bit conversion. So you can read a lot of voltages. a lot easier than before. And they go to the full range of 3.3 volts, the same as everything else on the chip. it has deck, it has capacitive touch as a temperature sensor. It has a Sigma Delta modulation output, which is a way to do something kind of like a deck by doing a very, very high frequency on and off. And then you can filter that to produce a deck output. It's how, if you've ever had a class day amplifies, it's sort of a similarity to how they work. There's tons of IO.

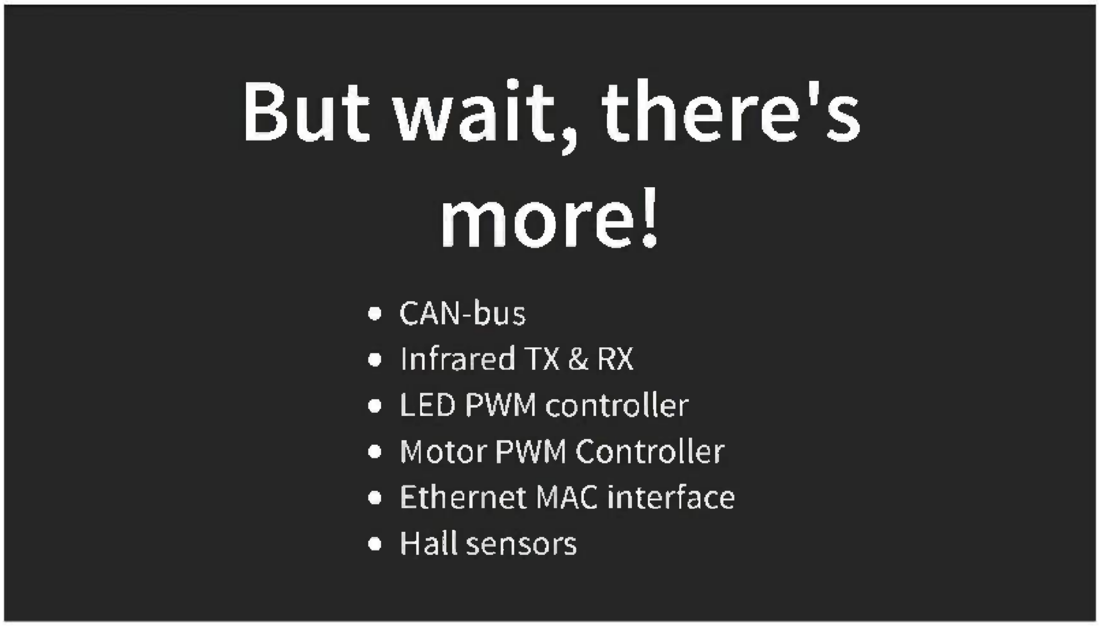


I/O Peripherals

- "GPIO Matrix", remap I/O to different pins
- "Interrupt Matrix", remap/share/split interrupts
- 4x SPI buses (3 easy to reuse)
- 2x I2S buses
- 2x I2C buses
- 3x serial UARTs
- 1x SD/MMC card host
- 1x SDIO slave

again, it's, kind of annoying sometimes, cause you only had a few pins. Certain peripherals had to be on certain pins. So you couldn't use all combinations. There's a GPI matrix crossbar. So you can choose where you want to put your pins. you don't have to look at the data sheet necessarily and say, okay, I can only use these two pins as these three pins for SBI. I tell you, I allocating their, you can say, well, I want to use these pins. Cause that's easier because I want to use that other thing for some. As an interrupt matrix as well, which is more of an internal feature to share and trumps between the cause and allocate the interrupts. Lots of flexibility. There there's a lot of ways to plug in connect things to ASP 32 that's by bus, forced by buses, which three are fairly easy to use.

And one of them is tricky. There's, two I two S buses. That's an audio data bus that was on the ASPA 2 66, but it was hard to use. and the I two S has a bunch of extra features. So. You just kind of, misuse it to do some other things. Like there's a demo project where somebody hooked up a camera, with a parallel interface and read all of the ports through the IQs interface. there's two hardware. I squared C buses. I squared C is a really common way to connect sensors to a microcontroller. for example, the Iotas has, I think, and I squared saved bus with full things on it. They, pressure and temperature and humidity sensor has an expanded pin. There's, an IRS deck that you can figure out.

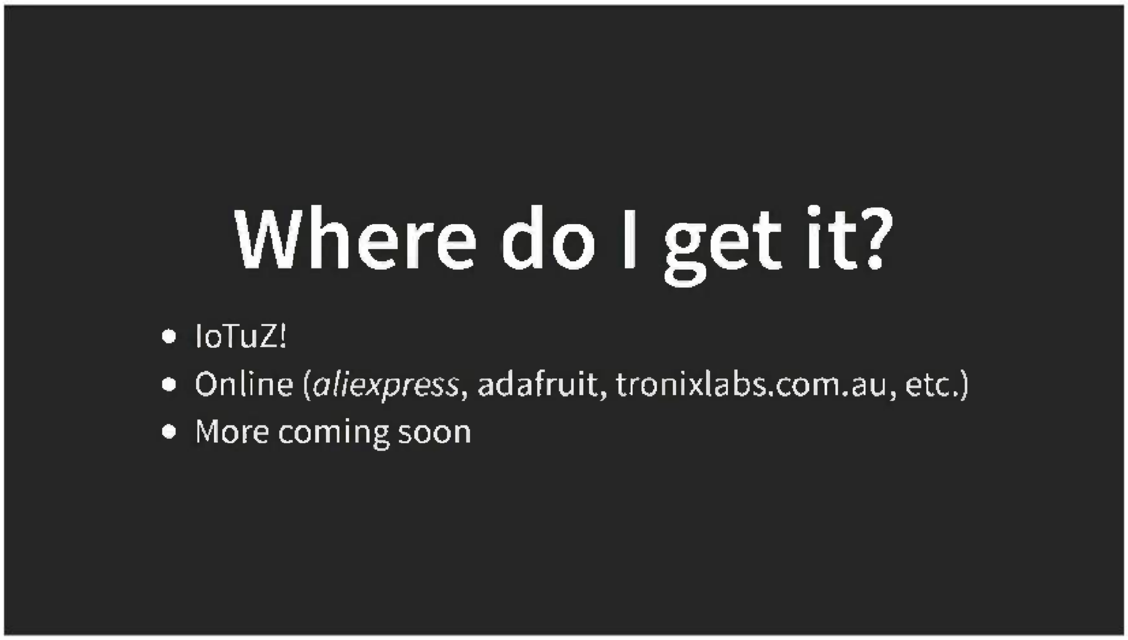


But wait, there's more!

- CAN-bus
- Infrared TX & RX
- LED PWM controller
- Motor PWM Controller
- Ethernet MAC interface
- Hall sensors

It's something else that I can't think of right now. yeah, the serial bullets, basically all of the IO that you could want. this has done to sound a little bit like a, late night television spiel, but there really is more, a lot of stuff got thrown in here is this canvas interface for automotive connections. There's an infrared transmitter and receiver for talking to sort of, this is supposed to be just for talking to sort of televisions and things like that. Sending I R remote credits, but it also kind of happens. What it is, is an engine for sending any stream of digital on and off pulses as a strain.

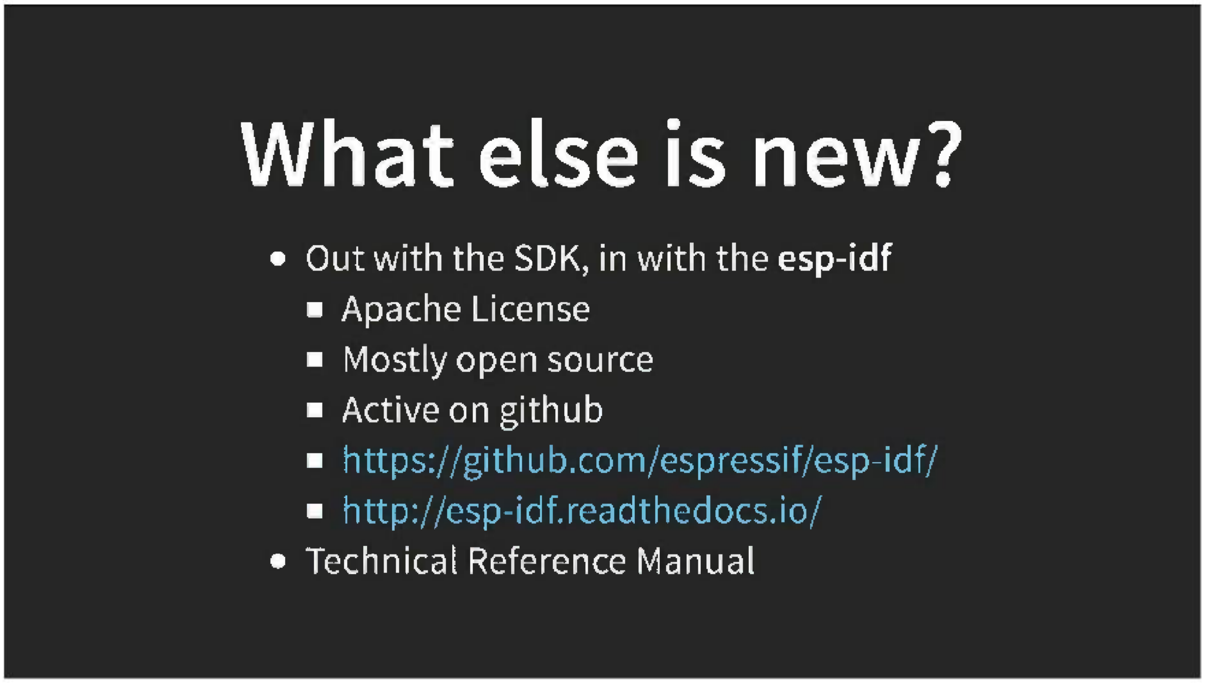
And there's actually a whole lot of places that that's useful. So somebody's already taken that remote control and device and worked out a way to drive a Neo pixel type WDS 20 I 12 LA days from it. So you can just it in a stream of pulses and program. Those, and that is a lot easier than on a 266 again, where you had to do that with software and software timing. And it was a little bit finicky deal hack. One of the other ones. there's various ways to make pulses for LA days. And murders is an ethernet interface. So you can have a wifi, Bluetooth AceNet internet of things, device, if that's what you want. there's a whole sensor, although I don't think there's software for the whole sensor yet.



Where do I get it?

- IoTuZ!
- Online (*aliexpress*, adafruit, tronixlabs.com.au, etc.)
- More coming soon

a lot of people asking where can they can get this thing. if you came along today, then you've got the IOT. So you have one already to take time. you can get them from sort of the usual online places where you bought . I think a lot of people have had bad experiences with Ali express and some of the other pre-orders there was some vendors that took pre-orders and never shipped. but that situation is hopefully over now.

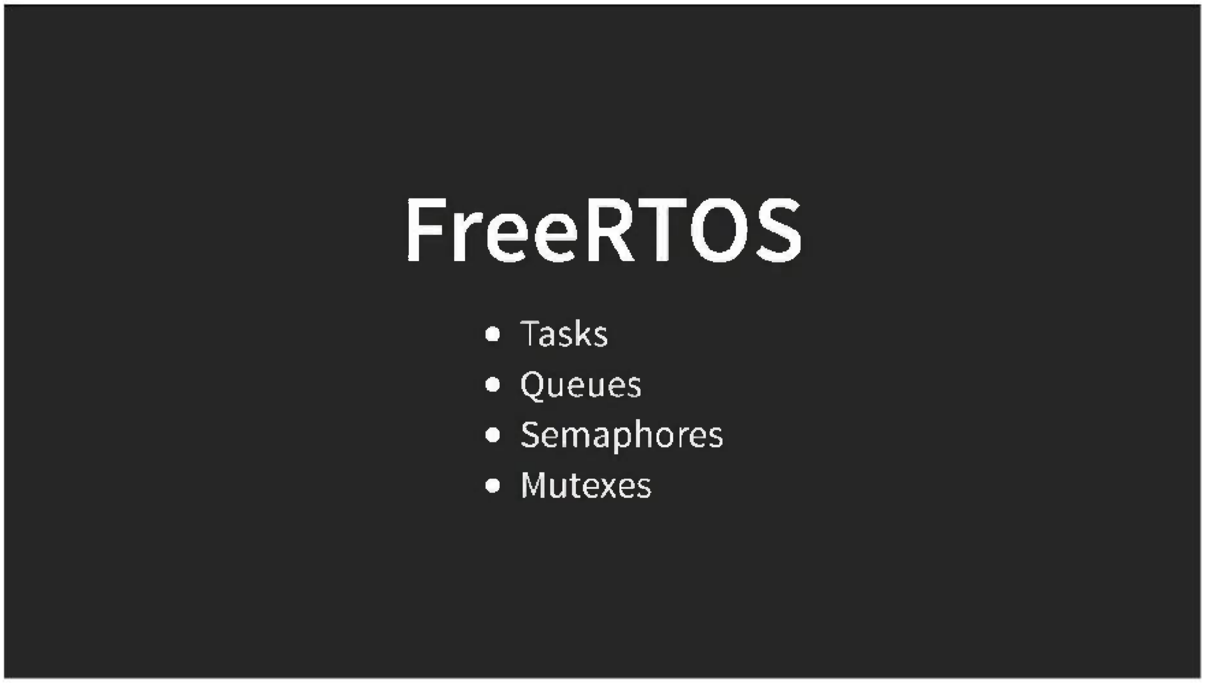


What else is new?

- Out with the SDK, in with the **esp-idf**
    - Apache License
    - Mostly open source
    - Active on github
    - https://github.com/espressif/esp-idf/
    - http://esp-idf.readthedocs.io/
- Technical Reference Manual

So that kind of is a real whirlwind tour of the hardware side. There's just so many phages. You kind of can't get into that much depth on the mall. but there's also quite a lot new on the software side, which I think is really exciting. the old days had two SDKs, there was a nano SSD. Where you sort of write your program, kind of like Arduino and you use callbacks to do networking. And there was a real time operating system. One where you could have multiple tasks and they could communicate with each other and then built on top of those. There were a lot of other ways to program at like Arduino and, smearing and various other interfaces. The new chip, because it's has dual calls already. So has concurrency always uses a real-time operating system. we named to avoid confusion. We've named that ESP IDF, which is the IOT development framework, really just because we didn't want to have yet another SDK. There were so many SDKs, it's under the Apache license, everything above the wifi layer is open source.

So you can go and get it on, get up. we plan to open source. There's still a few, a few little bits that are still in binary around that because we haven't moved it out yet. But the plan is to really only leave the low-level wifi. Connectivity, which is quite, it's quite chip specific to begin with, and quite proprietary in binary. And we want to put the rest under an open-source license and we're using GitHub pretty actively. we pushed to get hub from our internal get servers. Every couple of days, we accept four requests. We try to interact with people as best as we can and, and be good, good hub citizens. And there's also a technical reference manual, which was conspicuously missing on a 266. So the hardware is actually. there are some chip chapters that are still missing or coming, but it's being added to every month or two. And you can go and look up actually what the peripheral does, what the register map is, all of that information that you didn't get with ESPN 266, necessarily. So using this plan, if you're gonna use your patio after program and you have to, how do you actually go about during this, the chip, as you probably realized, there's a lot more complicated than the 8 266 there's lots of stuff.

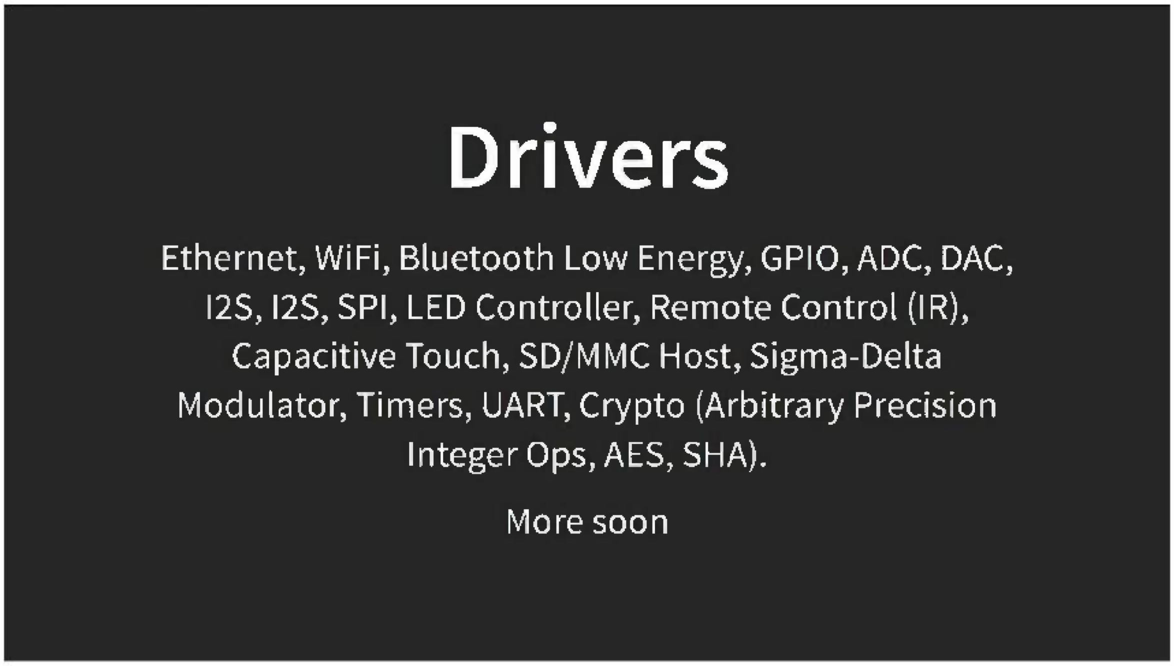


FreeRTOS

- Tasks
- Queues
- Semaphores
- Mutexes

it's not as complicated as Lennox house. but it is a lot more complicated than sort of simple microcontrollers. And the idea is that if you've got good distractions, then you can program for that without having to know everything down to the base level. And that's that goal. So one of the big ways to do that is free autos, which is a real-time operating system.

real-time operating system is a bit different to a general purpose operating system, but it gives you a framework where you can write programs that talk to each other and share data. you have tasks which are basically the same as. On a normal operating system, with some differences in scheduling, but the same general idea that you can have multiple things running concurrently. I mean, it's PDF, you can put those on one core or the other core, or you can allow them to be scheduled on whichever call is free. the threads can send data to each other and it's sort of safe way through cues. You push data into queue and one task evaded in the other task. you can do semaphores in new Texas, which is sort of basic ways to signal between tasks. at the moment is PRDF only supports three autos. There are some other artists is coming. at some point in the future, we're going to be able to support other ones and not X they're not X operating system. Everybody's interested in. This has a support for ASB 32 already. You can boot that. I'm not ex I think the peripheral support is still a work in progress, but if you're just doing not X, check that out. And the other big obstruction that real-time operating system lets you.



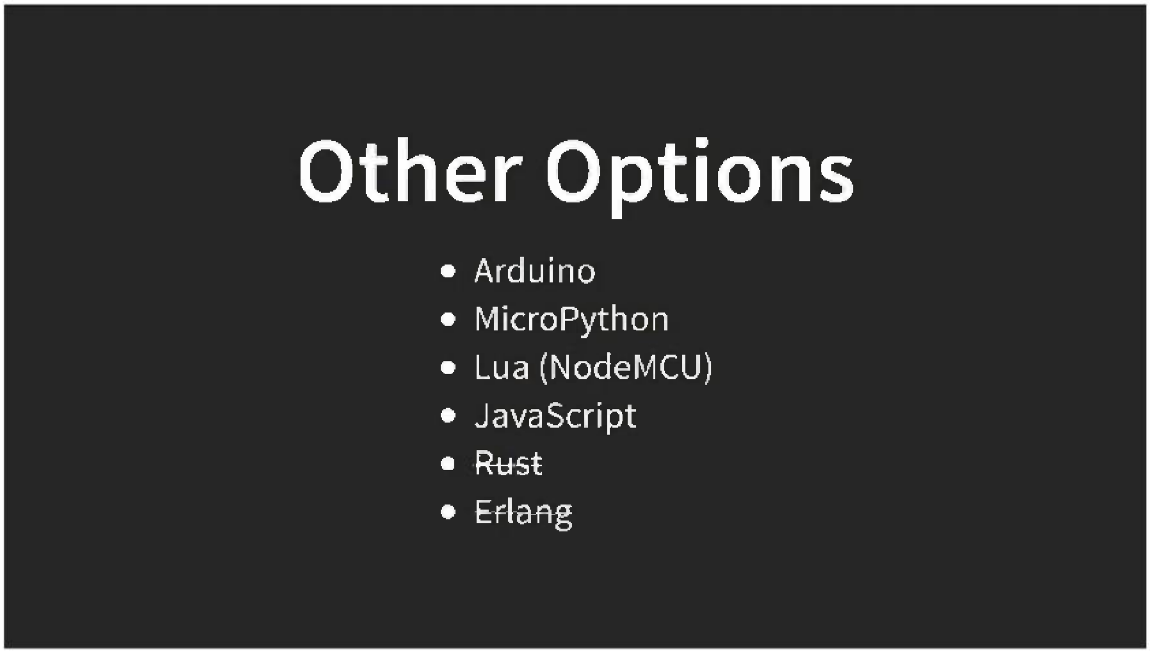
Drivers

Ethernet, WiFi, Bluetooth Low Energy, GPIO, ADC, DAC, I2S, I2S, SPI, LED Controller, Remote Control (IR), Capacitive Touch, SD/MMC Host, Sigma-Delta Modulator, Timers, UART, Crypto (Arbitrary Precision Integer Ops, AES, SHA).

More soon

schedule the tasks and talk to each other and do basic computing. It doesn't really give you anything for interacting to the outside world. It's kind of just a pure, computing kind of set of instructions. So the other restriction is drivers on his PRTF has drivers already in it for these peripherals and other peripherals of more or less on their way stuff is landing every couple of days. So most of the major features in hardware have a driver that you can use rather than having to poke at the hundred.

And also examples. There's plenty of examples there. if you think that there's an obvious bit of functionality that doesn't have an example, let us know, and we'll write an example for it. So you can take the example card and play with that. And if you look in the ASPI DF examples, directory, that's where the examples of the project are.



Other Options

- Arduino
- MicroPython
- Lua (NodeMCU)
- JavaScript
- ~~Rust~~
- ~~Erlang~~

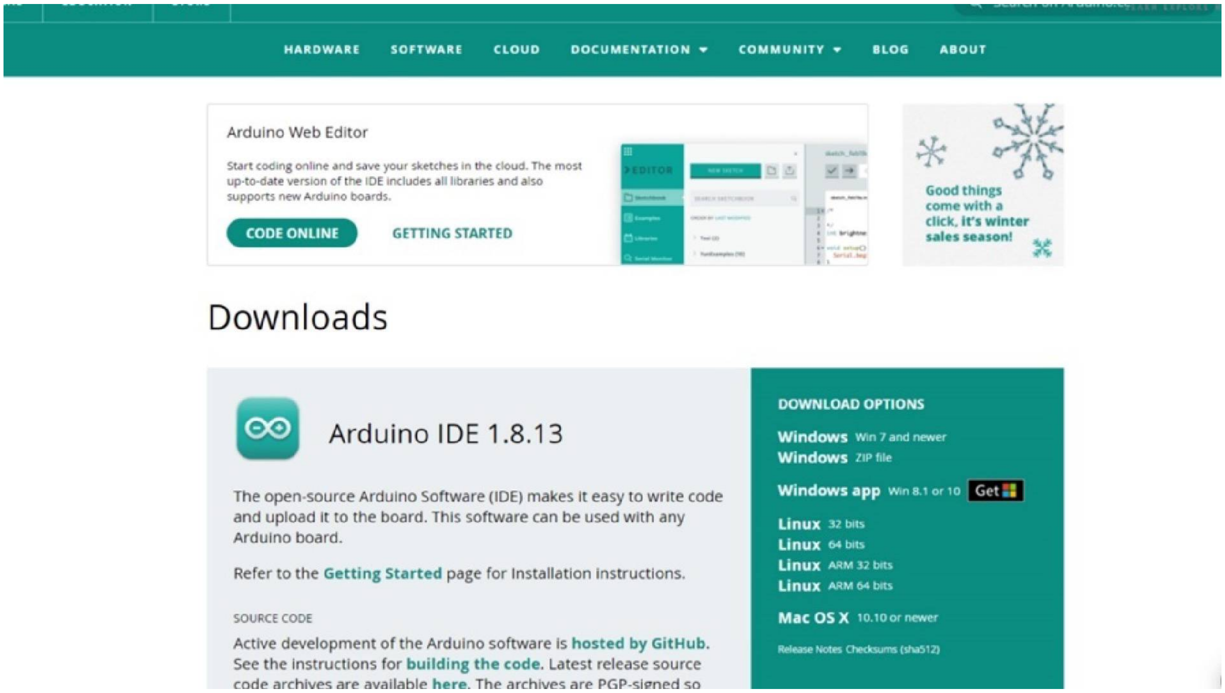
There are also a bunch of other programming options. If you're brand new to embedded, then Adriano is very popular and very good in very. Straightforward way to work with, microcontrollers, including ASP 32, you can also use the Arduino libraries in ASPI DF, because they built on top of it. And that's something that the Ayatollahs software is using. And I think mosque's going to talk a little bit more about strategies for joining those two together. So you can't even take, Iotas has a couple of Arduino libraries written for the Arduino environment that it just pulls in and users, in the ASPI DF environment. So it gives you a one. To match things up.

Go jump in!

There's a micropolitan port that we're going to hear about later this afternoon, the nerd MCU project, which ran a law on night, 2 66 is up and running on the ASP 32. There are a few JavaScript program projects. I'm not actually across where all of those are at. the bottom two I put with lines through them because that's my wishlist for things I want to run on ASP 32, rust is. A very interesting language for embedded. Unfortunately, you can't run rust unless you have LLVM support for your platform and extends it. Doesn't have LOVM support yet. so hopefully some people will get interested in adding LLVM support for extensor and then all kinds of interesting things might happen. the other one is really more, I don't know anybody else who shares this idea, but I really liked systems programming and Erlang, some time ago. A lot of airline concepts map to real-time always concepts.

TAKING THE ESP32 FOR A SPIN

we will cover the following topics software requirements, installing and setting up the Arduino. IDE, working with Serial monitor, interfacing LED and blinking the LED. There are many ways of programming the ESP32, but one of the easiest ways of programming, the ESP 32 is through the Arduino IDE. You can download the latest version of the Arduino IDE for both Mac and Windows. Everything explained, from now onwards will apply to both Windows and Mac OS unless explicitly stated otherwise.



Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

Downloads

Arduino IDE 1.8.13

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the Getting Started page for Installation instructions.

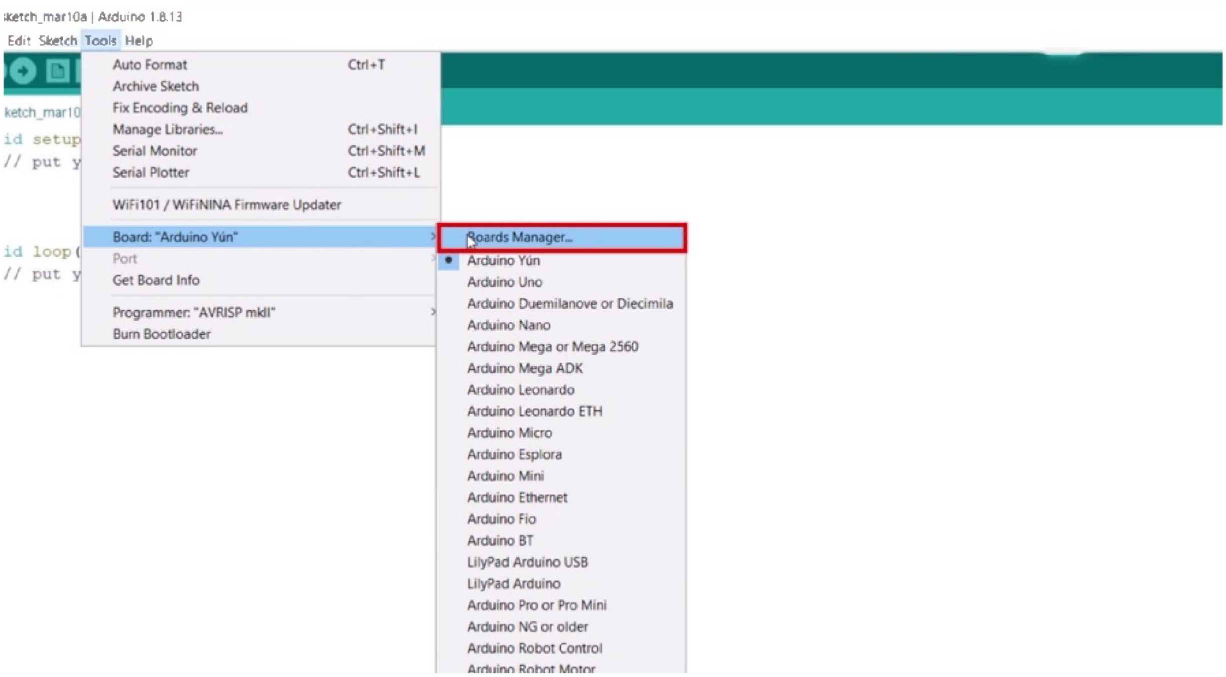
SOURCE CODE

Active development of the Arduino software is hosted by GitHub. See the instructions for building the code. Latest release source code archives are available here. The archives are PGP-signed so

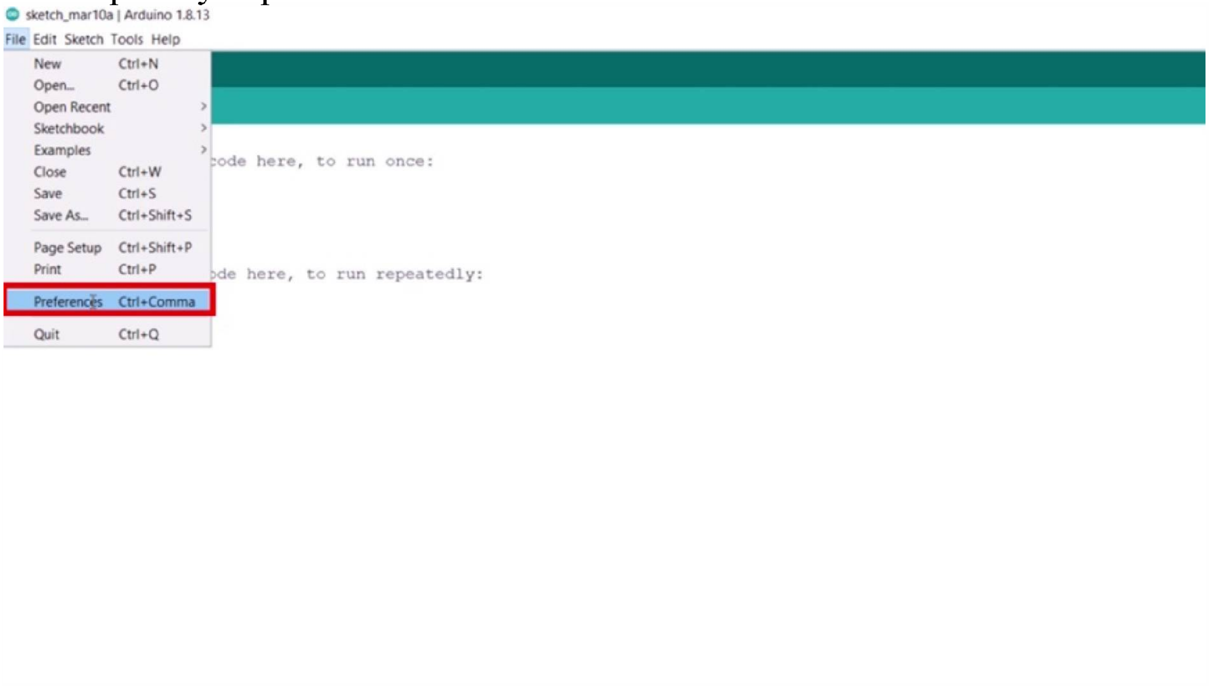
DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer
Release Notes Checksums (sha512)

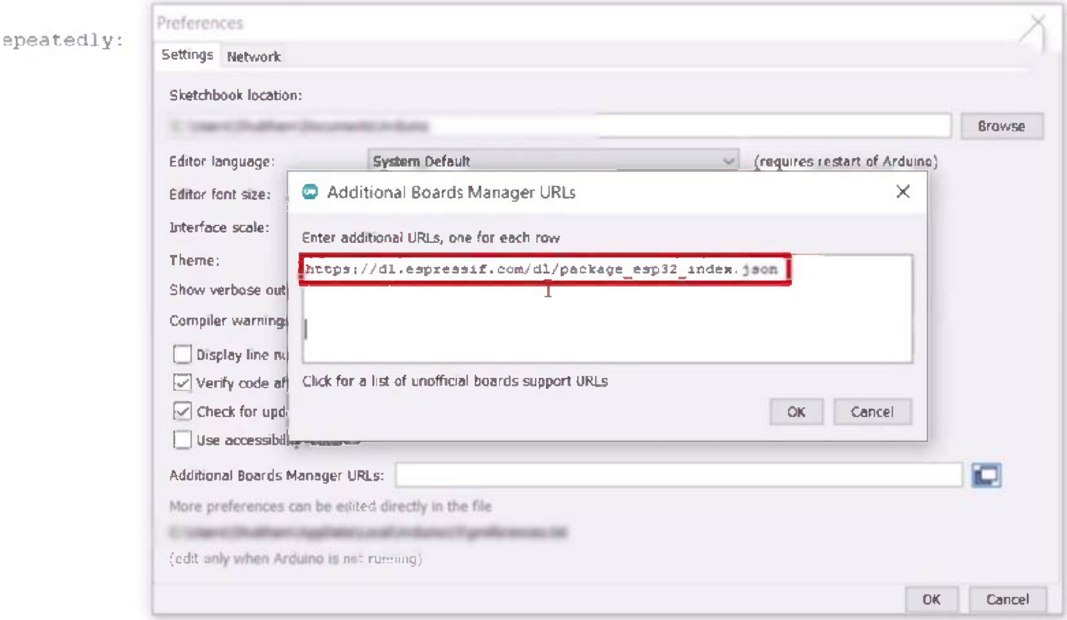
Go to this website from the link and select the appropriate version for your OS. Then download the software now double click on the .exe file and it will open a dialogue box. In case of Mac OS, it will be a different type of extension file. Click on OK and proceed further as you would install any software. After installing the Arduino IDE, you will have the development environment in front of you.



Now check the tool section where you will find the board section. If you select Arduino boards manager option, a dialog box will pop up, showing you a list of boards that come preinstalled. From Arduino version 1.6.2 all Arduino AVR boards are preinstalled. However, for other microcontrollers, as in our case the ESP32, we will have to install it separately. The Arduino board manager makes it pretty easy to install and Arduino core for the ESP32. If you try searching for the ESP32, you will not find it, this is because you need to install a third party core for the ESP32, which is developed by espressif.

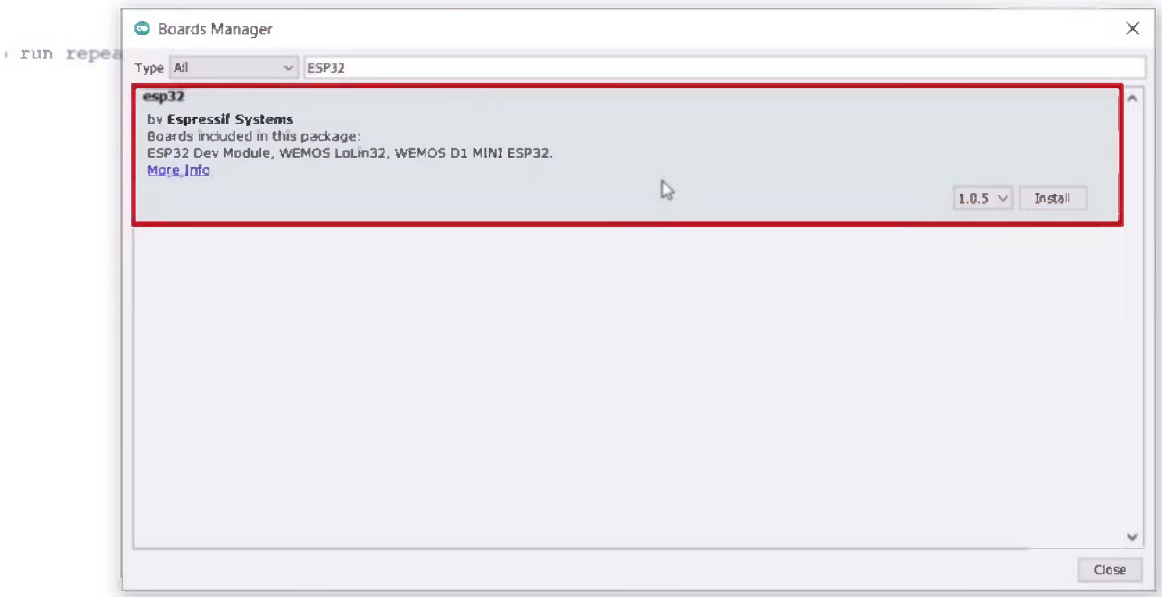


To do this, go to file and then preferences in Mac. You can find this option by going to Arduino and then preferences. You will find a field called the additional Boards manager URLs.


epeatedly:
Preferences
Settings
Network
Sketchbook location:
Browse
Editor language:
System Default
(requires restart of Arduino)
Editor font size:
Interface scale:
Theme:
Show verbose out
Compiler warning
Display line nu
Verify code af
Check for upd
Use accessibili
Additional Boards Manager URLs:
More preferences can be edited directly in the file
(edit only when Arduino is not running)
OK
Cancel
Additional Boards Manager URLs
Enter additional URLs, one for each row
Click for a list of unofficial boards support URLs
OK
Cancel

https://dl.espressif.com/dl/package_esp32_index.json

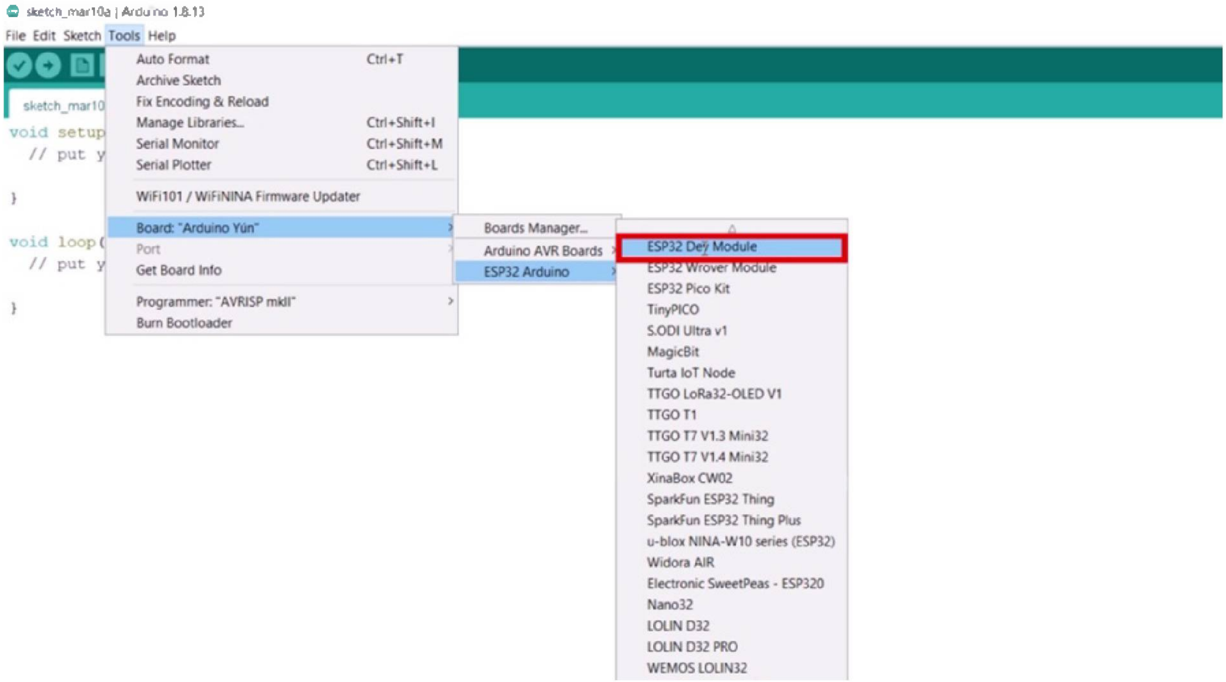
The field needs a specific type of file. This is a file written in JSON format which needs to be entered to fetch the list of third party cores. Now you can click on Ok to exit the preferences window.



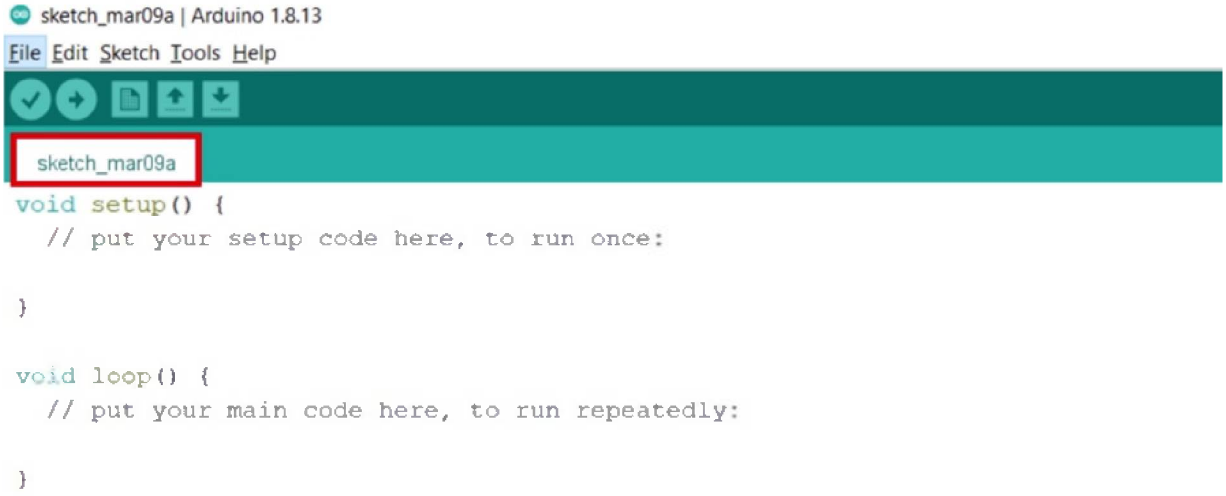
ESP32

esp32
by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

If you check for ESP32 in the board manager now. You can see that the packages include the ESP32 development module, which has the Arduino core for the ESP32. This will help us make the ESP32 compatible with the Arduino IDE You can download and install any release you want, but you should download the latest version as it includes bug fixes and updates for the latest issues. To check for all the releases for the ESP32 Arduino core, Now install and wait for it to download and install the packages once installed close to boards manager window.

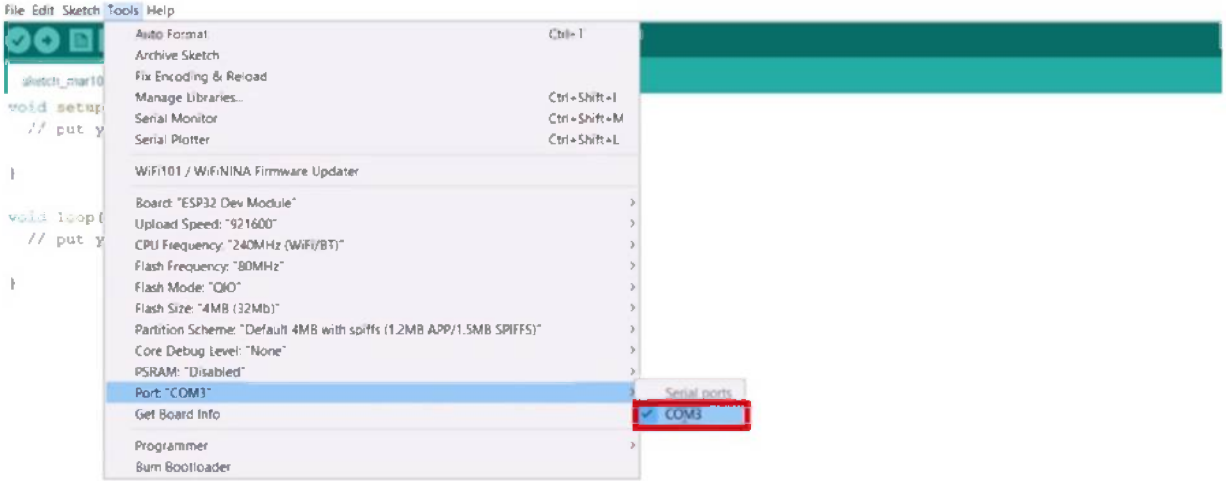


If you go to tools and scroll down to check for the boards, you will find the ESP32 dev module. Now you are all set to program the ESP32. This is the Arduino Development Environment. Arduino makes it very easy to write code. It does most of the work for you from importing libraries to configuring boards, pointing out errors and much more.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

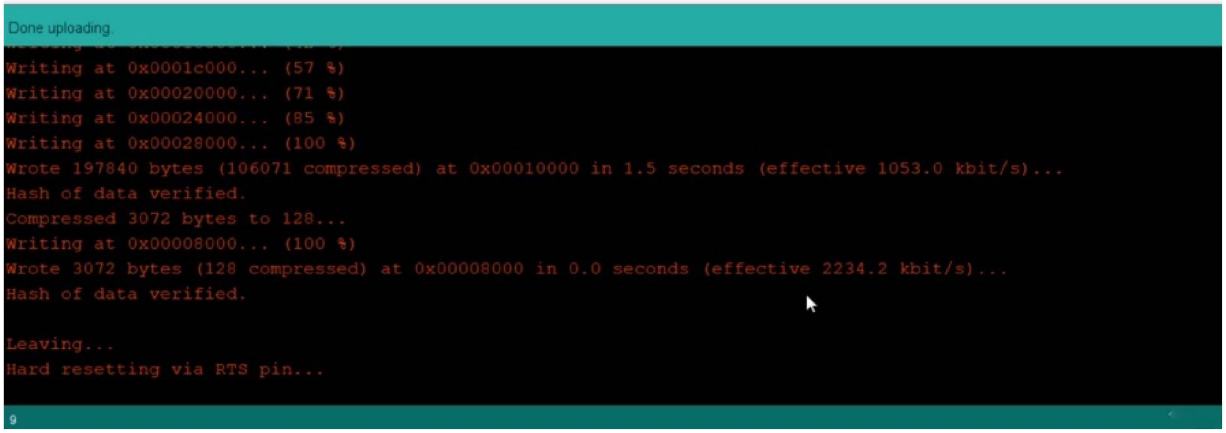
The Arduino programming language is a set of C and C++ functions that run every time you call the program In Arduino, the program you run is called a sketch. This is what gets uploaded to the microcontroller. board. You must have noticed that two functions are already on your IDE. These are functions that get called whenever you compile and run an Arduino sketch. The first one is the setup function. The setup function is used to initialize variables, libraries declare PinModes, etc. This runs only once in the program, now coming to the loop function. This is where you write your actual code. This function keeps on running the instructions inside it in a loop. It is basically an infinite loop. Let's look at a simple example to print. Hello World on the Arduino IDE Serial monitor. First connect the ESP32 to any of your computer's USB ports using the USB to micro USB cable to start programming, you need to select the correct serial communication port also referred to as the com port in the Arduino IDE.



```
void setup
  // put y

}

void loop(
  // put y

}
```

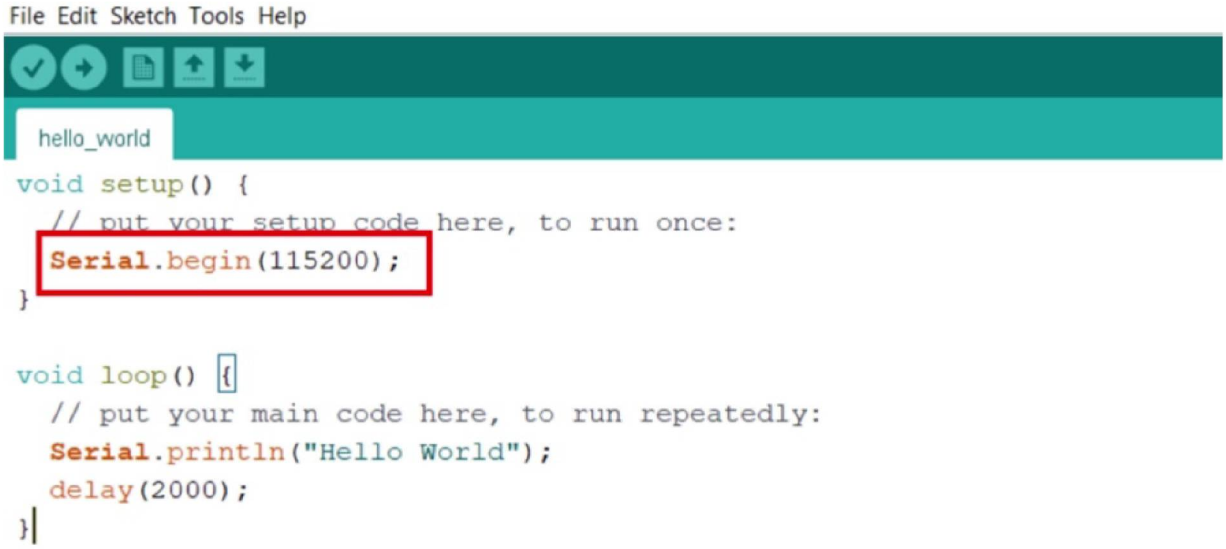
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Port: "COM3"

For Mac users The file, which maps to this port is the USB Serial port. Once you have selected the com port, you can upload a blank sketch to check if the sketch is being uploaded correctly.



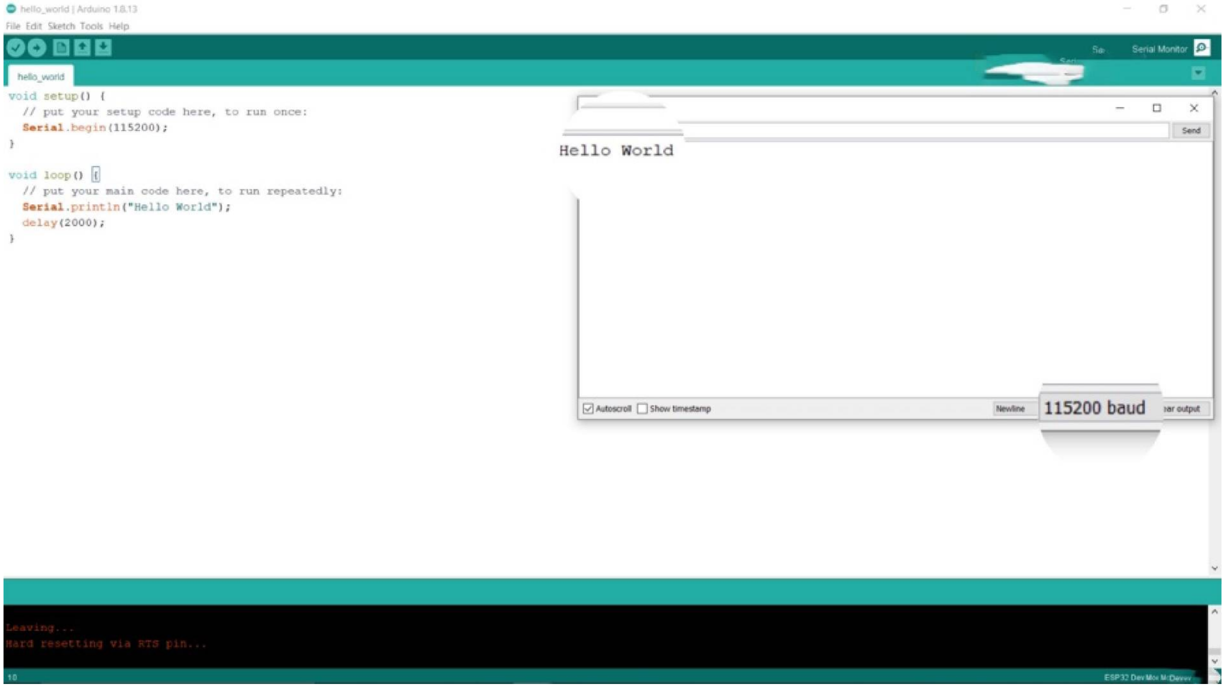
```
Writing at 0x0001c000... (57 %)
Writing at 0x00020000... (71 %)
Writing at 0x00024000... (85 %)
Writing at 0x00028000... (100 %)
Wrote 197840 bytes (106071 compressed) at 0x00010000 in 1.5 seconds (effective 1053.0 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 2234.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

To do this select upload here you will find your debug messages after hitting the upload button. You will also get information about the memory your whole code and variables are using. If you have successfully uploaded your code to the ESP32, you will get this message.



```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World");
  delay(2000);
}
```

Now, let's get back to our example. First, you have to set the baud rate to 115200, which tells the ESP32 to exchange messages with the serial monitor at the data rate of 115200 bits per second.

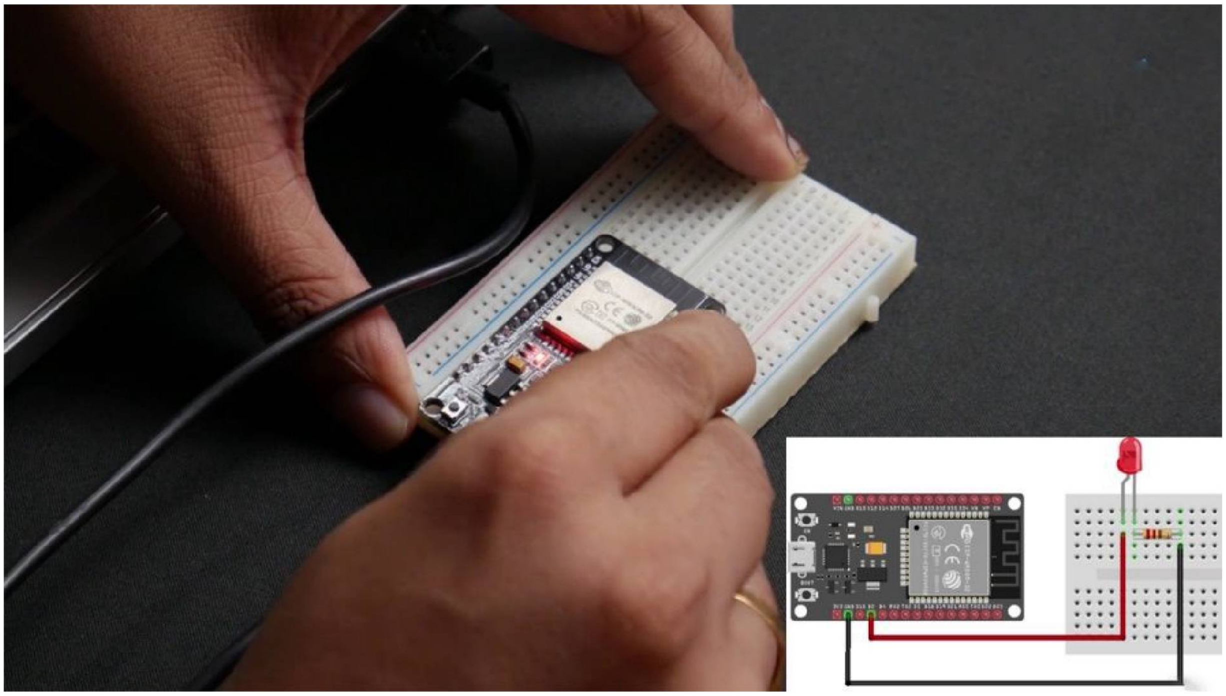


```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World");
  delay(2000);
}
```

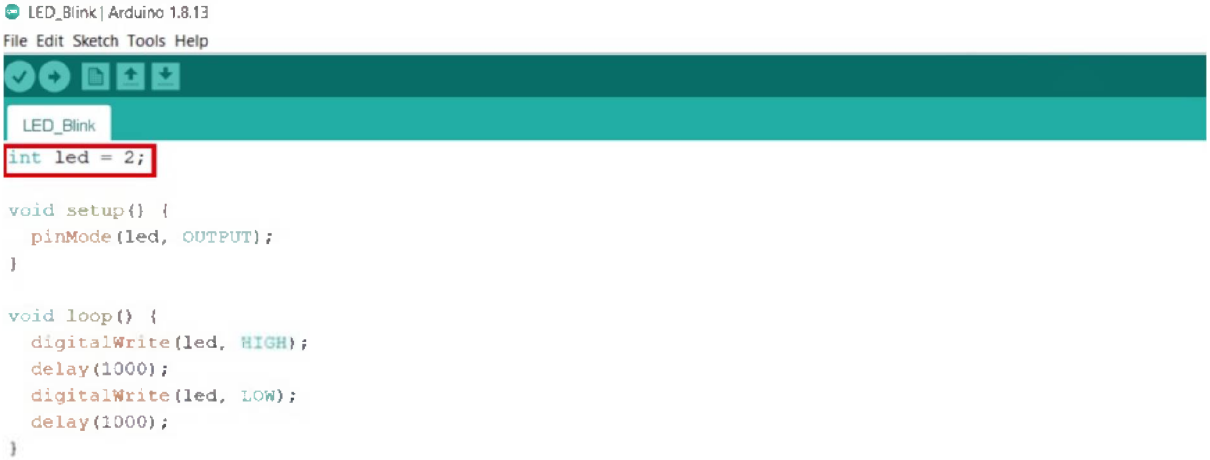
```
Hello World
```

```
Leaving...
Hard resetting via RTS pin...
```

Here, the println prints the text to be displayed on a new line. each time. We have used a delay of two thousand milliseconds or two seconds. After you upload the program to the ESP32 press the serial monitor icon here, Set the serial monitor to the same baud rate you have mentioned in the code, you will now be able to see your print statement text.

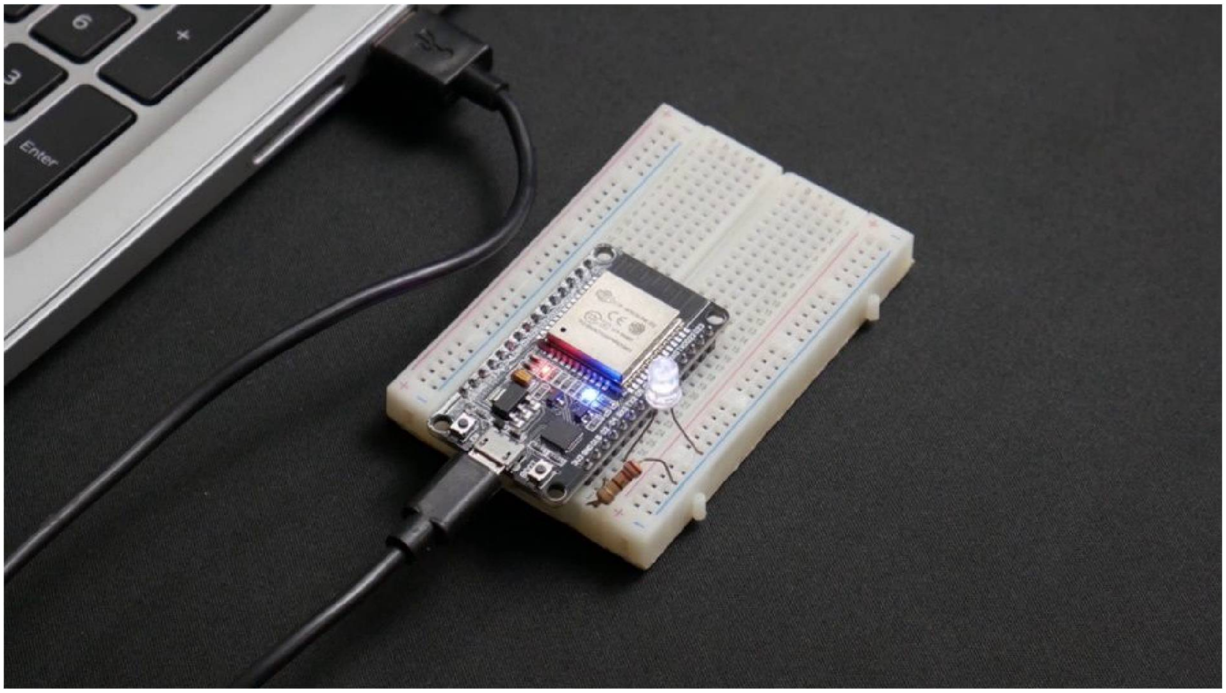


Now let us try interfacing and blinking an LED using the ESP32. First contact the positive leg of the LED to pin 2 , of the ESP32 with the 220 Ohm resistor in between and connect the negative terminal of the LED to the ground of the ESP 32.



```
int led = 2;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

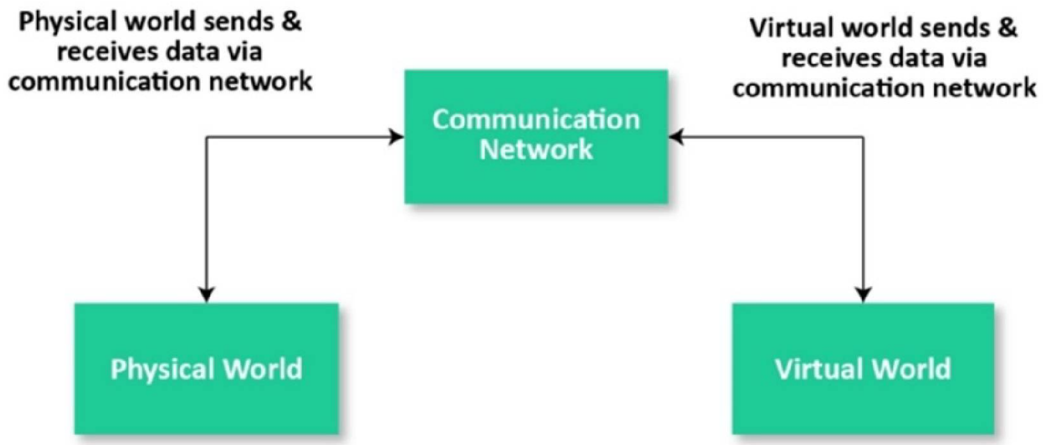
Let's checkout the code. First, we will initialize a variable for pin 2 as stated. Here we are setting as output here the digitalwrite sets the pin in our case pin 2 to high or low. The delay will help us to blink the LED at an interval of one second. Now, upload the sketch, you will be asked to save the file first, once uploaded, you will see the LED blinking.



Pin 2 is also the pin to access the onboard LED on the ESP32, you will see the onboard LED and the LED you connected, blinking simultaneously. The onboard LED on the ESP32 The two can be used as a test LED to debug certain code errors. we have covered the following topics software requirements, installing and setting up the Arduino IDE, working with serial monitor, interfacing LED and blinking the LED section summary. In this section we have covered the following topics.

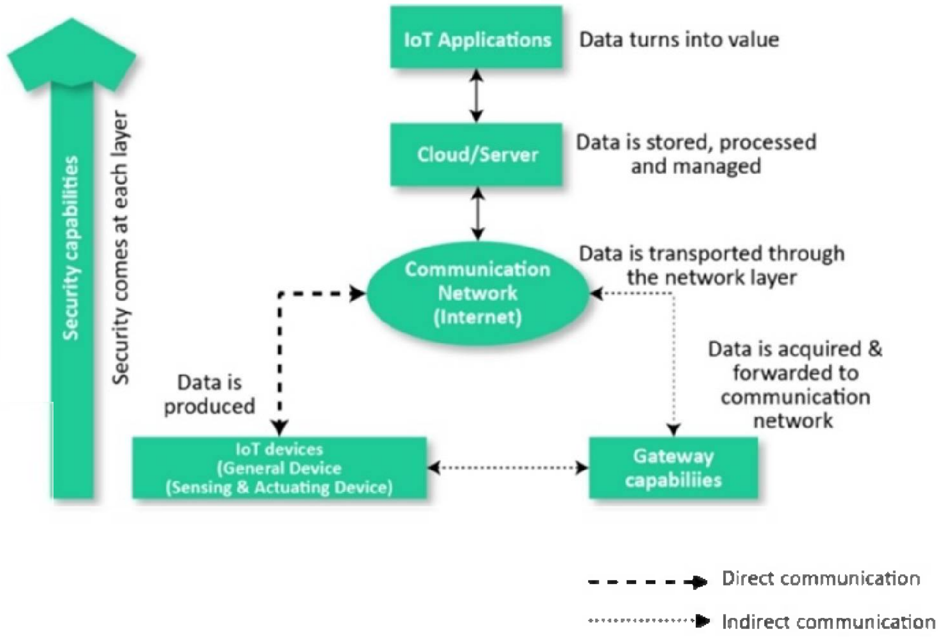
IOT CONCEPTS PART 1

we will cover the following topics, primary building blocks of IoT, IoT data lifecycle, case study.

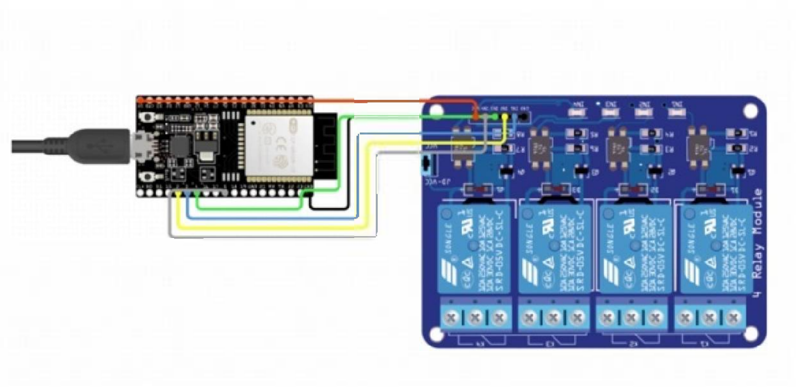

Physical world sends & receives data via communication network
Virtual world sends & receives data via communication network
Communication Network
Physical World
Virtual World

Now that you have a basic understanding of IoT, it is time to get into the details. If you pay close attention an IoT systems basic task is to connect the virtual world or the digital world with the physical world. IoT allows things or objects in the physical world to interact with the virtual world that is cloud services, platforms and applications through communication networks, enabling them to exchange information with each other. In context to the Internet of things, Any object of the physical world or virtual world with a unique identity that is integrated with the communication network is called a thing. A thing can be a physical thing or a virtual thing.

However, the term devices, when used, always refers to a physical thing. A simple example of a physical thing can be a temperature sensor. Cloud services, APIs and software applications are a few examples of virtual things. Since these technologies exchange and process data on their own and have their own identity, they are considered as things. So what comes to your mind when you think about IoT? sensors, lights, Bluetooth, Wi-Fi, cloud Right? You are absolutely right. And like you, each organization has its way of defining and implementing the Internet of Things. Nevertheless, the basic building blocks of IoT remains the same whatsoever.


IoT Applications
Data turns into value
Cloud/Server
Data is stored, processed and managed
Communication Network (Internet)
Data is transported through the network layer
Data is acquired & forwarded to communication network
Data is produced
IoT devices (General Device (Sensing & Actuating Device)
Gateway capabiliies
Security capabilities
Security comes at each layer
Direct communication
Indirect communication

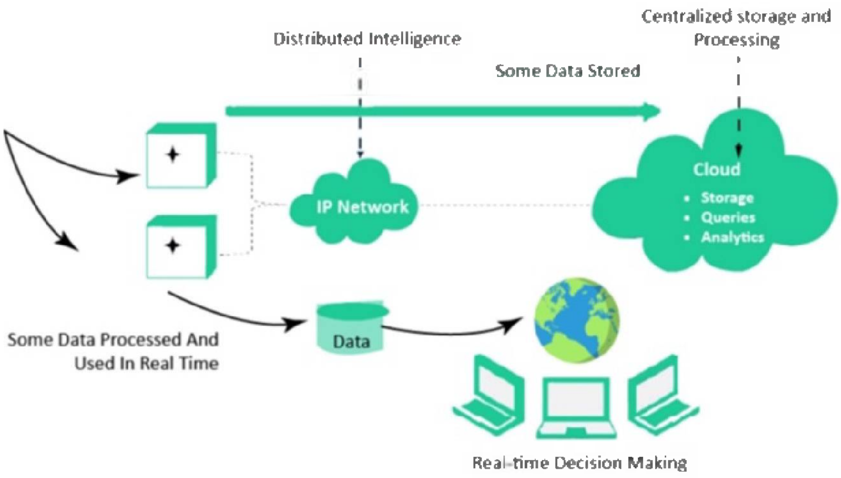
Let us take a look at the primary building blocks of IoT 1. IoT devices, 2. gateways, 3. communication network 4. cloud or server, 5. IoT application. Any device or equipment capable of communicating with other devices connected to the Internet, which is equipped with sensors and actuators to collect static or dynamic information and can act upon it, can be classified as an IoT device. As you know, we will be using the ESP32 board for building IoT applications in this project. Similarly, most of the IoT devices are built using embedded boards with wireless capabilities. These boards come with the microcontroller or a processor on board RAM and ROM and digital and analog GPIO pins.



These boards can be easily stacked upon other boards, sensors and actuators to form an IoT device.

Some of the other famous IoT boards are Raspberry Pi, Banana Pi, Arduino and many more. Let us get an insight into Gateways.

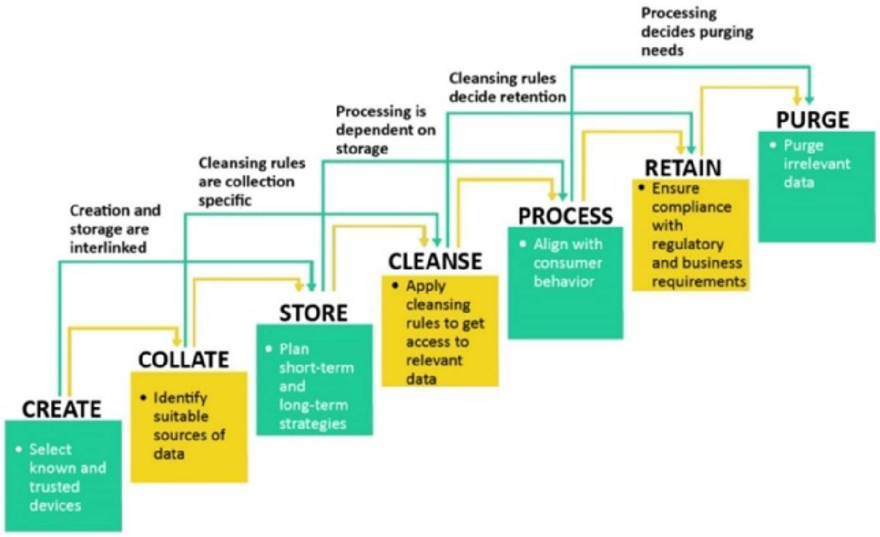
Now Suppose one of your IoT devices uses Bluetooth for communication while another device can only send or receive data using Wi-Fi. The problem is similar when you are trying to communicate with the person who cannot speak or understand your language. In such cases, we need a gateway for protocol conversion, just like a translator to facilitate smooth communication between the devices. Now the communication network has to make it possible for the devices to interact with each other. It allows IoT boards stacked with sensors and actuators to communicate with web or cloud servers or even allows them to communicate with each other. The communication network is generally the typical Internet network with different layers such as the physical layer, link layer, network layer, transport layer, application layer and communication protocols operating at different layers. Bluetooth, Wi-Fi, SigFox, MQTT, REST and CoAP are some of the communication network protocols.


Centralized storage and Processing
Distributed Intelligence
Some Data Stored
IP Network
Cloud
Storage
Queries
Analytics
Some Data Processed And Used In Real Time
Data
Real-time Decision Making

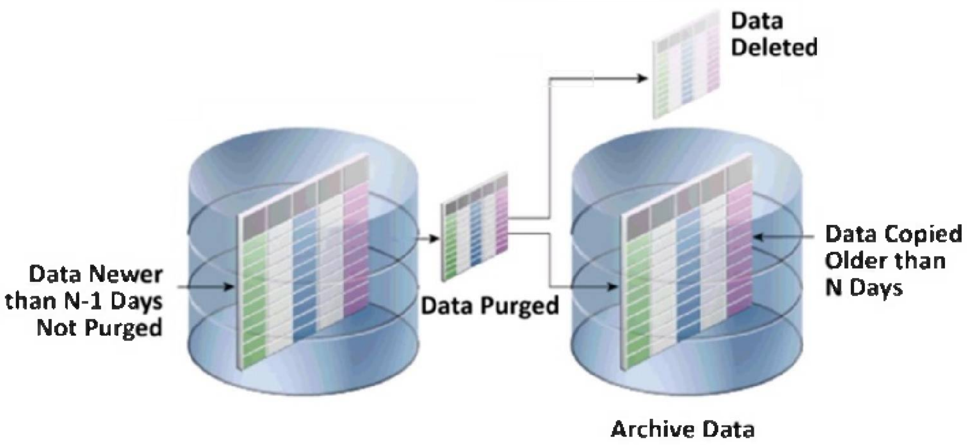
Edge Computing Architecture with IoT Endpoints

The data now goes to the cloud or server for storage, apart from storage, the cloud also performs data mining and analytics to derive useful information. It is also responsible for managing the connected devices and networks. The cloud can also communicate with other private or public cloud services to enable an IoT application. The IoT application is the part where we feel the presence of IoT and automation, the application does the processing, mining and analysis of the cloud data. The IoT application is the piece of software at the cloud server that extracts data, manipulates it to derive useful insights and manages to send insight to the target IoT devices securely.For example, an IoT application developed for home automation will process data from sensors and send necessary commands from the cloud to operate home appliances.

All the while talking about IoT. It is the data that flows through the system. The word data in the IoT world is of utmost importance. Since the data is generated from various sources, it is crucial to ensure that it is processed as per specific tasks.


Processing decides purging needs
Cleansing rules decide retention
Processing is dependent on storage
Cleansing rules are collection specific
Creation and storage are interlinked
PURGE
• Purge irrelevant data
RETAIN
• Ensure compliance with regulatory and business requirements
PROCESS
• Align with consumer behavior
CLEANSE
• Apply cleansing rules to get access to relevant data
STORE
• Plan short-term and long-term strategies
COLLATE
• Identify suitable sources of data
CREATE
• Select known and trusted devices

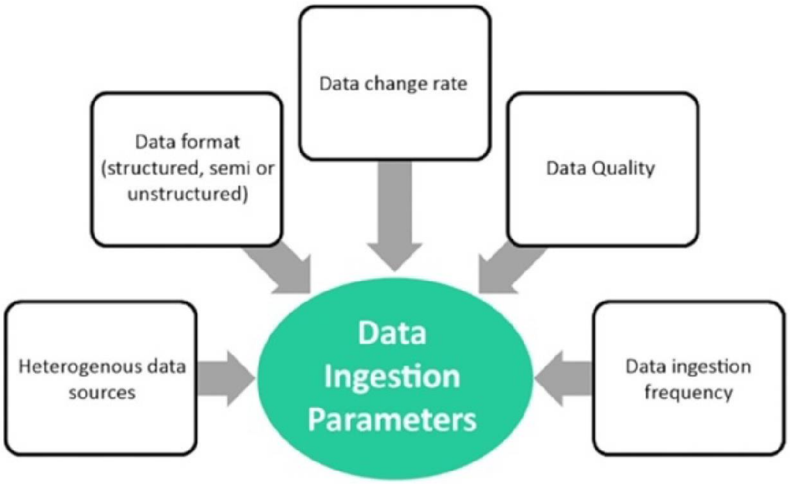
IoT data flows through many layers, unlike traditional data, which flows through fixed boundaries. So now let us take a look at the data lifecycle in IoT 1. creation. The process of data creation should be in context with the problem statement and should not generate unnecessary data. According to the needs the data can be generated using various sensors taken from the web or even location data. 2. collation - the raw data collected should be organized and structured according to the system's need. or organization. Factors such as frequency mode of data collection, batch real time, near real time or real time data need to be considered during this process. 3. storage - the storage mechanism in an IoT system should be scalable and accommodate and allocate growing data, but at the same time it should be cost effective. As we are dealing with high amounts of data in fewer time intervals. Generally, bigdata based solutions are preferred to store IoT data. 4. cleansing It is obvious that not all data collected is of use. Most of the time Only some parts or parameters of the data need to be used for a particular task. Generally, critical parameters are confined to not make the data irrelevant for any other use case. 5. processing - the processing logic should consider the use case and the data characteristics. For example, in retail industry, data regarding consumer behavior patterns can be processed according to the operation period. That is a holiday season or year end. 6. retention - once the data is used, it is preferred to only hold on to the relevant data or data that may come to use in the future, keeping unnecessary data will take up storage space, thereby increasing the cost. 7. archival - In this phase, the data that is no longer active but important is moved out of the production system to long term storage systems. This data, though, can be brought back to use at any time when needed.


Data Deleted
Data Newer than N-1 Days Not Purged
Data Purged
Data Copied Older than N Days
Archive Data

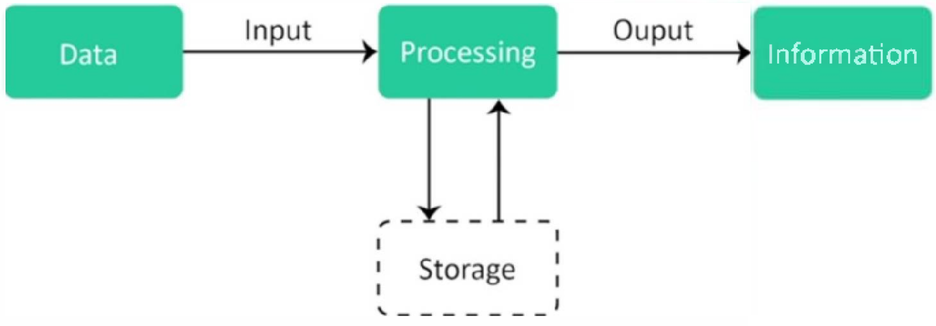
8. purge - as the world suggests, data purging is a term used to describe various methods which can be used to erase data from a storage space permanently. Purging however, stores are reference in the records for the future and should not be interchanged with the term data deletion. I know

that the data lifecycle was very technical. However, now with the knowledge you have, you will be easily able to identify and understand any Iot systems working. To make it clearer Let us look at a real world example of an IoT system. If you Google about Patras, a Greek city, it is famous as it is, the third largest city in Greece and has a port that connects Greece to Italy. But if you go deeper, you will find that it is one of the pioneers of smart city implementation. They have a dedicated, smart city hub and have partnered with Deutsche Telekom to deploy various IoT implementations using in NB-IoT. Now, you may have a question. How exactly is the city smart? Patras has used NB-IoT to power 2 IoT systems on the same network; a smart parking system and a smart lighting system. The smart parking system consists of sensors that are embedded into the street under the parking spaces. These sensors can sense if the space above them is empty or free based upon the vehicle's presence above them. The status is then communicated through Cosmotes NB-IoT radio access network, which is further linked to Deutsche Telekom. The users have an app where all the information for the parking spots are available. The app also guides you directly to the required parking spot. The smart lighting system also uses NB-IoT to control street lighting. The street lighting can be adjusted to different levels depending on the time of the day. This reduces electricity consumption and has also improved security in the local area. The lighting system uses the LED technology and FlashNet smart lighting controllers. Such real-life examples of successful deployment of IoT have paved a path for constant growth and development in the technology and shows that Internet of Things can help us develop our daily lives to a large extent.

we will cover the following topics. IoT architecture - device management, IoT architecture ingestion and processing, IoT architecture analysis, visualization and integration. Now that you are familiar with the basic building blocks of IoT and how data flows in the system. Let us get to know about the IoT architecture. connectivity, smart devices and cloud services are considered the core aspects of IoT. One of the most important and often ignored parts in IoT is device management, just like how you keep track of all the electronics you have by keeping them up to date and fixing them whenever needed. Similarly, IoT device management is a part of the IoT ecosystem, which is responsible for the management of various types of devices and sensors on a single unified platform. The importance of device management has been noticed lately, and companies are now developing device management platforms with advanced features for optimum performance and management of devices. Whatever the IoT system may be, the IoT device management is built on four fundamentals.1. provisioning and authentication. Every device is enrolled and authenticated in an IoT system to have its own identity, to be trusted and secure. 2. configuration and control a system should always be in control of all the devices it is managing. Remote resetting and configuration are essential in0020order to boost the systems control capabilities. 3.Monitoring and diagnostics. Many factors, including bugs in the software or some operational issues, can make the device not able to function properly. Monitoring and diagnostics thereby become an important part of the device management system 4. software updates and maintenance. Most developers often ignore software updates, but it is one of the most important parts of IoT device management, especially for security updates and maintenance of remote device software. Now that we have taken care of the devices, sensors and actuators in the IoT system, the next thing to look out for is the data. Data ingestion is moving unstructured data from where it originated in the system to where it can be stored and analyzed. To put in simple words, we can also say that data, ingestion, is all about collecting information from multiple sources and then putting it somewhere to be accessed later. Generally, data comes in from various sources in variable speeds and multiple formats. An effective data data ingestion process includes prioritizing data sources, validating individual files and routing data items to correct destinations.


Data change rate
Data format (structured, semi or unstructured)
Data Quality
Heterogenous data sources
Data Ingestion Parameters
Data ingestion frequency

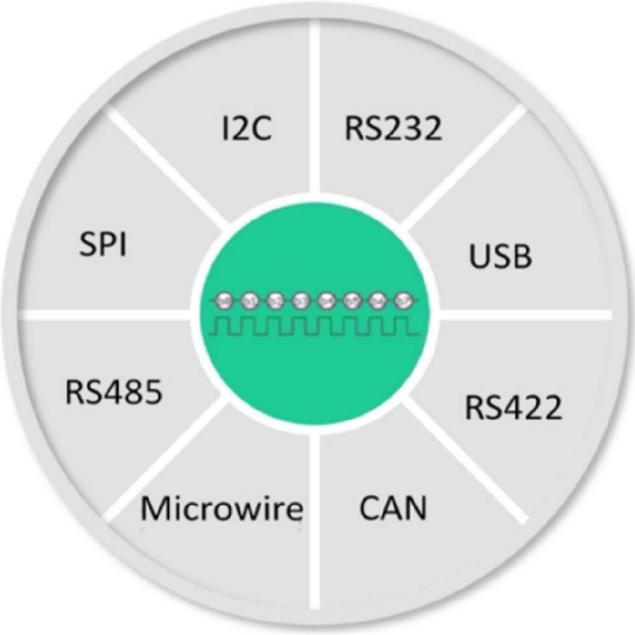
Some parameters need to be considered for a smooth data ingestion process. These parameters are 1. data velocity. It is the speed at which the data flows from sensors, machines, human The movement of data can be huge or continuous in the data ingestion process. 2. data size This refers to the enormous volumes of data that flows into the system. Data generated from various sources may increase with time. 3. data frequency - data received can be processed in real time or in batches. In real time as data is received simultaneously it is sent into the data ingestion pipeline. While for batches, the received data is stored in batches and sent to the pipeline at fixed time intervals. 4. data format - data ingestion can be done in different formats, such as a structured one that is a tabular form or unstructured format, including images, audios and videos. Most of the time, the semi structured format is used, which consists of JSON or CSS files. Imagine having tons of documentation and papers, but you are not able to find the exact solution to a problem in the same way. An IoT system generates a huge amount of data, but it will be of no use if it is not processed accurately. In order to make sense of the massive amount of data that the sensors collect, we need to process it.



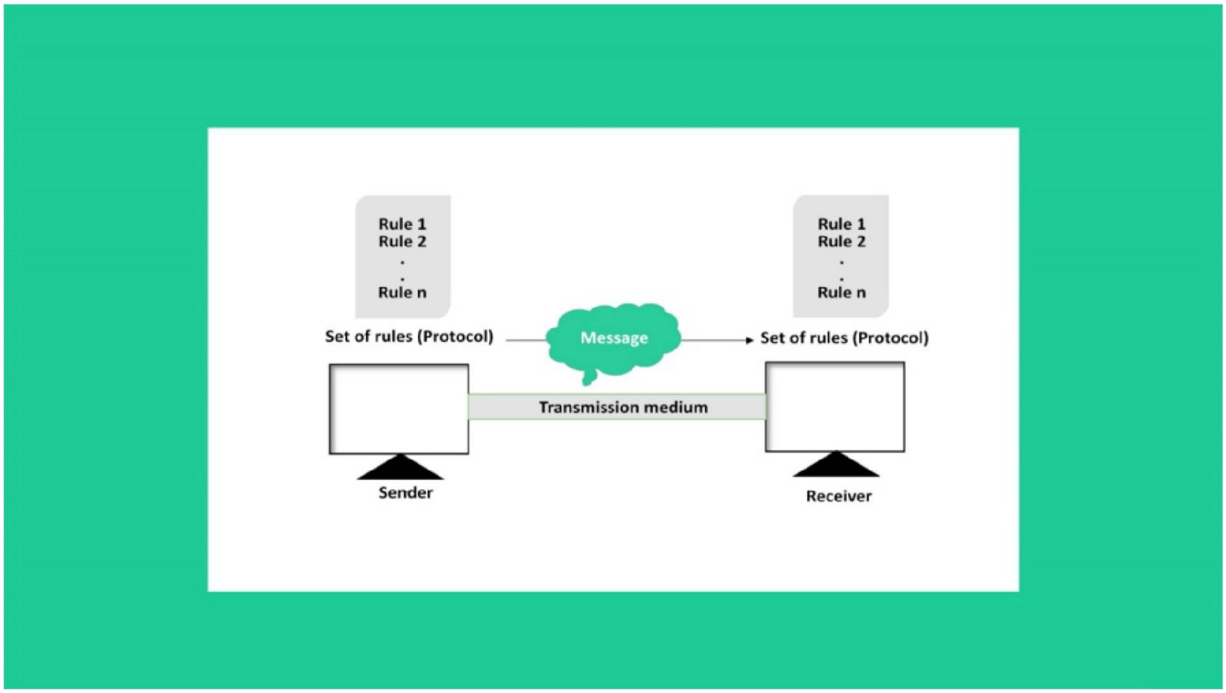
Data processing is a process of converting raw data to something meaningful that the end users can understand and react to. The difference between the two words, data and information lies in the processing. Data refers to raw, unorganized facts and is generally useless until it is processed after processing it is called information. So how does data processing work? The process usually follows a cycle that consists of input, processing and output. The input stage consists of the data being converted into a machine readable format. This step is essential as the output is completely dependent on the input. Alternatively, we can say garbage in, garbage out. in the processing stage the raw data is transformed into information by using various data manipulation techniques. Some basic techniques are 1. Classification here data is classified into various groups 2. sorting in this technique, data is arranged in some sort of order. 3. Calculation Arithmetic and logical operations are performed on numeric data. In the output stage, the process data is converted into a human readable format and presented to the end user as information. Now that we have information or data that makes sense, you need to make use of it in an appropriate manner for your needs. Data analytics and representation of the data is an important part of the IoT architecture. With the application of data analytics, tools and procedures real important and case specific data can be obtained.

You may ask what is the need for analytics? If IoT data analytics is evaluated and implemented properly, it can have a lot of benefits and some of them include 1. improved equipment maintenance after data analytics One can have an easy idea about the systems equipment in real time. For example, in the manufacturing industry, with the help of sensors, one can measure the important basic figures of the equipment, like vibration, heat, amount of time elapsed, etc. Data analysis can then help in sorting out the wear and tear of the equipment to particular causes. Nowadays, smart machines are deployed with sensor notification to operators regarding potential breakdowns and wear and tear of the machines. 2. Operations, optimization and automation. Analytics also helps organizations to keep track of processes that are difficult to keep track of manually. This allows the organization to keep track of the workflow and identify defects at the right time so that they are not carried away to the final product. For example, General Motors monitor humidity with IoT to optimize painting. If the conditions are unfavorable, details are sent to the operators. 3. Enhanced customer experience. Customer experience can be enhanced to a better personalized customer experience using data analytics, Customer behaviors and preferences can be analyzed to meet appropriate customer needs. Rather than numbers Visualization always gets you a better understanding of the data, and hence data visualization is as important as analytics. Realtime or analyzed data is generally visualized in the form of graphs or pie charts and presented to the end user for better understanding. If you are familiar with data analytics and visualization, you may have heard of Tableau, a visualization tool with a variety of intuitive charts. Now we have properly managed both IoT devices and IoT data. It is time to integrate both of them. The word integration can now be defined as making individually designed applications and data work well together. Implementing end to end IoT solutions with IoT devices, IoT data, various APIs and applications working together can be called as data integration.

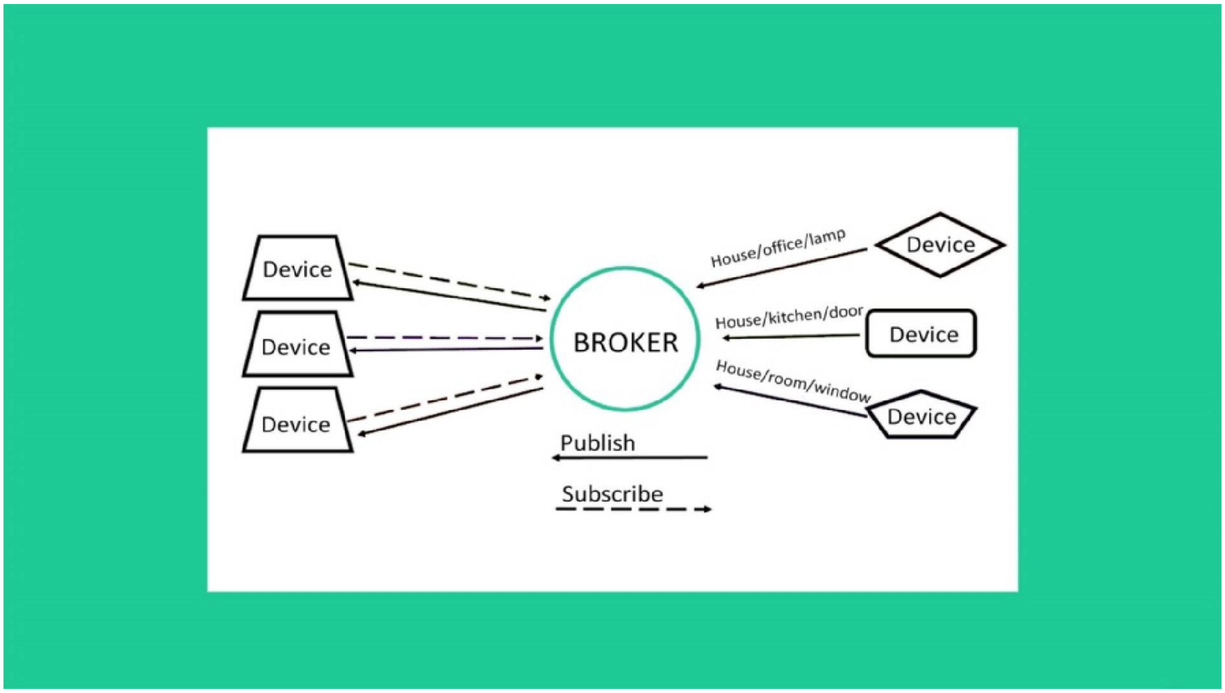
we will cover the following topics IoT communication protocols - MQTT, ZigBee, CoAP, IoT security, interoperability, Role of AI, ML in IoT. Now that you know how an end to end IoT device works. Let us check on some more factors that affect the working of an IoT system.



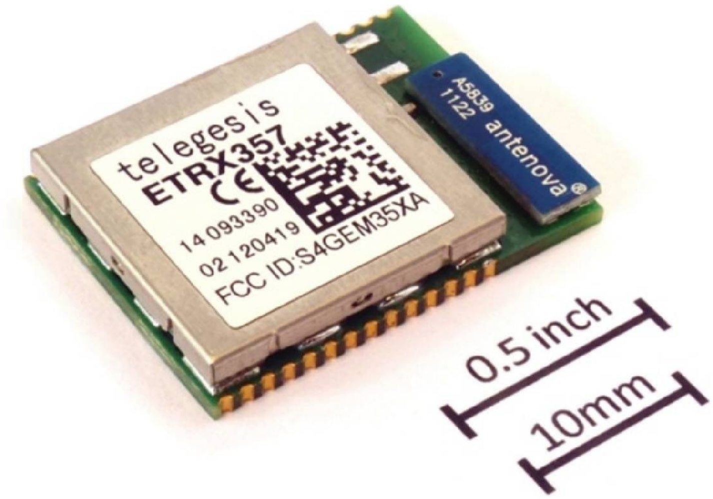
communication protocols are a very important part of the IoT industry. They are selected based on application and requirements such as range, power and memory. So why exactly do you think these communication protocols are important? If you think about the key differentiating factor between an ordinary device and a smart device, you will conclude that a smart device can talk to other devices.



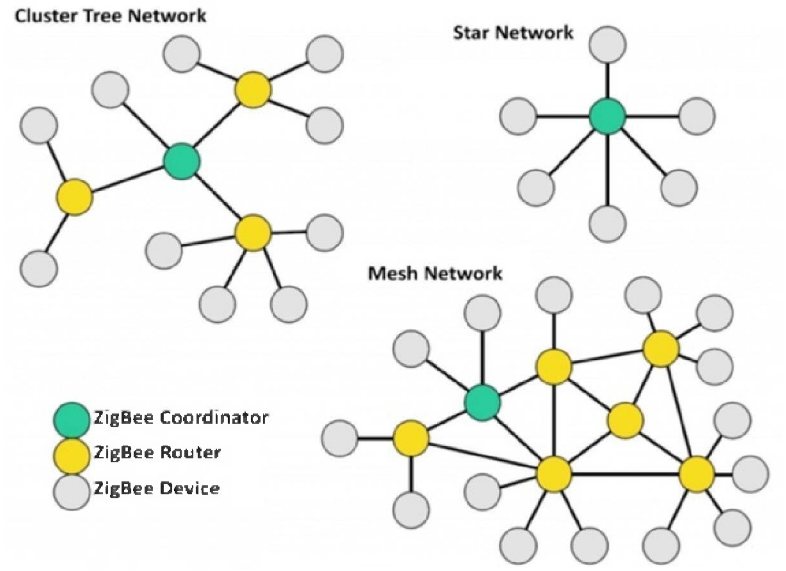
For every interaction these devices make, there needs to be a common language that all the devices should use in the IoT system. Communication protocols provide this common language. Let's now get to know about some of the important communication protocols in IoT. MQTT - MQTT stands for MQ Telemetry Transport. It is a lightweight message protocol for sending simple data flows from sensors to applications and middleware. It works on top of the TCP IP network for supplying reliable yet simple streams of data. MQTT can work on any network that provides ordered, lossless and bidirectional connections.



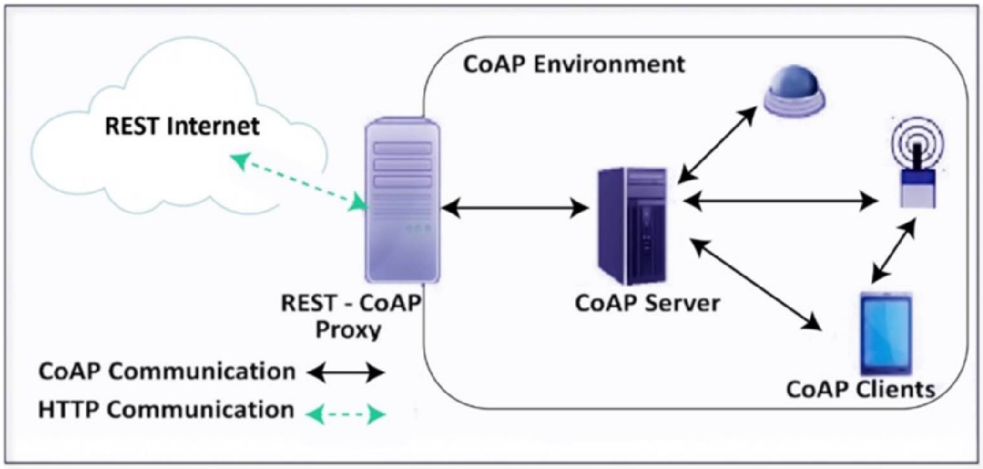
The MQTT protocol comprises three key elements subscriber, publisher and Broker. The publisher publishes to topics, and the subscribers subscribe to the topics they want. However, the publisher or the subscriber never contact each other directly as they don't even know the other exists. The connection between the publisher and subscriber is handled by a third component known as the broker. The broker filters all messages and distributes them correctly to the subscribers. You can understand the concept easily if you consider the youtube platform as a broker, the content creators, as the publishers, and we who watch the content as subscribers. ZigBee, it is one of the most important IoT communication protocol and has significant advantages in complex IoT systems. Like Bluetooth, ZigBee also has a wider user base due to its low power operation, high security, robustness and high scalability.



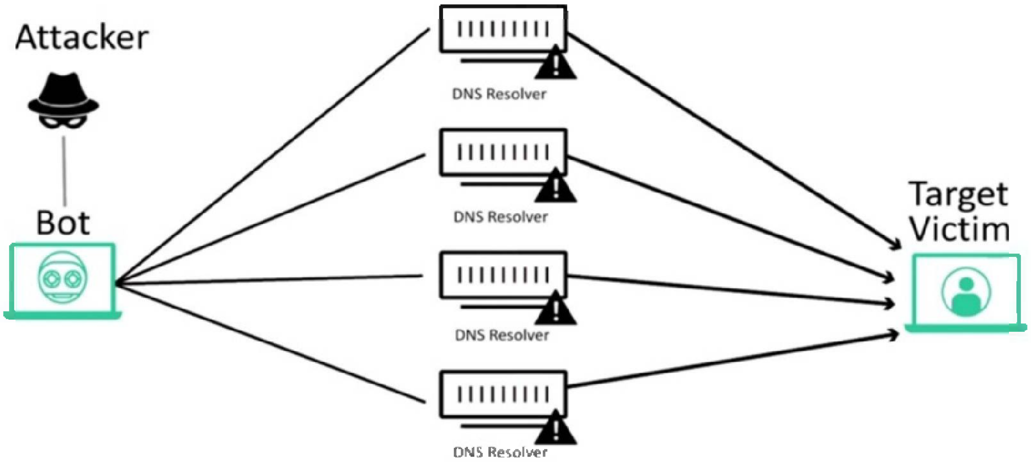
ZigBee is based on IEEE 802.15.4 standard, which lays down specification for the low rate wireless personal area network. ZigBee is mostly used for two way communication example between a sensor and a control system. A ZigBee network consists of three devices a coordinator, a router and an end device. An end device can be anything like a CCTV camera, smart thermostat, etc.



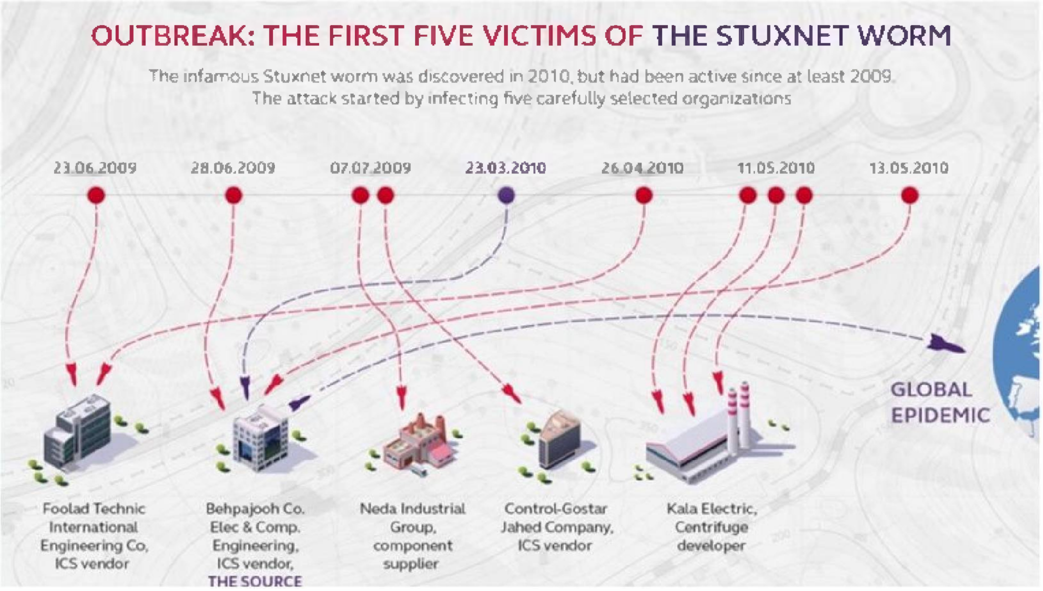
There should be at least one coordinator in the network as it acts as a bridge for the entire network. The coordinator transfers not only data but also stores and handles information. The routers act as intermediary devices, allowing the device to pass data to and fro from other devices. The number of coordinator's routers and end devices depends on the topology type implemented in the network. CoAP the constrained application protocol is a specialized Web transfer protocol used for constraint notes CoAP is designed so that constrained devices can join the IoT network through constrained networks with low bandwidth and low availability.



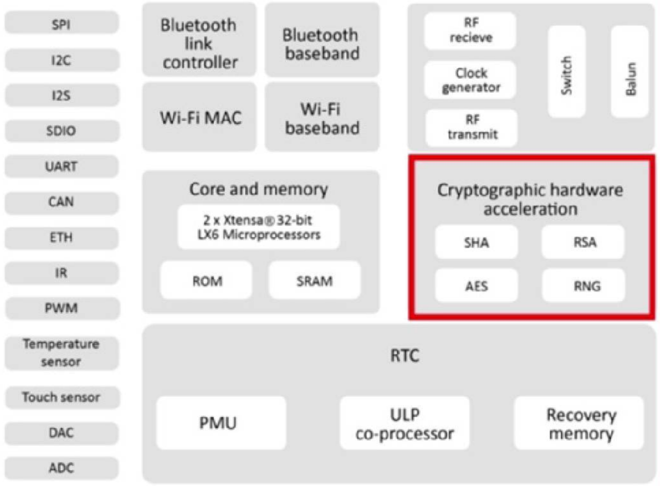
CoAP function somewhat like HTTP, but for restricted devices enabling equipment such as sensors or actuators to communicate on the IoT. These sensors and actuators contribute by passing along their data as part of a system. CoAP can continue to work even in networks with high congestion and limited connectivity where Protocols such as MQTT fail to exchange information. If you have been in touch with IoT technology, you would know that security is one of the most challenging parts in IoT. The topic of cybersecurity is huge and beyond the scope of this project. However, it is important to understand the three types of IoT based attacks and exploits.



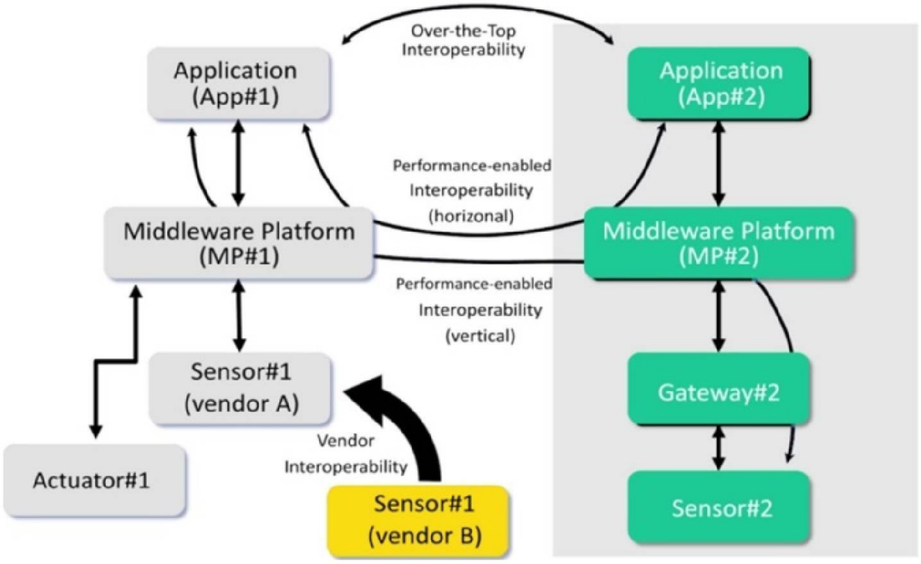
Mirai, the most damaging denial of service attack in history occurred in August of 2016. The main victims include curbs on security, which is a popular Internet security blog. Dyn, a very popular and widely used DNS provider for the Internet and Lone Star Cell, a large telecom operator in Liberia. Small targets of the attacks included Italian political sites, Minecraft servers in Brazil and Russian auction sites. And guess what, Mirai, the name of the malware spawned from the insecure IoT devices in remote areas. Stuxnet, Stuxnet was the first known documented cyber weapon that leads to permanently damage another nation's assets.


OUTBREAK: THE FIRST FIVE VICTIMS OF THE STUXNET WORM
The infamous Stuxnet worm was discovered in 2010, but had been active since at least 2009.
The attack started by infecting five carefully selected organizations
23.06.2009
28.06.2009
07.07.2009
23.03.2010
26.04.2010
11.05.2010
13.05.2010
GLOBAL EPIDEMIC
Foolad Technic International Engineering Co, ICS vendor
Behpajooh Co. Elec & Comp. Engineering, ICS vendor, THE SOURCE
Neda Industrial Group, component supplier
Control-Gostar Jahed Company, ICS vendor
Kala Electric, Centrifuge developer

It was a worm that was released to damage SCADA based Siemens, programmable logic controllers. The worm used rootkit to modify the rotational speeds of motor that were under direct control of the PLC. Chain reaction, this was unlikely, a research study to exploit PAN networks using just a light bulb. It was focused on Philips Hue light bulbs controlled by smart apps and were present in almost all consumer homes. It showed how vulnerable remote IoT systems could be. It was further predicted that these attacks could be scaled up to smart city attacks by inserting just one single infected smart light. Now that IoT is growing at a faster rate, the security aspect is taken into consideration while designing IoT applications. Special separate hardware is allocated to the IoT systems to run basic security functions.



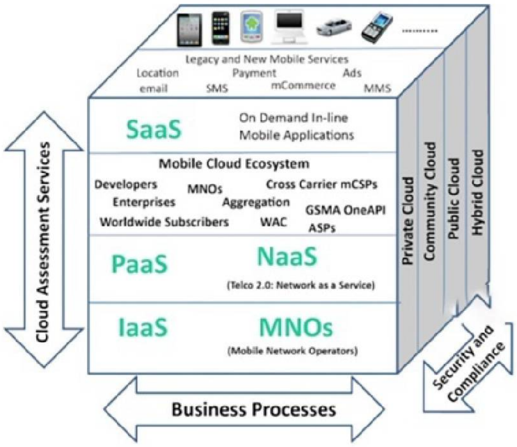
The ESP32 that we will be using also has cryptographic hardware acceleration to run security algorithms such as the following. Furthermore, the US government also passed a bill in 2017 named the Internet of Things Cybersecurity Improvement Act. The bill's intent is to regulate and formalize a standard for the security of IoT applications sold in USA. where you had to install a third party called Arduino IDE to make it compatible. Similarly, the scalability of IoT highly depends upon interoperability.


Over-the-Top Interoperability
Application (App#1)
Application (App#2)
Performance-enabled Interoperability (horizonal)
Middleware Platform (MP#1)
Middleware Platform (MP#2)
Performance-enabled Interoperability (vertical)
Sensor#1 (vendor A)
Gateway#2
Vendor Interoperability
Actuator#1
Sensor#1 (vendor B)
Sensor#2

One vital issue among smart objects is that they are proprietary and designed to operate only within a predefined hardware or infrastructure environment. Also, no single End-To-End application can be delivered by a single technology. This has hampered the scalability of IoT and has also incurred extra costs. Just three rules for IoT connectivity can overcome these challenges. Open industry standards. There should be a standard development organization that fixes and develops standards for IoT with the fix it assured quality of service. This will help in worldwide adoption, cross-vendor support and interoperability in the long run. Software driven technologies. Deploying wireless solutions with the hardware driven approach is challenging as you are bound to a certain device type and must depend on the respective vendor to go through the certification process. On the other hand, software driven technologies can be flexibly used in any legacy devices and infrastructure that already meet your operational requirements. Open interfaces IoT interoperability on the application layer involves effective data transfer to different user applications and servers. Open source messaging protocols like MQTT or CoAP and application programming interfaces based on restful principles are key drivers of cross application interoperability. Have you thought about making these are IoT enabled, smart devices even more smarter? Yes, a new technology called AIoT has originated and a new wave has started.

So where does artificial intelligence come into the picture. After the IoT system has collected data and sent it to the cloud through the Internet A.I., which is considered as brain of AIoT helps in decision making and simulating the machines to act or respond in a particular manner. Take a simple example of a smart air conditioner that adjusts its temperature, according to the temperature outside the sensor in the system can only sense the data. This doesn't serve the purpose of a smart system. It is the component that helps in making intelligent decisions and adjust the temperature accordingly. Some more examples to look up to can be collaborative robots or drones. You can also consider Tesla self-driving cars, for that matter. To be honest, it is a very exciting time to live for both humans and machines. With multiple advances in AI, communication, IoT and analytics IoT devices are taking over almost every technology domain.

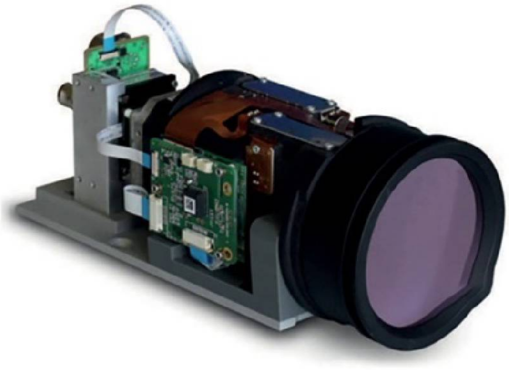
we will cover the following topics Introduction to cloud computing, IoT enabling technologies, edge computing and IoT. We have already mentioned that the data is sent to the cloud from an IoT device where it is stored and processed. Cloud computing has had a huge role in making this possible. Ever since the rise of cloud computing, there has been a massive shift and companies prefer it over the traditional technological patterns. Also because of the scalability and data dynamics provided by cloud computing. There is a lot of stress being given on the use of cloud computing to make data available remotely. So what is cloud computing? Cloud computing is the delivery of on demand computing services like applications to storage and processing power. This is generally over the Internet and on a pay as you go basis rather than owning their own cloud or data storage. A company can rent access to anything from applications, to storage from a cloud service provider. This benefits the organization by avoiding the upfront cost and complexity of owning an IT infrastructure. In return, the cloud service provider benefits from providing the same service to a wide range of customers.



Services provided by the cloud provider includes networking as a service, software as a service, platform as a service and infrastructure as a service. Networking as a service. Software defined networking and software defined perimeters are the types of services typically provided by NaaS.

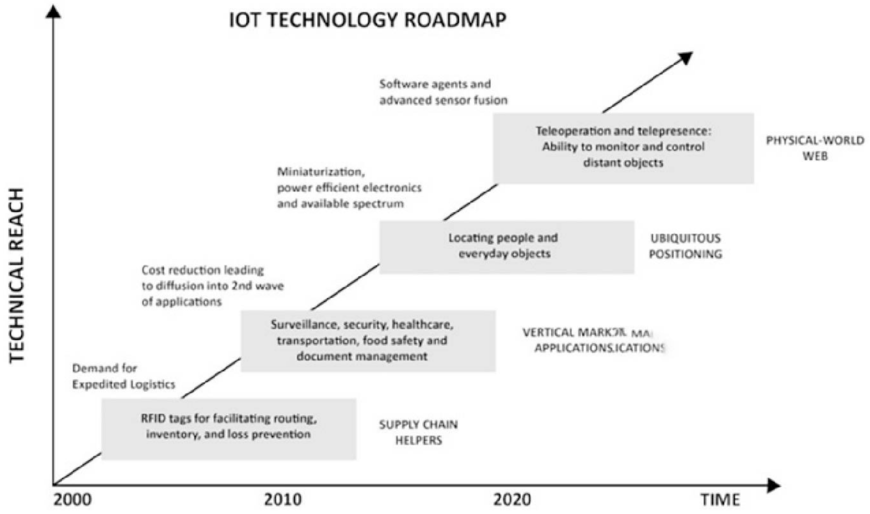
Rather than building a world wide infrastructure and capital to support an organization's communication, the cloud approach can be used to form a virtual network. This offers optimal use of resources and can be purchased and deployed rapidly. Amazon Web Services providing services for Web hosting is an excellent example of NaaS. Software as a service. SaaS is the foundation of cloud computing. The service provider usually offers services or applications that clients can run directly on the mobile devices or other clouds. From a user's perspective, the SaaS layer is virtually running on their client device. Big brand applications such as the Google Apps and Microsoft Office 365 works on SaaS services. Platform as a service. Platform as a service also known as PaaS refers to using the clouds underlying hardware and low-level software facilities. This implies that the end users only use the providers, hardware, OS, middleware and various frameworks to host their private applications and services. Swedbank and Toshiba are a few of the famous companies built using the PaaS model. IBM BlueMix, Microsoft Azure, and Google App Engine are some of the public platform as a service providers in the market. Infrastructure as a service in the IaaS, or infrastructure as a service model. The provider builds scalable hardware services in the cloud and also provides few software frameworks to build clients virtual machines. Due to this, a lot of flexibility can be obtained through the deployment, but at the same time the customer has to put in more effort to work with the system as the customer has to configure from the OS level up to the application level. Cisco Meta Cloud and the Google Compute engine are examples of the infrastructure as a service model. Cloud computing has aided in boosting the efficiency in day to day tasks without disturbing the quality of data to be stored or transferred. Since the relationship between cloud computing and IoT is mutual, both the services complement each other, which has aided the rapid growth in both fields. In IoT the other team that has also been in close contact to cloud computing is edge computing. Due to the distance between the users and the data centers where the cloud service is hosted, there may be latency. Edge computing is exactly based on this feature of minimizing the distance that data has to travel while still retaining the centralized nature of cloud computing. Bringing the computing aspect as close to the source of data aids and reducing latency and use of bandwidth.

The term edge in edge computing is a bit fuzzy and it is not well defined. A personal computer or a microcontroller in an IoT device can be considered as an edge, but the router or the Internet service provider can also be considered as an edge. Here, the important takeaway is that the edge of the network is geographically as close to the network as possible. To understand edge computing a bit more, let us take an example of IoT security video cameras. Consider a building that is secured by high definition cameras. These cameras are ordinary objects and they just record the data and send it to the cloud for processing. Now the cloud runs a motion detection application on the raw input obtained from the video cameras and then saves only those clips which have activity in them. Such a system brings in constant strain on the building's Internet infrastructure as bandwidth is used to transport high volumes of data to the cloud. Additionally, there is a heavy load on the cloud for processing data of several video cameras together.



However, if the motion detection application is moved on to the network edge each of the cameras, have their own internal computer to run the motion detection application and then send relevant video footage only to the cloud server.

This will significantly reduce bandwidth use because only necessary camera footage has to travel to the cloud server. Additionally, the cloud will also have less load and will only be responsible for the clips storage, meaning that the cloud can now communicate with the larger fleet of cameras. This is what edge computing looks like. However, you should not compare cloud computing and edge computing technologies directly as each of them is more useful and advantageous than the other in their own respective scenarios.


IOT TECHNOLOGY ROADMAP
TECHNICAL REACH
Software agents and advanced sensor fusion
Teleoperation and telepresence: Ability to monitor and control distant objects
PHYSICAL-WORLD WEB
Miniaturization, power efficient electronics and available spectrum
Locating people and everyday objects
UBIQUITOUS POSITIONING
Cost reduction leading to diffusion into 2nd wave of applications
Surveillance, security, healthcare, transportation, food safety and document management
VERTICAL MARKET APPLICATIONS
Demand for Expedited Logistics
RFID tags for facilitating routing, inventory, and loss prevention
SUPPLY CHAIN HELPERS
2000
2010
2020
TIME

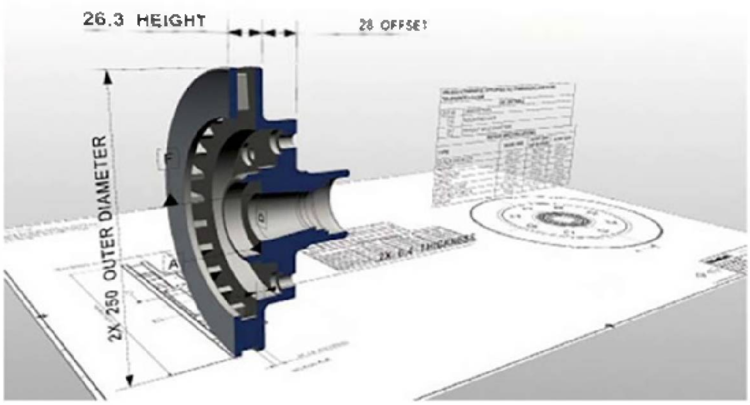
Apart from cloud computing and edge computing, IoT also depends on other enabling technologies to maximize its opportunities. Let us look at some of the technologies that have paved a path for enabling IoT. RFID radio frequency identification provides a simple, low energy and versatile option for identity and access tokens. RFID employs two way radio communication transmitters and receivers to identify and track information associated with the tags. RFID tags are primarily used to make everyday objects communicate through radio frequency and report their status. Retail, manufacturing, logistics, smart warehousing and banking are among the major industries where it is used. Bluetooth low energy. As the name suggests. It is the standard Bluetooth technology, but with lower power, which enables the devices to run for a longer period of time than usual.

As BLE is developed in conjunction with the standard Bluetooth, all of its functions can be used and there is no need to establish new support for other systems. This technology is well suited for short range communication, which involves low to medium output. Fitness, health devices and human interface devices are some of the key targets of this technology.

SENSORS

Sensors The development in the field of sensors has also boosted the IoT industry. Modern sensors can detect changes in a wide range of specific parameters like pressure, temperature and communicate that to the cloud or gateway.

Nowadays, microscopic sensors made using micro electromechanical systems can be easily embedded into clothing and other smart devices.



Digital twin. The concept of digital twin evolved from the idea of having a digital form of an engineering sketch or graphic. In the context of IoT a Digital twin is a digital representation of a physical operation or a system. The twins are designed so as to simulate real world use cases in a digital space. It can be constructed so as to receive input from sensors which gather data through a real world counterpart. This allows the team to simulate the physical object in real time in the process, thus offering insights into the performance and potential problems. This technology is heavily used in the automotive and healthcare sectors. Additionally, rapid growth in technology such as big data, communications and analytic software have also had their impact on the growth of IoT.

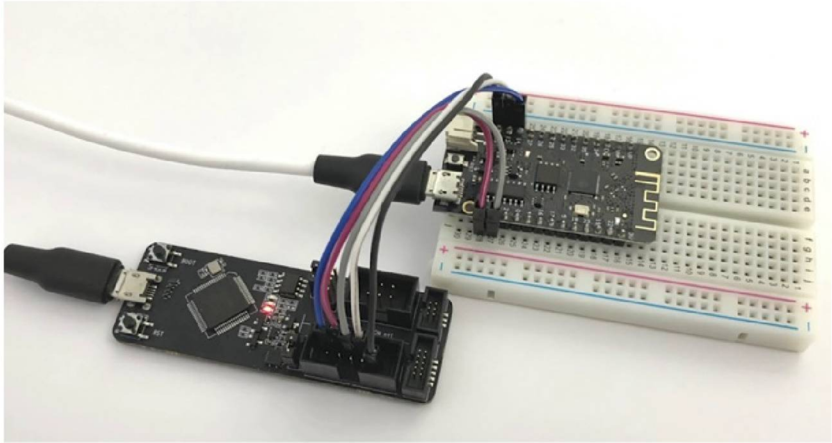
INTRODUCTION TO CAYENNE

we will learn the following topics in Introduction to Cayenne getting started with Cayenne part 1 getting started with Cayenne part 2 In-depth understanding of Cayenne MQTT library setting up IoT projects on

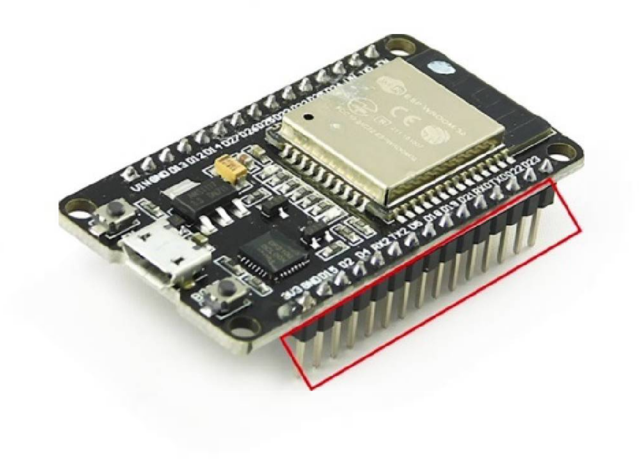
Cayenne and troubleshooting. What is Cayenne? How Cayenne works? Why Cayenne? Supported hardware, library support. Now that you are clear with your IoT concepts, it is time for some hands on experience. Let us start by learning about Cayenne. So what is Cayenne? Cayenne is the first of its kind. Drag and drop IoT project builder that helps developers connect and host their connected device projects quickly.

This drag and drop feature removes the ambiguity in hardware programming. It not only makes it possible to build programs using drag and drop, but it also standardizes the connection of devices such as sensors and actuators and make sure that the drivers are in place. You can say that Cayenne was designed for IoT. It can control hardware remotely. It can display sensor data and also store and analyze this data. This seems a bit complex, right? You might be wondering how does Cayenne work? Cayenne has three major components, the app, the online dashboard cloud and the agent.

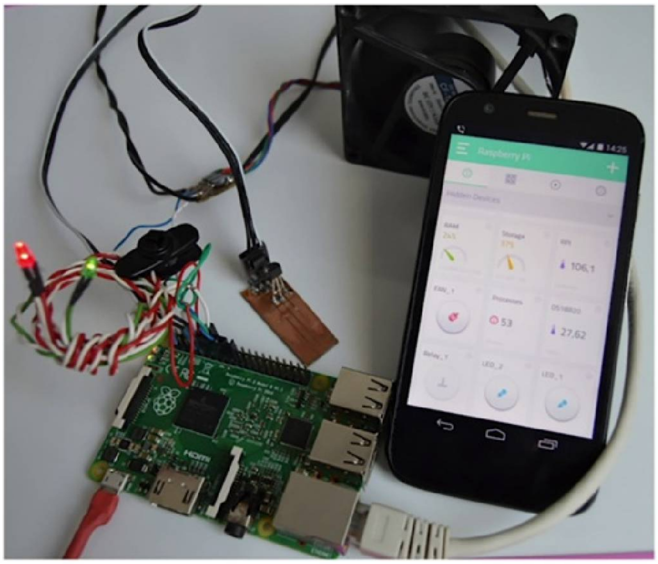
After you set up an account on the Cayenne platform, you will set up your device. After this, you can use the drag and drop feature of this platform to control everything. It has options to display CPU, RAM, storage and any other quantity a specific sensor can measure.



Of course, you have to wire up your circuit. That is no way a software can do this. You cannot use a sensor that Cayenne does not support, but the range of sensors supported is enough for most of the projects.



You can also get direct access to GPIO pins. So why do we need to use Cayenne.

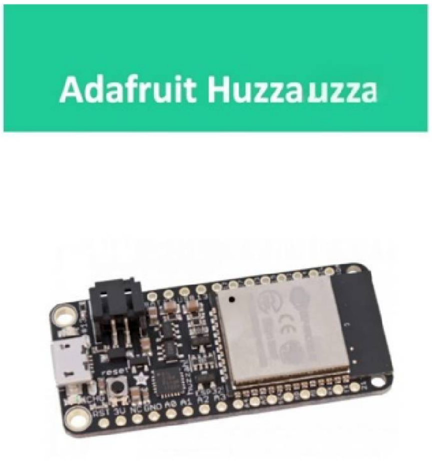


The simple yet effective build of the Cayenne platform makes it a great option for IoT project building. Also in the project building phase of any product, the first step is building the prototype using the Cayenne's drag and drop feature. You can easily build a prototype of your project.

As Cayenne takes care of everything from building a program to standardizing the connections of devices. It saves a lot of time and resources while giving you a comprehensive insight into the project. Cayenne allows you to quickly design, prototype and visualize IoT solutions. Thus, Cayenne is best suited for rapid prototyping.

Originally, the Cayenne platform only worked with the Raspberry Pi. However, now it is developed and expanded. Let us take a look at the hardware that the Cayenne platform supports.

Cayenne supports single board computers such as the Raspberry Pi microcontrollers such as the Arduino and ESP32 are also supported.

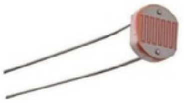


You can also add your different boards.

PIR Sensor

Thermistor

Photoresistor



This platform support sensor such as the PIR sensor, thermistor, photoresistor and many more.

Light Switch

Motor Switowitc

The light switch and the motors switch are some of the actuators supported by the platform.

AcSiP S76S

ADS 1015

MCP23008

Cayenne, also has support for LoRa WAN devices and extensions for PWM analog and digital inputs.

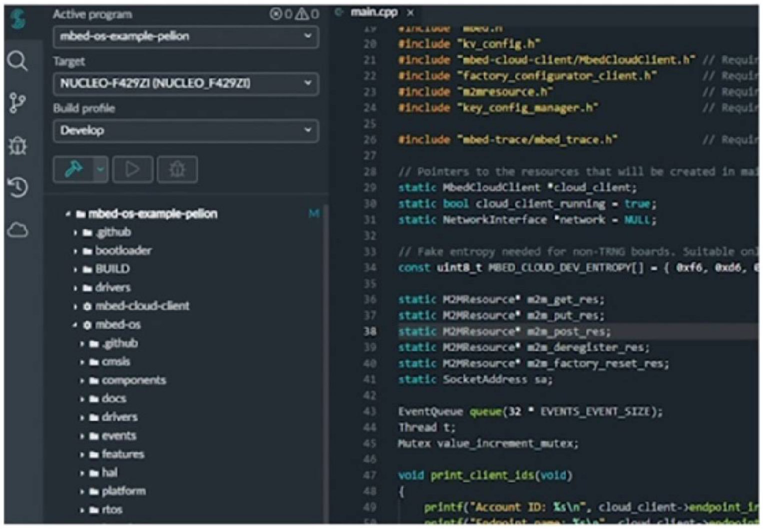
Along with hardware, Cayenne also has a good amount of library support. Cayenne is used by both beginners and experts and based on this Cayenne has made some libraries available.

Arduino UNO

ESP8266

ESP32

For beginners, the Arduino MQTT library is introduced. This library can be used with the Arduino IDE and boards such as the Arduino UNO, ESP8266 and the ESP32.



Active program: mbed-os-example-pelion

Target: NUCLEO-F429ZI (NUCLEO_F429ZI)

Build profile: Develop

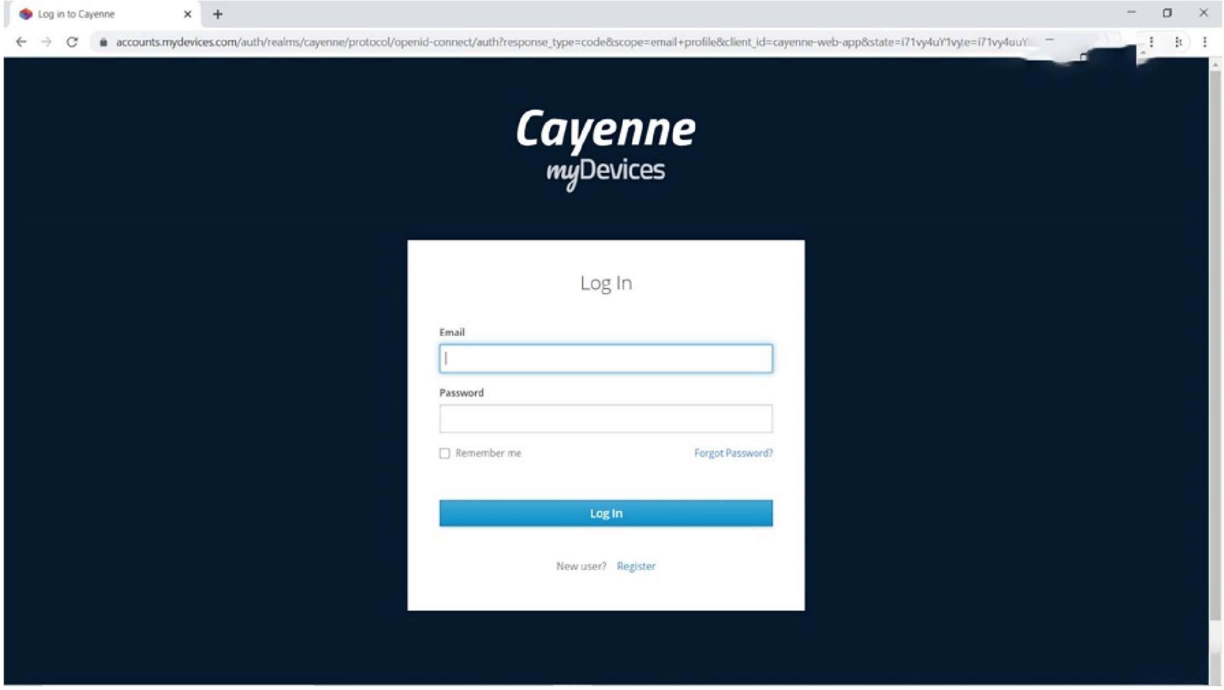
```
#include "kv_config.h"
#include "mbed-cloud-client/MbedCloudClient.h" // Requir
#include "factory_configurator_client.h"       // Requir
#include "m2mresource.h"                       // Requir
#include "key_config_manager.h"                // Requir

#include "mbed-trace/mbed_trace.h"             // Requir

// Pointers to the resources that will be created in mai
static MbedCloudClient *cloud_client;
static bool cloud_client_running = true;
static NetworkInterface *network = NULL;

// Fake entropy needed for non-TRNG boards. Suitable onl
const uint8_t MBED_CLOUD_DEV_ENTROPY[] = { 0xf6, 0xd6, 0

static M2MResource* m2m_get_res;
static M2MResource* m2m_put_res;
static M2MResource* m2m_post_res;
static M2MResource* m2m_deregister_res;
static M2MResource* m2m_factory_reset_res;
static SocketAddress sa;

EventQueue queue(32 * EVENTS_EVENT_SIZE);
Thread t;
Mutex value_increment_mutex;

void print_client_ids(void)
{
    printf("Account ID: %s\n", cloud_client->endpoint_in
```

For intermediate users The mbed MQTT library is also available. It can be used in conjunction with the mbed IDE. The embedded C MQTT library and the C++ MQTT Library are available for advanced users with more features. These libraries also include support from the Cayenne team to update and customize. support for new boards.

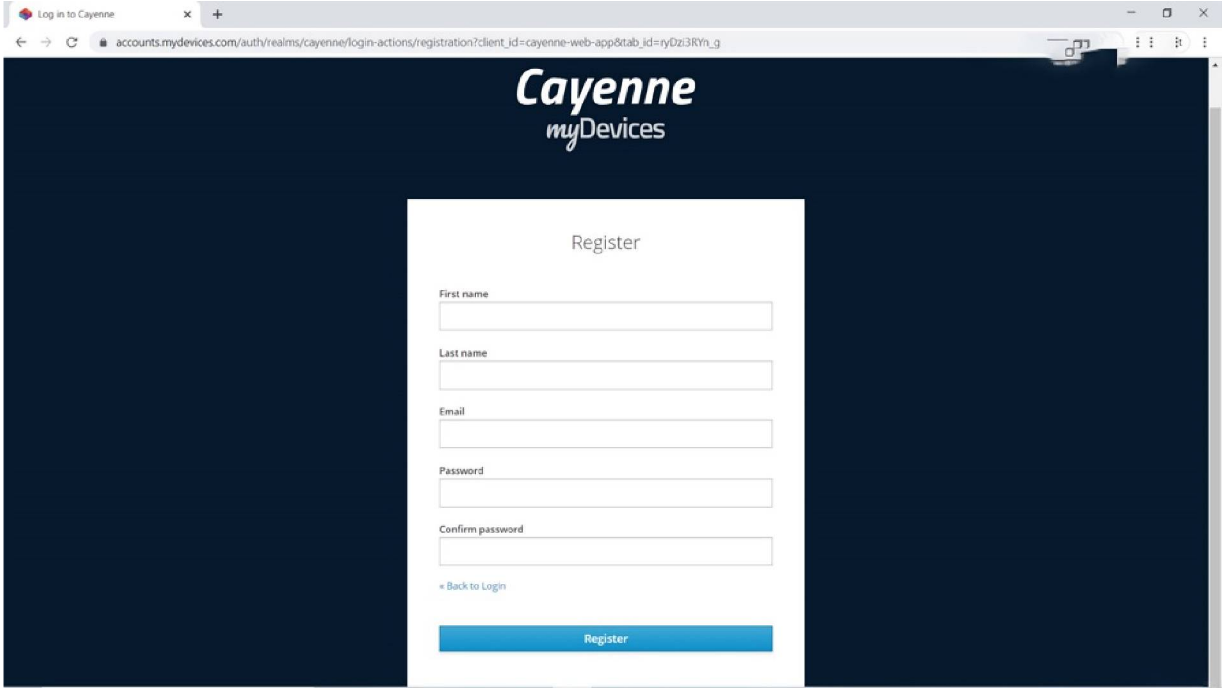
Cayenne is a cloud based end to end IoT solution that not only offers you the greatest tools and library support, but also handles the difficult part of device management of the actual IoT hardware and software, thereby making it the right platform to get started with while learning IoT and for rapid prototyping.

GETTING STARTED WITH CAYENNE SETUP

we will cover the following topics, create a Cayenne account, choose the device, what is meant by bring your own thing, Installing Cayenne MQTT ESP Library on Arduino IDE . Now that you have an overview of the Cayenne platform, let us set up your Cayenne account.


Log in to Cayenne
accounts.mydevices.com/auth/realms/cayenne/protocol/openid-connect/auth?response_type=code&scope=email+profile&client_id=cayenne-web-app&state=i71vy4uY1vyte=i71vy4uuY
Cayenne
myDevices
Log In
Email
Password
□ Remember me
Forgot Password?
Log In
New user? Register

Let's go to the Cayenne website now. Here you will be asked for your email address and password.



Log in to Cayenne

accounts.mydevices.com/auth/realms/cayenne/login-actions/registration?client_id=cayenne-web-app&tab_id=ryDzi3RYn_g

Cayenne
myDevices

Register

First name

Last name

Email

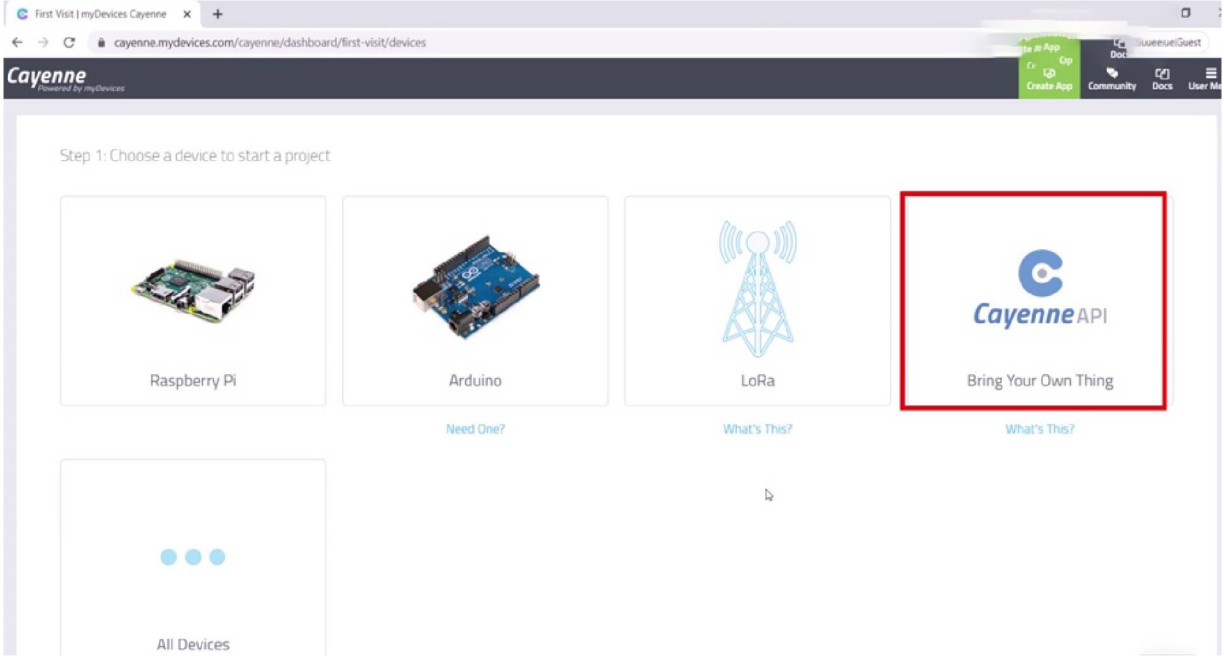
Password

Confirm password

« Back to Login

Register

However, you first need to create an account with Cayenne. Click here to register. Now, note down the email address and password you will need to log in later. Once you have created your account, you need to go to my devices Cayenne login page. Now enter your email address, password and then login.



First Visit | myDevices Cayenne

cayenne.mydevices.com/cayenne/dashboard/first-visit/devices

Cayenne
Powered by myDevices

Create App Community Docs User Me

Step 1: Choose a device to start a project

Raspberry Pi

Arduino

Need One?

LoRa

What's This?

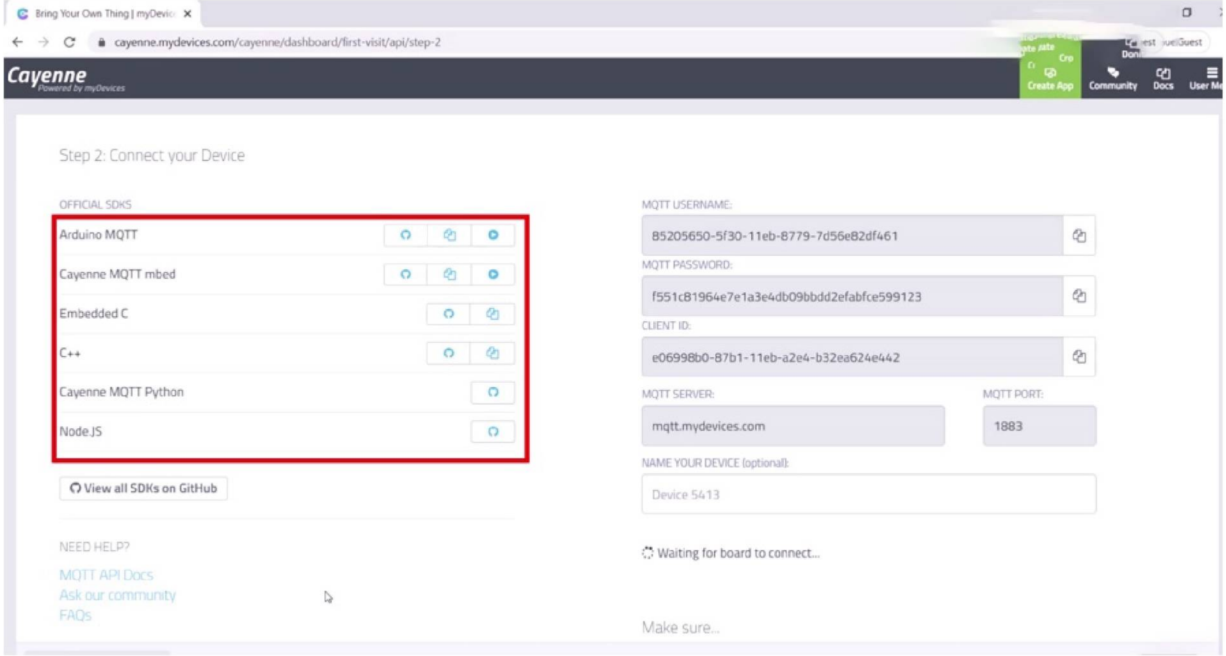
Cayenne API

Bring Your Own Thing

What's This?

All Devices

Firstly, you will be asked to choose your device which you want to interface and work on. As you can see, there is already pre-built support for Raspberry Pi, Arduino and LoRa, but you will not find the ESP32 here. You will need to manually add the ESP32 as a device using the Cayenne MQTT API.


Bring Your Own Thing | myDevic
cayenne.mydevices.com/cayenne/dashboard/first-visit/api/step-2
Cayenne
Powered by myDevices
Create App
Community
Docs
User Me

Step 2: Connect your Device

OFFICIAL SDKS

- Arduino MQTT
- Cayenne MQTT mbed
- Embedded C
- C++
- Cayenne MQTT Python
- Node.JS

MQTT USERNAME: 85205650-5f30-11eb-8779-7d56e82df461

MQTT PASSWORD: f551c81964e7e1a3e4db09bbdd2efabfce599123

CLIENT ID: e06998b0-87b1-11eb-a2e4-b32ea624e442

MQTT SERVER: mqtt.mydevices.com

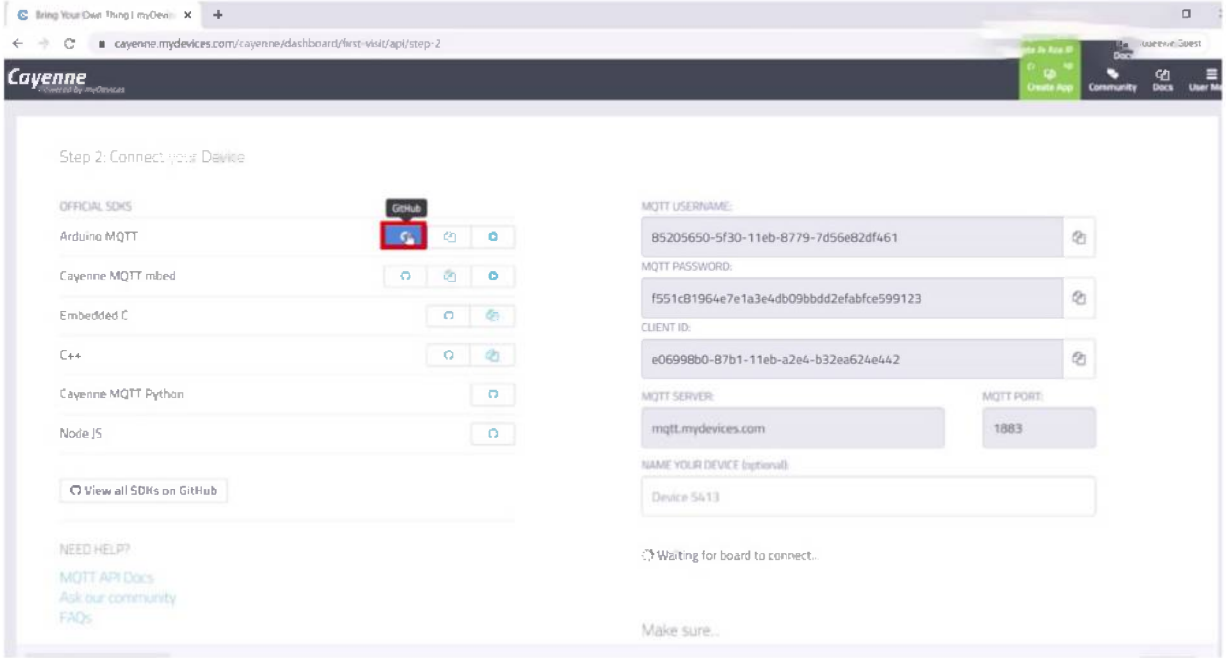
MQTT PORT: 1883

NAME YOUR DEVICE (optional): Device 5413

Waiting for board to connect...

Make sure...

Now, click here, as you can see, there are a number of official SDKs available for MQTT support.



Step 2: Connect your Device

OFFICIAL SDKS

- Arduino MQTT
- Cayenne MQTT mbed
- Embedded C
- C++
- Cayenne MQTT Python
- Node JS

MQTT USERNAME: 85205650-5f30-11eb-8779-7d56e82df461

MQTT PASSWORD: f551c81964e7e1a3e4db09bbdd2efabfce599123

CLIENT ID: e06998b0-87b1-11eb-a2e4-b32ea624e442

MQTT SERVER: mqtt.mydevices.com

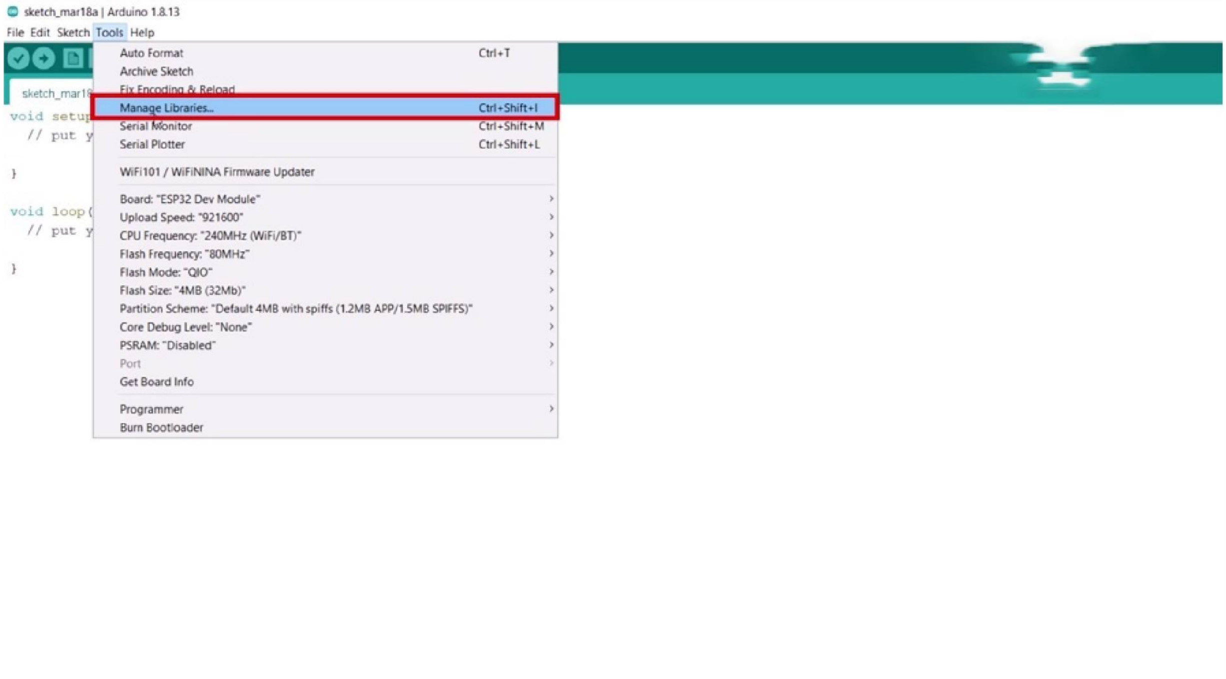
MQTT PORT: 1883

NAME YOUR DEVICE (optional): Device 5413

Waiting for board to connect...

Make sure...

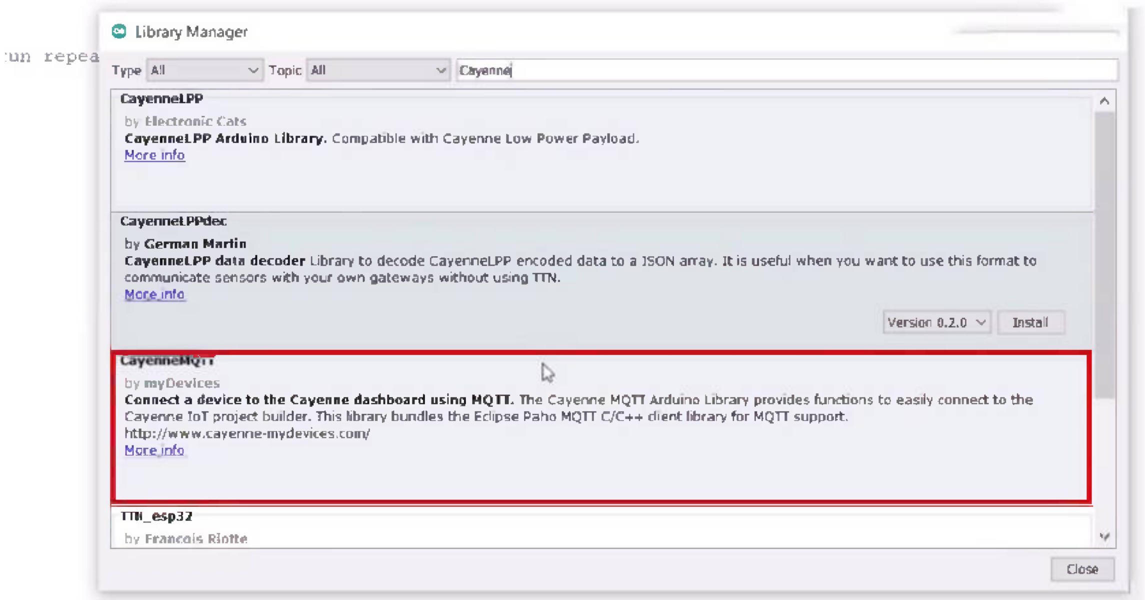
The Arduino MQTT SDK is one of them. The Cayenne MQTT Arduino Library contains functions and codes that could easily connect you to the Cayenne IoT dashboard. It has support for Arduino microcontroller boards as well as other ESP8266 and ESP32 based development boards. You can click here to access the GitHub page for the same.


sketch_mar18a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_mar18
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Port
Get Board Info
Programmer
Burn Bootloader

```
void setup
  // put y

}

void loop(
  // put y

}
```

Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"

Now let's get back to the Arduino IDE and see how we can install the Cayenne MQTT ESP library. First, you need to go to tools and then click manage libraries This will open up the library manager.



Cayenne

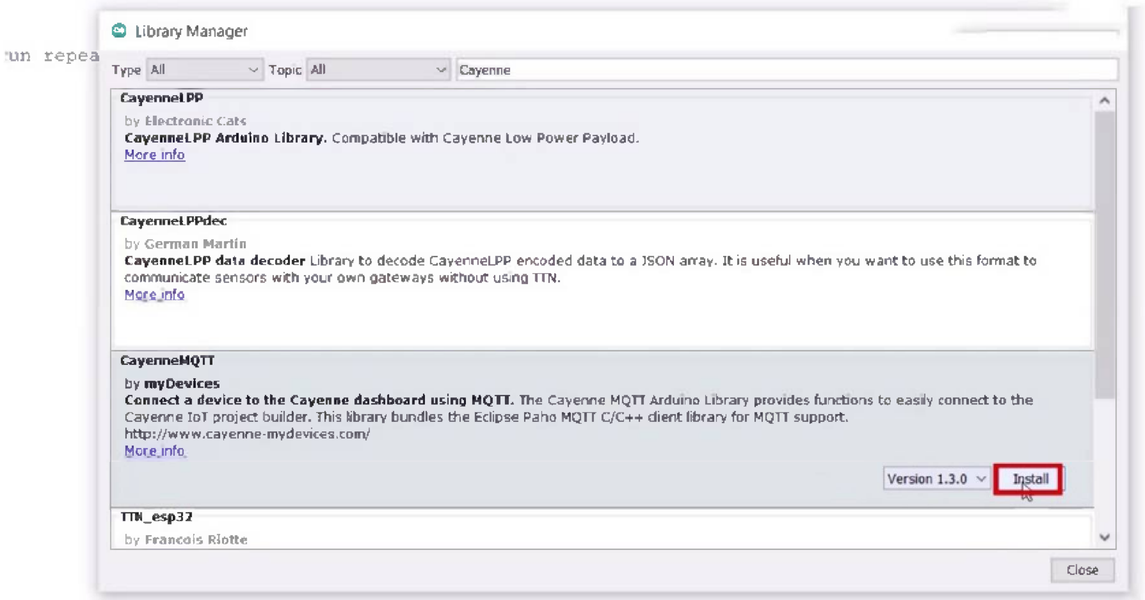
CayenneLPP
by Electronic Cats
CayenneLPP Arduino Library. Compatible with Cayenne Low Power Payload.
More info

CayenneLPPdec
by **German Martin**
CayenneLPP data decoder Library to decode CayenneLPP encoded data to a JSON array. It is useful when you want to use this format to communicate sensors with your own gateways without using TTN.
More info
Version 0.2.0 Install

CayenneMQTT
by myDevices
Connect a device to the Cayenne dashboard using MQTT. The Cayenne MQTT Arduino Library provides functions to easily connect to the Cayenne IoT project builder. This library bundles the Eclipse Paho MQTT C/C++ client library for MQTT support.
http://www.cayenne-mydevices.com/
More info

TTN_esp32
by Francois Riotte

Now search for Cayenne. You will find the Cayenne MQTT Library.



Cayenne

CayenneLPP
by Electronic Cats
CayenneLPP Arduino Library. Compatible with Cayenne Low Power Payload.
More info

CayenneLPPdec
by German Martin
CayenneLPP data decoder Library to decode CayenneLPP encoded data to a JSON array. It is useful when you want to use this format to communicate sensors with your own gateways without using TTN.
More info

CayenneMQTT
by **myDevices**
Connect a device to the Cayenne dashboard using MQTT. The Cayenne MQTT Arduino Library provides functions to easily connect to the Cayenne IoT project builder. This library bundles the Eclipse Paho MQTT C/C++ client library for MQTT support.
http://www.cayenne-mydevices.com/
More info
Version 1.3.0 Install

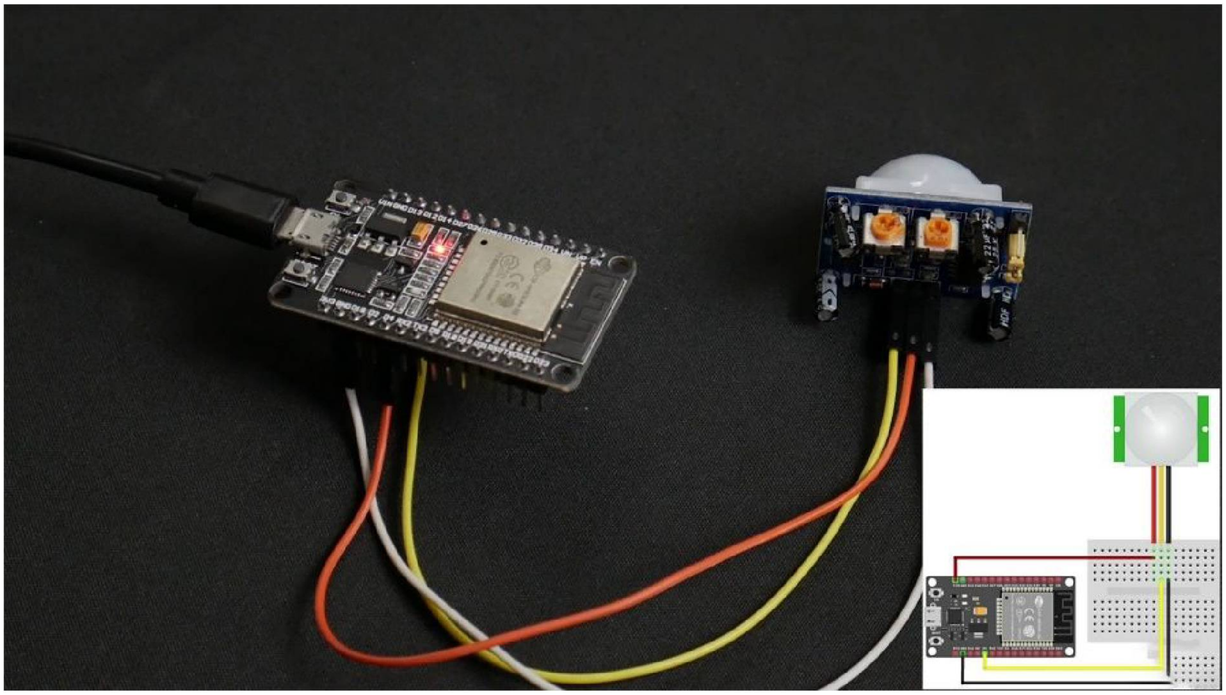
TTN_esp32
by Francois Riotte

Now install the latest version of this library. Once installed, you are now ready to program the ESP32 to connect it to Cayenne.

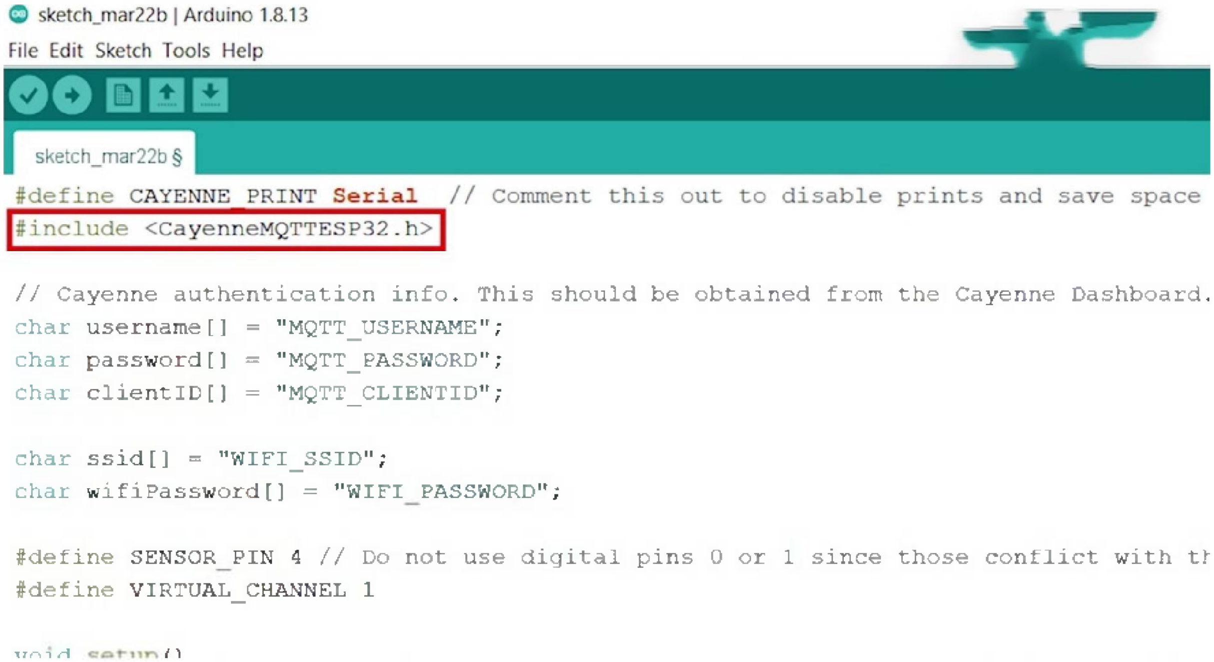
DASHBOARD OVERVIEW

we will cover the following topics, adding your device, dashboard overview, what projects, devices and widgets? What are events and triggers? Now that you have set up your own account. Let's get started with a small project using Cayenne and the ESP32.

You will also be needing a PIR sensor, if you want to learn about the PIR sensor and how it works. We recommend you to go through this before proceeding. First, let's complete the hardware connection part for the project. The PIR sensor has three pins, the VCC, the ground and the data out pin. It also consists of two potentiometers through which we can adjust the sensors delay time and sensitivity Using Jumper wires please connect the VCC pin and ground pin of the PIR sensor to the VCC pin and ground pin of the ESP32 respectively.

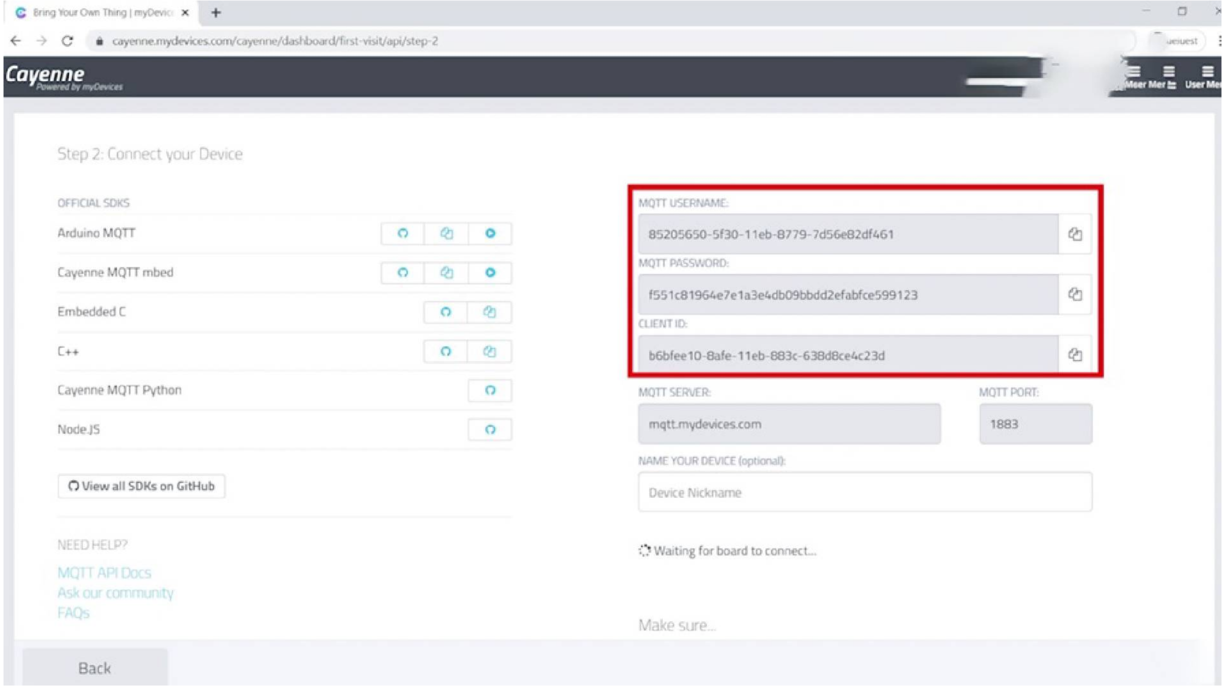


To connect the data out pin You can use any GPIO pin, but for now we will be using the GPIO pin 4. Connect the data out pin of the PIR sensor to the GPIO pin 4 of the ESP32.



```
#define CAYENNE_PRINT Serial  // Comment this out to disable prints and save space
#include <CayenneMQTTESP32.h>

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "MQTT_CLIENTID";

char ssid[] = "WIFI_SSID";
char wifiPassword[] = "WIFI_PASSWORD";

#define SENSOR_PIN 4 // Do not use digital pins 0 or 1 since those conflict with th
#define VIRTUAL_CHANNEL 1

void setup()
```

Now let's check and understand the code. This here is used to print the general Wi-Fi status. We will then be importing the MQTT ESP32 library into the code. It contains functions that enable you to send and receive data from Cayenne. The next step is to enter your Cayenne credentials, go back to your Cayenne account and click here.



Step 2: Connect your Device

MQTT USERNAME: 85205650-5f30-11eb-8779-7d56e82df461

MQTT PASSWORD: f551c81964e7e1a3e4db09bbdd2efabfce599123

CLIENT ID: b6bfee10-8afe-11eb-883c-638d8ce4c23d

MQTT SERVER: mqtt.mydevices.com

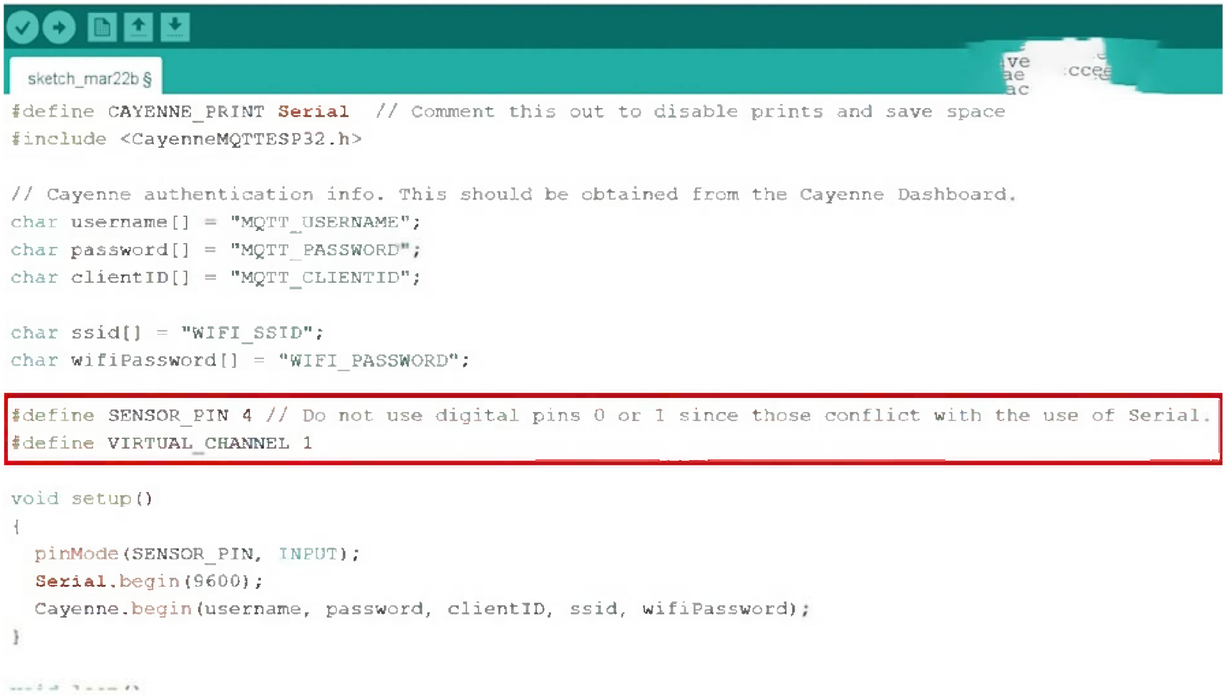
MQTT PORT: 1883

NAME YOUR DEVICE (optional): Device Nickname

Waiting for board to connect...

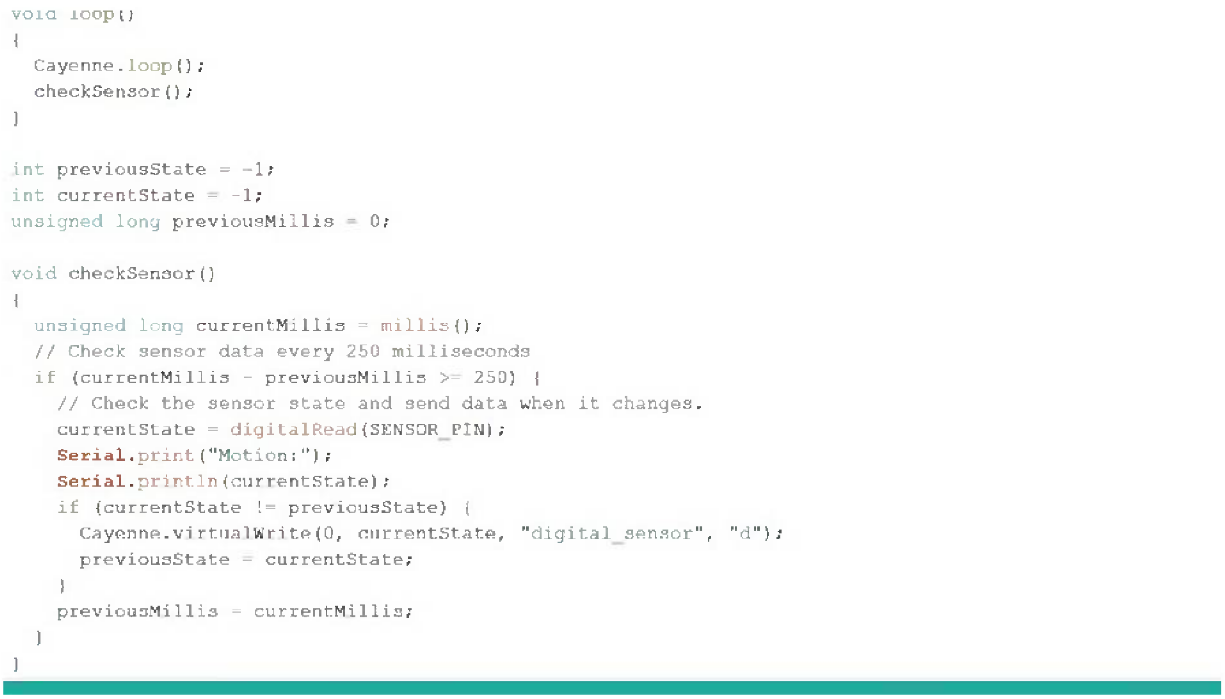
Make sure...

Now enter your Cayenne credentials in your code accordingly. The client ID is used to differentiate between different clients connected to Cayenne.



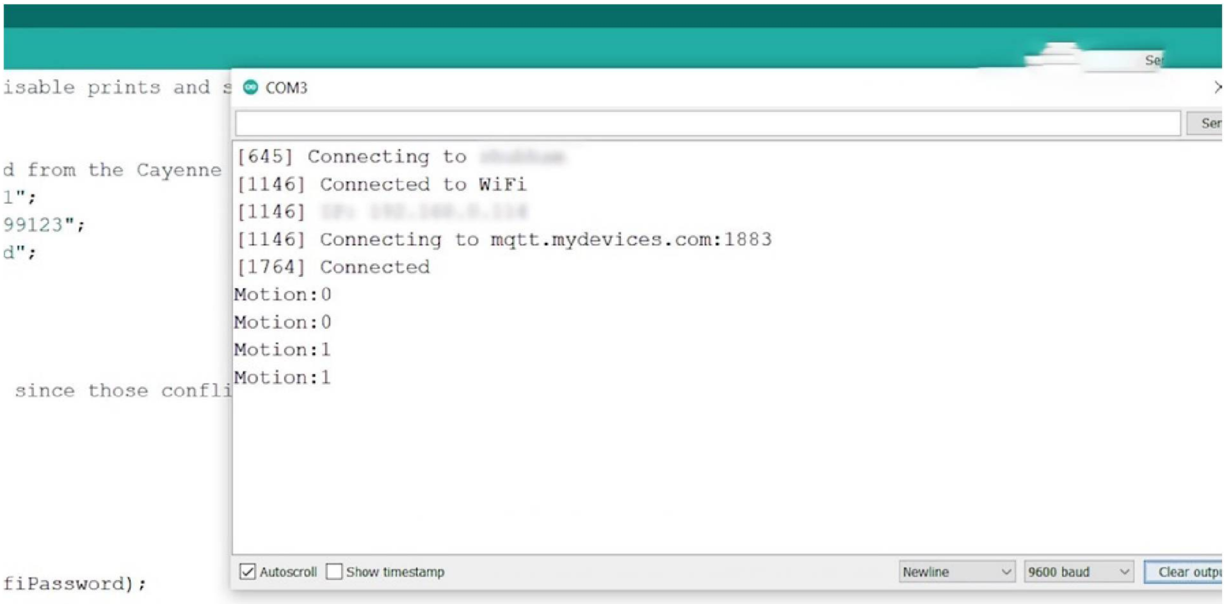
```
#define CAYENNE_PRINT Serial  // Comment this out to disable prints and save space
#include <CayenneMQTTESP32.h>

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "MQTT_CLIENTID";

char ssid[] = "WIFI_SSID";
char wifiPassword[] = "WIFI_PASSWORD";

#define SENSOR_PIN 4 // Do not use digital pins 0 or 1 since those conflict with the use of Serial.
#define VIRTUAL_CHANNEL 1

void setup()
{
  pinMode(SENSOR_PIN, INPUT);
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
}

void loop()
```

You will also need to provide your Wi-Fi credentials. Here we have defined Pin 4 as the sensor pin, and a virtual channel to which the data will be send in Cayenne. Now here we initialize the serial Baudrate as 9600. This is needed to connect Cayenne using Cayenne's username, password, client-ID, Wi-Fi SSID Wi-Fi password.



```
void loop()
{
  Cayenne.loop();
  checkSensor();
}

int previousState = -1;
int currentState = -1;
unsigned long previousMillis = 0;

void checkSensor()
{
  unsigned long currentMillis = millis();
  // Check sensor data every 250 milliseconds
  if (currentMillis - previousMillis >= 250) {
    // Check the sensor state and send data when it changes.
    currentState = digitalRead(SENSOR_PIN);
    Serial.print("Motion:");
    Serial.println(currentState);
    if (currentState != previousState) {
      Cayenne.virtualWrite(0, currentState, "digital_sensor", "d");
      previousState = currentState;
    }
    previousMillis = currentMillis;
  }
}
```

Now we include the Cayenne dot loop function here to execute all the Cayenne functions repeatedly. The check sensor function is used to sense data from the sensor and print it accordingly. Now the check sensor function is where you read and write the data from the sensor. Here we check sensor data every 250 milliseconds. Now here we read the sensor data from the data out pin and print it to the Serial Monitor in Arduino IDE. Here we check the sensor state and send data when it changes to Channel Zero.

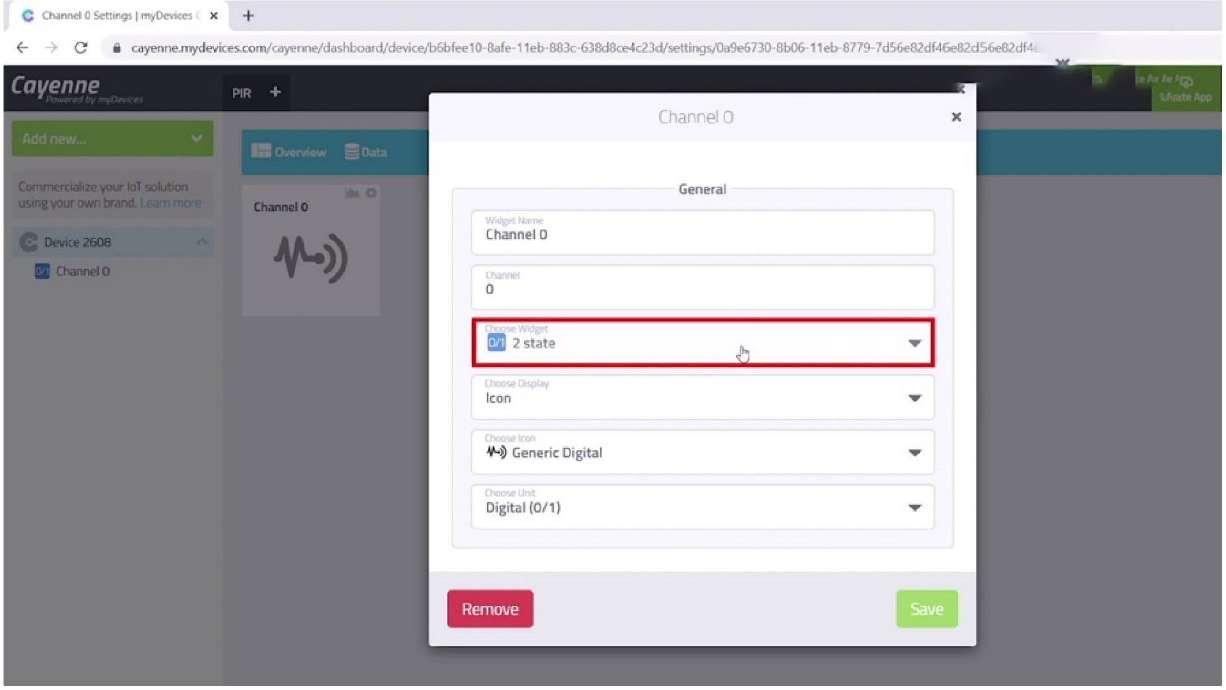


```
isable prints and s
d from the Cayenne
1";
99123";
d";
 since those confli
fiPassword);
```

```
[645] Connecting to [redacted]
[1146] Connected to WiFi
[1146] [redacted]
[1146] Connecting to mqtt.mydevices.com:1883
[1764] Connected
Motion:0
Motion:0
Motion:1
Motion:1
```

Now that you are familiar with the code, save it and click on the upload button. Click on the serial monitor. It will display the IP address and connection status of your device. It will then display if motion is detected or not. Zero signifies that there is no motion detected, while one signifies that motion is detected. If the serial monitor displays that your device is not connected or does not show any output values always check if you have entered all the credentials correctly and you have made the proper connections in the circuit.

Now go back to Cayenne. Your device will be connected and you will notice channel zero on your dashboard. Click on the plus sign on the top right corner of the channel. This will add the channel to your dashboard.



Channel 0 Settings | myDevices
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d/settings/0a9e6730-8b06-11eb-8779-7d56e82df46e82d56e82df4
Cayenne
Powered by myDevices
PIR
Add new...
Commercialize your IoT solution using your own brand. Learn more
Device 2608
Channel 0
Overview
Data
Remove
Save

Channel 0

General

Widget Name: Channel 0

Channel: 0

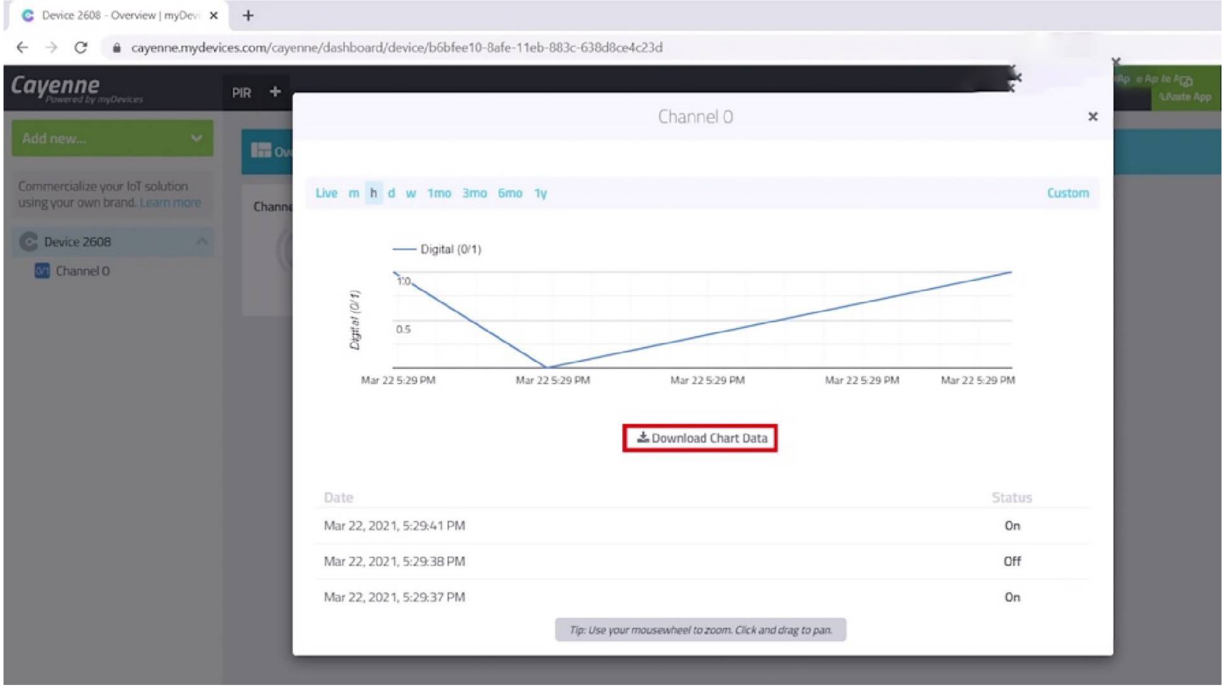
Choose Widget: 0/1 2 state

Choose Display: Icon

Choose Icon: Generic Digital

Choose Unit: Digital (0/1)

Now click on the setting options. here. Here, choose the widget to motion detector.



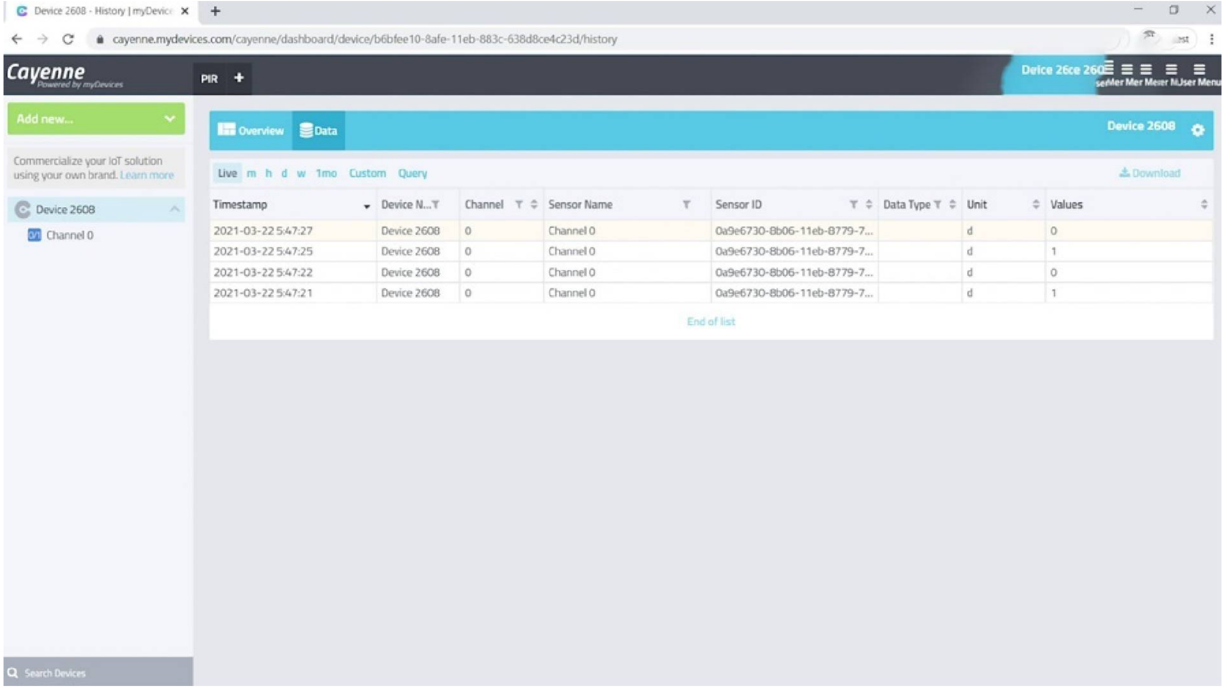
Date	Status
Mar 22, 2021, 5:29:41 PM	On
Mar 22, 2021, 5:29:38 PM	Off
Mar 22, 2021, 5:29:37 PM	On

Tip: Use your mousewheel to zoom. Click and drag to pan.

Now if you check whenever the motion is detected, the Channel 0 will show you an animation. Now click here. This will show you a chart about when motion was detected. You can download the chart data. Now that you have practically used Cayenne. Let's check onto the dashboard and various components of Cayenne. The Cayenne dashboard is the main screen where you can setup, customize, monitor, manage and control your connected devices.

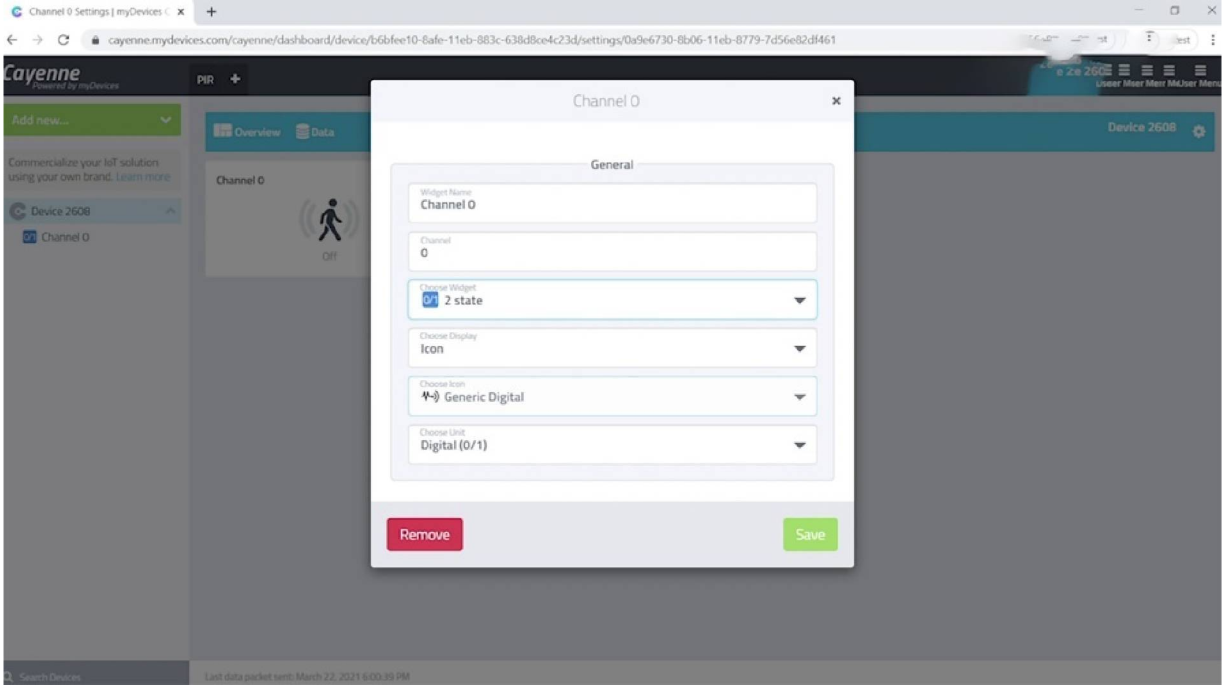
At the top right corner of the Cayenne dashboard, you can see some options. The create apps option will take you to a page. If you want to partner with my devices to create a commercial product. The community option takes you to the Cayenne community where you can post your queries in context to Cayenne. You can also find solutions through queries other people may have asked. Cayenne has a very active community. The Docs option will take you to a page where all the documentation regarding Cayenne is posted. Everything from hardware, Cayenne cloud API to Cayenne MQTT API is mentioned here. Lastly, the user menu gives you your account related information and options.

Exactly below the user menu You can see the settings option where you can configure and reset the dashboard.


Device 2608 - History | myDevic
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d/history
Cayenne
Powered by myDevices
PIR
Dеice 26ce 260
seMer Mer Meer NJser Menu
Add new...
Commercialize your IoT solution using your own brand. Learn more
Device 2608
Channel 0
Overview
Data
Device 2608
Live m h d w 1mo Custom Query
Download
End of list
Search Devices

Timestamp	Device N...	Channel	Sensor Name	Sensor ID	Data Type	Unit	Values
2021-03-22 5:47:27	Device 2608	0	Channel 0	0a9e6730-8b06-11eb-8779-7...		d	0
2021-03-22 5:47:25	Device 2608	0	Channel 0	0a9e6730-8b06-11eb-8779-7...		d	1
2021-03-22 5:47:22	Device 2608	0	Channel 0	0a9e6730-8b06-11eb-8779-7...		d	0
2021-03-22 5:47:21	Device 2608	0	Channel 0	0a9e6730-8b06-11eb-8779-7...		d	1

In the same blue bar you will see two options overview and data. The overview option pictures your project as a whole where different channels can be placed. The data option gives you an insight into the data collected in the project. You can see a summary of all the devices connected and the channels present in them on the left hand side. Still, you can find many terms on the dashboard, which may be unclear. So What are projects, devices and widgets? The projects feature allows you to use widgets from any combination of connected devices into one custom dashboard called a project. Each project has its own set of features that you can set up. Cayenne uses widgets to visualize devices, their data, status and actions. Every device, sensor and actuator added in Cayenne has one or more widgets associated. Depending on the hardware capabilities. Widgets can be rearranged on your dashboard simply by using the drag and drop feature of Cayenne. To move a widget place your mouse or finger at the top middle area of the widget, then tap and move the widget where you want it on your dashboard.



cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d/settings/0a9e6730-8b06-11eb-8779-7d56e82df461

Channel 0

General

Widget Name: Channel 0

Channel: 0

Choose Widget: 0/1 2 state

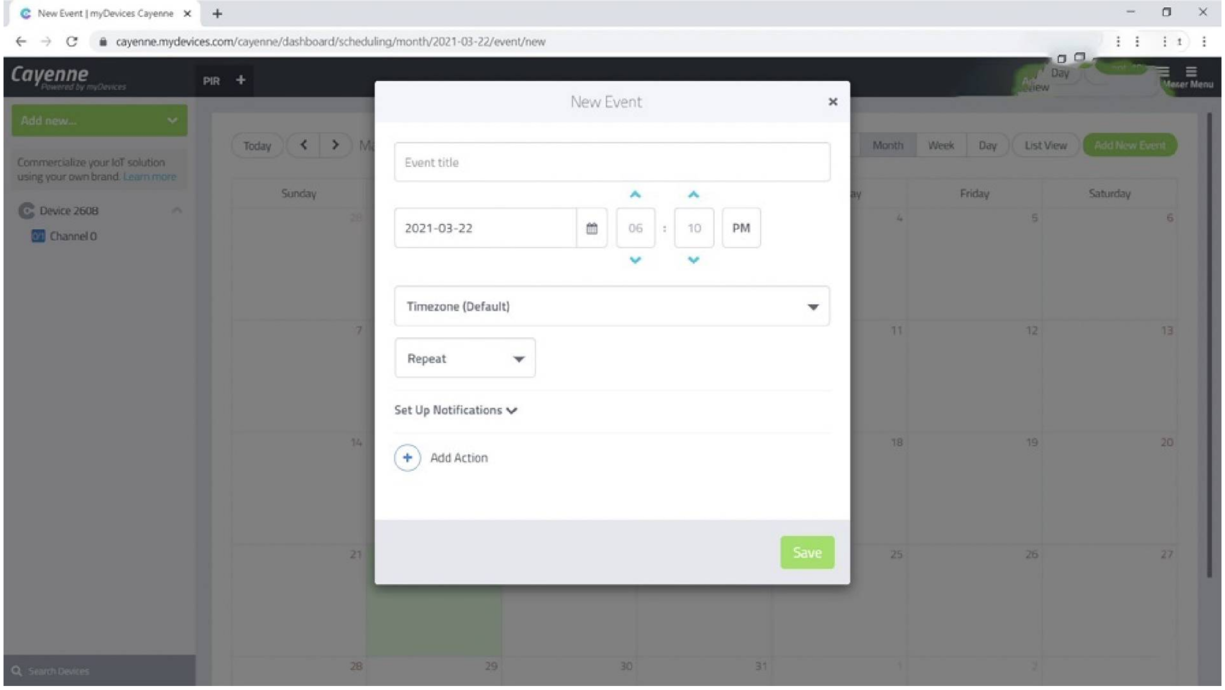
Choose Display: Icon

Choose Icon: Generic Digital

Choose Unit: Digital (0/1)

Last data packet sent: March 22, 2021 6:00:39 PM

A variety of widget parameters can be custom made. According to the user. You can access the widget settings by selecting the wheel in the upper right of the widget. Here you can change the widget type, change the minimum, maximum values and more. Let us also understand another important feature of Cayenne. On the top left corner. If you click on the ADD devices drop down menu, you will see device, widget, event, trigger and project.



New Event | myDevices Cayenne
cayenne.mydevices.com/cayenne/dashboard/scheduling/month/2021-03-22/event/new
Cayenne
Powered by myDevices
PIR
Add new...
Commercialize your IoT solution using your own brand. Learn more
Device 2608
Channel 0
Search Devices
Today
Month
Week
Day
List View
Add New Event
Sunday
Friday
Saturday
User Menu

New Event

Event title

2021-03-22 06 : 10 PM

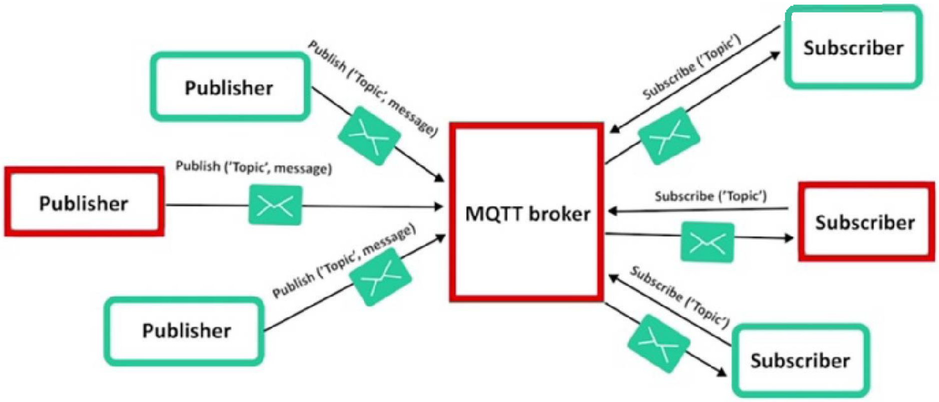
Timezone (Default)

Repeat

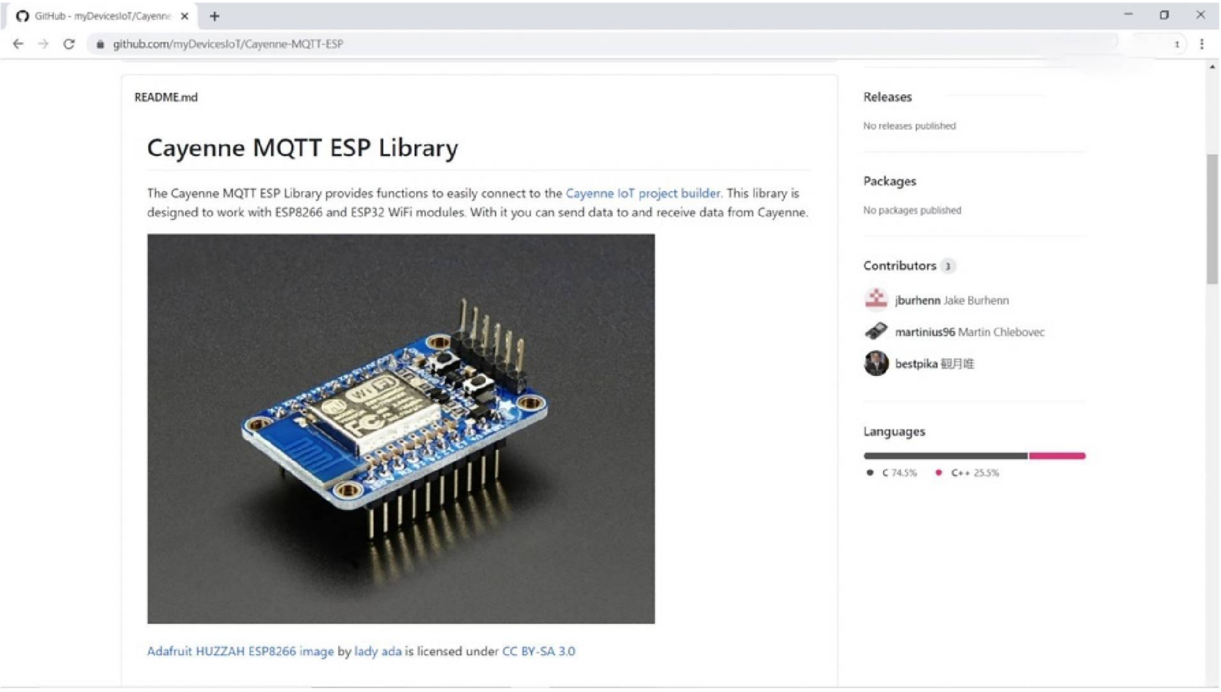
You already have an idea about three of the options, but what are triggers and events? Cayenne allows you to create scheduled events for the connected devices, sensors and actuators connected. A scheduled event can have one or more actions added to it. It is similar to you creating events or reminders on your smartphone. Cayenne also allows you to create triggered actions on and between your connected devices, sensors and actuators based on your device's state. Simply put, if an event takes place, a trigger response action will also take place. Using the event trigger option gives you an effective yet simple way to implement your IoT project.

IN-DEPTH UNDERSTANDING OF THE CAYENNE MQTT ESP LIBRARY

we sent PIR sensor data to the Cayenne dashboard. This was possible because of the Cayenne MQTT API. The Cayenne MQTT API is used to connect any device available to the Cayenne cloud. After connecting your device, you can easily send or receive data from the device to the Cayenne dashboard and display it using widgets. We can also receive commands from Cayenne allowing remote control and automation of your IoT device.



As you have studied about MQTT in the previous, you may remember three important terms publisher, broker and subscriber in MQTT. In Cayenne the Cayenne Cloud acts as a broker who manages various sensors, actuators and client devices that wish to send and receive data using the Cayenne Cloud. Cayenne MQTT is straightforward and easy to use. Offering several ways to send data to Cayenne. Cayenne offers three options to use MQTT. 1 Use the Cayenne MQTT libraries 2. Use raw MQTT API functions. 3. Use HTTP to push MQTT data.


GitHub - myDevicesIoT/Cayenne
github.com/myDevicesIoT/Cayenne-MQTT-ESP

README.md

Cayenne MQTT ESP Library

The Cayenne MQTT ESP Library provides functions to easily connect to the Cayenne IoT project builder. This library is designed to work with ESP8266 and ESP32 WiFi modules. With it you can send data to and receive data from Cayenne.

Adafruit HUZZAH ESP8266 image by lady ada is licensed under CC BY-SA 3.0

Releases

No releases published

Packages

No packages published

Contributors 3

jburhenn Jake Burhenn

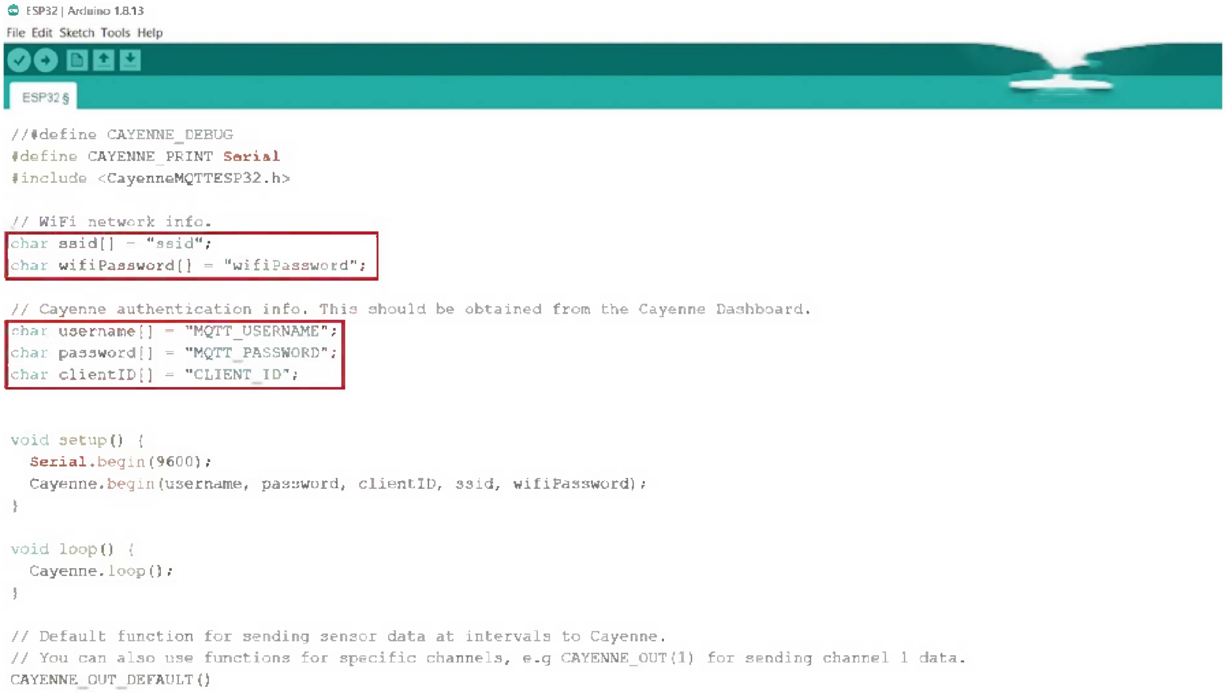
martinius96 Martin Chlebovec

bestpika 觀月唯

Languages

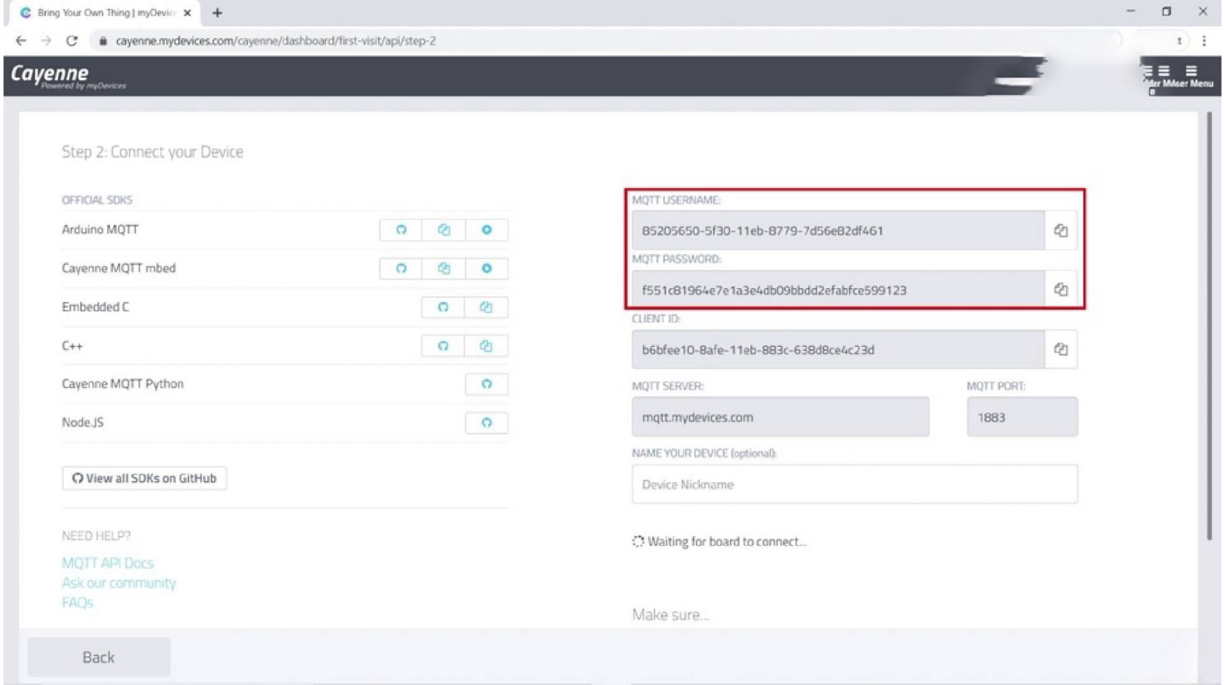
● C 74.5% ● C++ 25.5%

As we have already installed and used the MQTT library. Using one of the Cayenne MQTT libraries is one of the easiest ways to start using MQTT with Cayenne.



```
//#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP32.h>

// WiFi network info.
char ssid[] = "ssid";
char wifiPassword[] = "wifiPassword";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "CLIENT_ID";


void setup() {
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
}

void loop() {
  Cayenne.loop();
}

// Default function for sending sensor data at intervals to Cayenne.
// You can also use functions for specific channels, e.g CAYENNE_OUT(1) for sending channel 1 data.
CAYENNE_OUT_DEFAULT()
```

To understand the Cayenne MQTT ESP Library Let us take a small example. As done in the previous project We need to fill in all the credentials like Wi-Fi SSID and Wi-Fi password. And the username, password and client ID.



MQTT USERNAME:
85205650-5f30-11eb-8779-7d56e82df461

MQTT PASSWORD:
f551c81964e7e1a3e4db09bbdd2efabfce599123

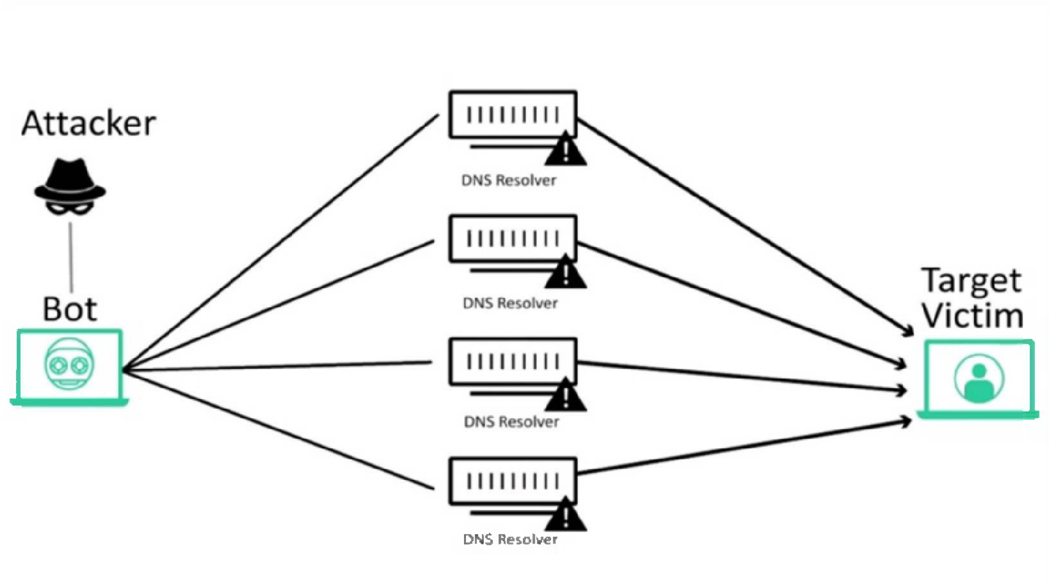
CLIENT ID:
b6bfee10-8afe-11eb-883c-638d8ce4c23d

MQTT SERVER:
mqtt.mydevices.com

MQTT PORT:
1883

NAME YOUR DEVICE (optional):
Device Nickname

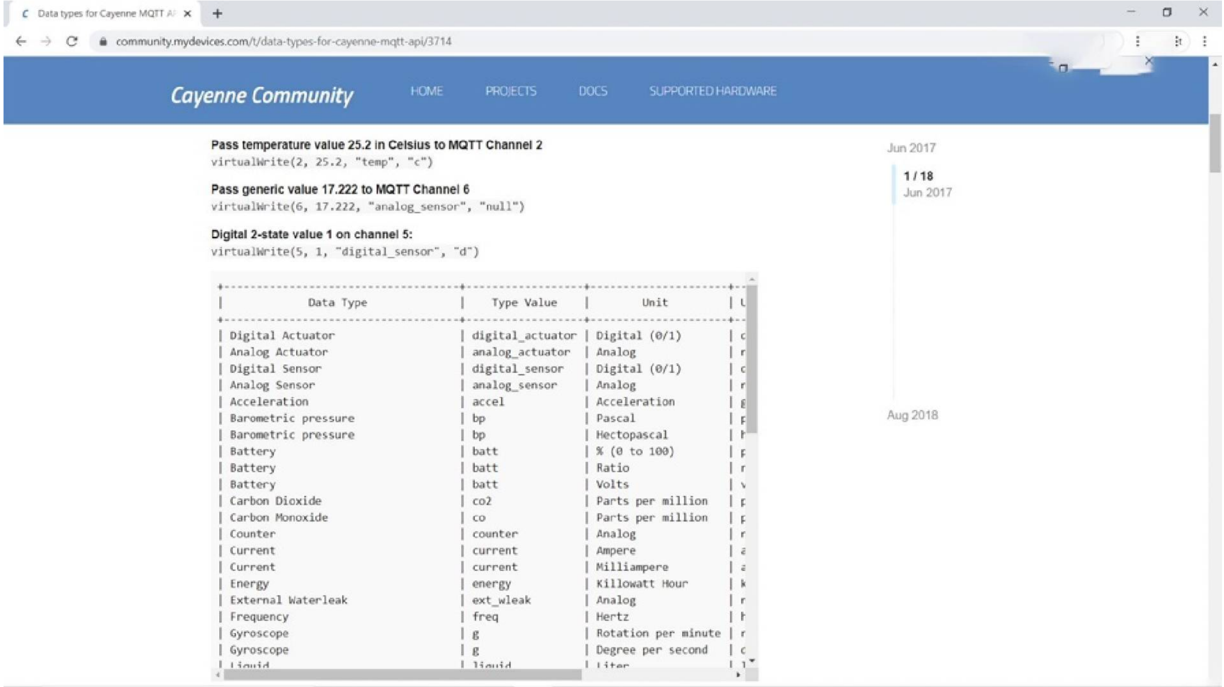
In Cayenne MQTT the user name and password for your account remains the same. It is the client ID that keeps changing. Suppose you have to connect three ESP32's to your Cayenne account. All of them will have the same username and password, but a different client ID.



Suppose you enter the same client ID. Only one of the devices will be connected and the other device will disconnect. This happens because from Cayenne's perspective, it seems like the devices credentials have been compromised and might be a part of DDoS attack. Now let's get back to the example. VWe have set the baud rate to 9600 and the Cayenne.begin function is used to connect to the Cayenne platform.

Now we include the Cayenne.loop function, which executes all the Cayenne functions repeatedly. Now the Cayenne out default function is where you write the data to the client. It is used to publish data in messages to the Cayenne broker and update the channel. The channel is where your actual data gets displayed. Here we will try to publish our messages to respective channels on Cayenne. We will be sending the time elapsed since the program started running to Channel 0. This is done using the millis() function, which returns the time in milliseconds. We have used the celsiusWrite() function to display temperature on the dashboard. We have manually passed a value of 22 degrees to the first channel. For the second channel we will be using the luxWrite() function, which denotes the luminosity. Lastly, we will be using the virtualWrite() function again to publish value to Channel three. Here we have also specified the type of sensor and the unit of measurement.

The general parameter format for the virtualWrite() function is this. If you need the full list of data type supported by the Cayenne, you can refer to this community page.



Data types for Cayenne MQTT A
community.mydevices.com/t/data-types-for-cayenne-mqtt-api/3714
Cayenne Community
HOME
PROJECTS
DOCS
SUPPORTED HARDWARE
Jun 2017
1 / 18
Jun 2017
Aug 2018

Pass temperature value 25.2 in Celsius to MQTT Channel 2

```
virtualWrite(2, 25.2, "temp", "c")
```

Pass generic value 17.222 to MQTT Channel 6

```
virtualWrite(6, 17.222, "analog_sensor", "null")
```

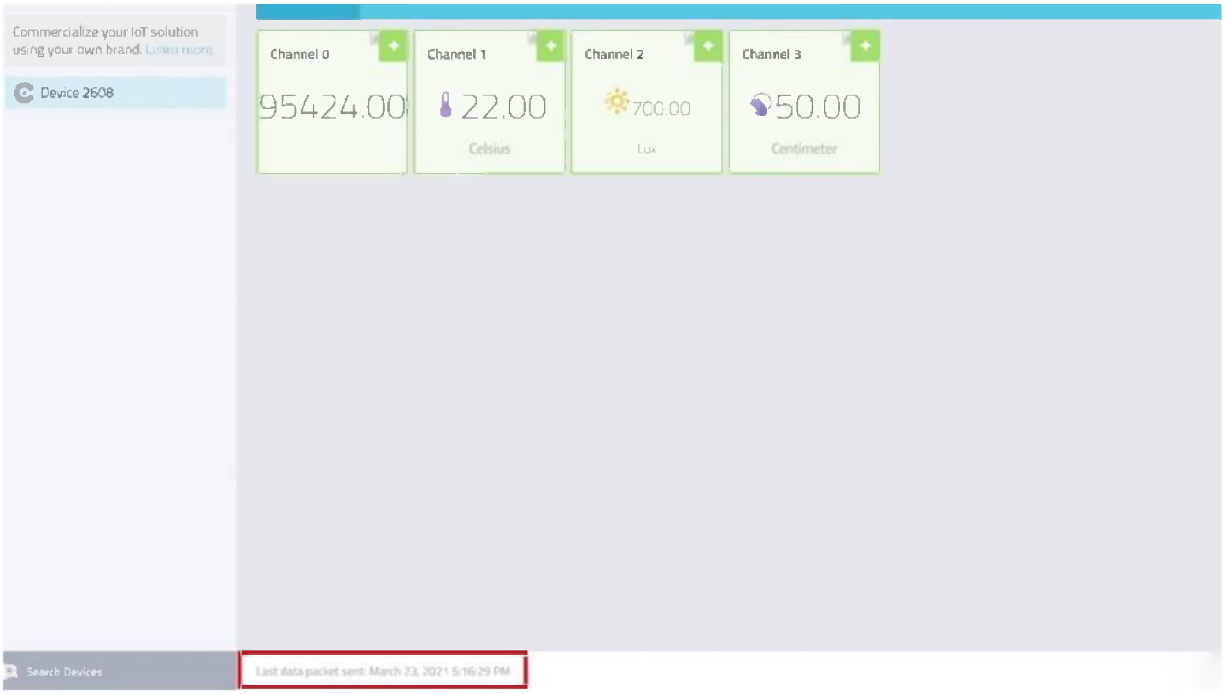
Digital 2-state value 1 on channel 5:

```
virtualWrite(5, 1, "digital_sensor", "d")
```

```
+-------------------------------------+------------------+-----------------------+--
|              Data Type              |    Type Value    |         Unit          | U
+-------------------------------------+------------------+-----------------------+--
| Digital Actuator                    | digital_actuator | Digital (0/1)         | c
| Analog Actuator                     | analog_actuator  | Analog                | r
| Digital Sensor                      | digital_sensor   | Digital (0/1)         | c
| Analog Sensor                       | analog_sensor    | Analog                | r
| Acceleration                        | accel            | Acceleration          | g
| Barometric pressure                 | bp               | Pascal                | p
| Barometric pressure                 | bp               | Hectopascal           | h
| Battery                             | batt             | % (0 to 100)          | p
| Battery                             | batt             | Ratio                 | r
| Battery                             | batt             | Volts                 | v
| Carbon Dioxide                      | co2              | Parts per million     | p
| Carbon Monoxide                     | co               | Parts per million     | p
| Counter                             | counter          | Analog                | r
| Current                             | current          | Ampere                | a
| Current                             | current          | Milliampere           | a
| Energy                              | energy           | Killowatt Hour        | k
| External Waterleak                  | ext_wleak        | Analog                | r
| Frequency                           | freq             | Hertz                 | h
| Gyroscope                           | g                | Rotation per minute   | r
| Gyroscope                           | g                | Degree per second     | c
| Liquid                              | liquid           | Liter                 | l
```

To read commands from the Cayenne dashboard, we need to use the Cayenne In default function.

Now upload the code. If you check your Cayenne dashboard, you will notice all channels with the published values. Similarly, you can publish data to n number of channels using Cayenne MQTT library. Each channel has a set of specifications that can be changed according to the need of the user.



Commercialize your IoT solution using your own brand. Learn more

Device 2608

Channel 0: 95424.00

Channel 1: 22.00 Celsius

Channel 2: 700.00 Lux

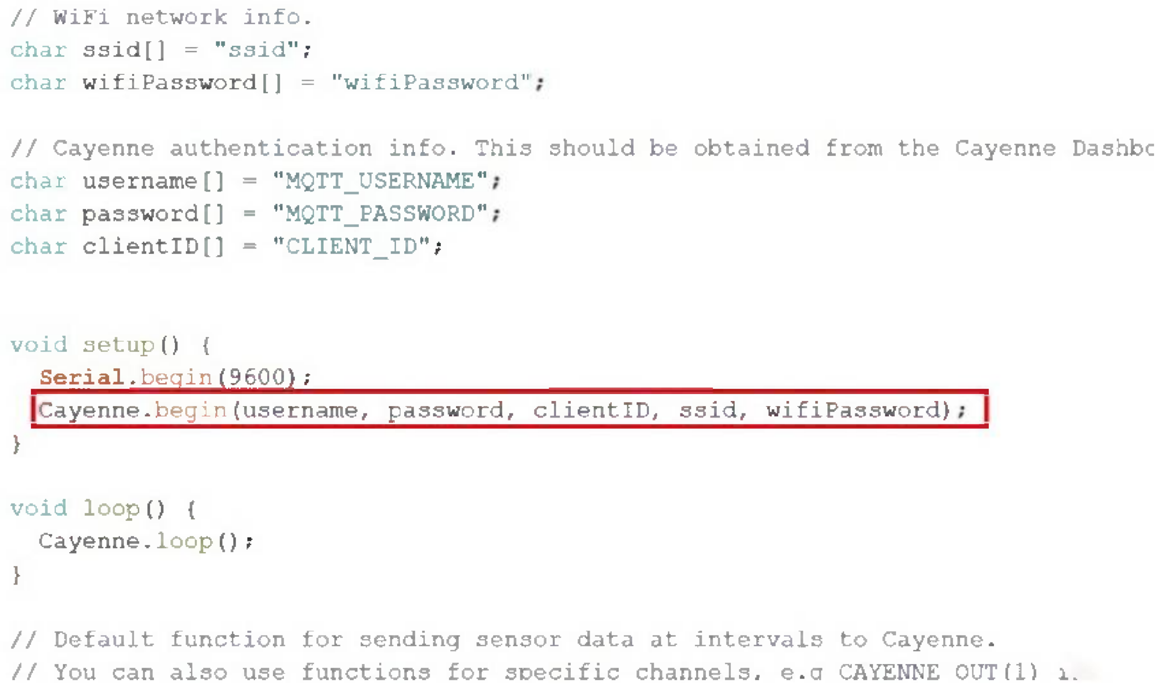
Channel 3: 50.00 Centimeter

Last data packet sent: March 23, 2021 5:16:29 PM

Also, if you look closely, you will see this message at the bottom. It shows the date and the time at which the last data packet was received. Now, you might notice one thing. We have sent data to Channel one, two and three manually, but for Channel 0, it receives data about every fifteen seconds. The reason is that the Cayenne Out default function is designed to run every fifteen seconds. Hence, the MQTT messages are received every fifteen seconds to Cayenne. Currently Cayenne MQTT messages are rate limited to sixty messages per minute. If messages are published above this rate, it may cause the messages to be dropped or the device to disconnect automatically. Due to the ease of use and reliability the Cayenne MQTT ESP library is a dependable method to start MQTT with Cayenne.

CAYENNE AND TROUBLESHOOTING

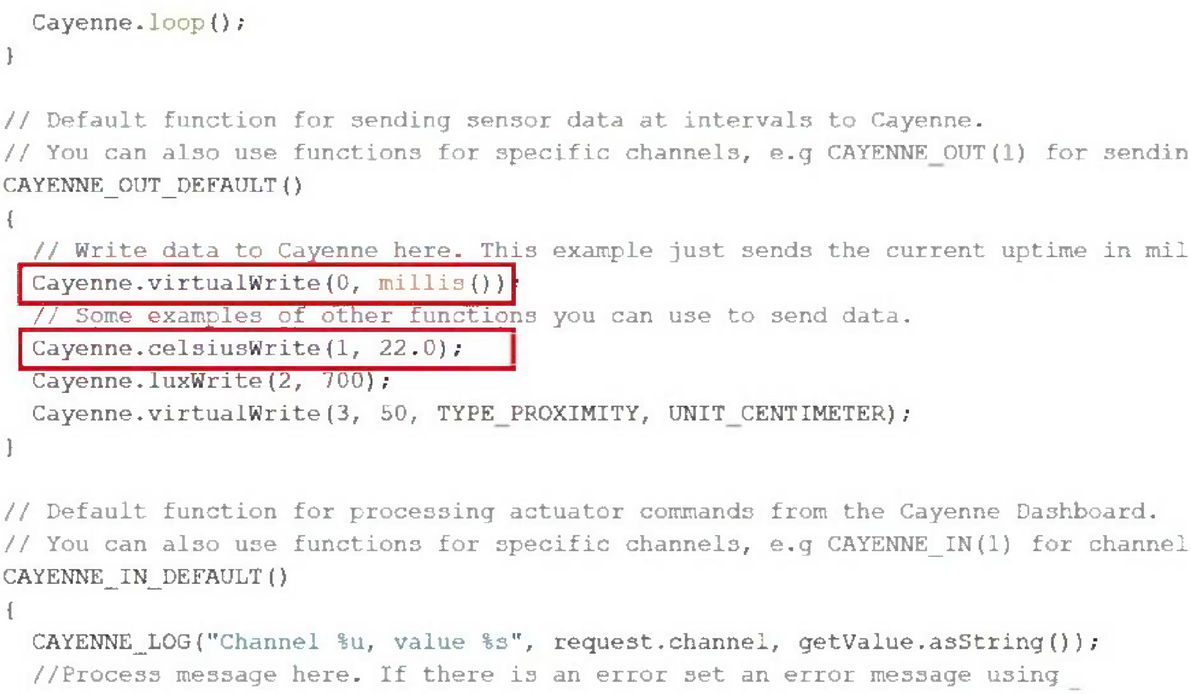
we will cover the following topics API method explanation, Set up for any IoT projects troubleshooting. you must have got a basic understanding of how to use the Cayenne MQTT library with the ESP32.



```
// WiFi network info.
char ssid[] = "ssid";
char wifiPassword[] = "wifiPassword";

// Cayenne authentication info. This should be obtained from the Cayenne Dashbc
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "CLIENT_ID";


void setup() {
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
}

void loop() {
  Cayenne.loop();
}

// Default function for sending sensor data at intervals to Cayenne.
// You can also use functions for specific channels, e.g CAYENNE OUT(1) i.
```

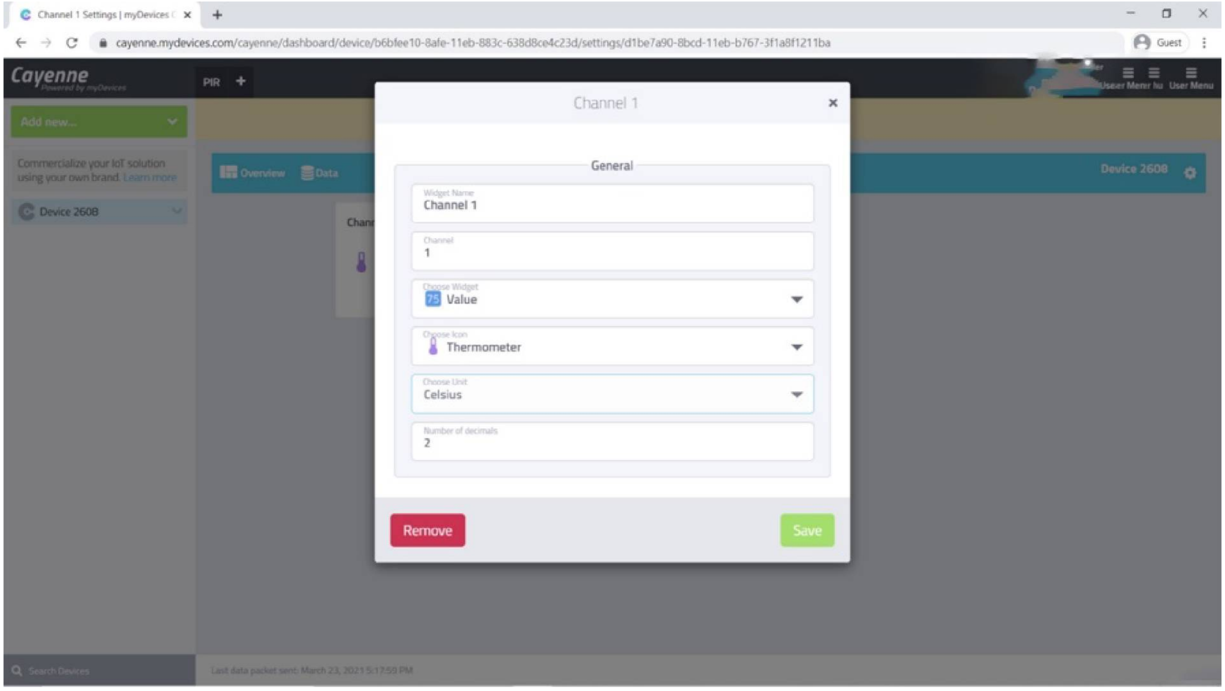
To use the Cayenne MQTT ESP library to its full potential we should know some key functions we have used the Cayenne.begin() function. This is a setup function. We specify our Wi-Fi SSID and Wi-Fi password along with the MQTT credentials from the Cayenne page.

This function is used to start communication with the Cayenne dashboard using MQTT. We had also used the Cayenne.loop() function. This function helps to send data to Cayenne continuously.



```
  Cayenne.loop();
}

// Default function for sending sensor data at intervals to Cayenne.
// You can also use functions for specific channels, e.g CAYENNE_OUT(1) for sendin
CAYENNE_OUT_DEFAULT()
{
  // Write data to Cayenne here. This example just sends the current uptime in mil
  Cayenne.virtualWrite(0, millis());
  // Some examples of other functions you can use to send data.
  Cayenne.celsiusWrite(1, 22.0);
  Cayenne.luxWrite(2, 700);
  Cayenne.virtualWrite(3, 50, TYPE_PROXIMITY, UNIT_CENTIMETER);
}

// Default function for processing actuator commands from the Cayenne Dashboard.
// You can also use functions for specific channels, e.g CAYENNE_IN(1) for channel
CAYENNE_IN_DEFAULT()
{
  CAYENNE_LOG("Channel %u, value %s", request.channel, getValue.asString());
  //Process message here. If there is an error set an error message using
```

Various functions such as Cayenne.virtualWrite() Cayenne.celsiusWrite() are used to send necessary data to the Cayenne dashboard. Here the main thing to notice is that the special functions like the CelsiusWrite() and luxWrite() directly publishes to a channel with the channel settings matching the data.



cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d/settings/d1be7a90-8bcd-11eb-b767-3f1a8f1211ba

Channel 1

General

Widget Name: Channel 1

Channel: 1

Choose Widget: Value

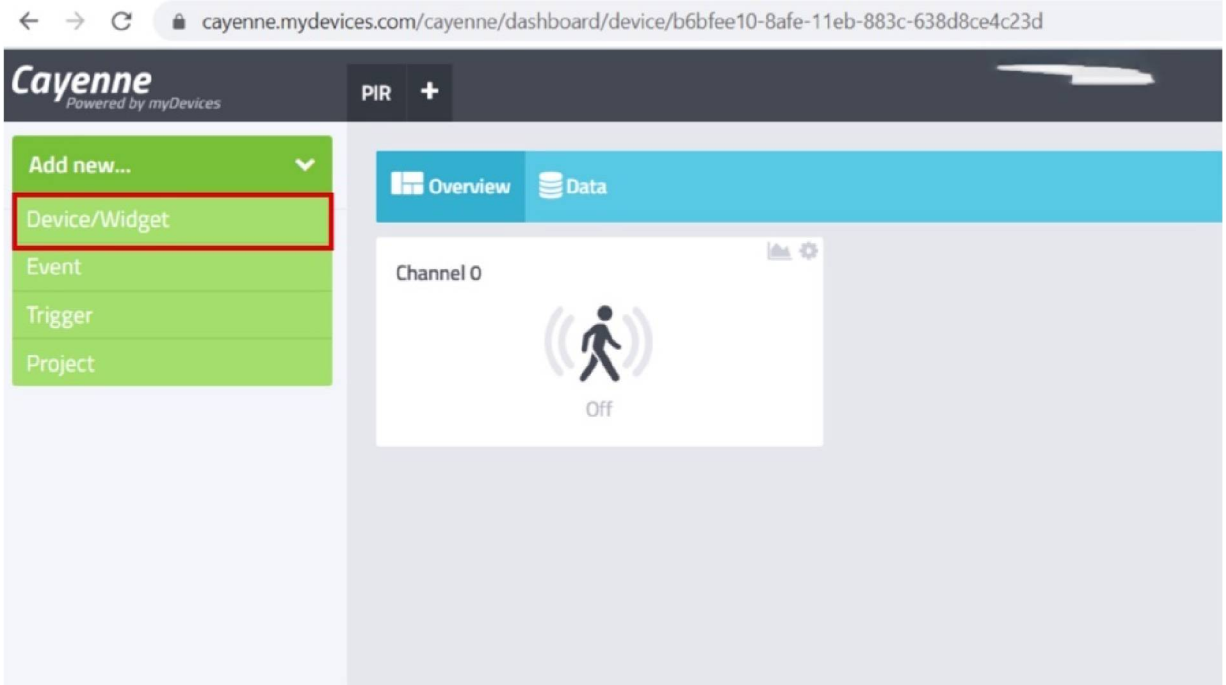
Choose Icon: Thermometer

Choose Unit: Celsius

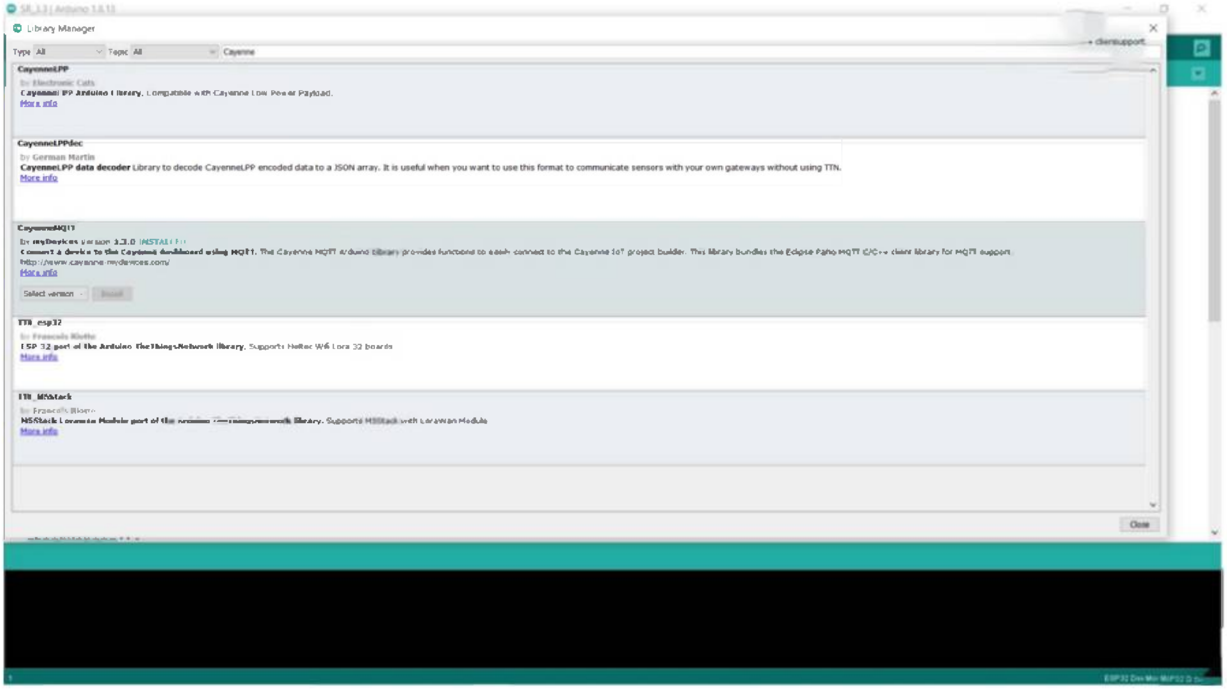
Number of decimals: 2

Channel 1 Settings | myDevices
Cayenne
Powered by myDevices
PIR
Add new...
Commercialize your IoT solution using your own brand. Learn more
Device 2608
Overview
Data
Device 2608
Guest
User Menu
Remove
Save
Search Devices
Last data packet sent: March 23, 2021 5:17:59 PM

For example, when you use the CelsiusWrite() function, the channel it is publishing to will already have a thermometer logo denoting temperature and unit set to Celsius. If you need such a specific widget icon and unit, the virtualWrite() function can be used and you can specify the type of sensor and unit. For this the virtualWrite() function expects a parameter like this. To help build virtual write statements, according to your use, you can check out the following community page, the link to which is given in the resources. In this section, we just discovered some mini projects to understand concepts and got a hands on with Cayenne and ESP32. But to develop an IoT project with Cayenne, you should keep some basic steps in mind. You should always be clear about whether you want to send data from the device to Cayenne or whether you want to control your device remotely.



After this, the first step is to configure your device on Cayenne. To do this, you need to move to the Cayenne dashboard and add a new and device widget option. Then you should select your device, which will guide you to a page with all the necessary credentials needed to connect the device. Always check beforehand if the device is compatible with Cayenne.



SR_1.3 | Arduino 1.8.13

Library Manager

Type All Topic All Cayenne

CayenneLPP
by Electronic Cats
CayenneLPP Arduino Library. Compatible with Cayenne Low Power Payload.
More info

CayenneLPPdec
by German Martin
CayenneLPP data decoder Library to decode CayenneLPP encoded data to a JSON array. It is useful when you want to use this format to communicate sensors with your own gateways without using TTN.
More info

CayenneMQTT
by **myDevices** version **1.3.0** INSTALLED
Connect a device to the Cayenne dashboard using MQTT. The Cayenne MQTT Arduino library provides functions to easily connect to the Cayenne IoT project builder. This library bundles the Eclipse Paho MQTT C/C++ client library for MQTT support.
http://www.cayenne-mydevices.com/
More info

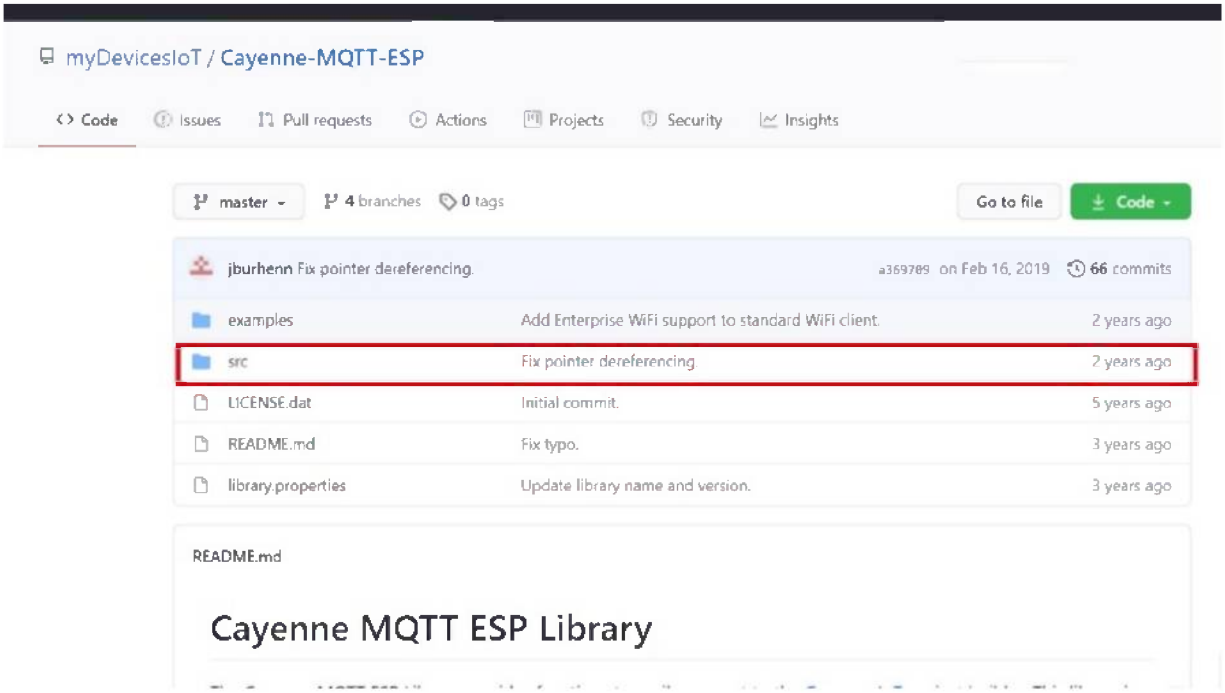
Select version Install

TTN_esp32
by Francois Riotte
ESP 32 port of the Arduino TheThingsNetwork library. Supports Heltec Wifi Lora 32 boards
More info

TTN_M5Stack
by Francois Riotte
M5Stack LoraWan Module port of the Arduino TheThingsNetwork library. Supports M5Stack with LoraWan Module
More info

Close

You need to check if you have installed the necessary libraries needed for Cayenne, if not install them. You are now ready to connect the device to Cayenne. You can now code, according to the library functions, to establish a connection with the Cayenne. Before sending the data You can either configure the channel according to the data or mention some specific code lines to do this for you.



myDevicesIoT / Cayenne-MQTT-ESP

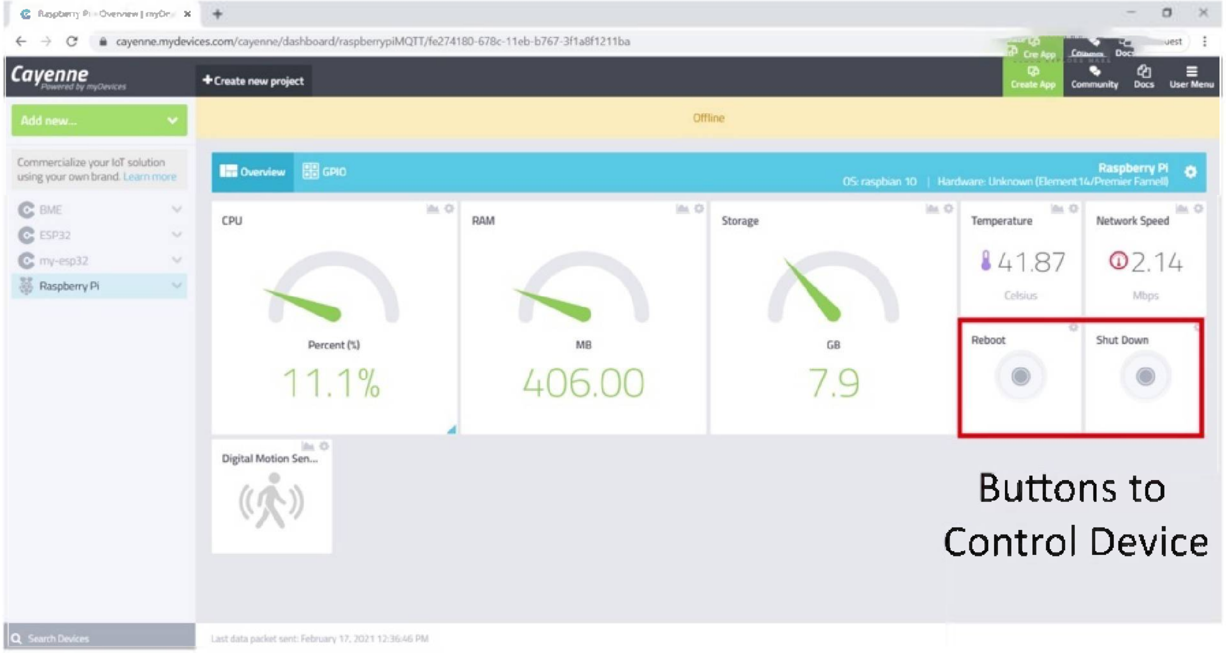
jburhenn Fix pointer dereferencing. a369789 on Feb 16, 2019 66 commits

examples	Add Enterprise WiFi support to standard WiFi client.	2 years ago
src	Fix pointer dereferencing.	2 years ago
LICENSE.dat	Initial commit.	5 years ago
README.md	Fix typo.	3 years ago
library.properties	Update library name and version.	3 years ago

README.md

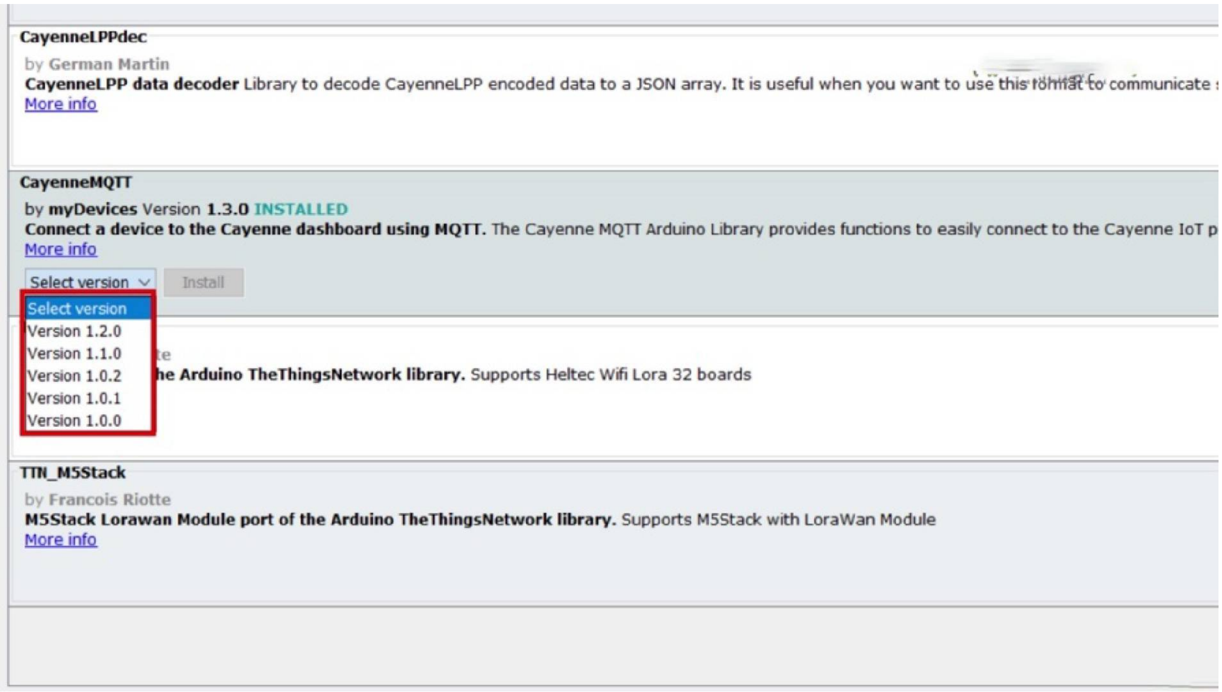
Cayenne MQTT ESP Library

Like the celsiusWrite() function, you can refer to the source code and modify it accordingly to send sensor data instead of sending data manually.


Raspberry Pi Overview | myDe...
cayenne.mydevices.com/cayenne/dashboard/raspberrypiMQTT/fe274180-678c-11eb-b767-3f1a8f1211ba
Cayenne
Powered by myDevices
Create new project
Create App
Community
Docs
User Menu
Add new...
Offline
Commercialize your IoT solution using your own brand. Learn more
BME
ESP32
my-esp32
Raspberry Pi
Overview
GPIO
Raspberry Pi
OS: raspbian 10 | Hardware: Unknown (Element14/Premier Farnell)
CPU
Percent (%)
11.1%
RAM
MB
406.00
Storage
GB
7.9
Temperature
41.87
Celsius
Network Speed
2.14
Mbps
Reboot
Shut Down
Digital Motion Sen...
Search Devices
Last data packet sent: February 17, 2021 12:36:46 PM

Buttons to Control Device

If you wish to control your remote device through Cayenne first, you will have to connect the actuator to your IoT board. Moreover, install the necessary libraries. You then need to create your Cayenne dashboard with widgets like sliders to send data to your device to control it remotely. This electronics domain pertaining to embedded systems that deals with the hardware and software is not easy as it looks. Each one of us may face different problems while developing a project. We will now cover some of the basic troubleshooting methods to use to solve your issues.



CayenneLPPdec
by **German Martin**
CayenneLPP data decoder Library to decode CayenneLPP encoded data to a JSON array. It is useful when you want to use this format to communicate
More info

CayenneMQTT
by **myDevices** Version **1.3.0** **INSTALLED**
Connect a device to the Cayenne dashboard using MQTT. The Cayenne MQTT Arduino Library provides functions to easily connect to the Cayenne IoT p
More info

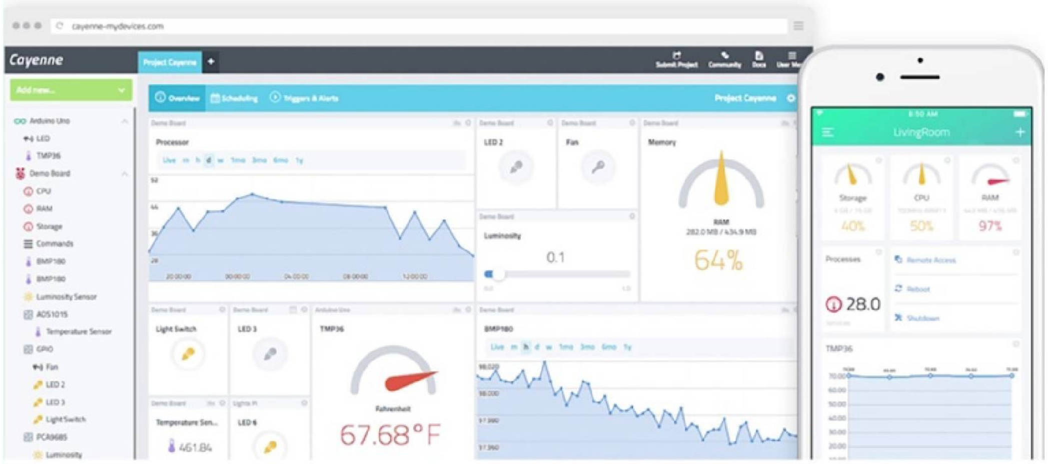
Select version | Install

Select version
Version 1.2.0
Version 1.1.0
Version 1.0.2
Version 1.0.1
Version 1.0.0

te
he Arduino TheThingsNetwork library. Supports Heltec Wifi Lora 32 boards

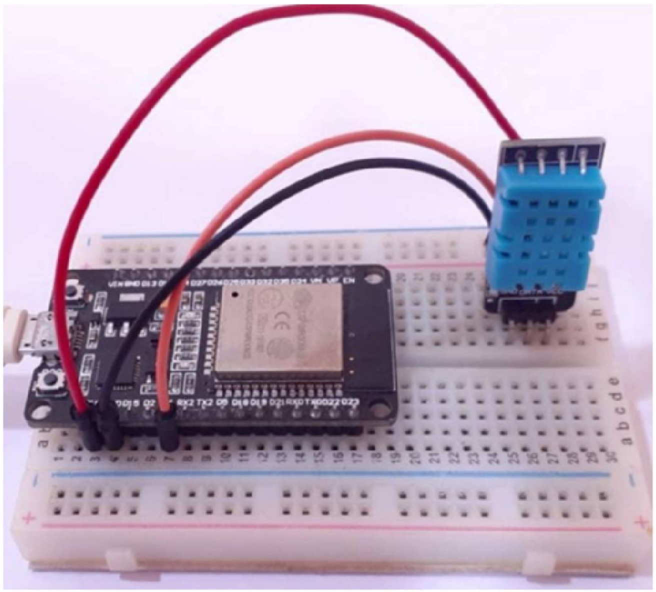
TTN_M5Stack
by **Francois Riotte**
M5Stack Lorawan Module port of the Arduino TheThingsNetwork library. Supports M5Stack with LoraWan Module
More info

For the software part, always install the latest version of the library as it contains bug fixes from the previous one.



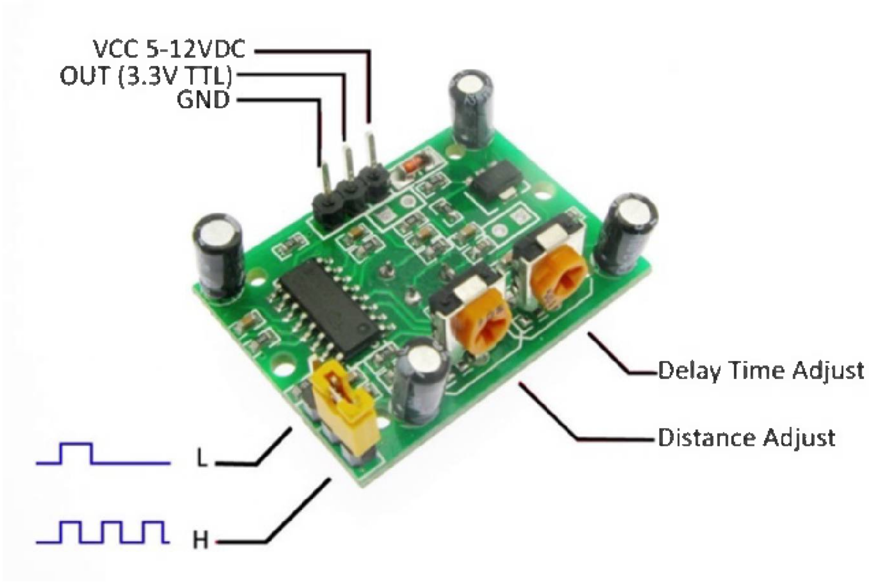
Cayenne is very user friendly and has a simple GUI. Unless you make a mistake in connecting your devices, Cayenne will not give you errors.

Always connect the device properly following the necessary steps. Make sure to copy and paste your Cayenne credentials properly.



For the hardware part. Firstly, check for your circuit connections if you have connected your device, sensor and actuator to the proper pin as mentioned in your code.

Use proper jumper wires and good breadboards to avoid loose connections, you can check if the jumper wire is not broken by using a multimeter. Always check your sensor or actuator beforehand. Some sensors may be broken and will not produce output even if they are receiving proper power.

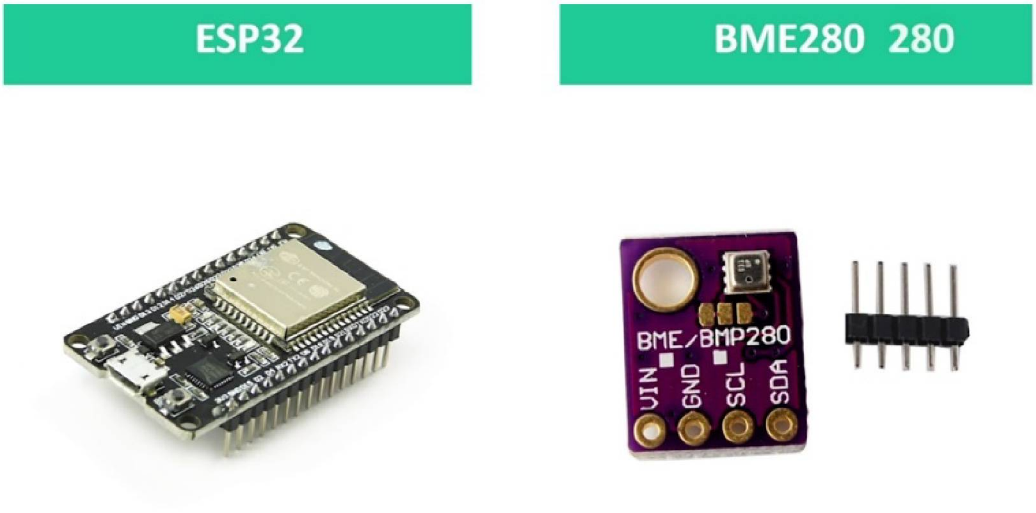


Taking your hardware beforehand reduces your time to solve an issue. If it occurs. Always provide only necessary and specified voltage to your peripherals.

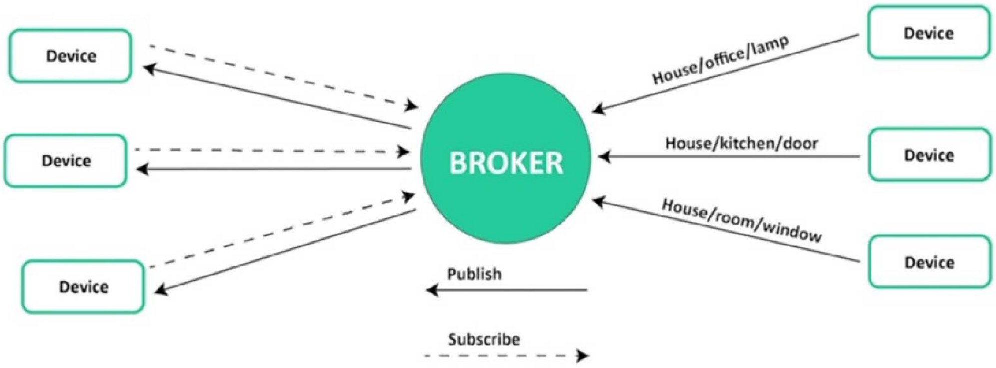
IOT WEATHER MONITORING SYSTEM THEORY

we will cover the following topics IoT architecture for the project. What is a BME280 Sensor? Working principle of BME280 sensor interfacing BME280 sensor with ESP32. Now that you have a basic understanding of working with the ESP32 and Cayenne, let's start with an end to end project of making a weather monitoring system.

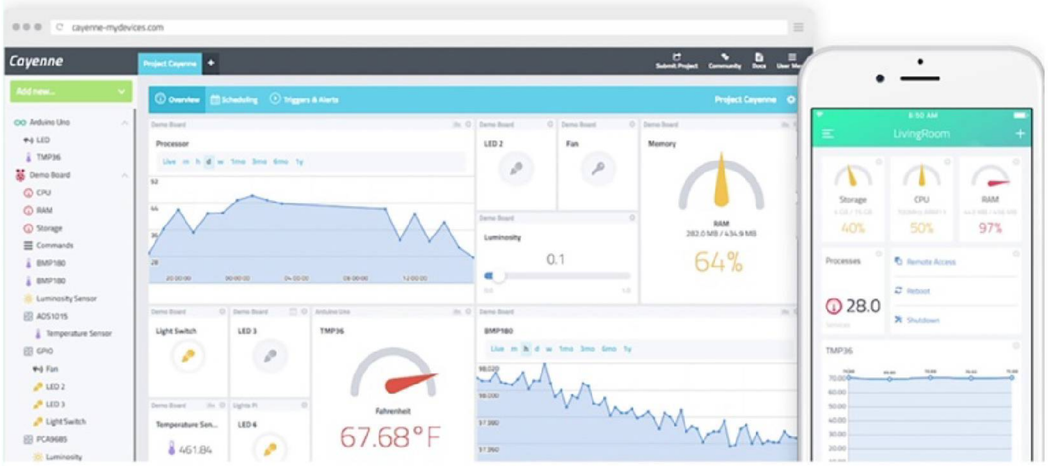
An IoT system, as you know, consists of an IoT device, gateway, communication network, cloud and IoT application. For our weather monitoring system. Can you guess what all components will be needed?



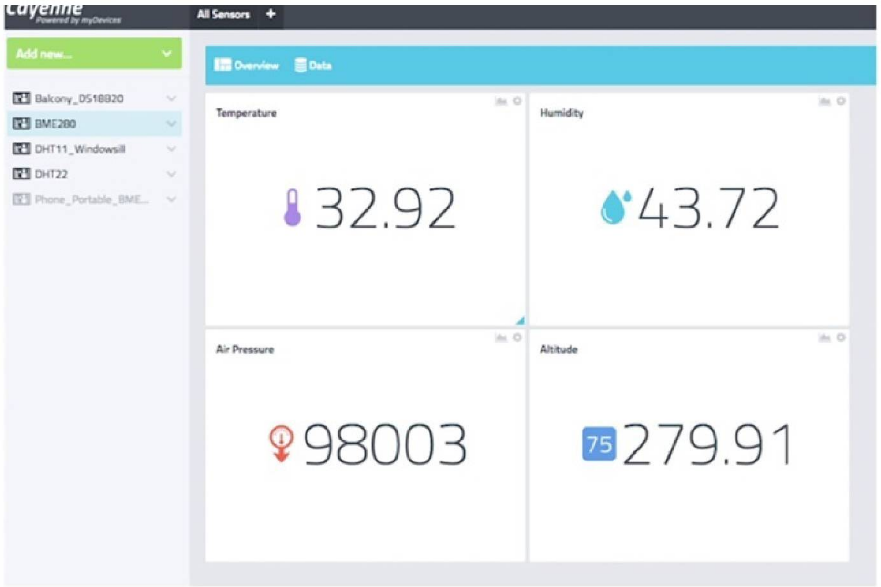
The ESP32 interfaced with the BME280 sensor will act as an IoT device.



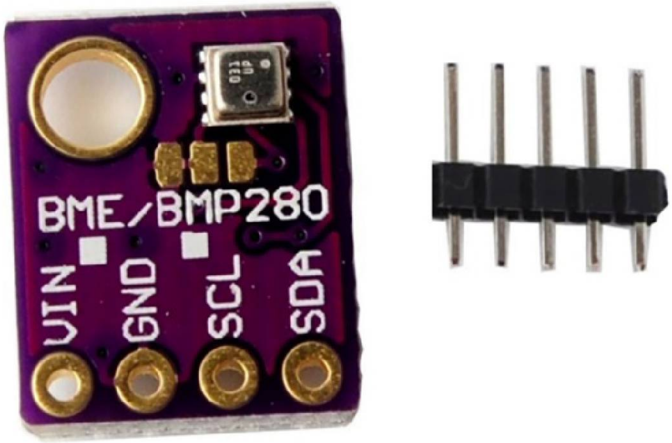
As we will not be using multiple communication protocols. We will not need a gateway. We will only use the MQTT protocol for data transfer.



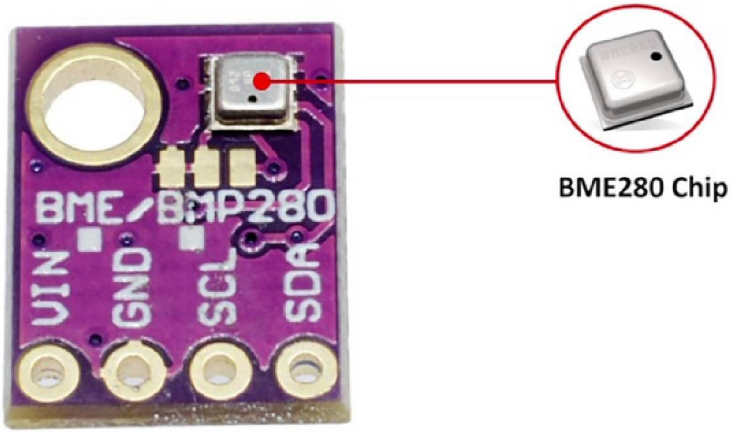
We will use the Cayenne cloud and the Cayenne dashboard as the IoT application.



In our weather monitoring system, the BME280 sensor will measure physical quantities and with the help of ESP32 we will send the measured data to Cayenne.

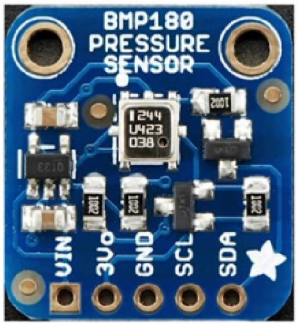


You now must be wondering what is BME280.



The BME280 is an environmental sensor that can measure temperature, barometric pressure and humidity. At the heart of the BME 280 module, you will find a BME280 sensor chip manufactured by BOSCH.

BME180



BME085 085

It is a successor to sensor like BME180 and BME085. This sensor is one of the best low cost sensing solutions to precisely measure humidity from 0 to 100, with plus or minus 3% accuracy. Barometric pressure from 300 hecto Pascal to 1100 hecto Pascal with plus or minus 1% absolute accuracy and temperature from minus 40 to 85 degrees Celsius with plus or minus 1 degree Celsius accuracy. The pressure measurements are very precise, so we can also use the BME280 to measure altitude with plus or minus one meter accuracy.

So how does the BME 280 work? At any given point in time, the sensor can measure the air's moisture, the air temperature and the weight of the air. The weight of the air above the sensor is the pressure of the air at that moment. Similarly, the ratio of moisture in the air to the highest amount of moisture at a particular temperature is calculated as the humidity by the sensor.

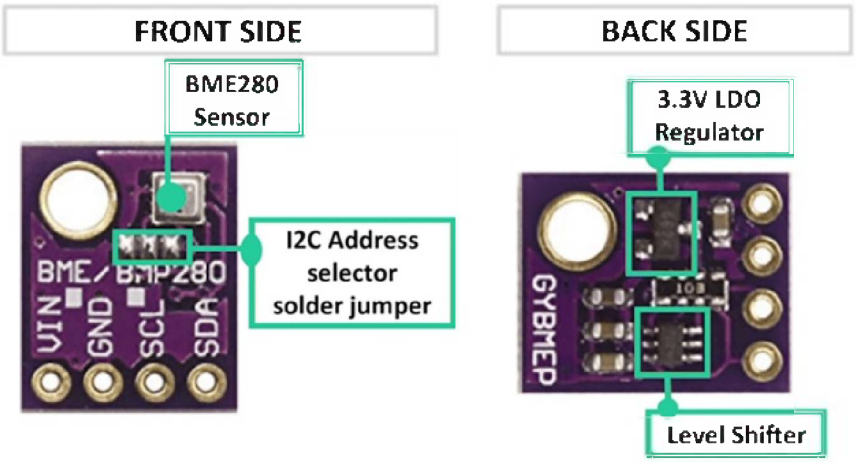
$$\text{altitude} = 44330 \times \left(1 - \left(\frac{P}{P_0}\right)^{\frac{1}{5.255}}\right)$$

altitude = altitude in meters

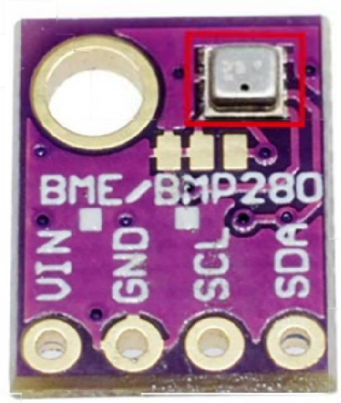
P **= measured pressure from the sensor in hectopascals**

P_0 **= reference pressure at sea level in hectopascals**

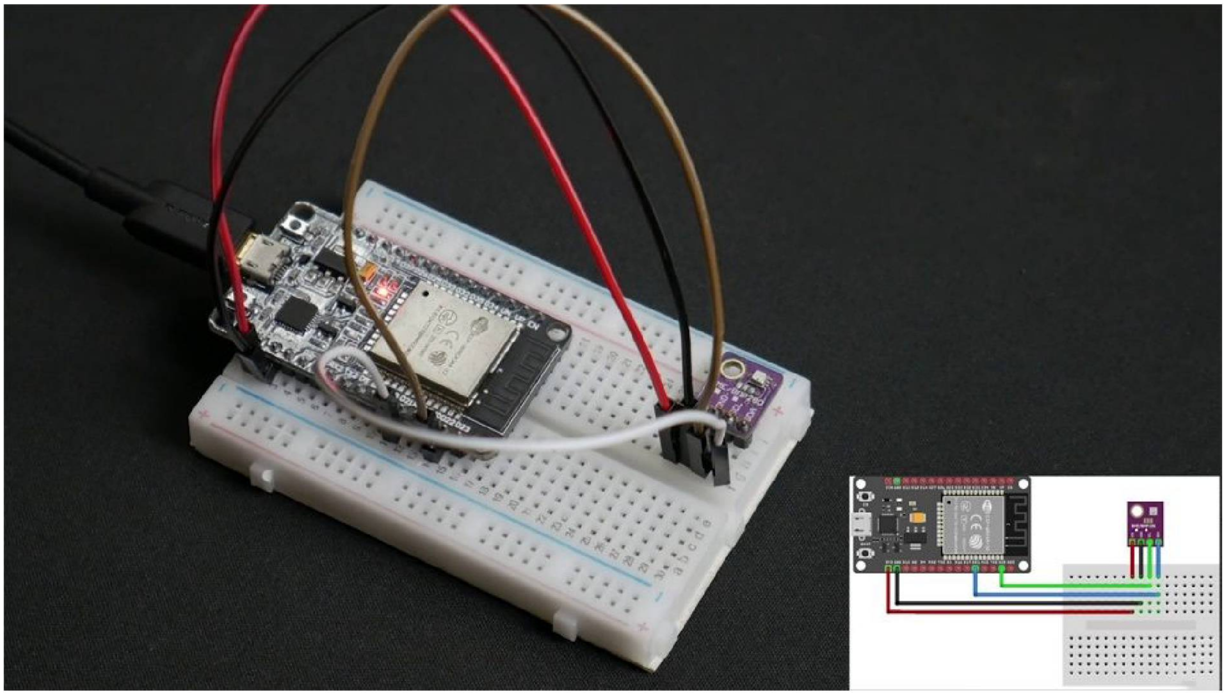
As pressure changes with altitude and the BME280 sensor is very accurate with its measurements, we can use the BME280 to also measure altitude. Altitude can be calculated by using this formula where P is measured pressure from the sensor and P0 is the reference pressure at sea level.



The BME280 sensor module needs up to 3.3V to power up and has a built in LM 6206 3.3V voltage regulator. These two features make it possible for the BME280 to interface with any microcontroller of your choice.



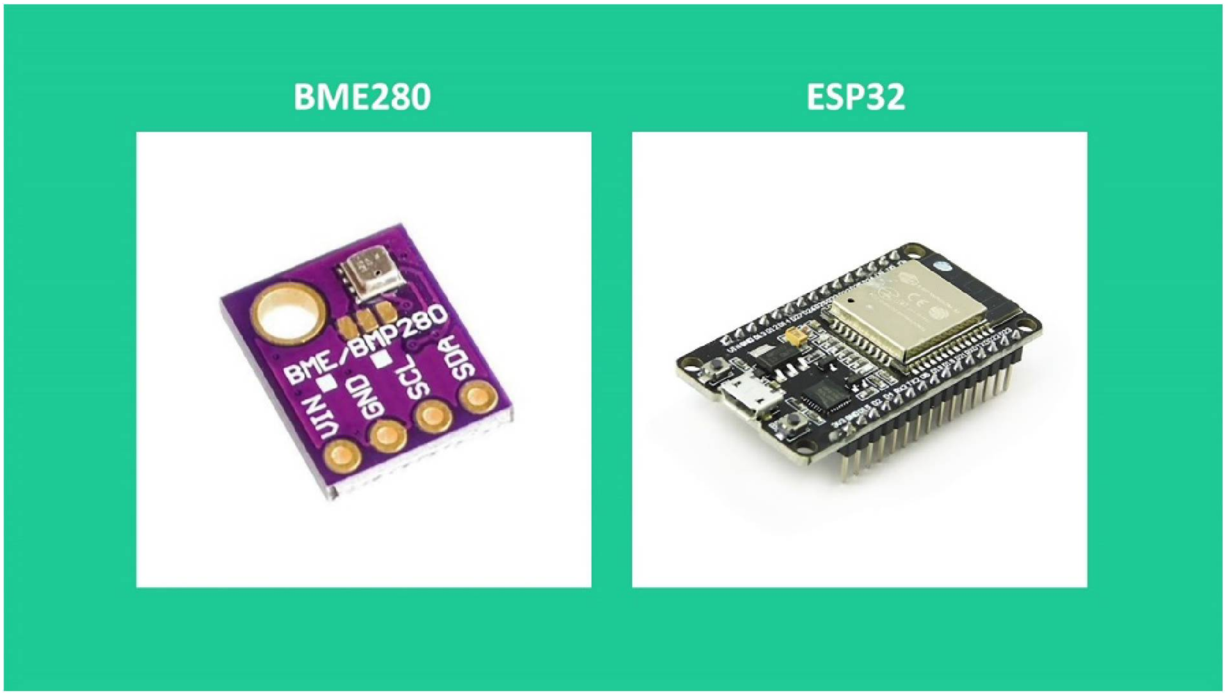
Now let's interface the sensor to the ESP32. First, let's take a look at the pinout diagram of the BME280. The sensor module only consists of 4 pins, the VCC, the ground, the Serial clock and the serial data pin.



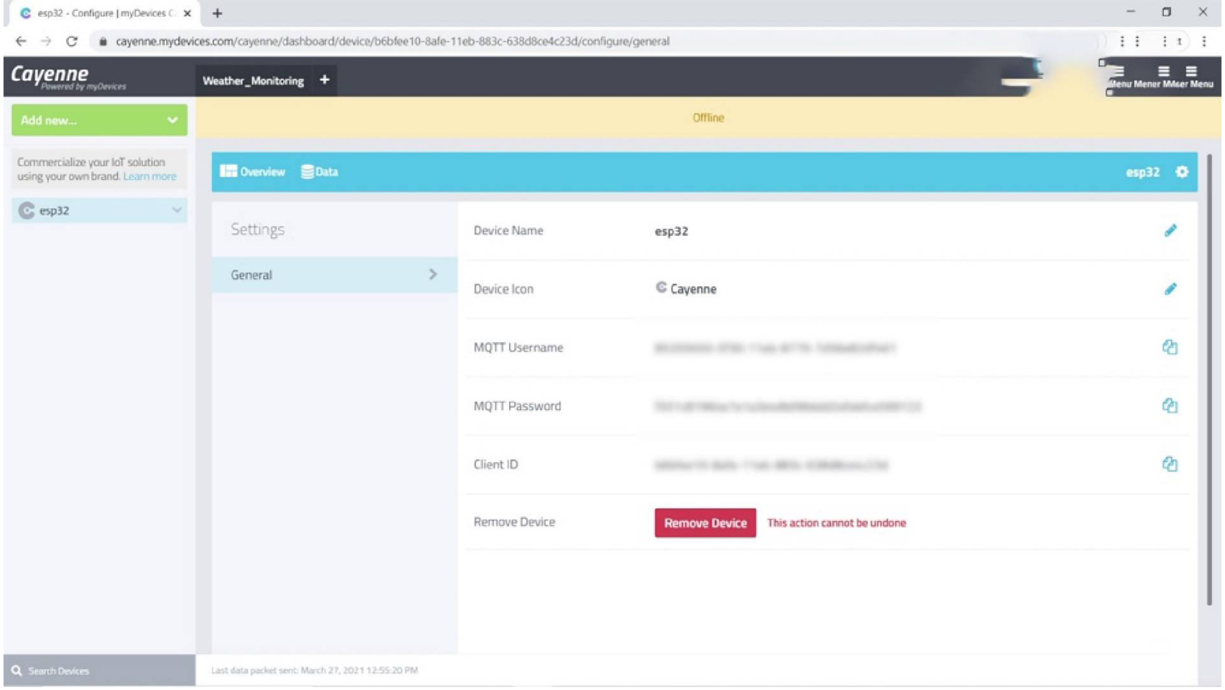
Now connect the VCC of the sensor to the 3.3V power supply pin of the ESP32. Connect the ground of the sensor module to the ground of the ESP32. Connect the Serial clock pin of the module to GPIO22 of the ESP32. GPIO 22 also acts like an SCL pin, or serial clock pin for the I2C data transfer. Similarly, connect the serial data pin of the sensor module to GPIO 21 of the ESP32. Always recheck your connections before proceeding to avoid any damage to the sensor as well as the ESP32.

CONFIGURATION AND IMPLEMENTATION

we will cover the following topics Cayenne device management and dashboard configuration, installing sensor libraries, code configuration and explanation, project implementation, visualization.



Now that you have interfaced the BME280 sensor with the ESP32. Let's check out how to send sensors data using the ESP32 to the Cayenne dashboard. You will need to login to your Cayenne account to get information about your credentials.



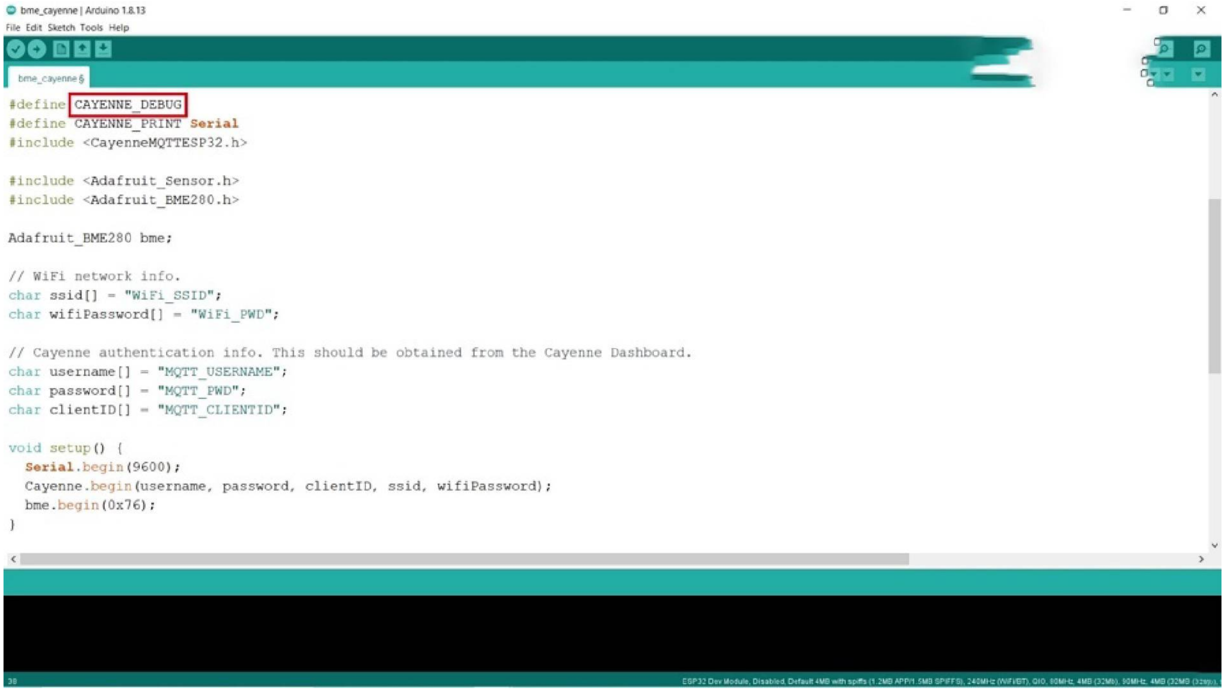
esp32 - Configure | myDevices C
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d/configure/general
Cayenne
Powered by myDevices
Weather_Monitoring
Add new...
Offline
Commercialize your IoT solution using your own brand. Learn more
esp32
Overview
Data
esp32
Settings
General
Search Devices

Device Name	esp32
Device Icon	Cayenne
MQTT Username	[redacted]
MQTT Password	[redacted]
Client ID	[redacted]
Remove Device	Remove Device This action cannot be undone

Last data packet sent: March 27, 2021 12:55:20 PM

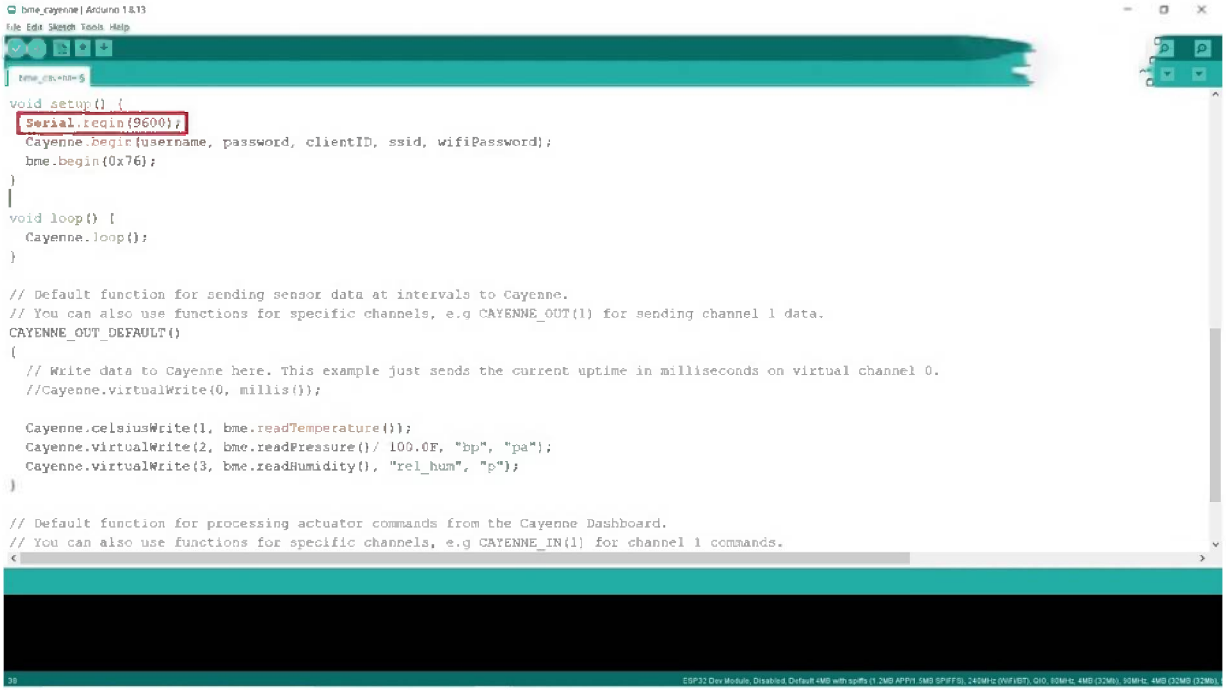
As we have already connected the ESP32 to the Cayenne dashboard, you can reuse its credentials. All you need to add a new device, as we did in the previous section.

Let's first install the necessary sensor libraries, open your Arduino IDE, go to tools, manage libraries and now search for BME280. You need to install the Adafruit BME280 library. When you click on install a dialog box, asking you to install missing dependencies will appear, click on install all. This will install the Adafruit BME280 library and also the Adafruit Unified Sensor Library. The Adafruit Unified Sensor Library is an abstraction layer library used by many sensor libraries.


bme_cayenne | Arduino 1.8.13
File Edit Sketch Tools Help
bme_cayenne §
38
ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 90MHz, 4MB (32MB (32Mb))

```
#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP32.h>

#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme;

// WiFi network info.
char ssid[] = "WiFi_SSID";
char wifiPassword[] = "WiFi_PWD";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PWD";
char clientID[] = "MQTT_CLIENTID";

void setup() {
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
  bme.begin(0x76);
}
```

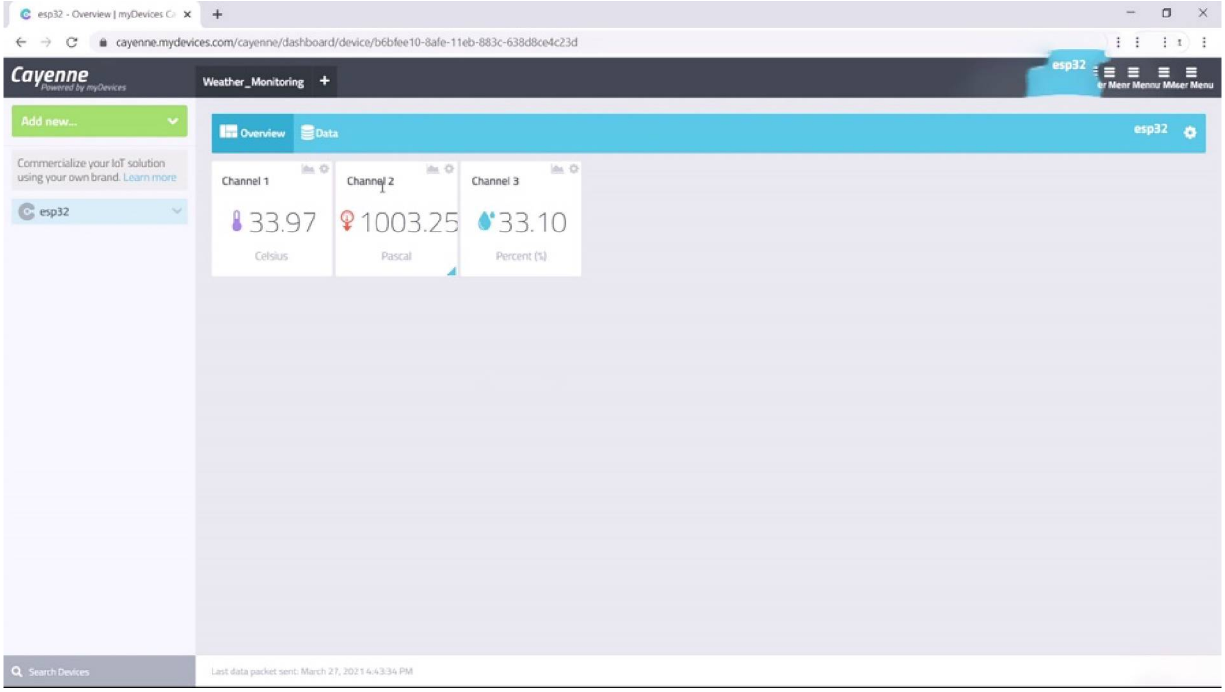
We are now ready to code. Let's check out the code to send the sensor data to Cayenne. This is used to print out the debug messages like the connection status, topic number and channel number and actually data itself. Now this here is used to print out the general Wi-Fi credentials and the port number to which the client is connected. You can comment both of these if you do not wish to print debug messages and general Wi-Fi status here we import the Cayenne MQTT Library for the ESP32, it contains functions that will enable you to send data to Cayenne as well as receive data. Here we have imported the two libraries we just installed. The Adafruit BME280 library contains functions to read the sensor data. This here is used to create an instance of the sensor. Now here you need to enter your Wi-Fi credentials. Here you need to enter your Cayenne authentication information like the username, password and client ID.



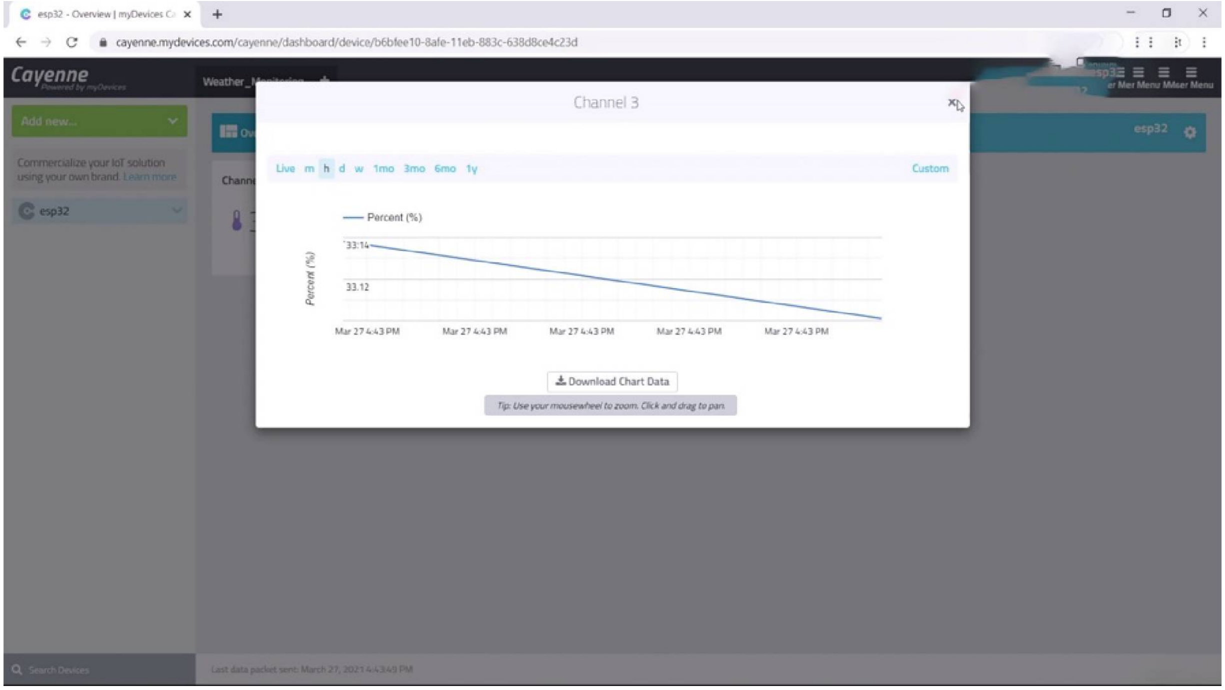
```
void setup() {
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
  bme.begin(0x76);
}

void loop() {
  Cayenne.loop();
}

// Default function for sending sensor data at intervals to Cayenne.
// You can also use functions for specific channels, e.g CAYENNE_OUT(1) for sending channel 1 data.
CAYENNE_OUT_DEFAULT()
{
  // Write data to Cayenne here. This example just sends the current uptime in milliseconds on virtual channel 0.
  //Cayenne.virtualWrite(0, millis());

  Cayenne.celsiusWrite(1, bme.readTemperature());
  Cayenne.virtualWrite(2, bme.readPressure()/ 100.0F, "bp", "pa");
  Cayenne.virtualWrite(3, bme.readHumidity(), "rel_hum", "p");
}

// Default function for processing actuator commands from the Cayenne Dashboard.
// You can also use functions for specific channels, e.g CAYENNE_IN(1) for channel 1 commands.
```

38
ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 90MHz, 4MB (32MB (32Mb),

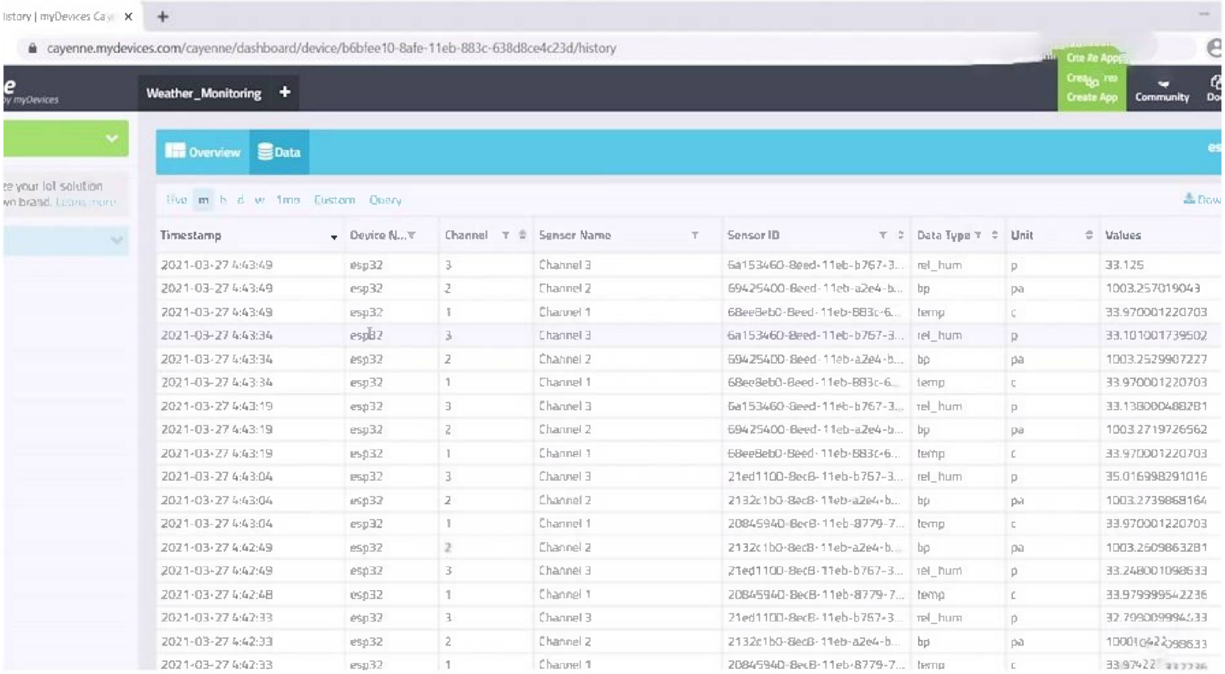
Now here we initialize the serial baud rate as 9600. This is, as you know, is needed to connect with Cayenne. This initializes the I2C interface with the given I2C address and checks if the chip ID is Correct. It then resets the chip using a soft reset and waits for the sensor for calibration. Here we are using the bme.read() functions to read the sensor data for particular quantities. The Cayennewrite() functions are then used to send the data to the Cayenne dashboard. Now save your code and click on the upload button.


esp32 - Overview | myDevices C
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d
Cayenne
Powered by myDevices
Weather_Monitoring
Add new...
Commercialize your IoT solution using your own brand. Learn more
esp32
Overview
Data
esp32
Channel 1
33.97
Celsius
Channel 2
1003.25
Pascal
Channel 3
33.10
Percent (%)
Search Devices
Last data packet sent: March 27, 2021 4:43:34 PM

After the code is done. Uploading Open your Cayenne Dashboard. You will have 3 channels displaying the published data. Click here to add them to your dashboard. You can now monitor your data.


esp32 - Overview | myDevices C
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d
Cayenne
Channel 3
Live
m
h
d
w
1mo
3mo
6mo
1y
Custom
Percent (%)
33.14
33.12
Percent (%)
Mar 27 4:43 PM
Mar 27 4:43 PM
Mar 27 4:43 PM
Mar 27 4:43 PM
Mar 27 4:43 PM
Download Chart Data
Tip: Use your mousewheel to zoom. Click and drag to pan.
Last data packet sent: March 27, 2021 4:43:49 PM

You can also check the charts for each channel and also a spreadsheet of all your published data.

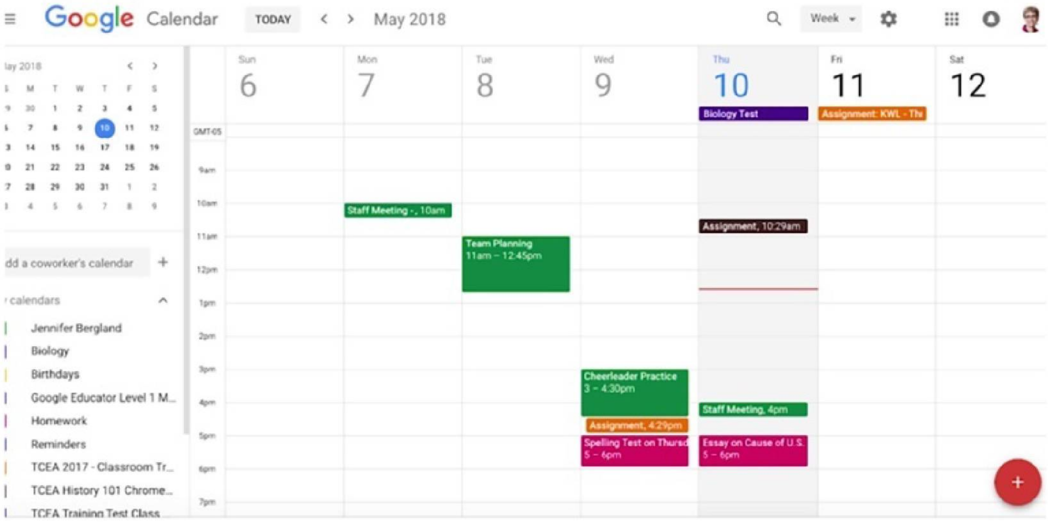

istory | myDevices Ca
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d/history
by myDevices
Weather_Monitoring
Create App
Community
Overview
Data
live m h d w 1mo Custom Query

Timestamp	Device N...	Channel	Sensor Name	Sensor ID	Data Type	Unit	Values
2021-03-27 4:43:49	esp32	3	Channel 3	6a153460-8eed-11eb-b767-3...	rel_hum	p	33.125
2021-03-27 4:43:49	esp32	2	Channel 2	69425400-8eed-11eb-a2e4-b...	bp	pa	1003.257019043
2021-03-27 4:43:49	esp32	1	Channel 1	68ee8eb0-8eed-11eb-883c-6...	temp	c	33.970001220703
2021-03-27 4:43:34	esp32	3	Channel 3	6a153460-8eed-11eb-b767-3...	rel_hum	p	33.101001739502
2021-03-27 4:43:34	esp32	2	Channel 2	69425400-8eed-11eb-a2e4-b...	bp	pa	1003.2529907227
2021-03-27 4:43:34	esp32	1	Channel 1	68ee8eb0-8eed-11eb-883c-6...	temp	c	33.970001220703
2021-03-27 4:43:19	esp32	3	Channel 3	6a153460-8eed-11eb-b767-3...	rel_hum	p	33.138000488281
2021-03-27 4:43:19	esp32	2	Channel 2	69425400-8eed-11eb-a2e4-b...	bp	pa	1003.2719726562
2021-03-27 4:43:19	esp32	1	Channel 1	68ee8eb0-8eed-11eb-883c-6...	temp	c	33.970001220703
2021-03-27 4:43:04	esp32	3	Channel 3	21ed1100-8ec8-11eb-b767-3...	rel_hum	p	35.016998291016
2021-03-27 4:43:04	esp32	2	Channel 2	2132c1b0-8ec8-11eb-a2e4-b...	bp	pa	1003.2739868164
2021-03-27 4:43:04	esp32	1	Channel 1	20845940-8ec8-11eb-8779-7...	temp	c	33.970001220703
2021-03-27 4:42:49	esp32	2	Channel 2	2132c1b0-8ec8-11eb-a2e4-b...	bp	pa	1003.2609863281
2021-03-27 4:42:49	esp32	3	Channel 3	21ed1100-8ec8-11eb-b767-3...	rel_hum	p	33.248001098633
2021-03-27 4:42:48	esp32	1	Channel 1	20845940-8ec8-11eb-8779-7...	temp	c	33.979999542236
2021-03-27 4:42:33	esp32	3	Channel 3	21ed1100-8ec8-11eb-b767-3...	rel_hum	p	32.799009994633
2021-03-27 4:42:33	esp32	2	Channel 2	2132c1b0-8ec8-11eb-a2e4-b...	bp	pa	1000[illegible]98633
2021-03-27 4:42:33	esp32	1	Channel 1	20845940-8ec8-11eb-8779-7...	temp	c	33.97422[illegible]

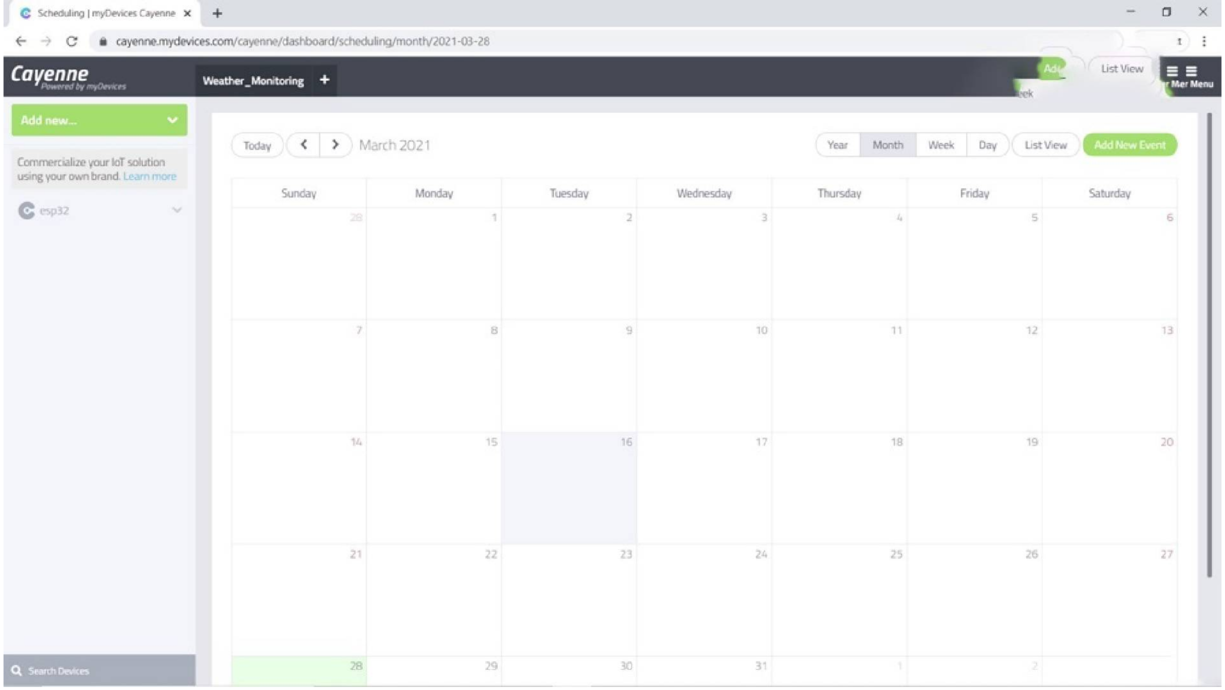
This can be downloaded and used further.

EVENTS AND TRIGGERS

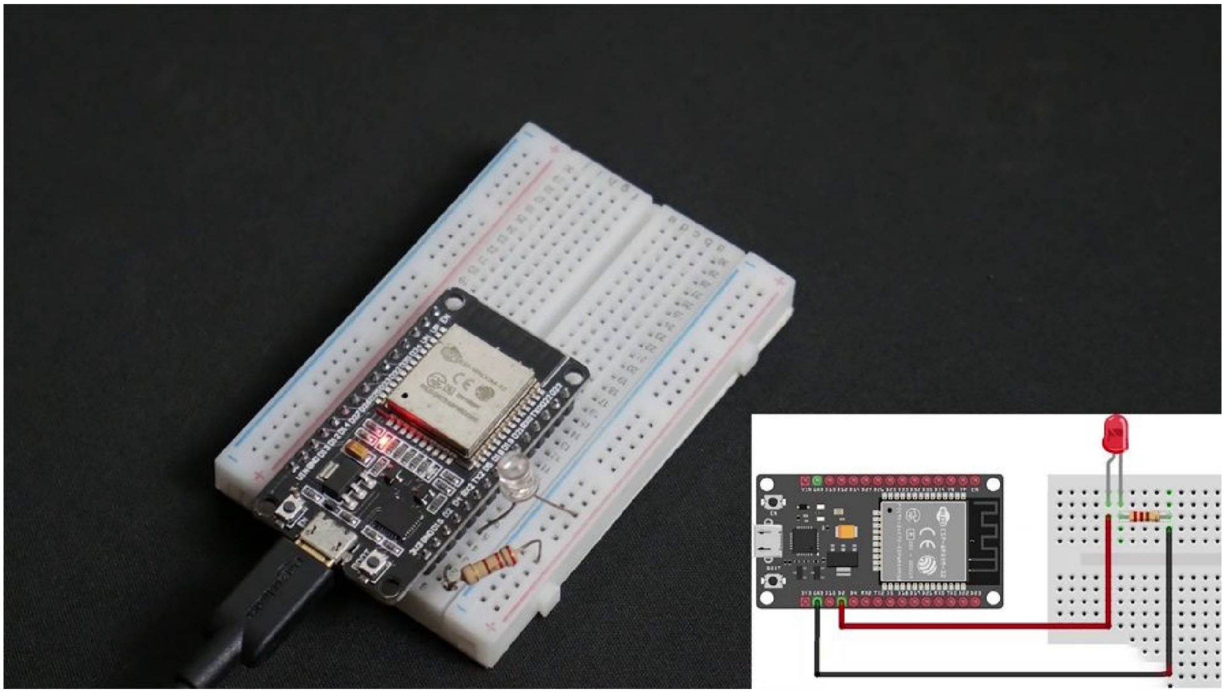
we will cover the following topics introduction to Cayenne event scheduling, implementation of event scheduling, introduction to trigger notifications, implementation of trigger notification based on sensor parameters. In the last project we sent sensor data to Cayenne. However, Cayenne has some more automation features.



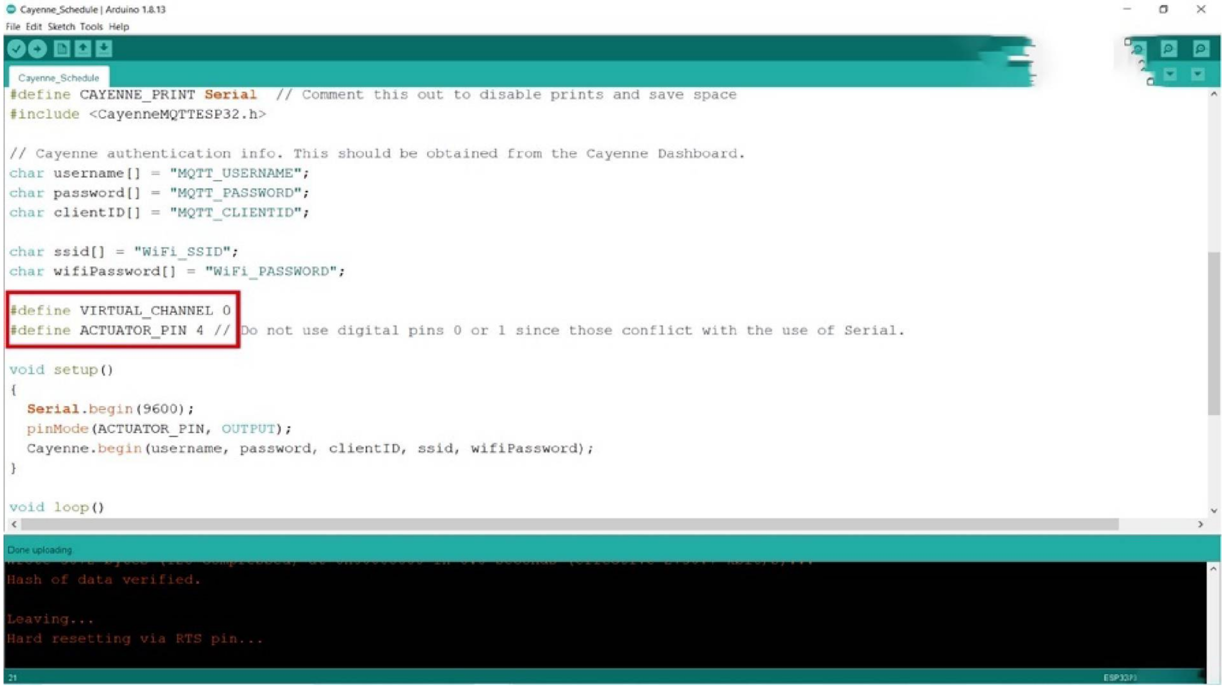
Let us get to know about this amazing event and trigger feature of Cayenne. So how do you complete your work daily? Don't you use applications like Google Calendar or some planner and schedule the tasks to be completed. Similarly, Cayenne has this amazing feature of event scheduling. Cayenne allows you to create scheduled events for the connected devices, sensors and actuators.



A scheduled event can have one or more actions added to it. It is similar to you creating events or reminders on your smartphone. Let's do a small project to understand the event scheduling feature of Cayenne.



We will turn on an LED according to schedule. Let's make the circuit connection, connect the positive leg of the LED to GPIO 4 and the negative leg of the LED to the ground of the ESP32, with the 220 ohm resistor in between.

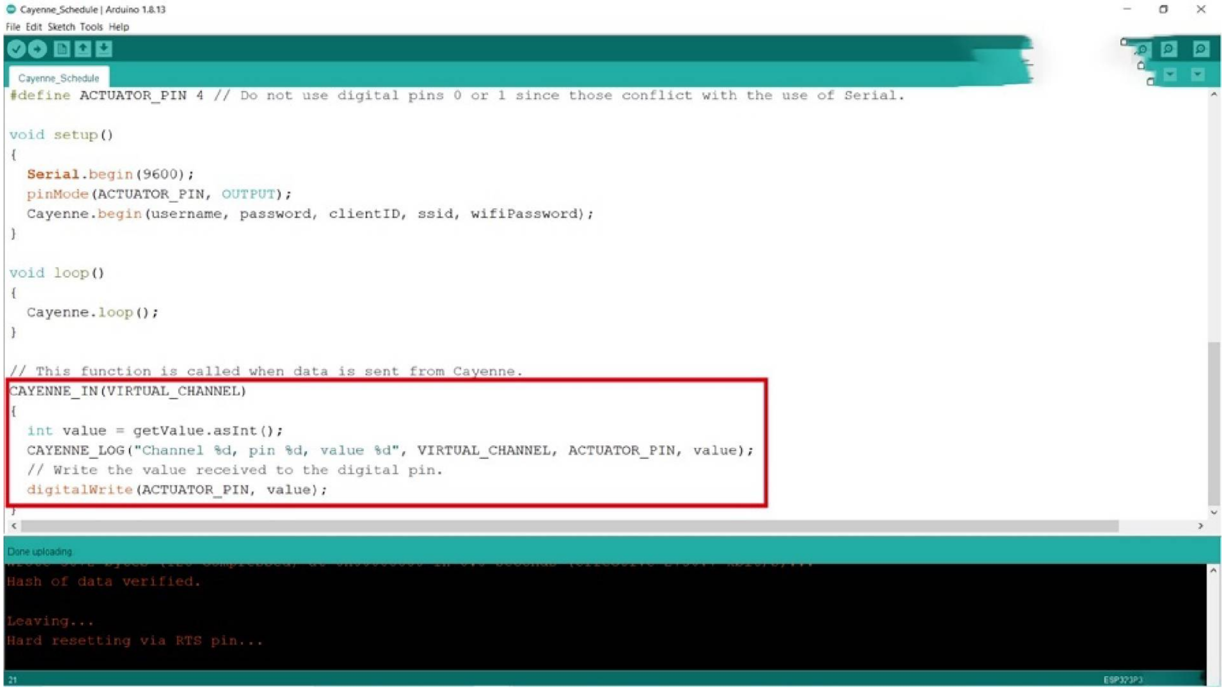

Cayenne_Schedule | Arduino 1.8.13
File Edit Sketch Tools Help
Cayenne_Schedule

```
#define CAYENNE_PRINT Serial  // Comment this out to disable prints and save space
#include <CayenneMQTTESP32.h>

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "MQTT_CLIENTID";

char ssid[] = "WiFi_SSID";
char wifiPassword[] = "WiFi_PASSWORD";

#define VIRTUAL_CHANNEL 0
#define ACTUATOR_PIN 4 // Do not use digital pins 0 or 1 since those conflict with the use of Serial.

void setup()
{
  Serial.begin(9600);
  pinMode(ACTUATOR_PIN, OUTPUT);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
}

void loop()
```

Done uploading.

```
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Now, let's check out the code here. We have defined the virtual channel as zero and an actuator pin 4. Here we have set up the actuator pin output.

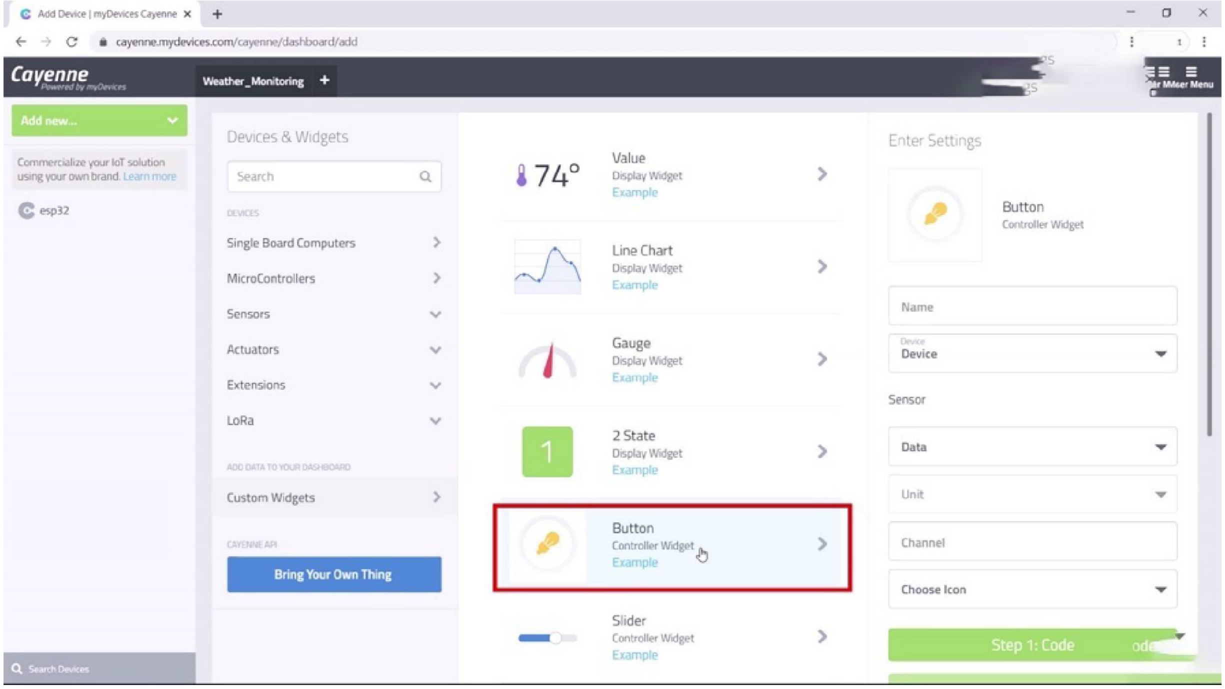

Cayenne_Schedule | Arduino 1.8.13
File Edit Sketch Tools Help
Cayenne_Schedule

```
#define ACTUATOR_PIN 4 // Do not use digital pins 0 or 1 since those conflict with the use of Serial.

void setup()
{
  Serial.begin(9600);
  pinMode(ACTUATOR_PIN, OUTPUT);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
}

void loop()
{
  Cayenne.loop();
}

// This function is called when data is sent from Cayenne.
CAYENNE_IN(VIRTUAL_CHANNEL)
{
  int value = getValue.asInt();
  CAYENNE_LOG("Channel %d, pin %d, value %d", VIRTUAL_CHANNEL, ACTUATOR_PIN, value);
  // Write the value received to the digital pin.
  digitalWrite(ACTUATOR_PIN, value);
}
```

Done uploading.

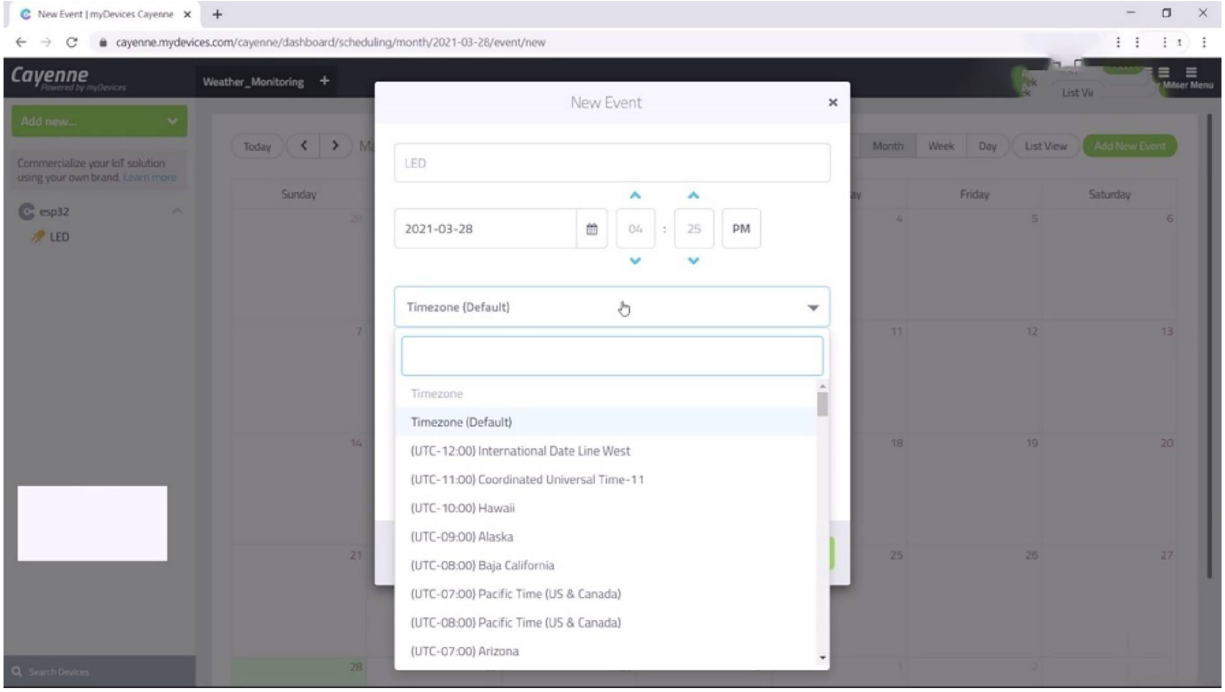
```
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

After connecting to Cayenne here, we will take input from the Cayenne and write the value to ESP32 to make it glow. Now save and upload the code. Open your Cayenne dashboard. We know how to add a new widget to your dashboard.



Click on Add New and the select device widgets. Now click on Custom Widgets, select the button controller widget and fill in the necessary details for the widget.

Now click on Add Widget, you will now have a LED control widget on your dashboard,



New Event

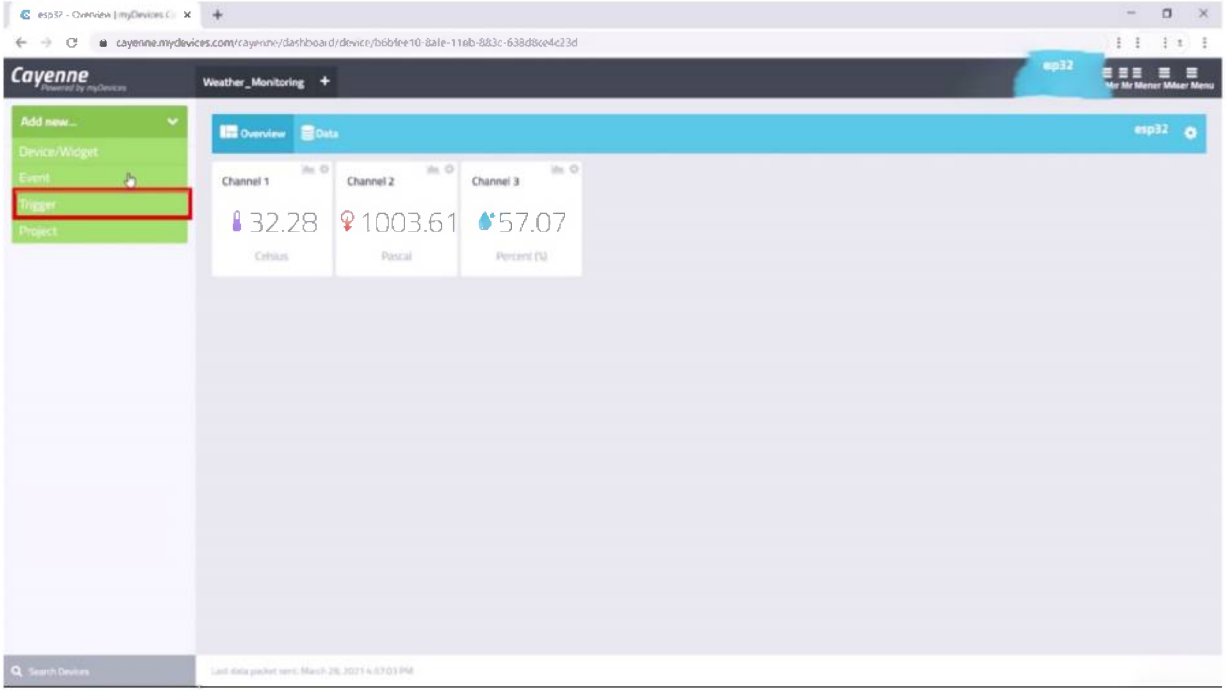
LED

2021-03-28 04 : 25 PM

Timezone (Default)

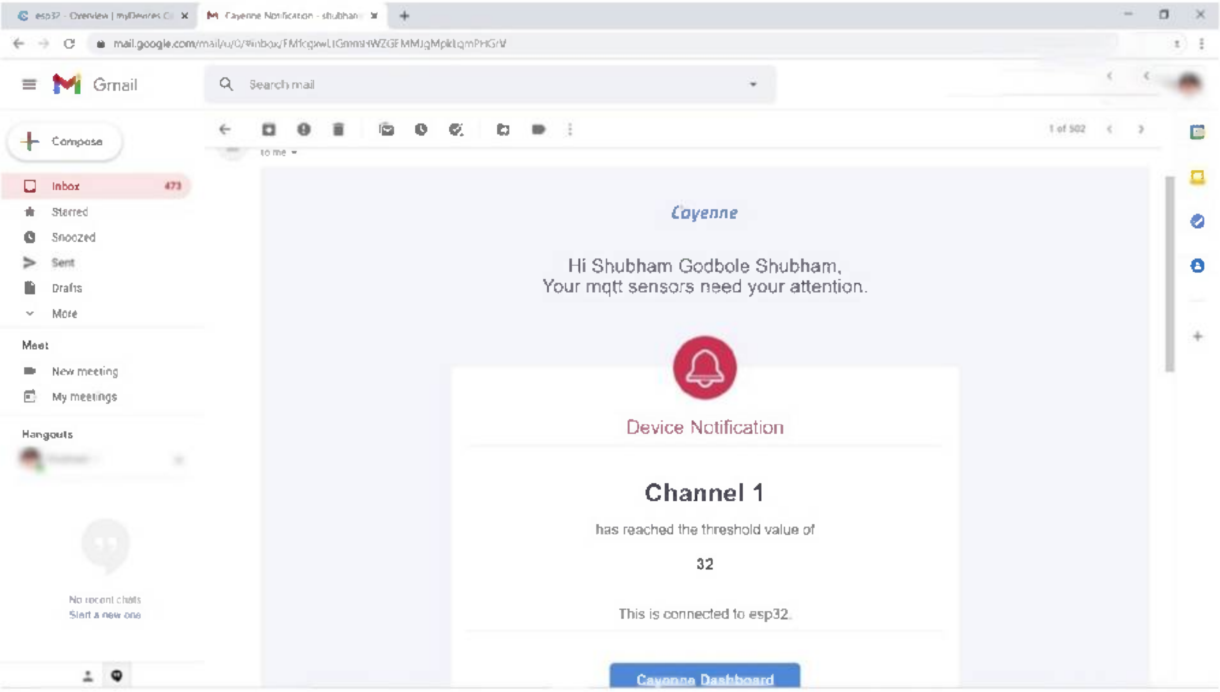
New Event | myDevices Cayenne
cayenne.mydevices.com/cayenne/dashboard/scheduling/month/2021-03-28/event/new
Cayenne
Weather_Monitoring
Add new...
Commercialize your IoT solution using your own brand. Learn more
esp32
LED
Today
Month
Week
Day
List View
Add New Event
Sunday
Friday
Saturday
Timezone
Timezone (Default)
(UTC-12:00) International Date Line West
(UTC-11:00) Coordinated Universal Time-11
(UTC-10:00) Hawaii
(UTC-09:00) Alaska
(UTC-08:00) Baja California
(UTC-07:00) Pacific Time (US & Canada)
(UTC-08:00) Pacific Time (US & Canada)
(UTC-07:00) Arizona
Search Devices

you can now control your LED with the controller widget. Now we need to schedule an event, click on Add New and select event, a dialog box will appear to create a new event, fill in the necessary information. We will choose the action to turn on the LED at the specified time. Now click on save, an event will show up on the calendar. At the specified time, your LED will glow up. Using the event scheduling feature You can automate all the devices, sensors and actuators connected to Cayenne. Along with event scheduling feature in Cayenne you can also automate a task if a certain event occurs. This has the term triggering in-Cayenne. Simply put, if an event takes place, a trigger response action will also take place. To understand the feature better, let's move on to a small project. We will try to send a notification to your email or mobile phone every time the temperature goes beyond a certain threshold. We will use our already pre-built project where we have sent the sensor data collected by the BME280 to Cayenne. Make the necessary circuit connections and upload your code to the ESP32. Now go to the Cayenne dashboard. You will see 3 channels as usual displaying your sensor data.



Now click on Add New and select trigger. Give your trigger a name and then drag and drop your connected device in the IF box.

You will now have an option to select triggers, click on the drop down menu and select channel 1. We have selected Channel One as this channel is responsible for the temperature data from the sensor Set the threshold value according to your use case. Now click on setup notifications. Add your email and mobile number here and click on save you have successfully created a trigger for an event in Cayenne.



Cayenne

Hi Shubham Godbole Shubham,
Your mqtt sensors need your attention.

Device Notification

Channel 1

has reached the threshold value of

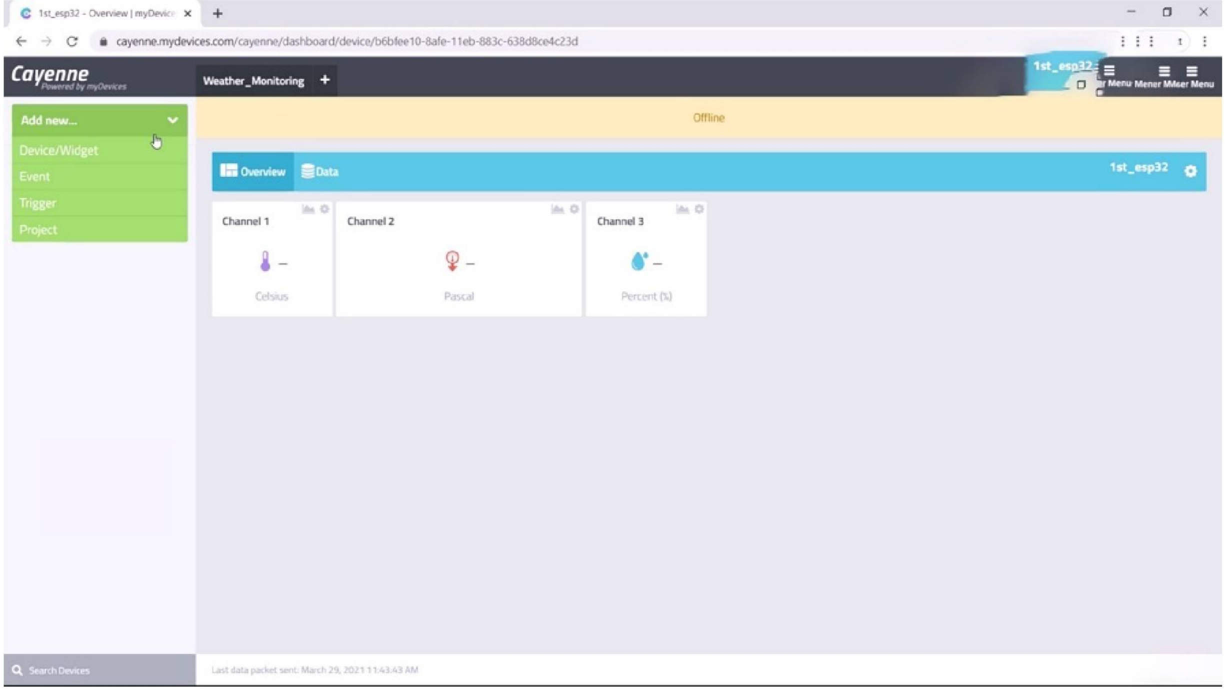
32

This is connected to esp32.

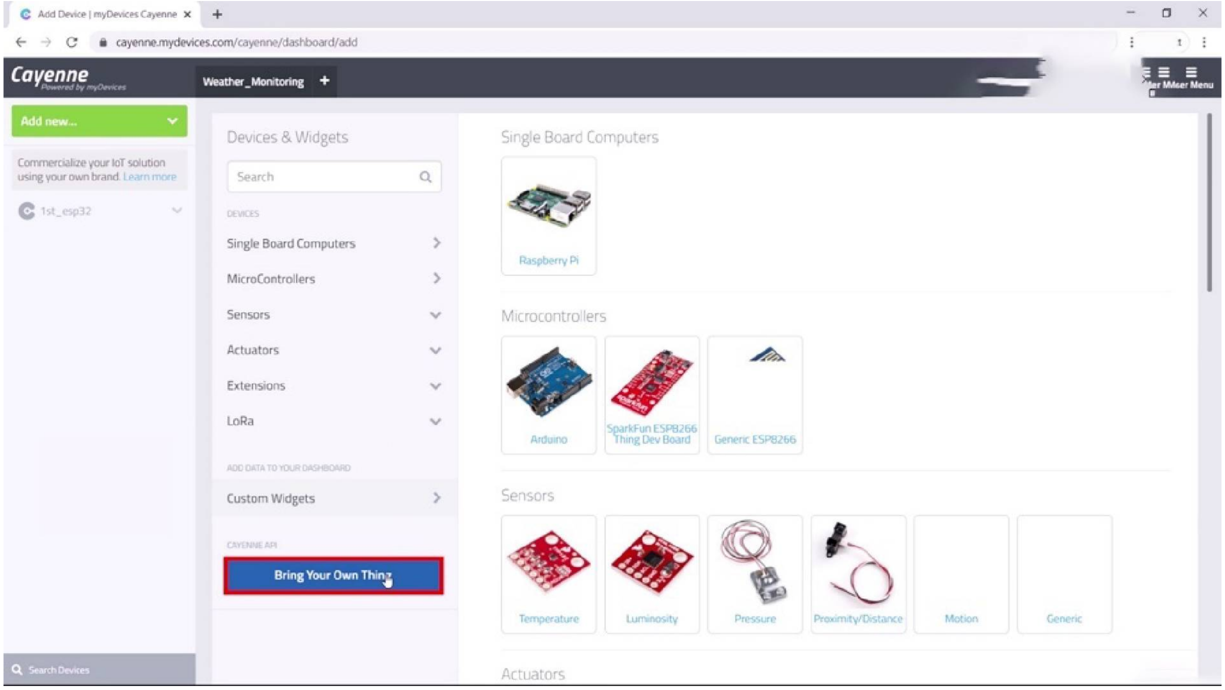
Every time the temperature is more than the threshold value, you will get a notification mentioning the same. You can also get details about the event trigger here. Using the event trigger option gives you an effective yet simple way to implement your IoT Project summary.

ACTUATOR NODE

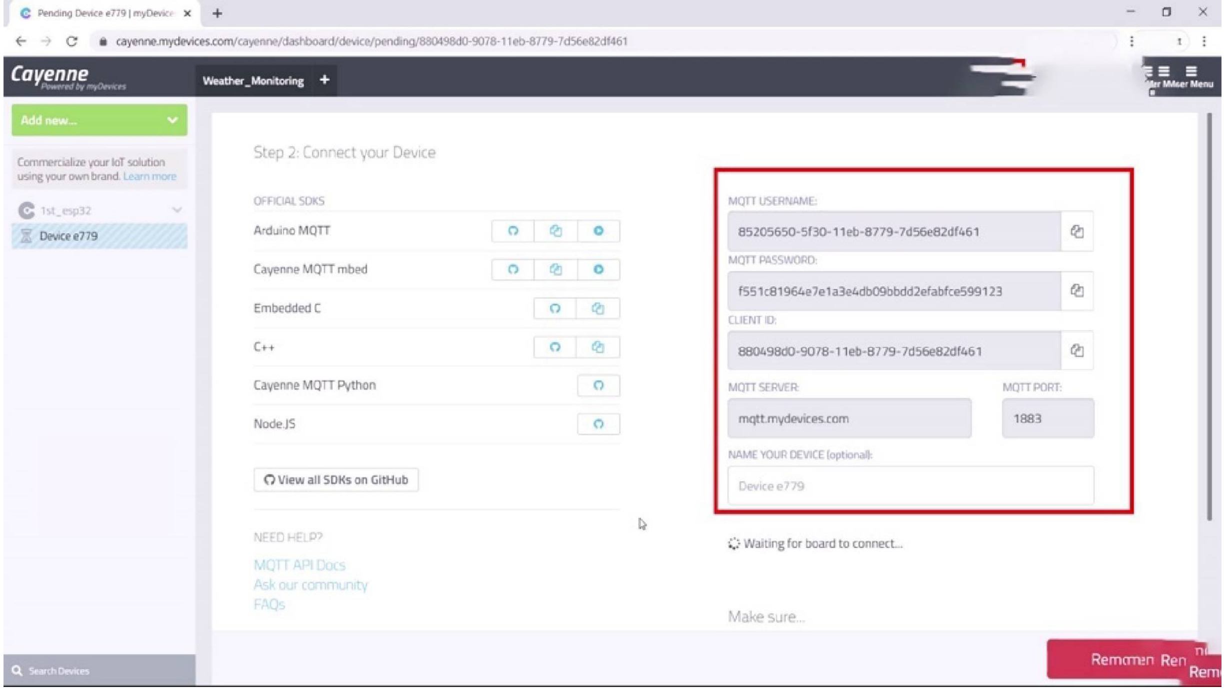
we have seen how a trigger notification can be sent using Cayenne. Now we will introduce a new ESP32 in our weather monitoring system. You have to follow the same steps to set up the new ESP32 with Cayenne.



Click on Add new device widget and click. Bring your own thing.



Use the MQTT credentials from this page to add a second ESP32, you can try toggling on and off an LED to check if your device is connected.



Step 2: Connect your Device

MQTT USERNAME: 85205650-5f30-11eb-8779-7d56e82df461

MQTT PASSWORD: f551c81964e7e1a3e4db09bbdd2efabfce599123

CLIENT ID: 880498d0-9078-11eb-8779-7d56e82df461

MQTT SERVER: mqtt.mydevices.com

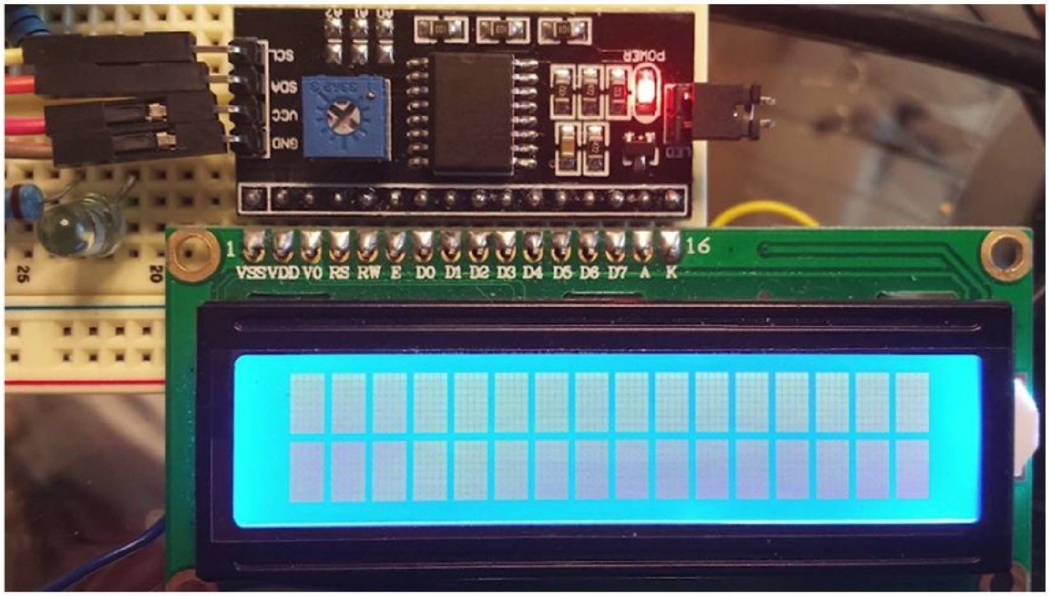
MQTT PORT: 1883

NAME YOUR DEVICE (optional): Device e779

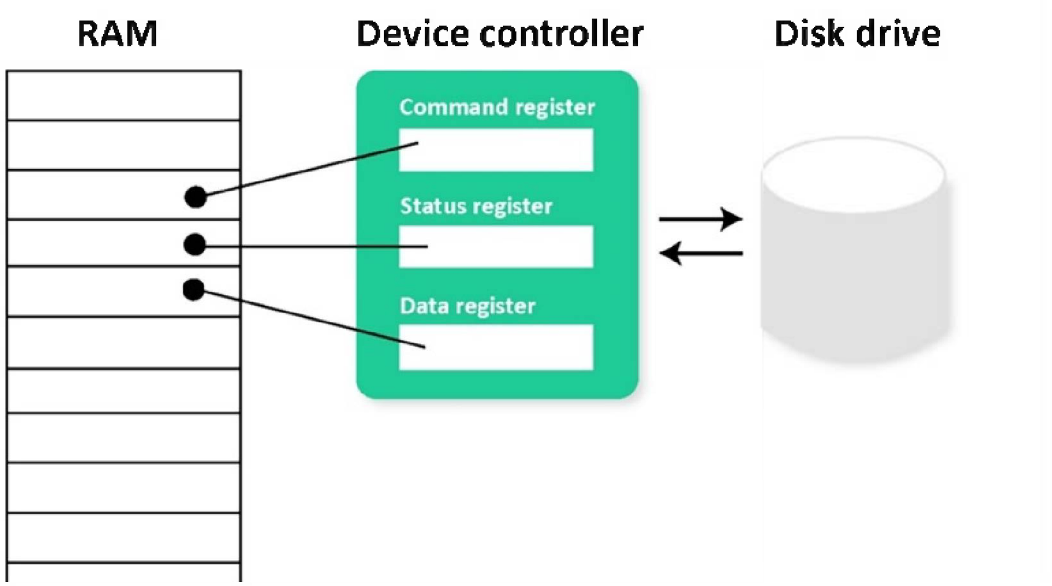
Waiting for board to connect...

Don't you think we should add a display to the system? Yes, we surely will be adding one.

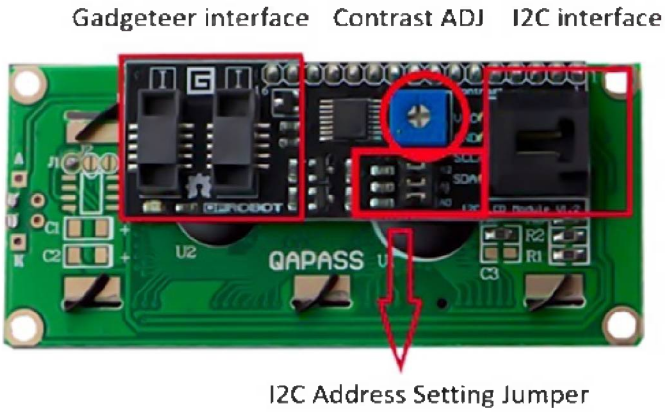
But first, let's get to know more about the display. The 16x2 LCD is one of the more standard alphanumeric displays.



They are extremely common and one of the fastest ways to have your project show status messages. As the name suggests, the liquid crystal display display 16 characters per line and there are two such lines . Each character is displayed in a 5x7 matrix, the 16x2 LCD is capable of displaying 224 different characters and symbols.

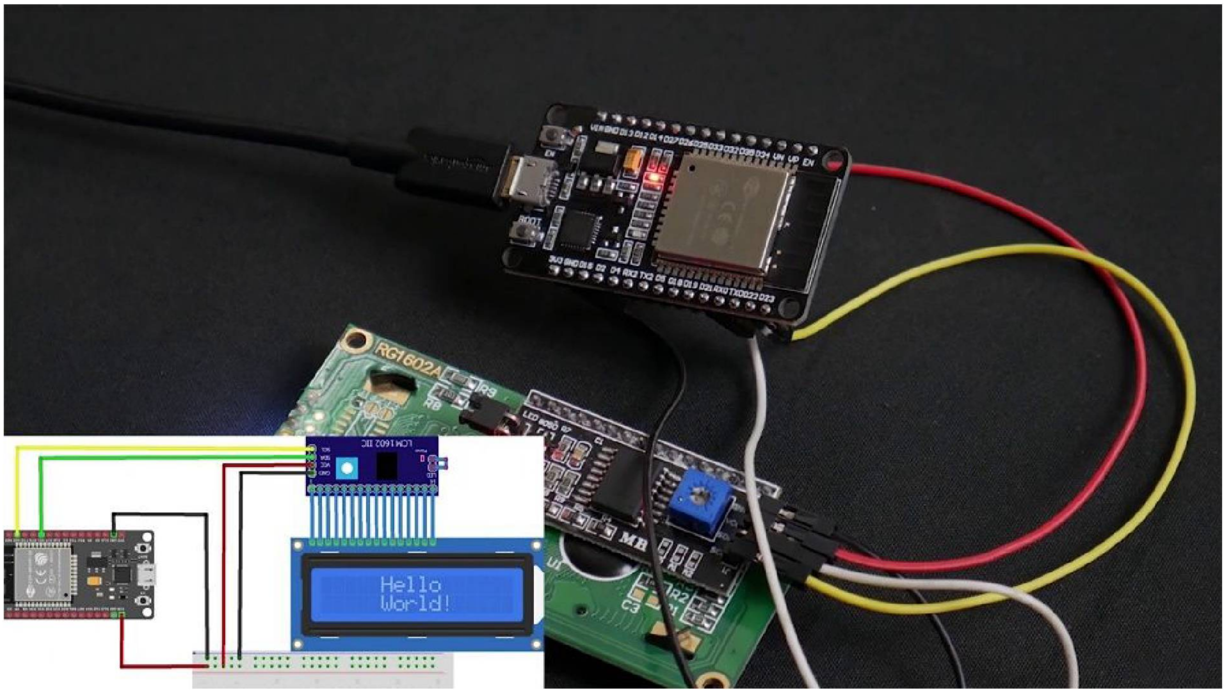


So how does the LCD work? The LCD has two registers, namely command and data. The command registers stores various commands given to the display. The data registers stores data to be displayed. The process of controlling the display involves putting the data that form the image of what you want to display in the data registers, then putting instructions in the instruction register.

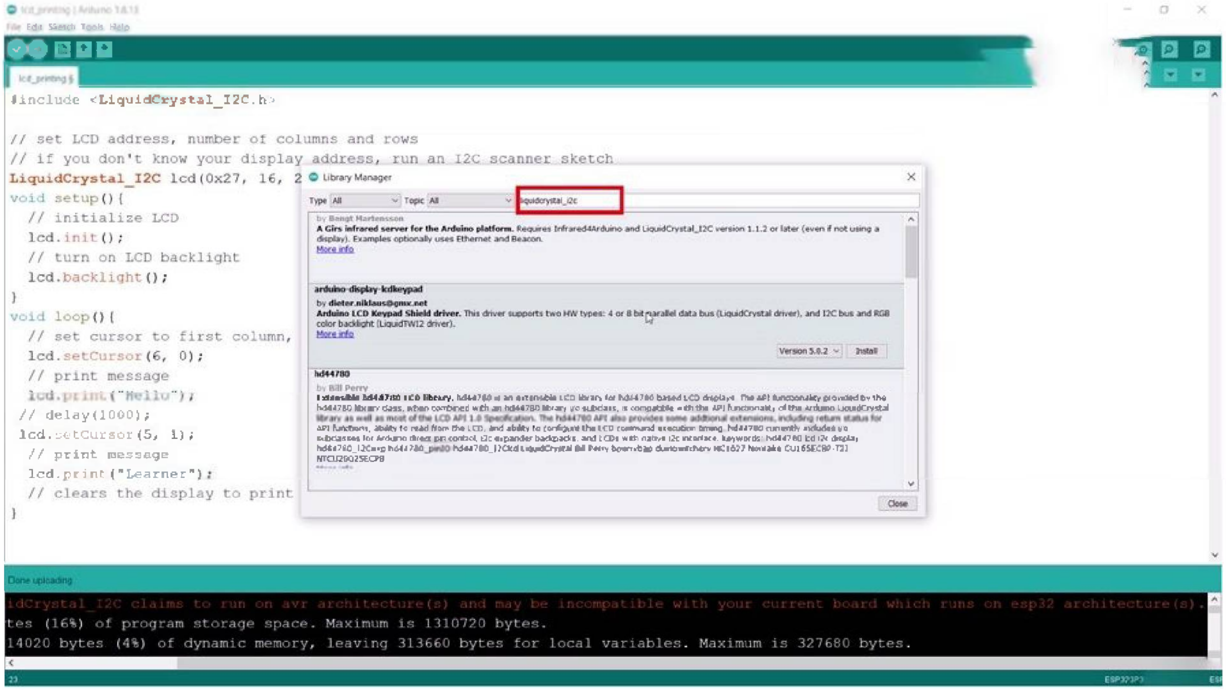


The LCD then displays white characters on a blue background. The contrast of the LCD can also be adjusted by adjusting the potentiometer.

Let's do a small project to understand how the 16x2 LCD works. First, let's make the circuit connections one main advantage of using the 16x2 I2C LCD is that you will not run out of pins on your microcontroller. As the I2C communication interface is used, we will have to connect only four pins for LCD. They are VCC, ground, SCL and the SDA pins.



Connect the VCC of the LCD to the 5V power supply of the ESP32. Connect ground of the LCD to the ground of the ESP32. Now connect the SCL pin of the LCD to GPIO 22 and the SDA pin of the LCD to GPIO 21. You can refer to the circuit diagram in the resources.



```
#include <LiquidCrystal_I2C.h>

// set LCD address, number of columns and rows
// if you don't know your display address, run an I2C scanner sketch
LiquidCrystal_I2C lcd(0x27, 16, 2
void setup(){
  // initialize LCD
  lcd.init();
  // turn on LCD backlight
  lcd.backlight();
}
void loop(){
  // set cursor to first column,
  lcd.setCursor(6, 0);
  // print message
  lcd.print("Hello");
 // delay(1000);
 lcd.setCursor(5, 1);
  // print message
  lcd.print("Learner");
  // clears the display to print
}
```

Library Manager

Type: All Topic: All liquidcrystal_i2c

by Bengt Martensson
A Girs infrared server for the Arduino platform. Requires Infrared4Arduino and LiquidCrystal_I2C version 1.1.2 or later (even if not using a display). Examples optionally uses Ethernet and Beacon.
More info

arduino-display-lcdkeypad
by **dieter.niklaus@gmx.net**
Arduino LCD Keypad Shield driver. This driver supports two HW types: 4 or 8 bit parallel data bus (LiquidCrystal driver), and I2C bus and RGB color backlight (LiquidTWI2 driver).
More info

Version 5.0.2 Install

hd44780
by Bill Perry
Extensible hd44780 LCD library. hd44780 is an extensible LCD library for hd44780 based LCD displays. The API functionality provided by the hd44780 library class, when combined with an hd44780 library i/o subclass, is compatible with the API functionality of the Arduino LiquidCrystal library as well as most of the LCD API 1.0 Specification. The hd44780 API also provides some additional extensions, including return status for API functions, ability to read from the LCD, and ability to configure the LCD command execution timing. hd44780 currently includes i/o subclasses for Arduino direct pin control, i2c expander backpacks, and LCDs with native i2c interface. keywords: hd44780 lcd i2c display hd44780_I2Cexp hd44780_pinIO hd44780_I2Clcd LiquidCrystal Bill Perry bperrybap duinowitchery MC1627 Newhaven CU165ECBP-T2J NTCU20025ECPB

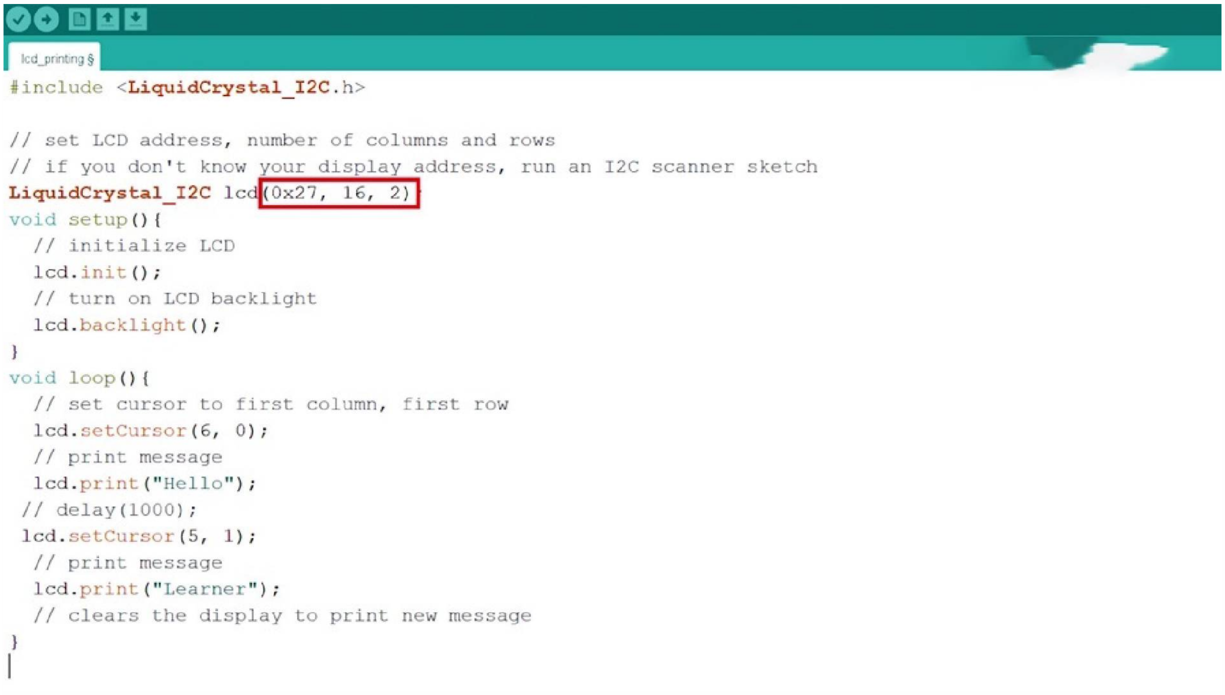
Close

Done uploading

```
idCrystal_I2C claims to run on avr architecture(s) and may be incompatible with your current board which runs on esp32 architecture(s).
tes (16%) of program storage space. Maximum is 1310720 bytes.
14020 bytes (4%) of dynamic memory, leaving 313660 bytes for local variables. Maximum is 327680 bytes.
```

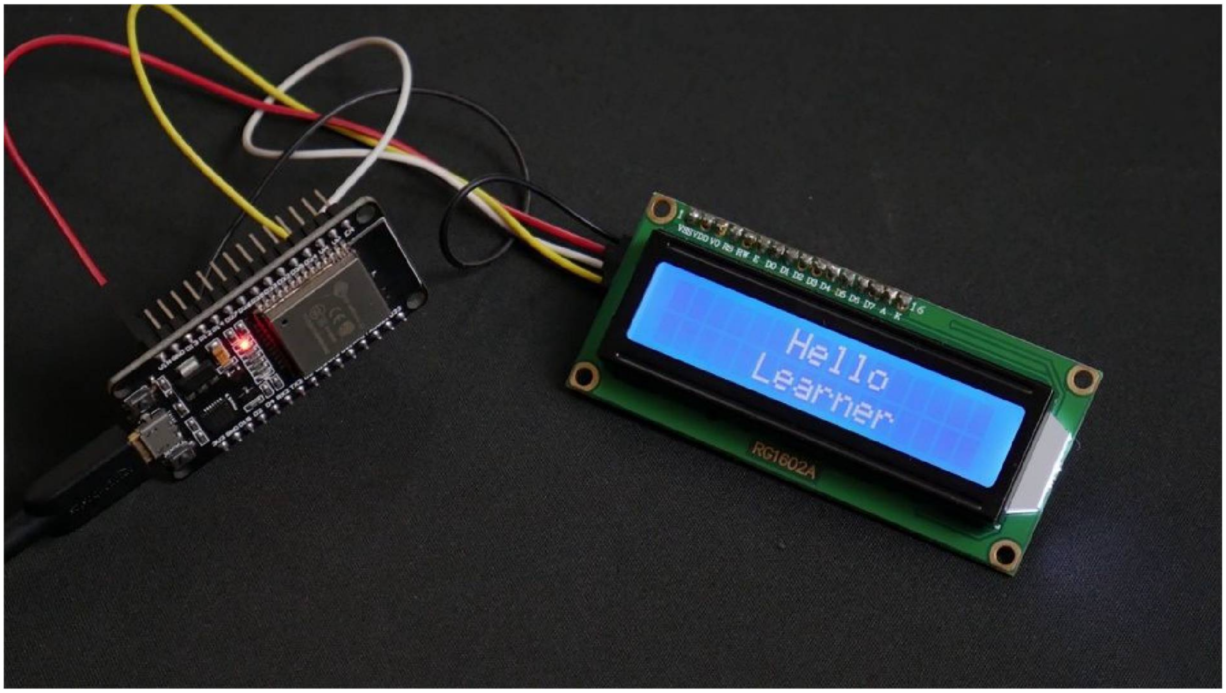
Now let's install the 16x2 LCD library, open your Arduino IDE and click on tools and click manage libraries. This will open your library manager now search for Liquid Crystal I2C library and click on install.

We are now ready to code. Let's check out the code now.



```
#include <LiquidCrystal_I2C.h>

// set LCD address, number of columns and rows
// if you don't know your display address, run an I2C scanner sketch
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup(){
  // initialize LCD
  lcd.init();
  // turn on LCD backlight
  lcd.backlight();
}
void loop(){
  // set cursor to first column, first row
  lcd.setCursor(6, 0);
  // print message
  lcd.print("Hello");
 // delay(1000);
 lcd.setCursor(5, 1);
  // print message
  lcd.print("Learner");
  // clears the display to print new message
}
```

First we have important the liquid crystal I2C library we just installed. Here we have set the I2C address and numbers of columns and rows in our LCD. Now we will initialize the LCD and turn on the backlight of the LCD. That is the blue background light. As we set the cursor to type anything on your computers. Similarly, here we have set the cursor to the sixth character on the first row. Using this, we will print our message. Similarly, we will print another message in the second row also Now save and upload your code.



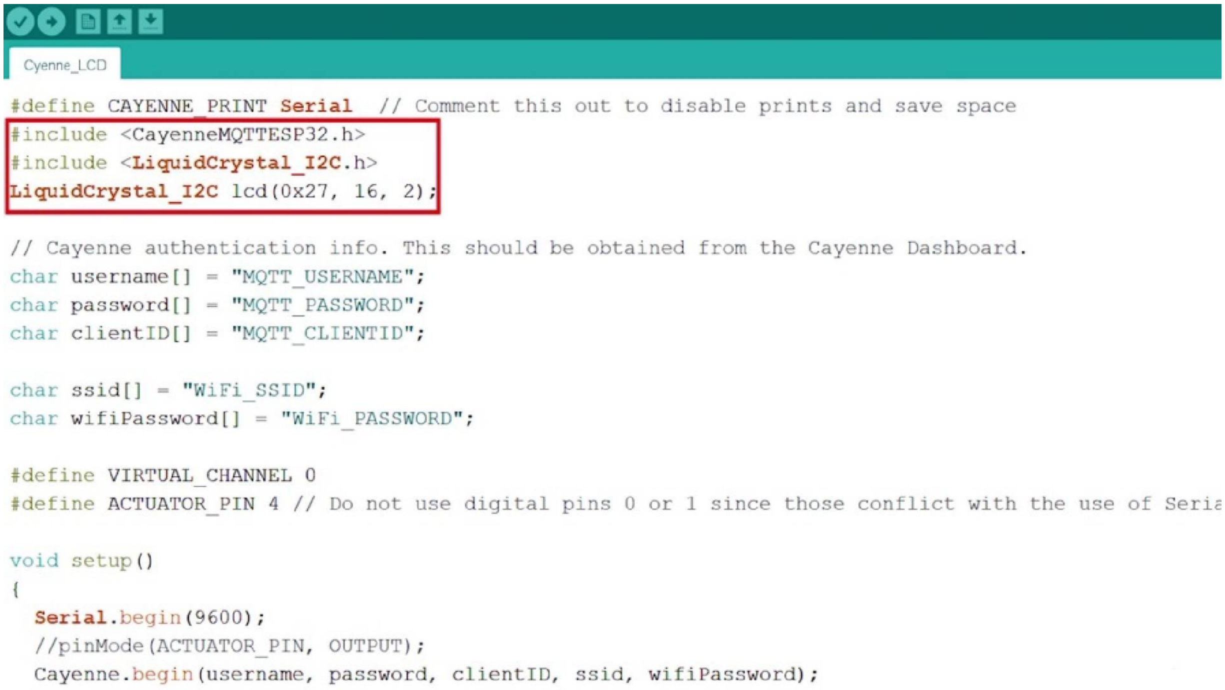
You will now see the display message you have printed on your LCD if you cannot see the messages clearly and just the contrast of the LCD accordingly using the inbuilt potentiometer.

Now that we have an idea of how to use the 16x2 LCD, we need to integrate it into our Weather monitoring system now. However, Cayenne has a drawback, you cannot directly interface and LCD with the Cayenne platform, nor can you send data from one device to another directly. You can check out the link in the resources for a discussion regarding this on the Cayenne community page. The Thing Speak IoT platform service that we will look in the next section does not have this issue.

END TO END IMPLEMENTATION

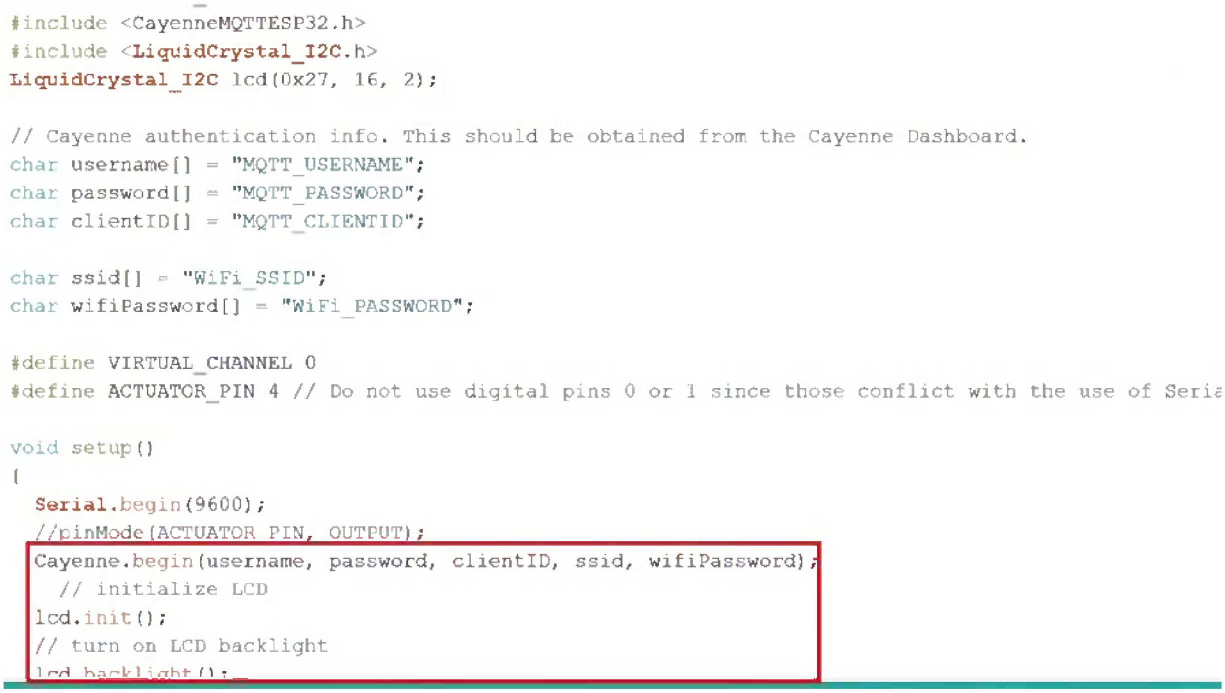
we mentioned a limitation of Cayenne and how we cannot directly interface the LCD with Cayenne. However, we have figured out a way around this to implement our end to end project successfully. You must be wondering how we will be doing it. We will use the LED control widget from Cayenne. This will provide us with the binary values from Cayenne. We can then code our LCD to display notifications based on these values.

Let's get into the practical implementation of the project to understand the concept and working of our system more clearly. As you did in the last project, we have already added the second ESP32 into the system. Let's add a button control widget, go to add new devices and widgets. Now click on the custom widget icon and select button controller widget, filling the necessary details and then click on Add Widget. You will now have a button controller widget on your dashboard.



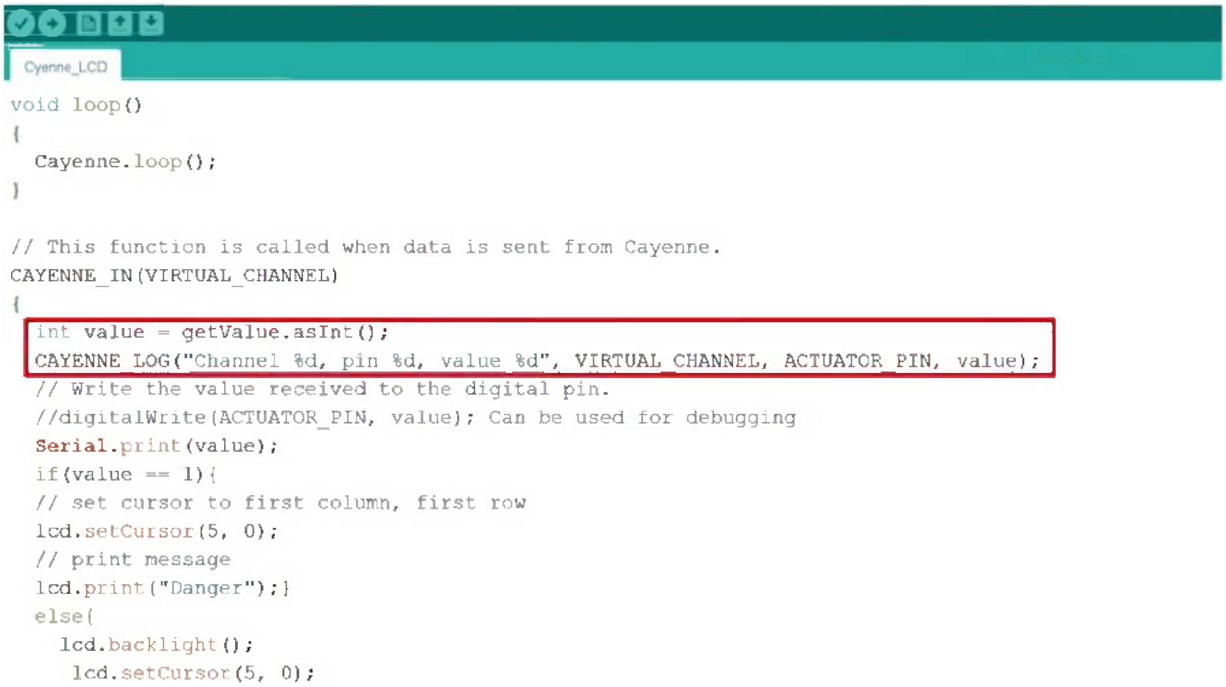
```
#define CAYENNE_PRINT Serial  // Comment this out to disable prints and save space
#include <CayenneMQTTESP32.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "MQTT_CLIENTID";

char ssid[] = "WiFi_SSID";
char wifiPassword[] = "WiFi_PASSWORD";

#define VIRTUAL_CHANNEL 0
#define ACTUATOR_PIN 4 // Do not use digital pins 0 or 1 since those conflict with the use of Seria

void setup()
{
  Serial.begin(9600);
  //pinMode(ACTUATOR_PIN, OUTPUT);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
```

Now let's take out the code here. We have imported the necessary libraries and defined the I2C address of the LCD. Now we will enter all the necessary credentials needed to connect to Cayenne and your Wi-Fi. Here we have defined a virtual Channel zero and an actuator pin four. This actuator pin will control the LED and can be used for debugging. It is OK even if you do not use the LED actuator pin.



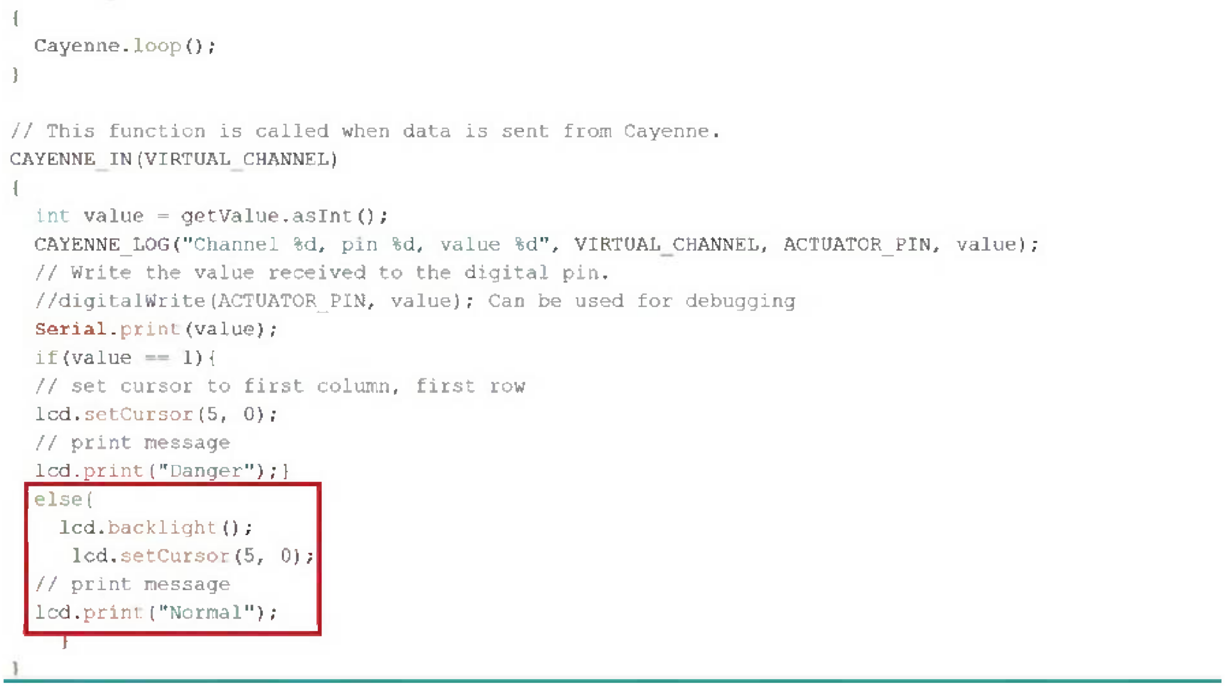
```
#include <CayenneMQTTESP32.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "MQTT_USERNAME";
char password[] = "MQTT_PASSWORD";
char clientID[] = "MQTT_CLIENTID";

char ssid[] = "WiFi_SSID";
char wifiPassword[] = "WiFi_PASSWORD";

#define VIRTUAL_CHANNEL 0
#define ACTUATOR_PIN 4 // Do not use digital pins 0 or 1 since those conflict with the use of Seria

void setup()
{
  Serial.begin(9600);
  //pinMode(ACTUATOR_PIN, OUTPUT);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
    // initialize LCD
  lcd.init();
  // turn on LCD backlight
  lcd.backlight();
```

Here We will connect to Cayenne and initialize our LCD here. We will get values from Cayenne.



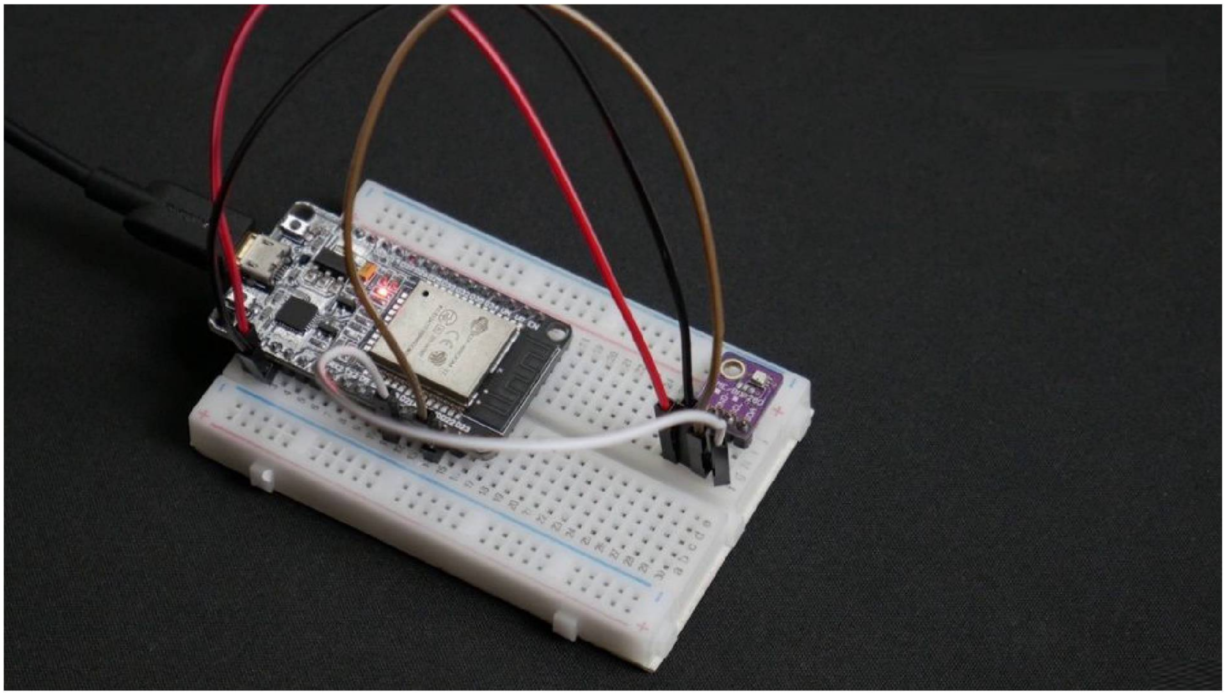
```
void loop()
{
  Cayenne.loop();
}

// This function is called when data is sent from Cayenne.
CAYENNE_IN(VIRTUAL_CHANNEL)
{
  int value = getValue.asInt();
  CAYENNE_LOG("Channel %d, pin %d, value %d", VIRTUAL_CHANNEL, ACTUATOR_PIN, value);
  // Write the value received to the digital pin.
  //digitalWrite(ACTUATOR_PIN, value); Can be used for debugging
  Serial.print(value);
  if(value == 1){
  // set cursor to first column, first row
  lcd.setCursor(5, 0);
  // print message
  lcd.print("Danger");}
  else{
    lcd.backlight();
     lcd.setCursor(5, 0);
```

We will store the value received from Cayenne in the variable value. This value will either be zero or one as the button control widget can only go on or off.

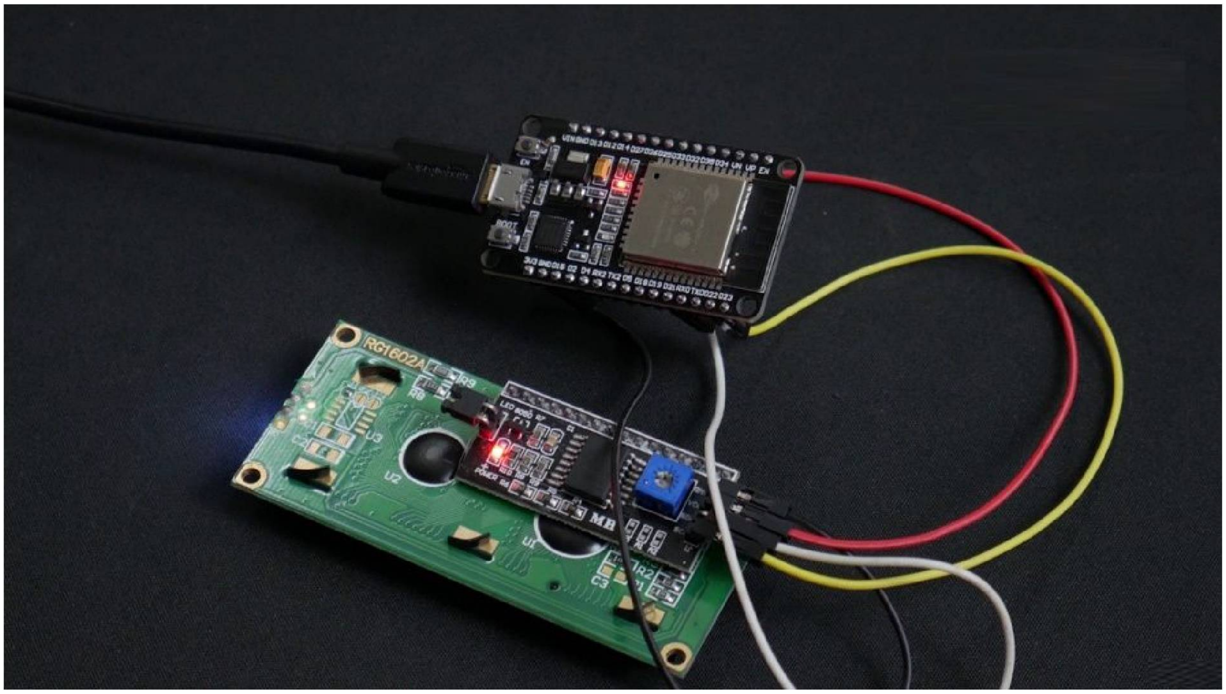


```
{
  Cayenne.loop();
}

// This function is called when data is sent from Cayenne.
CAYENNE_IN(VIRTUAL_CHANNEL)
{
  int value = getValue.asInt();
  CAYENNE_LOG("Channel %d, pin %d, value %d", VIRTUAL_CHANNEL, ACTUATOR_PIN, value);
  // Write the value received to the digital pin.
  //digitalWrite(ACTUATOR_PIN, value); Can be used for debugging
  Serial.print(value);
  if(value == 1){
  // set cursor to first column, first row
  lcd.setCursor(5, 0);
  // print message
  lcd.print("Danger");}
  else{
    lcd.backlight();
     lcd.setCursor(5, 0);
  // print message
  lcd.print("Normal");
    }
}
```

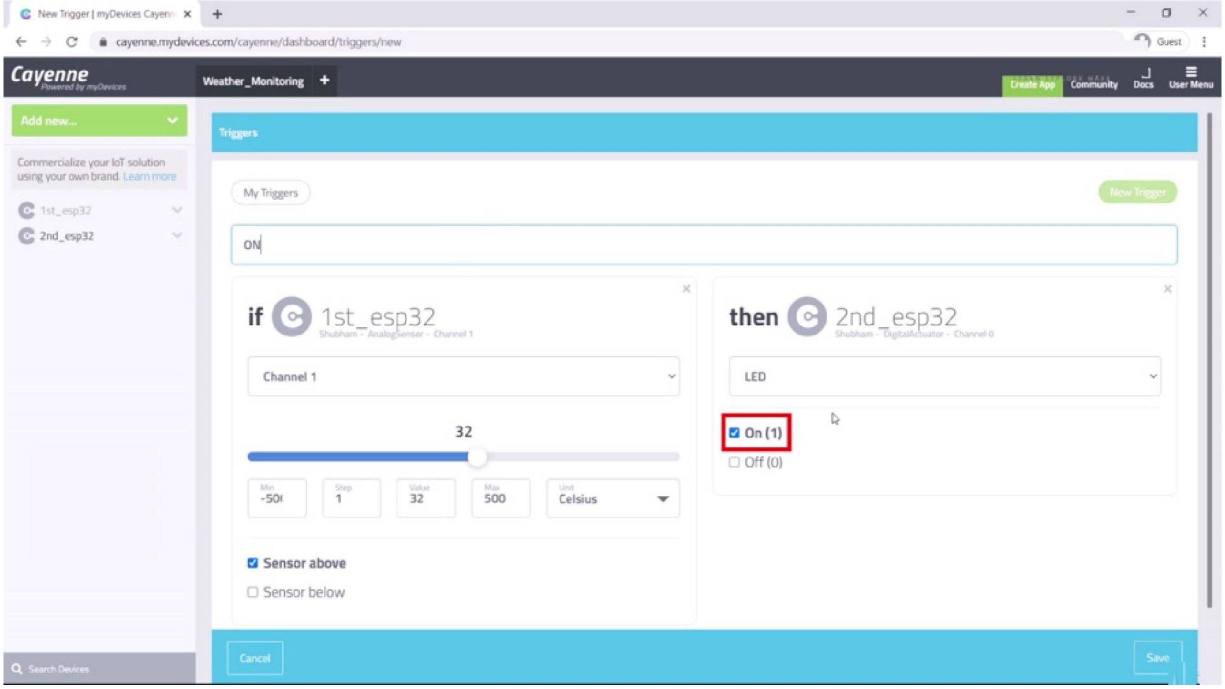
Now is the interesting part based on these values. Based on these values received, we will now program the LCD to display notifications accordingly. So here whenever the value received from the Cayenne will be one or in other words, the button controller widget will be on, then we will display the word danger on our LCD. Now, when the value received from Cayenne will be zero, then we will display the word normal. Now save and upload this code to the second ESP 32.



Now let's come to the hardware part. Connect the first ESP32 to the BME280 sensor and upload the code, which sends sensor data to Cayenne.



Now connect the second ESP32 with the LCD. Your hardware side is now good to go. Now open your Cayenne account. Here We have to set triggers accordingly. Go to add new, triggers now drag and drop your first ESP32 to the if box. Now set the trigger. We will set the trigger for temperature like this. We are selecting Channel one as Channel One is responsible for temperature values, select sensor above. This implies that the triggered action will take place if the sensor value is above the specified value.



ON

if 1st_esp32
Shubham - AnalogSensor - Channel 1

Channel 1

32

Min	Step	Value	Max	Unit
-50(	1	32	500	Celsius

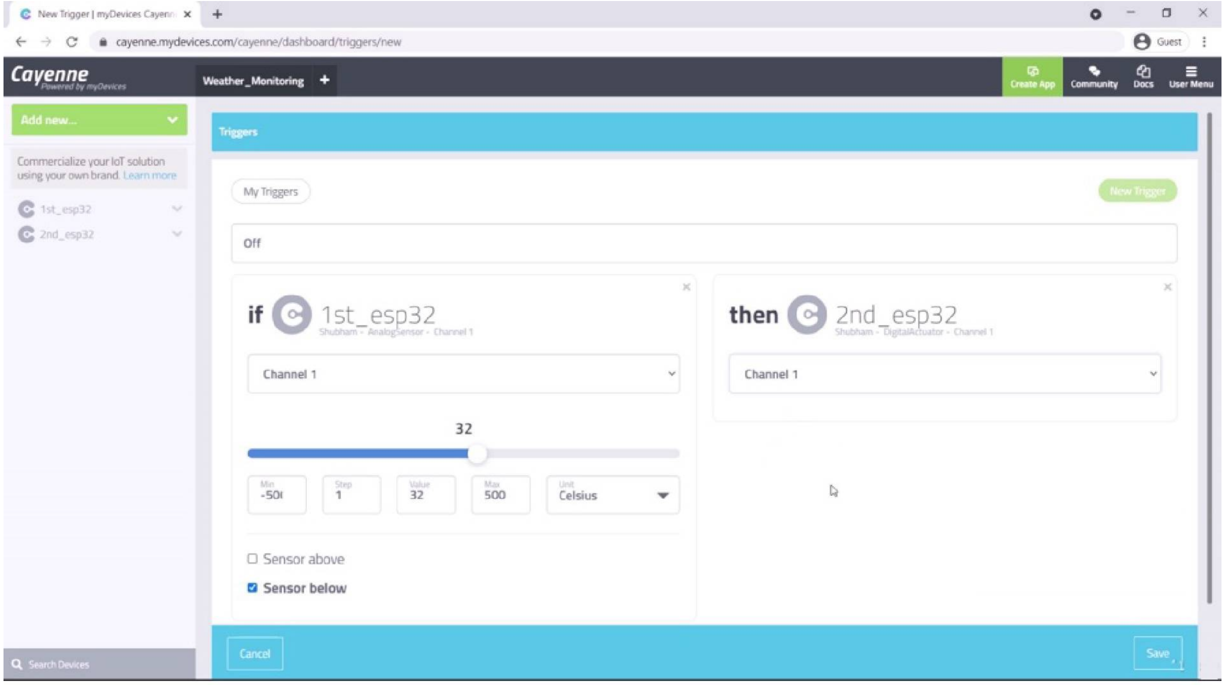
☑ Sensor above
☐ Sensor below

then 2nd_esp32
Shubham - DigitalActuator - Channel 0

LED

☑ On (1)
☐ Off (0)

Now drag and drop your second ESP32. Now select action and click on LED and select the on option. Here we have created a trigger where each time the temperature goes above the threshold, it will trigger a little button controller widget to go on and thereby displaying danger on our LCD. To automate the system fully, we will have to add another trigger.


New Trigger | myDevices Cayenne
cayenne.mydevices.com/cayenne/dashboard/triggers/new
Cayenne
Weather_Monitoring
Create App
Community
Docs
User Menu
Add new...
Commercialize your IoT solution using your own brand. Learn more
1st_esp32
2nd_esp32
Triggers
My Triggers
New Trigger
Search Devices
Cancel
Save

Off

if 1st_esp32 — Shubham - AnalogSensor - Channel 1

Channel 1

32

Min -50(· Step 1 · Value 32 · Max 500 · Unit Celsius

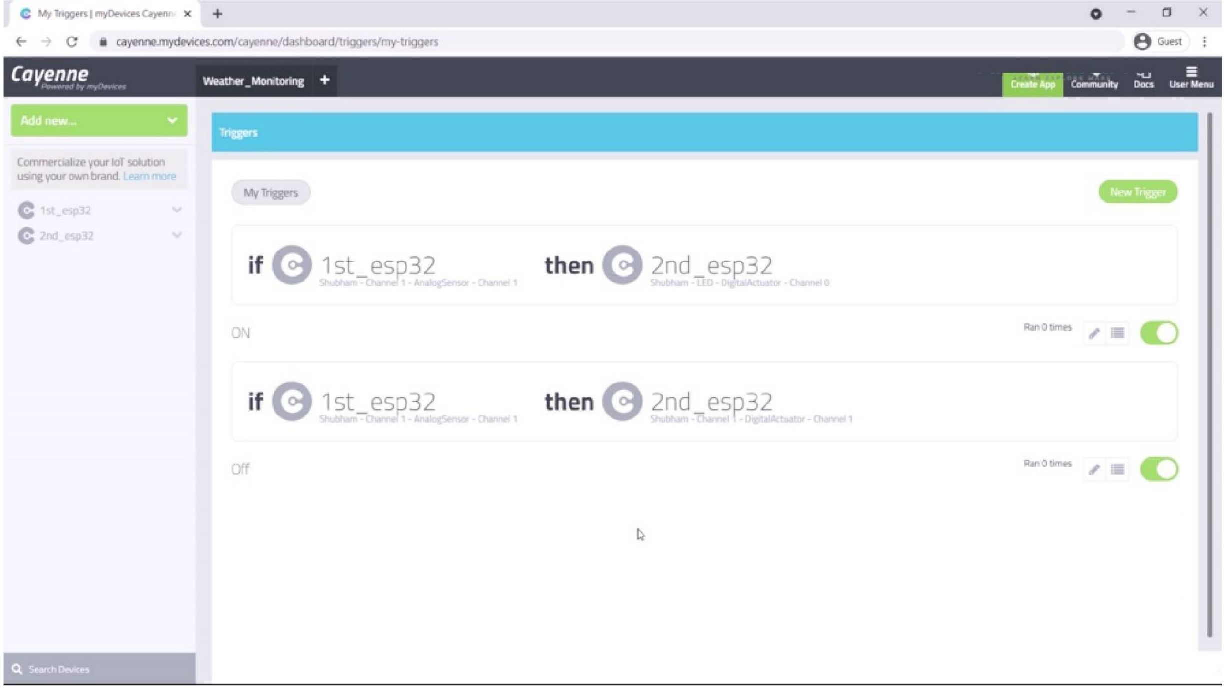
☐ Sensor above

☑ Sensor below

then 2nd_esp32 — Shubham - DigitalActuator - Channel 1

Channel 1

For this Go to the trigger option from add new and drag and drop your first ESP32 and input the same threshold value as before. But now, instead of the sensor above select sensor below and for the second ESP32 select LED off.


My Triggers | myDevices Cayenne
cayenne.mydevices.com/cayenne/dashboard/triggers/my-triggers
Cayenne
Weather_Monitoring
Create App
Community
Docs
User Menu
Add new...
Commercialize your IoT solution using your own brand. Learn more
1st_esp32
2nd_esp32
Triggers
My Triggers
New Trigger
Search Devices

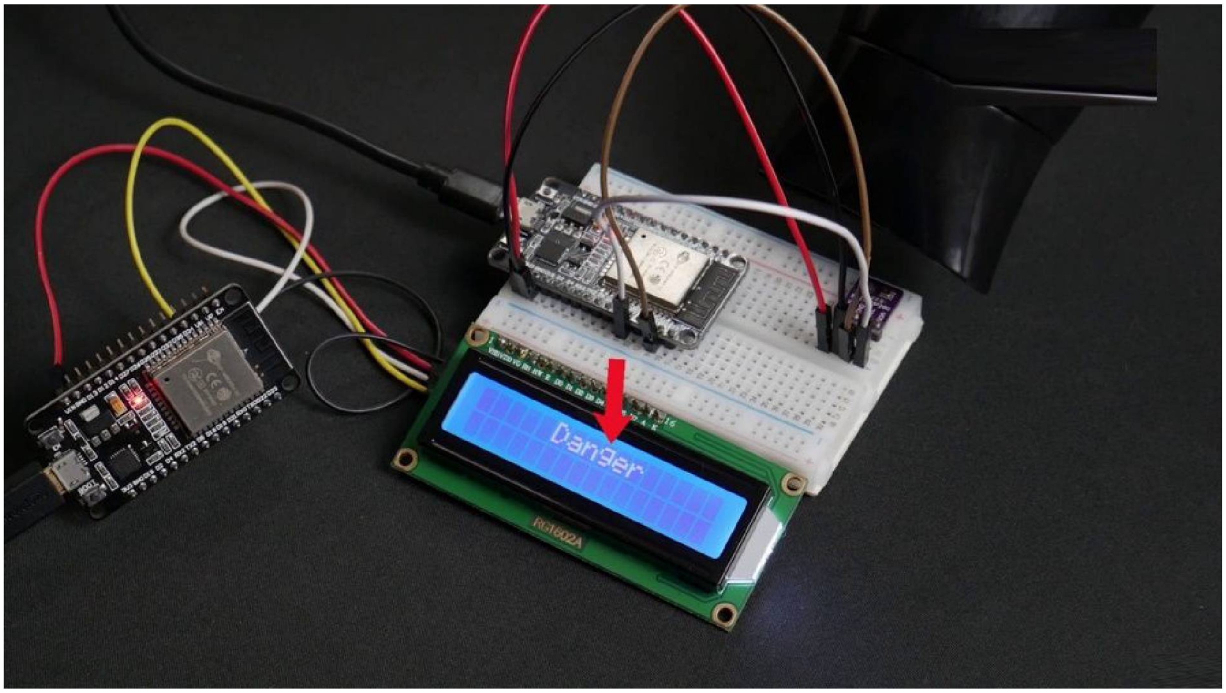
if 1st_esp32 — Shubham - Channel 1 - AnalogSensor - Channel 1 then 2nd_esp32 — Shubham - LED - DigitalActuator - Channel 0

ON — Ran 0 times

if 1st_esp32 — Shubham - Channel 1 - AnalogSensor - Channel 1 then 2nd_esp32 — Shubham - Channel 1 - DigitalActuator - Channel 1

Off — Ran 0 times

This will automate the system to display the text as normal whenever the system has not crossed the threshold value.



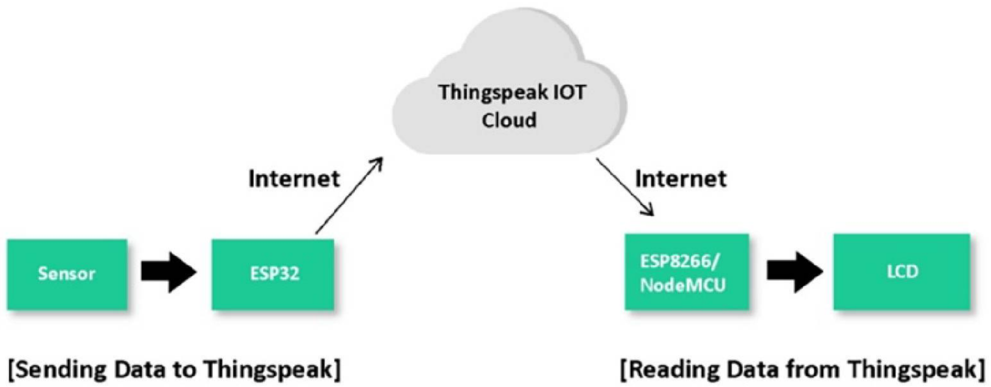
Now, power both the ESP32 and you have your own weather monitoring system.

INTRODUCTION TO THINGSPEAK

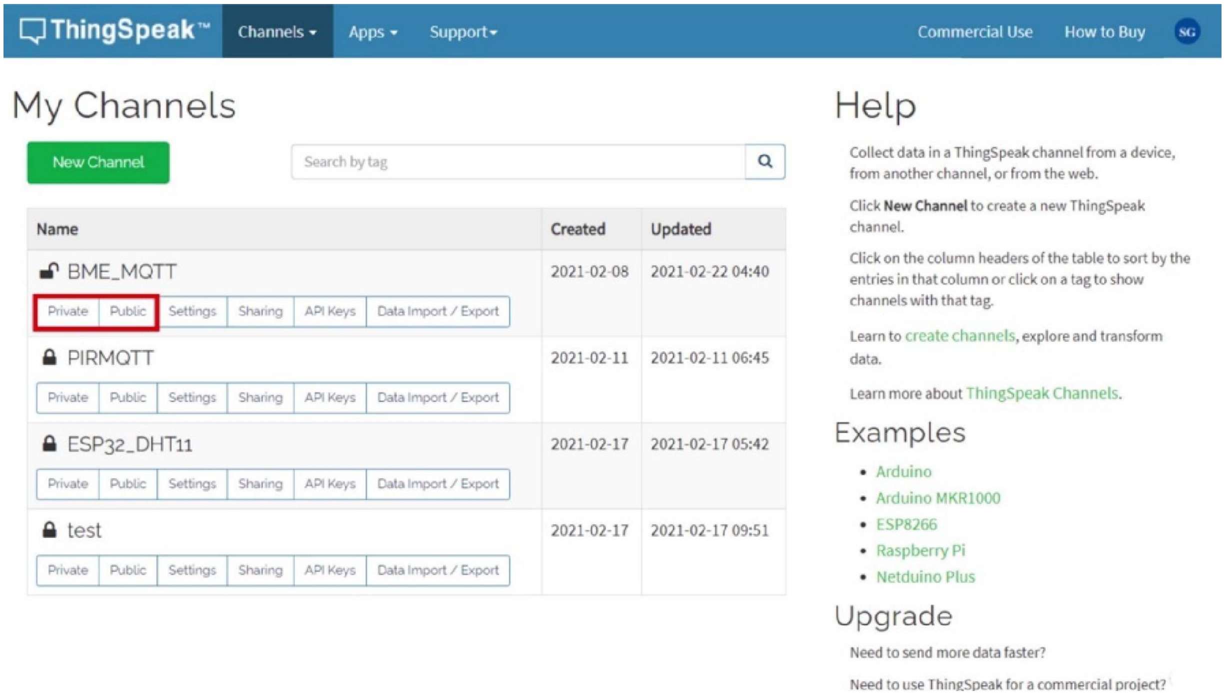
we will cover the following topics. What is Thing Speak? How does Thing Speak work? Thing Speak versus Cayenne, supported hardware, library support. Now that you have used Cayenne and you have an overall idea of developing a project, let's dive in a little deeper and learn more about Thing Speak, which is an another IoT service platform. Now you may be wondering what is Thing Speak?

Thing Speak is an open source IoT analytics platform that allows you to aggregate, visualize and analyze live data streams in the cloud. Thing Speak provides instant visualization of the data posted by your connected devices, Thing Speak has the ability to execute Matlab code through which you can perform online analysis and processing of data as soon as it comes in. Thing

Speak is a go to option to build prototypes and proof of concept IoT systems that require analytics.



So how does Thing Speak work? So as we know, sensors or things sense data and generally act locally. Thing Speak enables sensors, instruments and even websites to send data to the Thing Speak cloud.



My Channels

Name	Created	Updated
BME_MQTT Private Public Settings Sharing API Keys Data Import / Export	2021-02-08	2021-02-22 04:40
PIRMQTT Private Public Settings Sharing API Keys Data Import / Export	2021-02-11	2021-02-11 06:45
ESP32_DHT11 Private Public Settings Sharing API Keys Data Import / Export	2021-02-17	2021-02-17 05:42
test Private Public Settings Sharing API Keys Data Import / Export	2021-02-17	2021-02-17 09:51

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

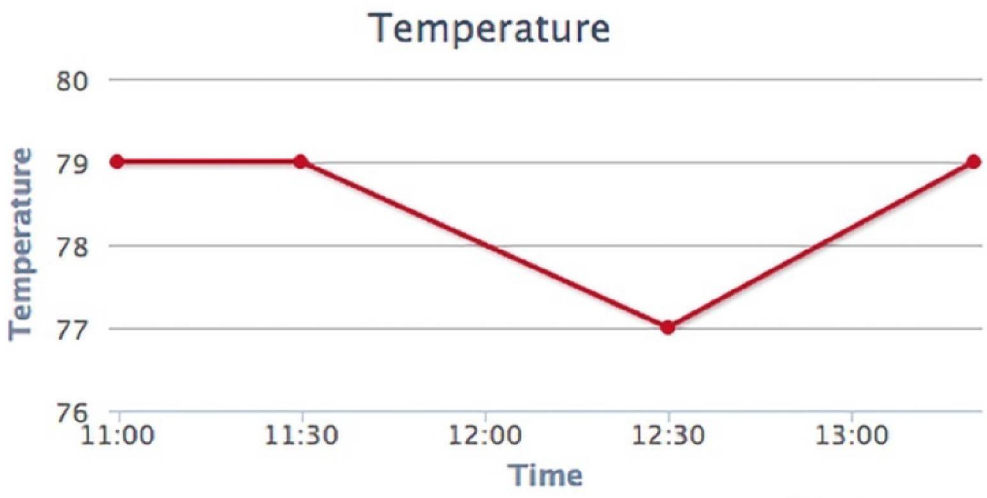
- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

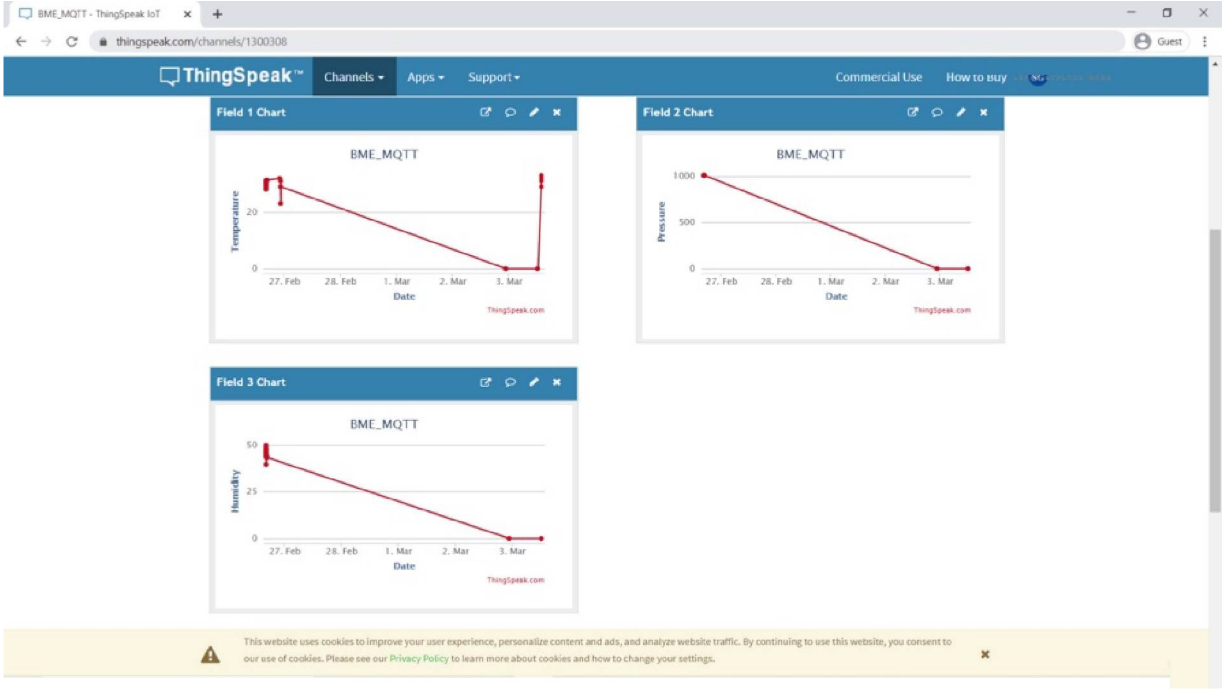
Need to send more data faster?

Need to use ThingSpeak for a commercial project?

Thing Speak stores the data in its predefined private or public channels. By default, your data will be stored in a private channel, but public channels can be used to share data with others.



Once the data is in the respective channel, you can analyze, visualize or even calculate new data as per the data received. You also have an option to interact with social media, web services and even other devices. We have already used Cayenne to develop an IoT system. So why do we need Thing Speak ? Each of these platforms have their pros and cons. Cayenne's Drag and drop feature makes it very easy to use for prototype building. However, as we saw in our IoT weather monitoring system, we had to develop a workaround to interface the LCD. This makes Cayenne only suitable for beginner level projects.



This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

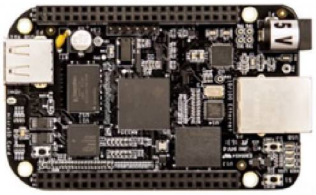
Thing Speak is a little more advanced and more suitable for those who can write code for functions. Due to this, Thing Speak can be extensively used for high end IoT applications. Thing Speak also offers data analytics with math works, which is an additional feature that Cayenne lacks.

One more advantage of Thing Speak is that it is compatible with more hardware as compared to Cayenne.

Arduino

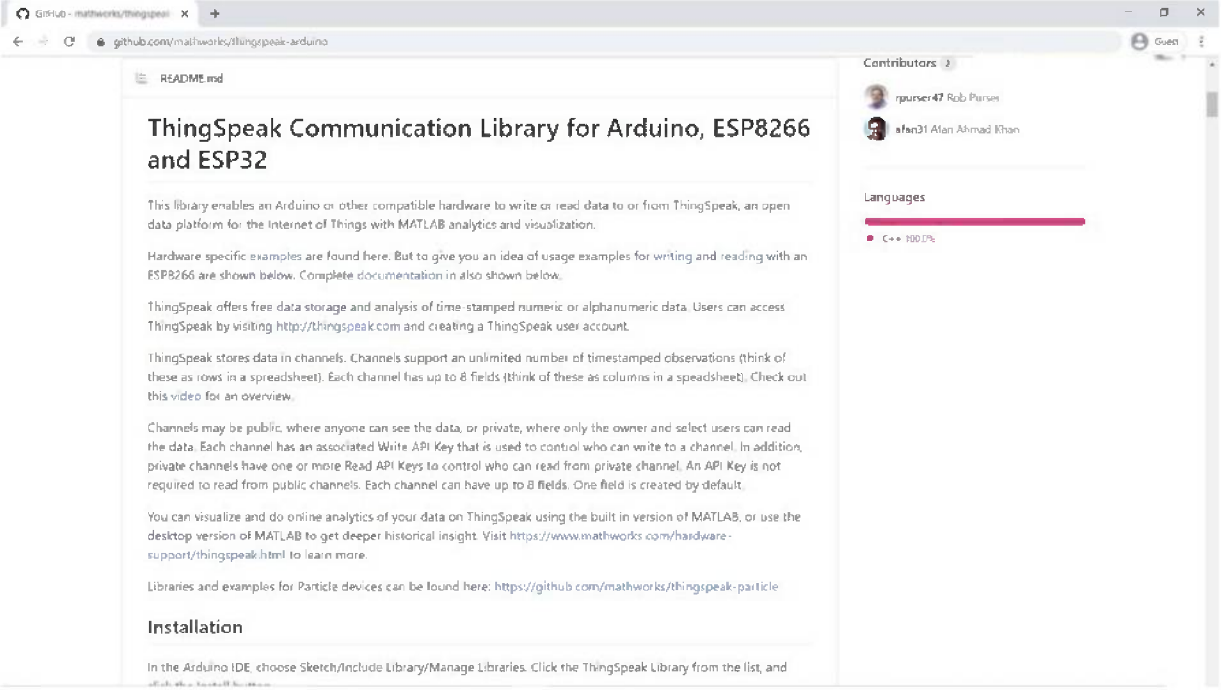
Raspberry Pi

Beaglebone



Thing Speak is compatible with the wide variety of boards and can send sensor data from devices such as Arduino, Raspberry Pi and Beagle bone.

You can also connect other hardware devices to communicate with Thing Speak. The only thing you need to keep in mind is that they should support protocols such as TCP/IP, HTTP or MQTT.



GitHub - mathworks/thingspeak
github.com/mathworks/thingspeak-arduino
Guest
README.md

ThingSpeak Communication Library for Arduino, ESP8266 and ESP32

This library enables an Arduino or other compatible hardware to write or read data to or from ThingSpeak, an open data platform for the Internet of Things with MATLAB analytics and visualization.

Hardware specific examples are found here. But to give you an idea of usage examples for writing and reading with an ESP8266 are shown below. Complete documentation in also shown below.

ThingSpeak offers free data storage and analysis of time-stamped numeric or alphanumeric data. Users can access ThingSpeak by visiting http://thingspeak.com and creating a ThingSpeak user account.

ThingSpeak stores data in channels. Channels support an unlimited number of timestamped observations (think of these as rows in a spreadsheet). Each channel has up to 8 fields (think of these as columns in a speadsheet). Check out this video for an overview.

Channels may be public, where anyone can see the data, or private, where only the owner and select users can read the data. Each channel has an associated Write API Key that is used to contol who can write to a channel. In addition, private channels have one or more Read API Keys to control who can read from private channel. An API Key is not required to read from public channels. Each channel can have up to 8 fields. One field is created by default.

You can visualize and do online analytics of your data on ThingSpeak using the built in version of MATLAB, or use the desktop version of MATLAB to get deeper historical insight. Visit https://www.mathworks.com/hardware-support/thingspeak.html to learn more.

Libraries and examples for Particle devices can be found here: https://github.com/mathworks/thingspeak-particle

Installation

In the Arduino IDE, choose Sketch/Include Library/Manage Libraries. Click the ThingSpeak Library from the list, and

Contributors 2

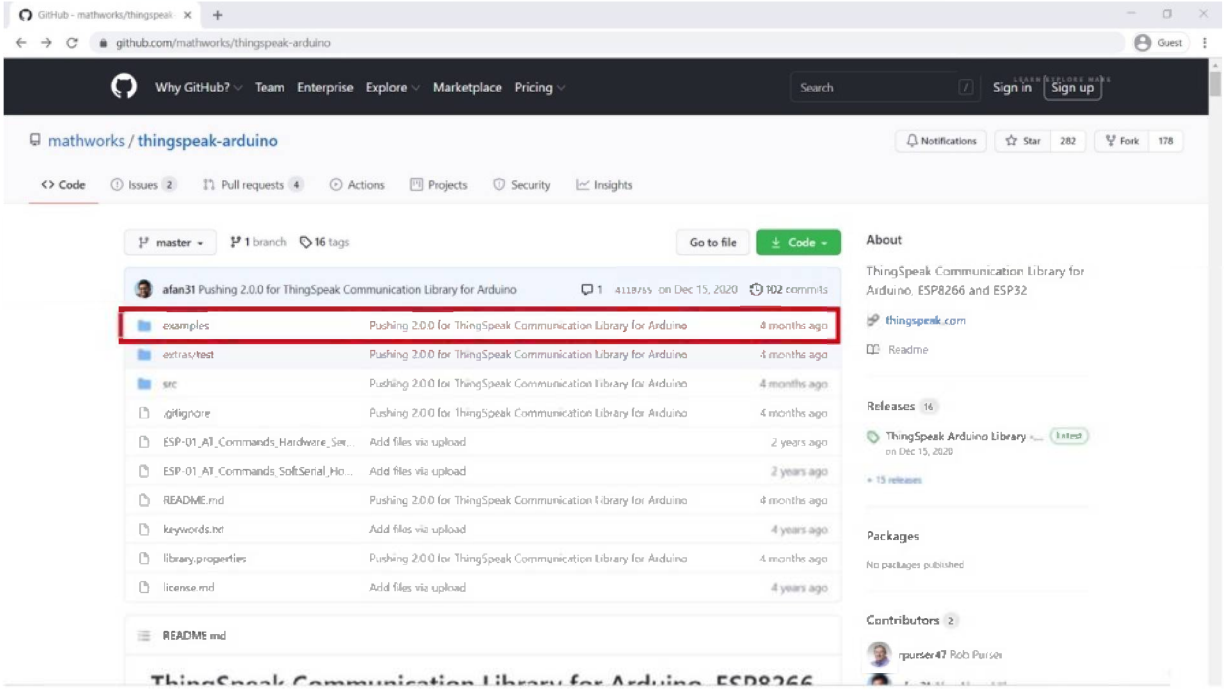
rpurser47 Rob Purser

afan31 Afan Ahmad Khan

Languages

● C++ 100.0%

The ESP32 can be easily connected to Thing Speak using its very own Thing Speak. h library. This library enables the ESP32 and other compatible hardware to read and write data from Thing Speak>



afan31 Pushing 2.0.0 for ThingSpeak Communication Library for Arduino 1 4118755 on Dec 15, 2020 102 commits

examples	Pushing 2.0.0 for ThingSpeak Communication Library for Arduino	4 months ago
extras/test	Pushing 2.0.0 for ThingSpeak Communication Library for Arduino	4 months ago
src	Pushing 2.0.0 for ThingSpeak Communication Library for Arduino	4 months ago
.gitignore	Pushing 2.0.0 for ThingSpeak Communication Library for Arduino	4 months ago
ESP-01_AT_Commands_Hardware_Ser...	Add files via upload	2 years ago
ESP-01_AT_Commands_SoftSerial_Ho...	Add files via upload	2 years ago
README.md	Pushing 2.0.0 for ThingSpeak Communication Library for Arduino	4 months ago
keywords.txt	Add files via upload	4 years ago
library.properties	Pushing 2.0.0 for ThingSpeak Communication Library for Arduino	4 months ago
license.md	Add files via upload	4 years ago

README.md

ThingSpeak Communication Library for Arduino, ESP8266

About

ThingSpeak Communication Library for Arduino, ESP8266 and ESP32

thingspeak.com

Readme

Releases 16

ThingSpeak Arduino Library ... Latest
on Dec 15, 2020

+ 15 releases

Packages

No packages published

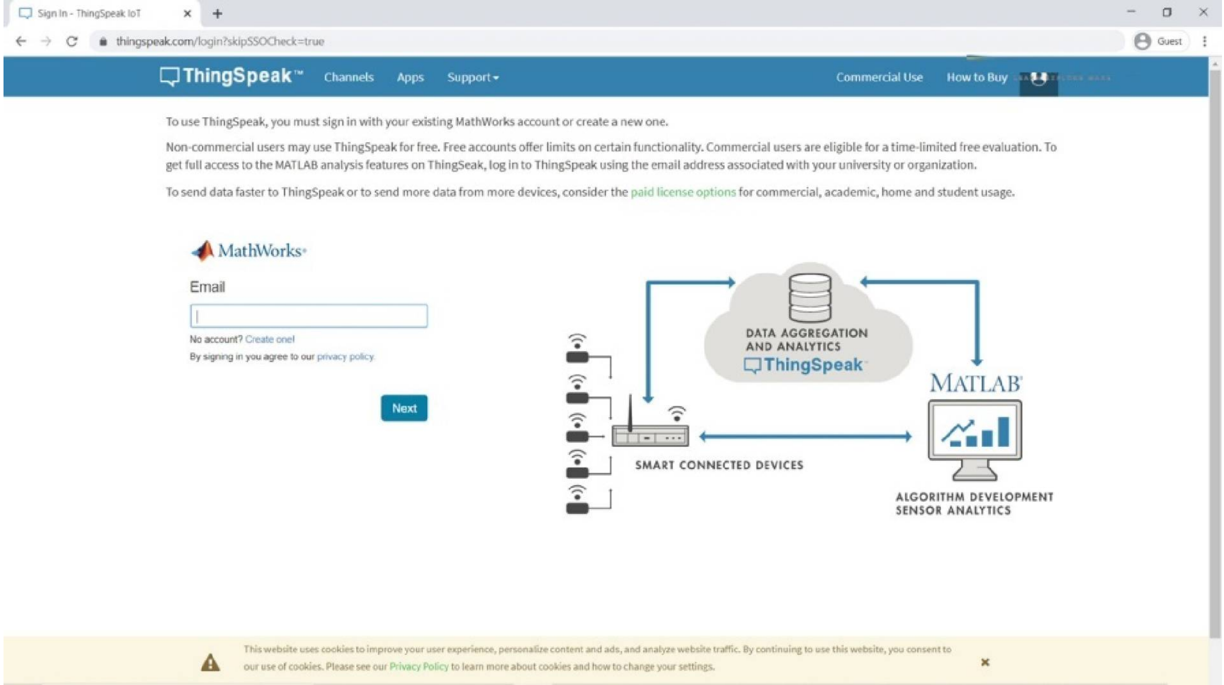
Contributors 2

rpurser47 Rob Purser

Hardware specific examples related to ESP32, ESP8266 and Arduino are given in this library. These examples can be used according to personal preferences to develop an IoT system.

GETTING STARTED WITH THINGSPEAK SETUP

Let's go to the Thing Speak website now. The link for the same is given in the resources of this project. You will now arrive at the Thing Speak homepage click on get started for free.



To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the paid license options for commercial, academic, home and student usage.

MathWorks

Email

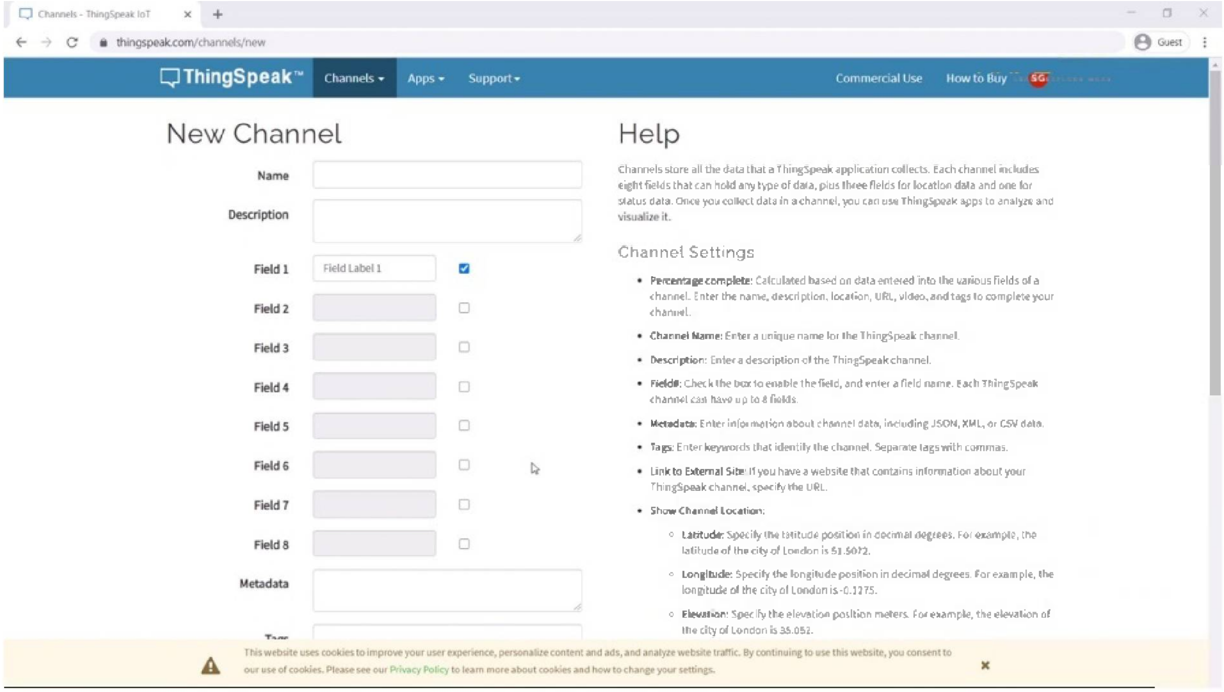
No account? Create one!

By signing in you agree to our privacy policy.

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Now you will be asked for the email. If you already have a Math works account, sign in with it or click on create one, fill in all the necessary details and create your Thing Speak account. Note down the email address and password. You will need it later to log into your account.

After you have logged in, you'll see a blank window with some options on the screen. One of the important options that you need to know about is the new channel option. A channel is where you send your data to store.



Field 1: Field Label 1 ☑

Field 2 ☐ Field 3 ☐ Field 4 ☐ Field 5 ☐ Field 6 ☐ Field 7 ☐ Field 8 ☐

Help

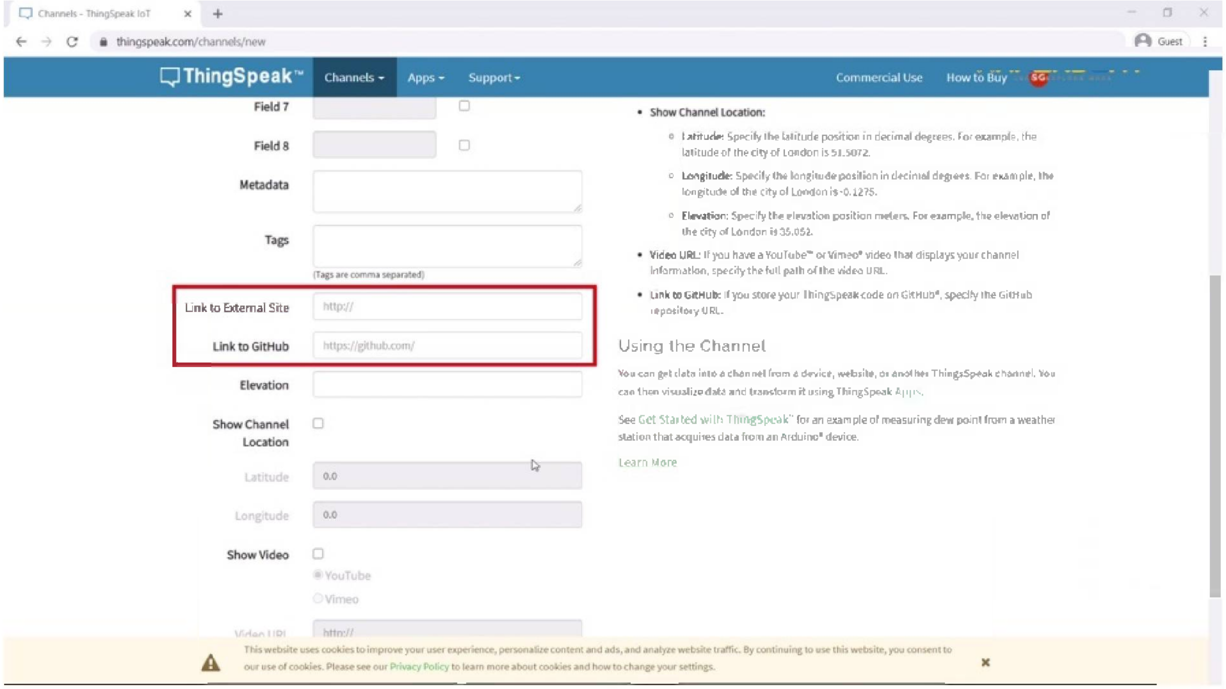
Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Now click on the new channel button. Here you will see various options that can be filled. Each channel includes a name and description you can personalize according to you. There are eight fields available where data can be pushed.



Field 7 ☐ Field 8 ☐

Link to External Site: http://

Link to GitHub: https://github.com/

Show Channel Location ☐

Latitude: 0.0

Longitude: 0.0

Show Video ☐

● YouTube ○ Vimeo

Video URL: http://

- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- **Video URL:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- **Link to GitHub:** If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

Using the Channel

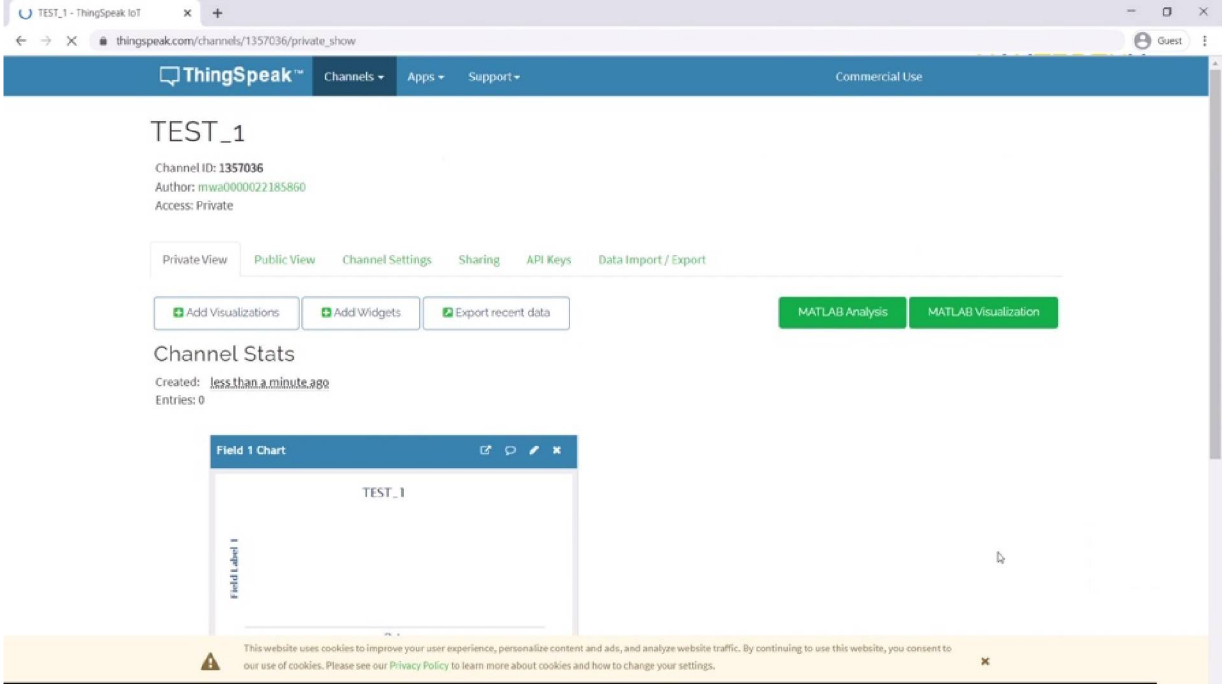
You can get data into a channel from a device, website, or another ThingsSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

See Get Started with ThingSpeak™ for an example of measuring dew point from a weather station that acquires data from an Arduino® device.

Learn More

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Additionally, more options are available, including YouTube URL or GitHub URL, which can be specified. Now under a name to your channel select field one and click on the same channel option.



TEST_1

Channel ID: 1357036
Author: mwa0000022185860
Access: Private

Channel Stats

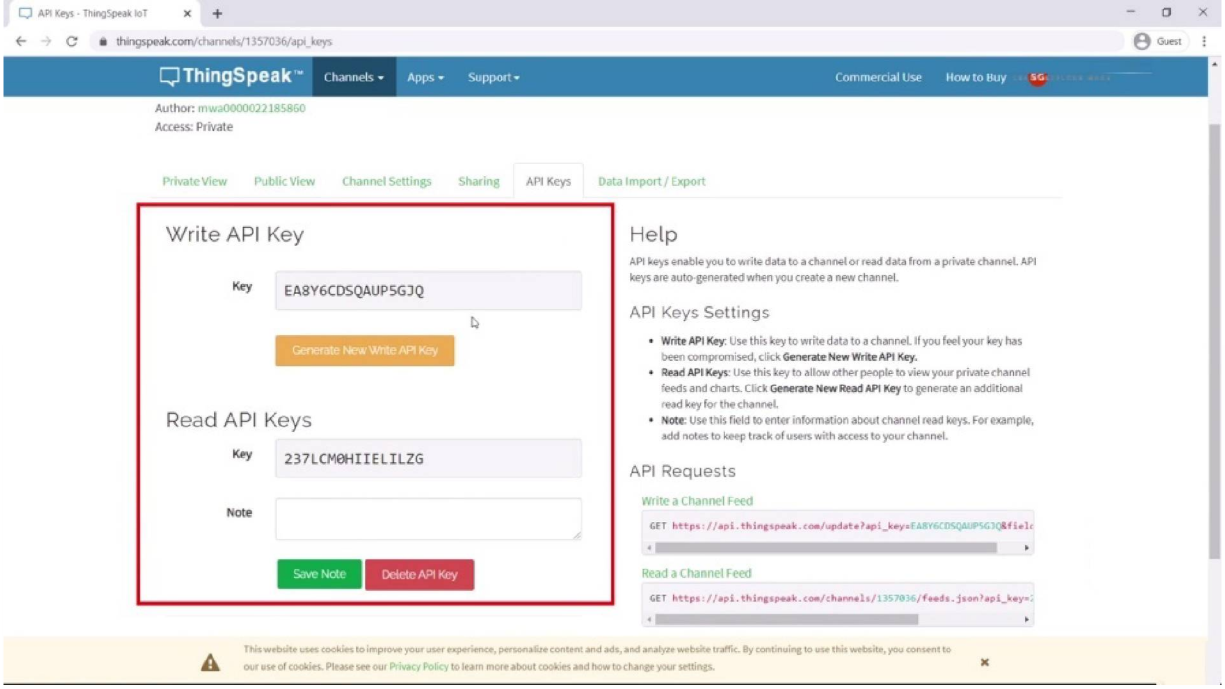
Created: less than a minute ago
Entries: 0

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

You have now successfully created a channel. To read or write data from this channel. You will need API keys. Now you may be wondering what is an API and what are these API keys.

So an API is an acronym for application program interface. It is a software intermediary which allows two applications to talk to each other. Each time we use an app like Facebook to send messages or to check weather conditions on your smartphone, you are using an API.

To understand more about how an API works, let's take a familiar example. Consider yourself sitting at a restaurant with the menu of choices to order from. The kitchen is a part of a system that will prepare your order. However, the critical link is the waiter who communicates your order to the kitchen and brings the ordered food to your table. The waiter is like an API that takes your request to the system and responds back to you. In our example. The request is food. Now to read and write data from your These API keys are sixteen digit codes that provide with appropriate read and right permissions to your channel. You will have separate, read and write API keys.



Author: mwa0000022185860
Access: Private

Write API Key

Key: EA8Y6CDSQAUP5GJQ

Read API Keys

Key: 237LCM0HIIELILZG

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

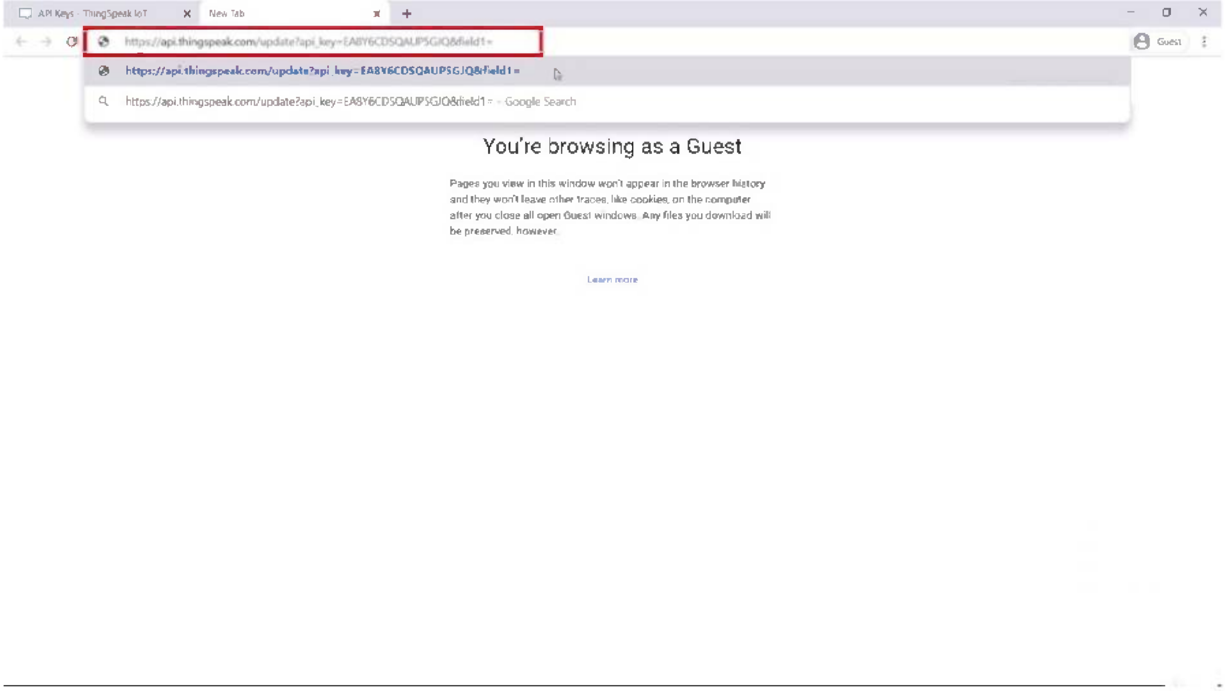
```
GET https://api.thingspeak.com/update?api_key=EA8Y6CDSQAUP5GJQ&fielc
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1357036/feeds.json?api_key=
```

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Click on API keys here you will find the read and write API keys. Let's now write data to a channel, copy the write a channel from here and paste it in your browser.



https://api.thingspeak.com/update?api_key=EA8Y6CDSQAUP5GJQ&field1=

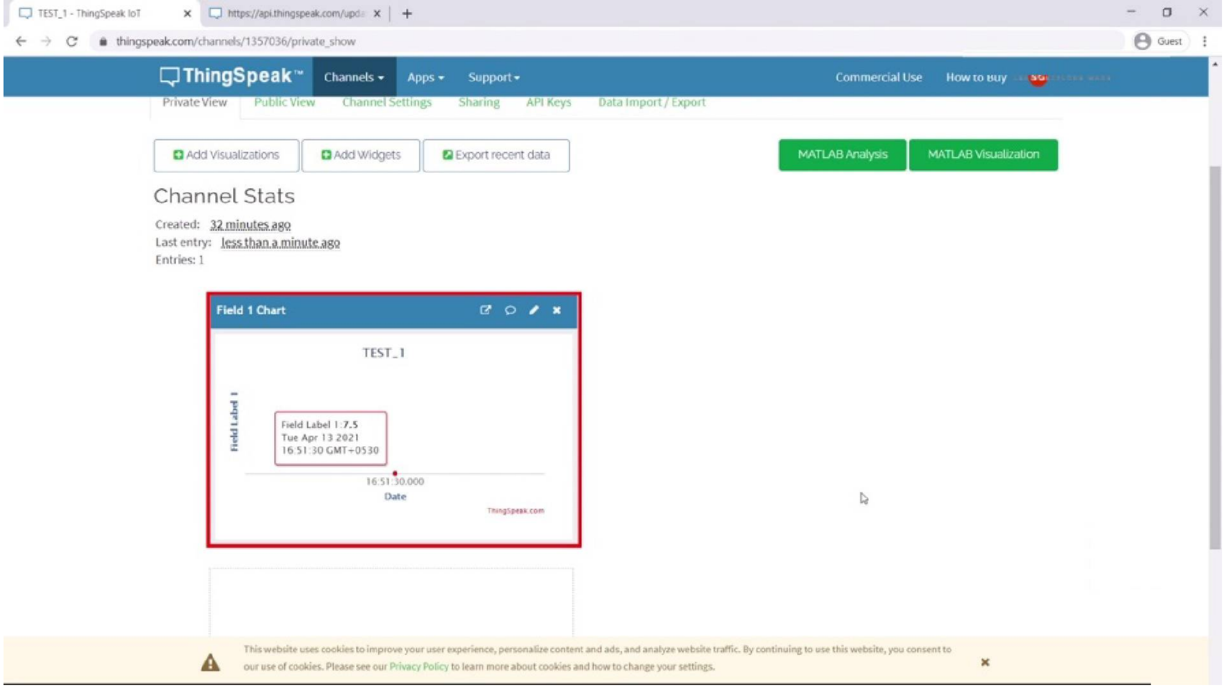
https://api.thingspeak.com/update?api_key=EA8Y6CDSQAUP5GJQ&field1=

https://api.thingspeak.com/update?api_key=EA8Y6CDSQAUP5GJQ&field1= - Google Search

You're browsing as a Guest

Pages you view in this window won't appear in the browser history and they won't leave other traces, like cookies, on the computer after you close all open Guest windows. Any files you download will be preserved, however.

You can change the data to be posted. Now press enter this will post your data to field one of your channel Here you can see the data posted.



Channel Stats

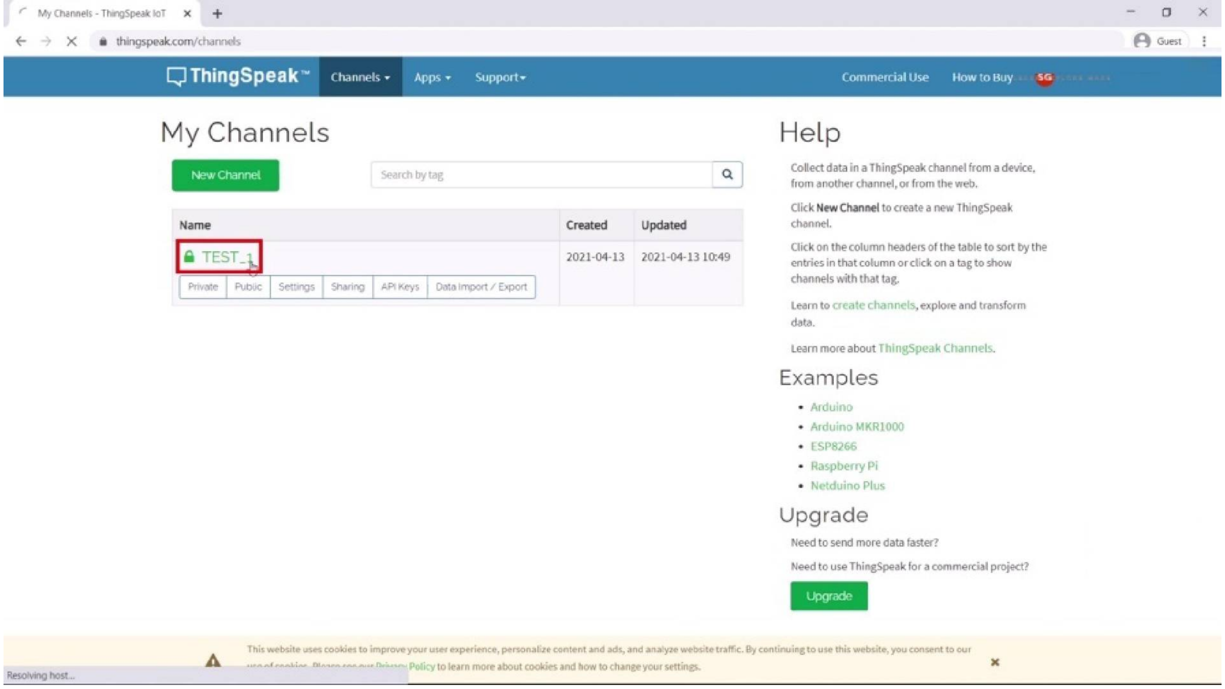
Created: 32 minutes ago
Last entry: less than a minute ago
Entries: 1

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

To increase the security aspect of your system. You can generate new API keys.

DASHBOARD OVERVIEW

we will cover the following topics. Dashboard overview, features of Thing Speak. Now that you have setup your Thing Speak account lets, get to know more about Thing Speak the dashboard and other features of Thing Speak.



My Channels

Name	Created	Updated
TEST_1	2021-04-13	2021-04-13 10:49

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

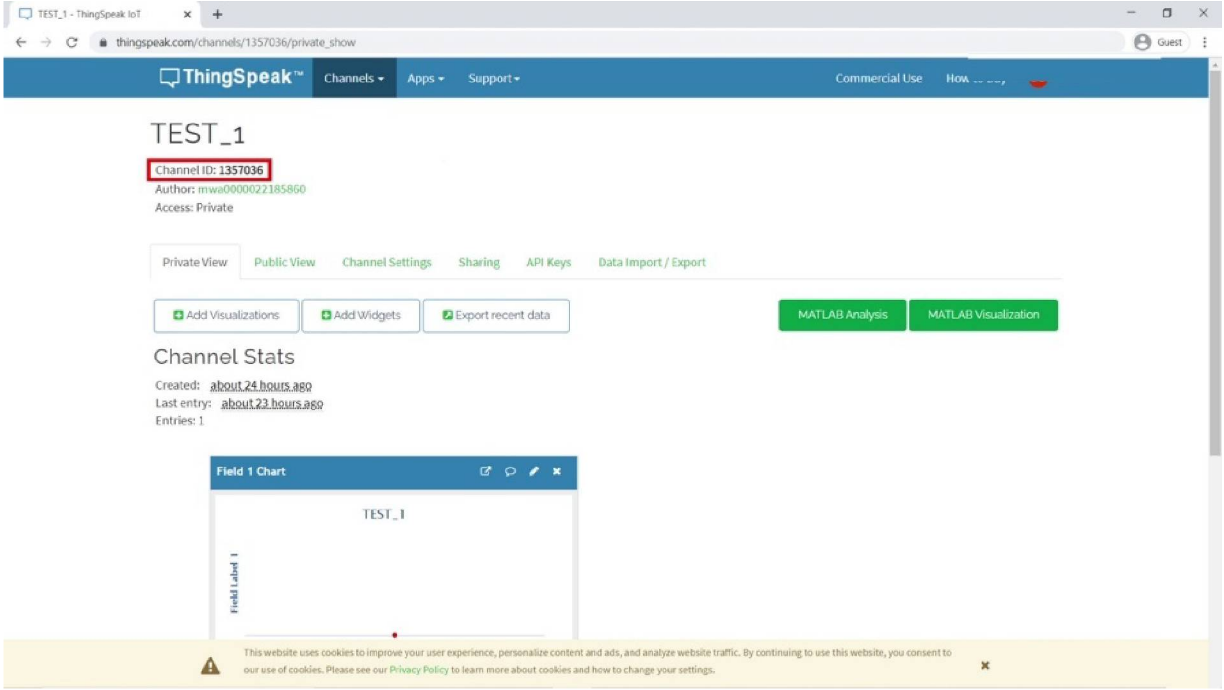
Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

To move to the dashboard of a particular channel. Click on the channel you want to check. Lets select the Test Channel we created in the previous project you will see the dashboard for the particular channel once you click on it.



TEST_1

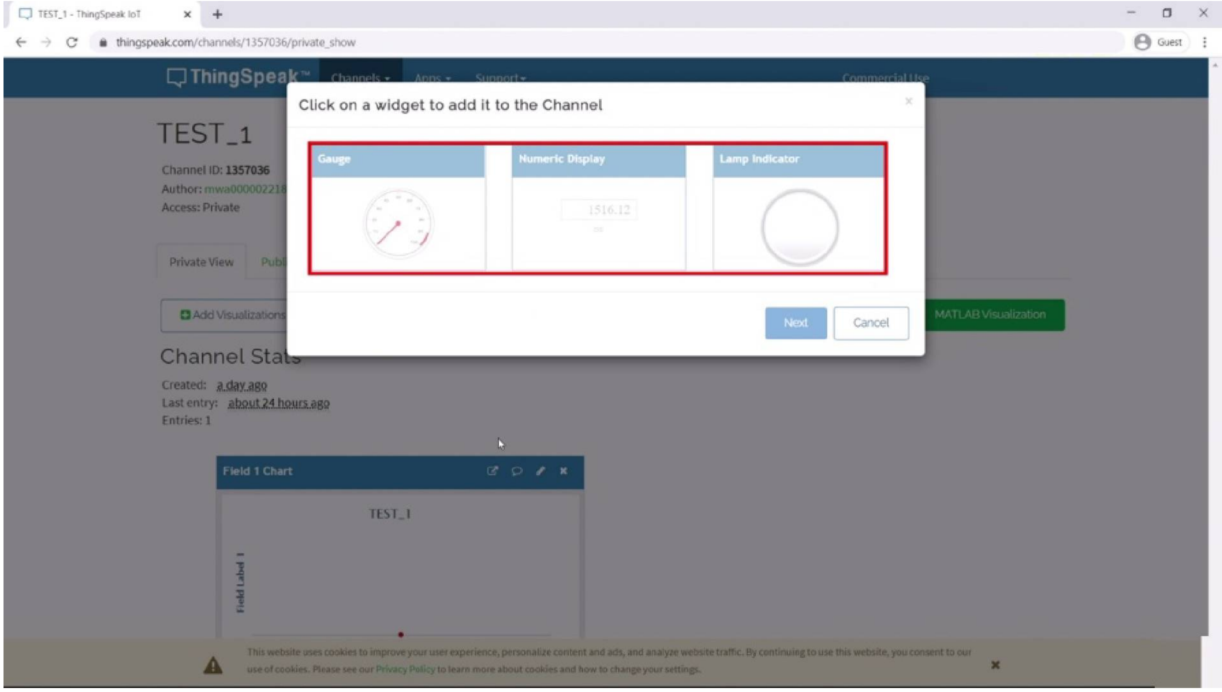
Channel ID: 1357036
Author: mwa0000022185860
Access: Private

Channel Stats

Created: about 24 hours ago
Last entry: about 23 hours ago
Entries: 1

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

At the top, it displays the channel name you had entered at first. Below it the channel ID is mentioned. Remember how in Cayenne we had a different client ID for every connected device. Similarly, each channel you make has a different channel ID. This channel ID is used to read and write data from the particular channel. Below the channel ID your author code. Each time someone signs up for math works - Thing Speak, they are allocated an author code. This code helps matlab and Thing Speak track all your contributions to the community. The channel status is mentioned below whether it is a private channel or a public channel. The private channel can only be accessed by you while on the other hand, data from a public channel is available in the public domain and anyone can see and use it. If you scroll below, you can see all the details regarding the channel, such as when the channel was created, when was the last time data was sent to the channel and how many total entries have been made to the channel. By using the add visualization button, you can add a Matlab visualization to your dashboard. The add widget button helps in adding widgets to your dashboard.

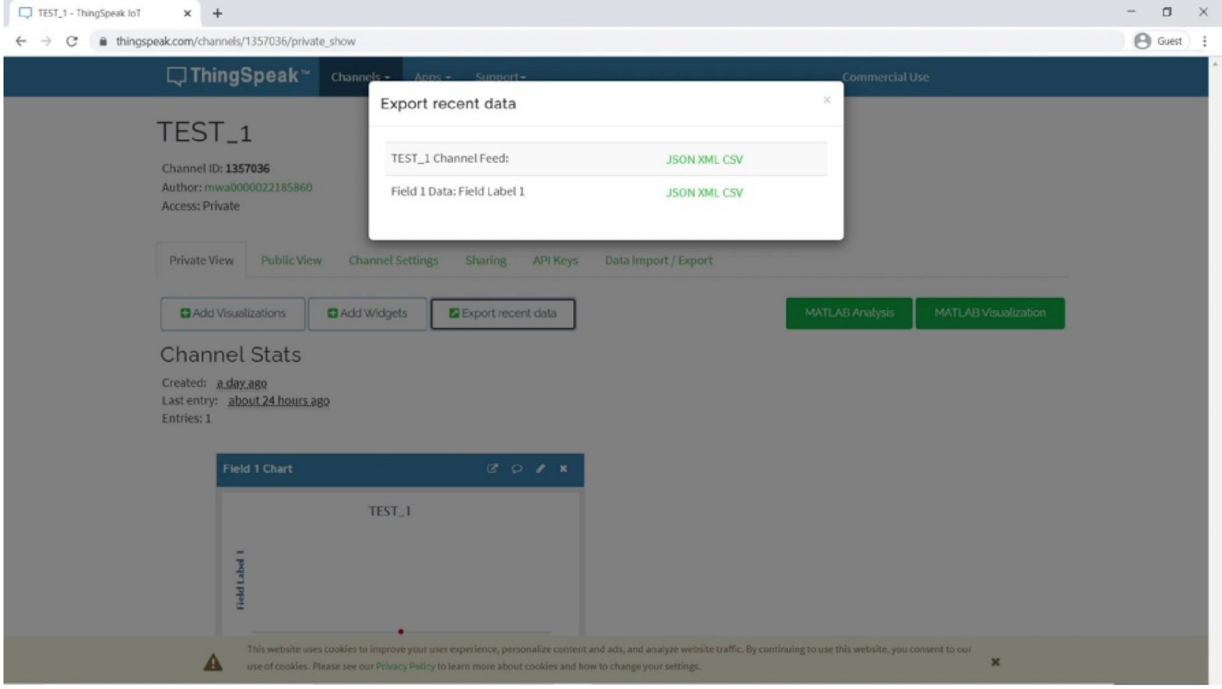


Channel ID: 1357036
Author: mwa000002218
Access: Private

Created: a day ago
Last entry: about 24 hours ago
Entries: 1

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

It has three options to choose from a gauge, a numeric display and a lamp indicator. All these widgets can be used according to personal preference and purpose.



Export recent data

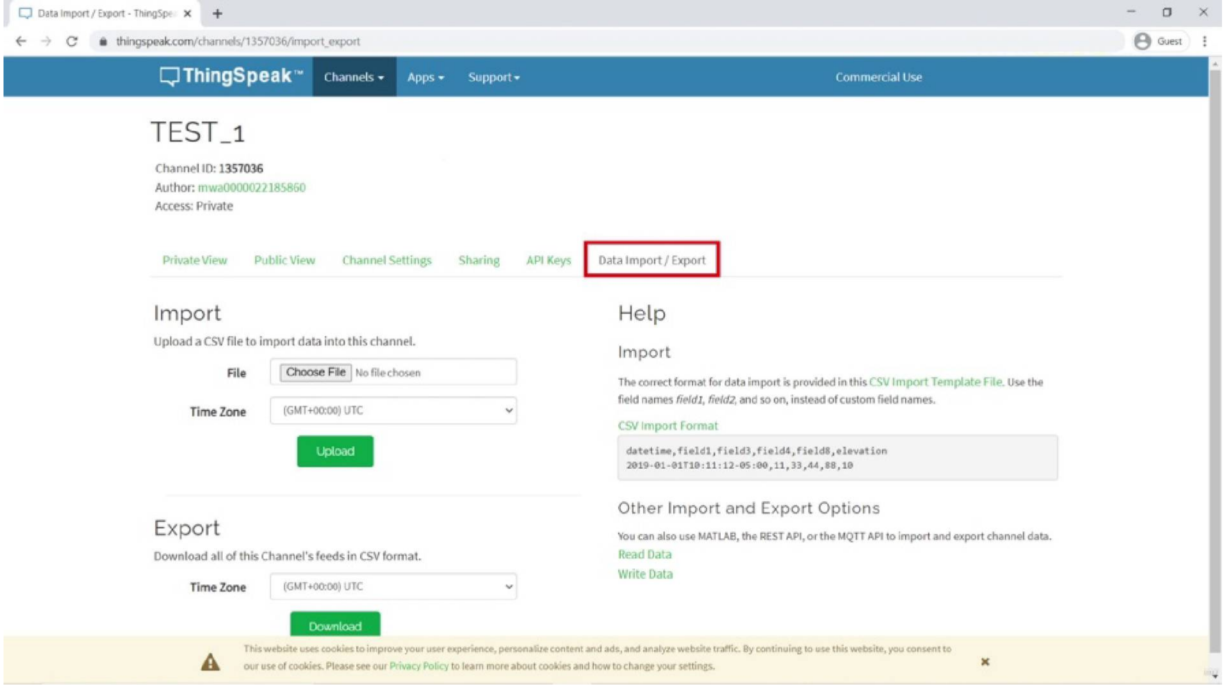
TEST_1 Channel Feed:	JSON XML CSV
Field 1 Data: Field Label 1	JSON XML CSV

Channel ID: 1357036
Author: mwa0000022185860
Access: Private

Created: a day ago
Last entry: about 24 hours ago
Entries: 1

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Just like Cayenne, you can export all the data from a particular channel or a particular field in the channel. Thing Speak offers us to export this data in three formats JSON, XML and CSV format. Now let's see the other options of the dashboard. As our channel is private, we can only view it in the private view and not in the public view. However, you can share your channel data with certain users or make it public for everyone else to see. Using the sharing feature here. You can update all the channel settings here. All the settings that were available for you when you created the channel are available here. You will have all your information regarding the API keys here.



TEST_1

Channel ID: 1357036
Author: mwa0000022185860
Access: Private

Import

Upload a CSV file to import data into this channel.

File: No file chosen

Time Zone: (GMT+00:00) UTC

Export

Download all of this Channel's feeds in CSV format.

Time Zone: (GMT+00:00) UTC

Help

Import

The correct format for data import is provided in this CSV Import Template File. Use the field names *field1*, *field2*, and so on, instead of custom field names.

CSV Import Format

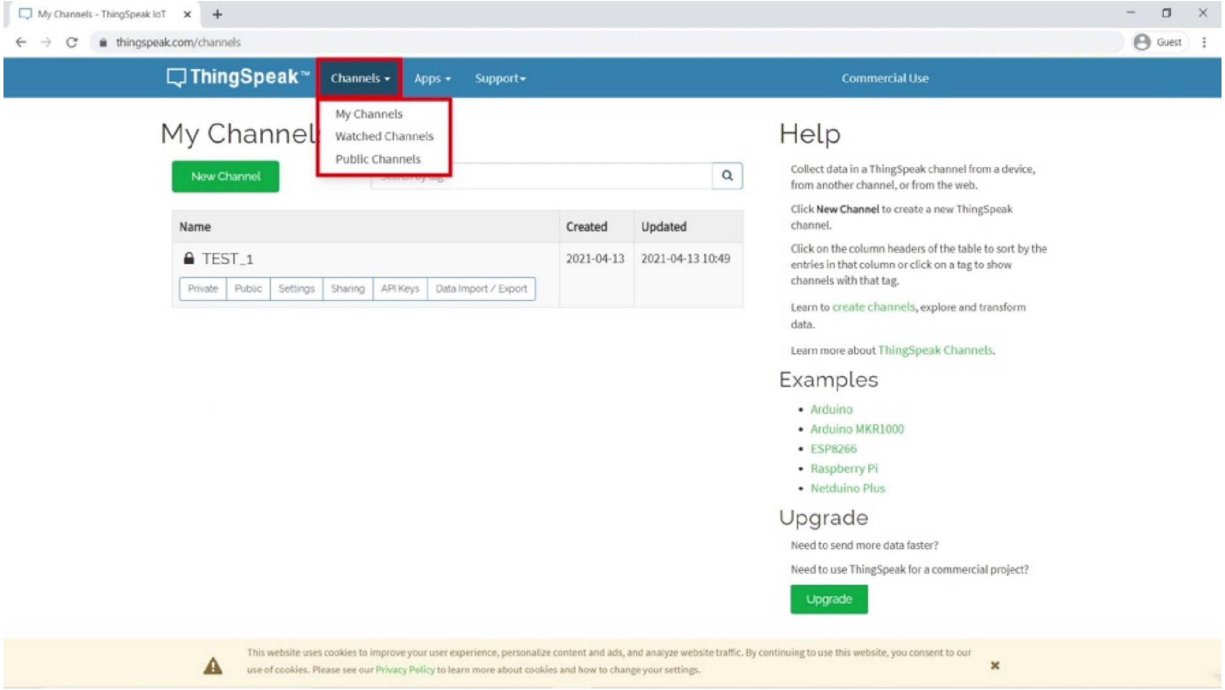
```
datetime,field1,field3,field4,field8,elevation
2019-01-01T10:11:12-05:00,11,33,44,88,10
```

Other Import and Export Options

You can also use MATLAB, the REST API, or the MQTT API to import and export channel data.

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Here you have options to import and export data from the particular channel at a specified time. You can mention it in the options. Now let's check some other features of Thing Speak.



My Channels

Name	Created	Updated
TEST_1 Private · Public · Settings · Sharing · API Keys · Data Import / Export	2021-04-13	2021-04-13 10:49

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

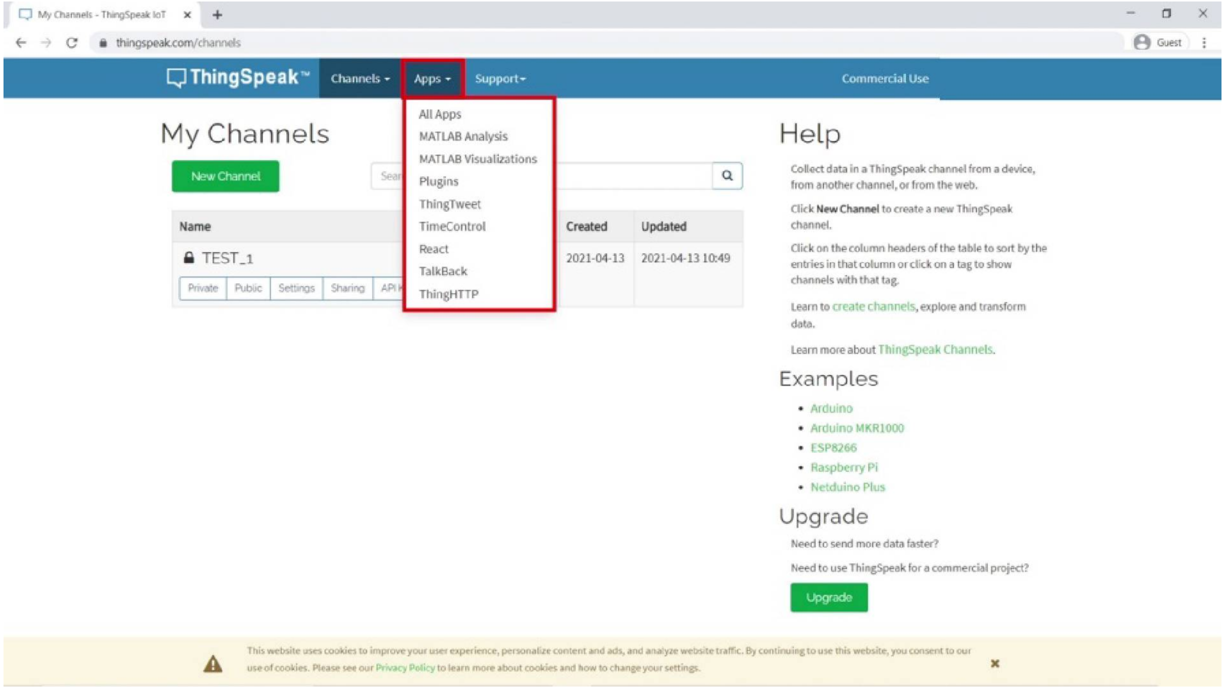
Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Here You can check all the channels that you have created. The free Thing Speak account allows you to have four channels at a time. You need to upgrade to a higher license to get more channels available for you. The channel options also show all the channels you have watched and all the public channels available on Thing Speak.



My Channels

Name	Created	Updated
TEST_1 Private · Public · Settings · Sharing · API	2021-04-13	2021-04-13 10:49

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

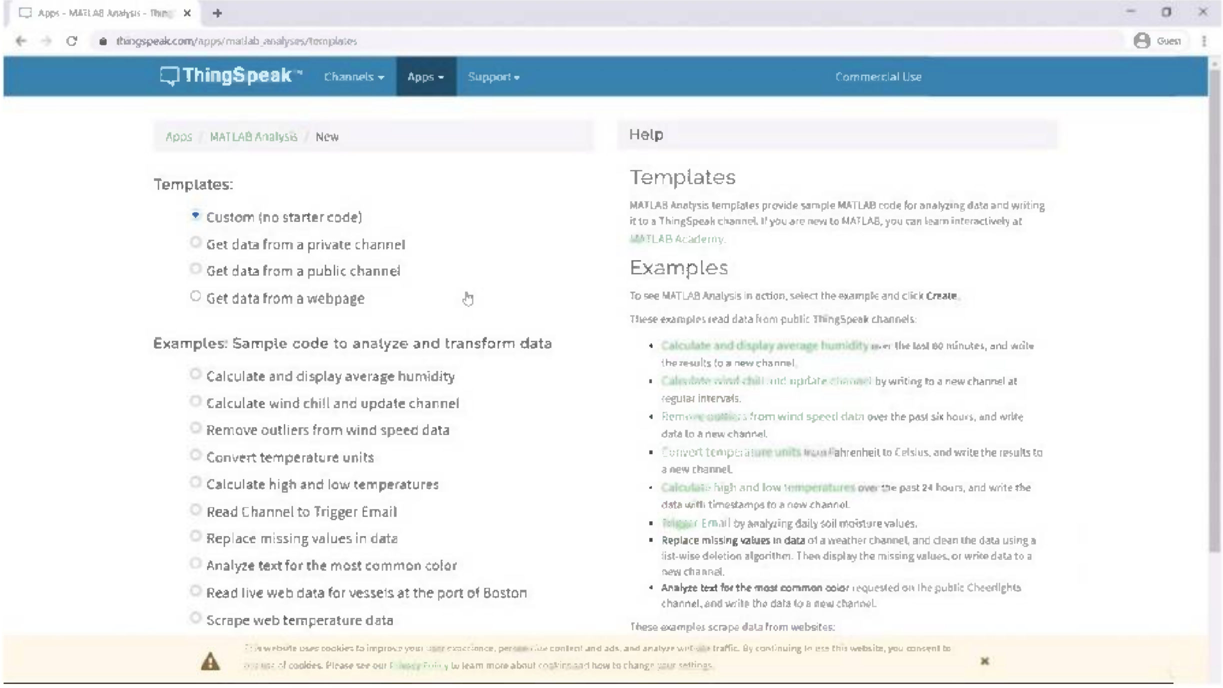
Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Thing Speak has provided us with some apps. These apps can be used to analyze and transform data or trigger an action. These apps include Matlab analysis and Matlab visualization.


Apps - MATLAB Analysis - Thing
thingspeak.com/apps/matlab_analyses/templates
Guest
ThingSpeak™
Channels ▾
Apps ▾
Support ▾
Commercial Use
Apps / MATLAB Analysis / New

Templates:

- ● Custom (no starter code)
- ○ Get data from a private channel
- ○ Get data from a public channel
- ○ Get data from a webpage

Examples: Sample code to analyze and transform data

- ○ Calculate and display average humidity
- ○ Calculate wind chill and update channel
- ○ Remove outliers from wind speed data
- ○ Convert temperature units
- ○ Calculate high and low temperatures
- ○ Read Channel to Trigger Email
- ○ Replace missing values in data
- ○ Analyze text for the most common color
- ○ Read live web data for vessels at the port of Boston
- ○ Scrape web temperature data

Help

Templates

MATLAB Analysis templates provide sample MATLAB code for analyzing data and writing it to a ThingSpeak channel. If you are new to MATLAB, you can learn interactively at MATLAB Academy.

Examples

To see MATLAB Analysis in action, select the example and click **Create**.

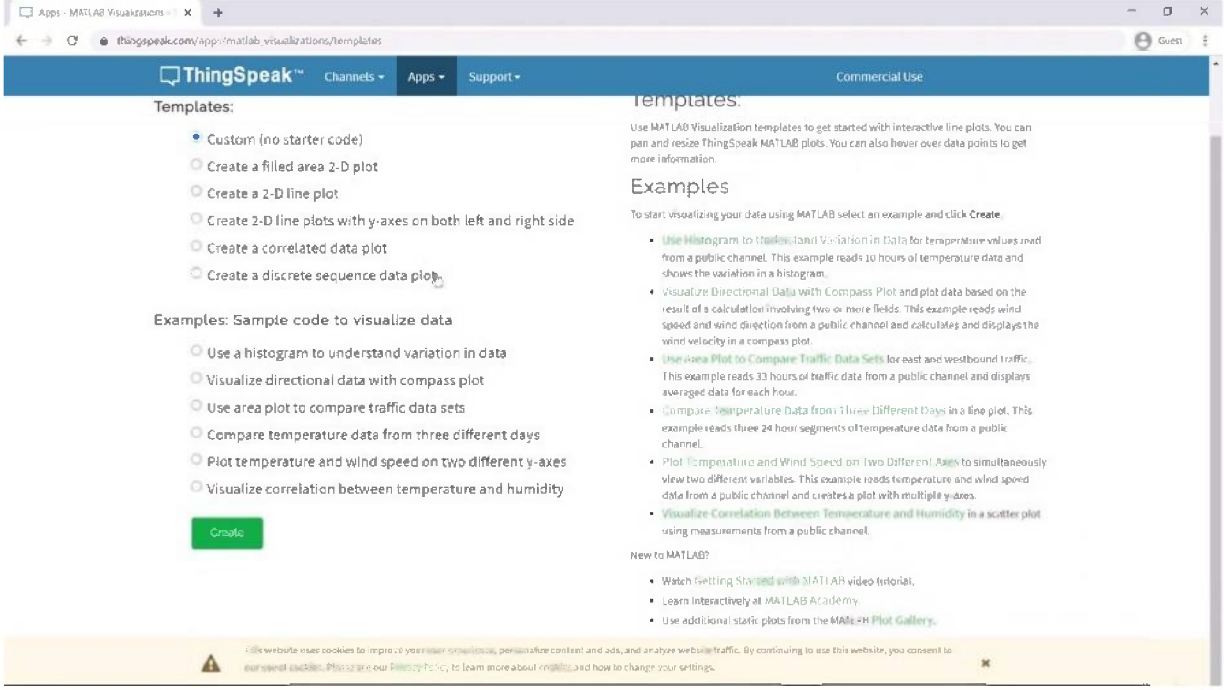
These examples read data from public ThingSpeak channels:

- Calculate and display average humidity over the last 60 minutes, and write the results to a new channel.
- Calculate wind chill and update channel by writing to a new channel at regular intervals.
- Remove outliers from wind speed data over the past six hours, and write data to a new channel.
- Convert temperature units from Fahrenheit to Celsius, and write the results to a new channel.
- Calculate high and low temperatures over the past 24 hours, and write the data with timestamps to a new channel.
- Trigger Email by analyzing daily soil moisture values.
- **Replace missing values in data** of a weather channel, and clean the data using a list-wise deletion algorithm. Then display the missing values, or write data to a new channel.
- **Analyze text for the most common color** requested on the public Cheerlights channel, and write the data to a new channel.

These examples scrape data from websites:

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Matlab analysis can be used to explore data, convert data to different units and build data models. Sample codes for the same have been provided here.


Apps - MATLAB Visualizations
thingspeak.com/apps/matlab_visualizations/templates
Guest
ThingSpeak™
Channels ▾
Apps ▾
Support ▾
Commercial Use

Templates:

- ● Custom (no starter code)
- ○ Create a filled area 2-D plot
- ○ Create a 2-D line plot
- ○ Create 2-D line plots with y-axes on both left and right side
- ○ Create a correlated data plot
- ○ Create a discrete sequence data plot

Examples: Sample code to visualize data

- ○ Use a histogram to understand variation in data
- ○ Visualize directional data with compass plot
- ○ Use area plot to compare traffic data sets
- ○ Compare temperature data from three different days
- ○ Plot temperature and wind speed on two different y-axes
- ○ Visualize correlation between temperature and humidity

Create

Templates:

Use MATLAB Visualization templates to get started with interactive line plots. You can pan and resize ThingSpeak MATLAB plots. You can also hover over data points to get more information.

Examples

To start visualizing your data using MATLAB select an example and click **Create**.

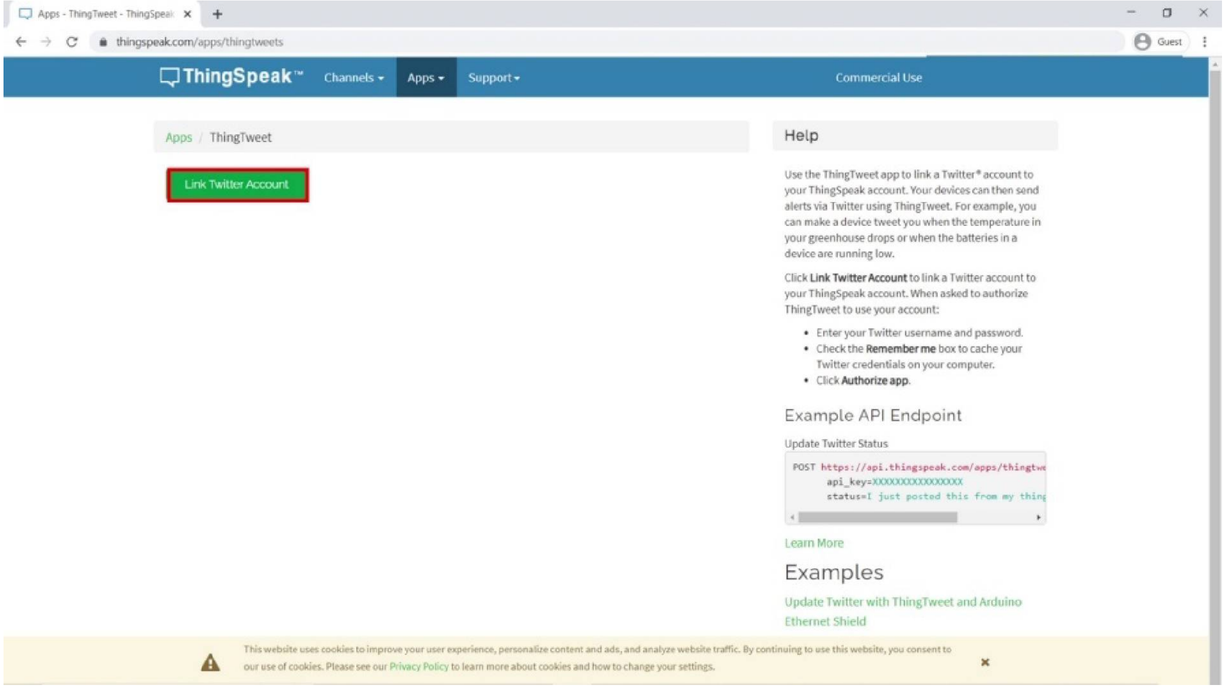
- Use Histogram to Understand Variation in Data for temperature values read from a public channel. This example reads 10 hours of temperature data and shows the variation in a histogram.
- Visualize Directional Data with Compass Plot and plot data based on the result of a calculation involving two or more fields. This example reads wind speed and wind direction from a public channel and calculates and displays the wind velocity in a compass plot.
- Use Area Plot to Compare Traffic Data Sets for east and westbound traffic. This example reads 33 hours of traffic data from a public channel and displays averaged data for each hour.
- Compare Temperature Data from Three Different Days in a line plot. This example reads three 24 hour segments of temperature data from a public channel.
- Plot Temperature and Wind Speed on Two Different Axes to simultaneously view two different variables. This example reads temperature and wind speed data from a public channel and creates a plot with multiple y-axes.
- Visualize Correlation Between Temperature and Humidity in a scatter plot using measurements from a public channel.

New to MATLAB?

- Watch Getting Started with MATLAB video tutorial.
- Learn interactively at MATLAB Academy.
- Use additional static plots from the MATLAB Plot Gallery.

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

The Matlab visualizations app can be used to plot and visualize data from a particular channel. Various templates and sample codes to explore and visualize the data are given here.



Help

Use the ThingTweet app to link a Twitter® account to your ThingSpeak account. Your devices can then send alerts via Twitter using ThingTweet. For example, you can make a device tweet you when the temperature in your greenhouse drops or when the batteries in a device are running low.

Click **Link Twitter Account** to link a Twitter account to your ThingSpeak account. When asked to authorize ThingTweet to use your account:

- Enter your Twitter username and password.
- Check the **Remember me** box to cache your Twitter credentials on your computer.
- Click **Authorize app**.

Example API Endpoint

Update Twitter Status

```
POST https://api.thingspeak.com/apps/thingtwe
     api_key=XXXXXXXXXXXXXXXX
     status=I just posted this from my thing
```

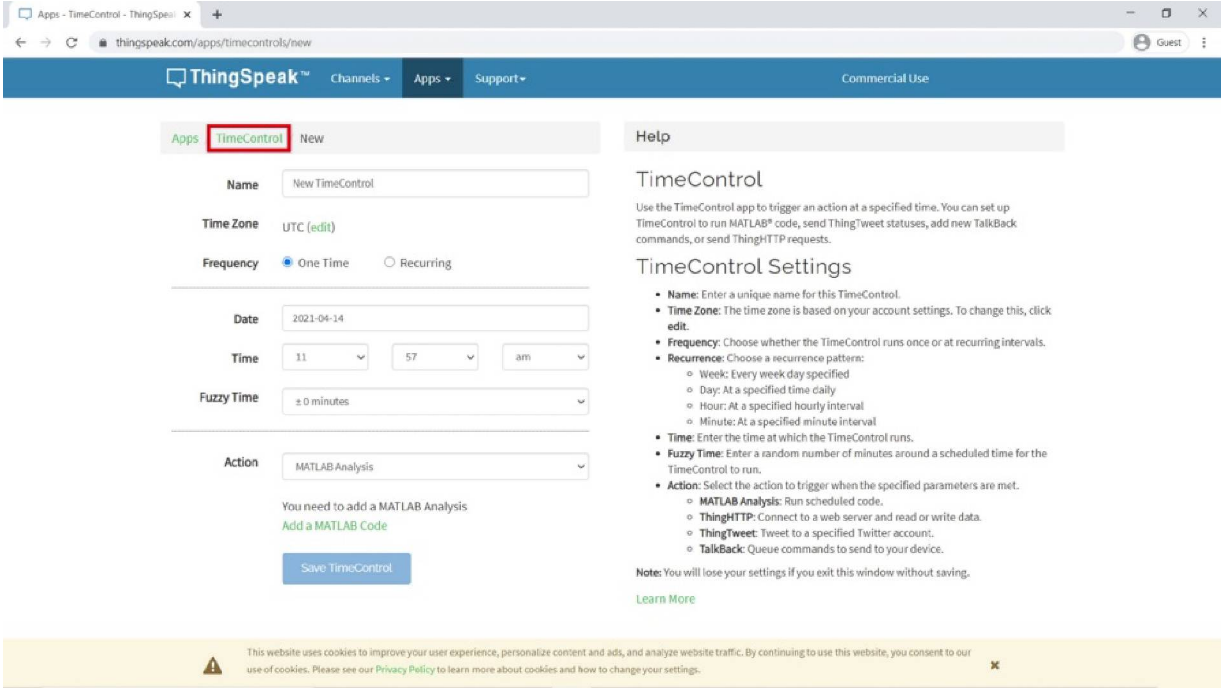
Learn More

Examples

Update Twitter with ThingTweet and Arduino Ethernet Shield

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

One of the interesting applications provided by Thing Speak is the ThingTweet. First, you need to link your Twitter account with your Thing Speak account. You can now use this app to send alerts to Twitter.



Name: New TimeControl

Time Zone: UTC (edit)

Frequency: ● One Time ○ Recurring

Date: 2021-04-14

Time: 11 57 am

Fuzzy Time: ± 0 minutes

Action: MATLAB Analysis

You need to add a MATLAB Analysis

Add a MATLAB Code

Help

TimeControl

Use the TimeControl app to trigger an action at a specified time. You can set up TimeControl to run MATLAB® code, send ThingTweet statuses, add new TalkBack commands, or send ThingHTTP requests.

TimeControl Settings

- **Name**: Enter a unique name for this TimeControl.
- **Time Zone**: The time zone is based on your account settings. To change this, click **edit**.
- **Frequency**: Choose whether the TimeControl runs once or at recurring intervals.
- **Recurrence**: Choose a recurrence pattern:
  - Week: Every week day specified
  - Day: At a specified time daily
  - Hour: At a specified hourly interval
  - Minute: At a specified minute interval
- **Time**: Enter the time at which the TimeControl runs.
- **Fuzzy Time**: Enter a random number of minutes around a scheduled time for the TimeControl to run.
- **Action**: Select the action to trigger when the specified parameters are met.
  - **MATLAB Analysis**: Run scheduled code.
  - **ThingHTTP**: Connect to a web server and read or write data.
  - **ThingTweet**: Tweet to a specified Twitter account.
  - **TalkBack**: Queue commands to send to your device.

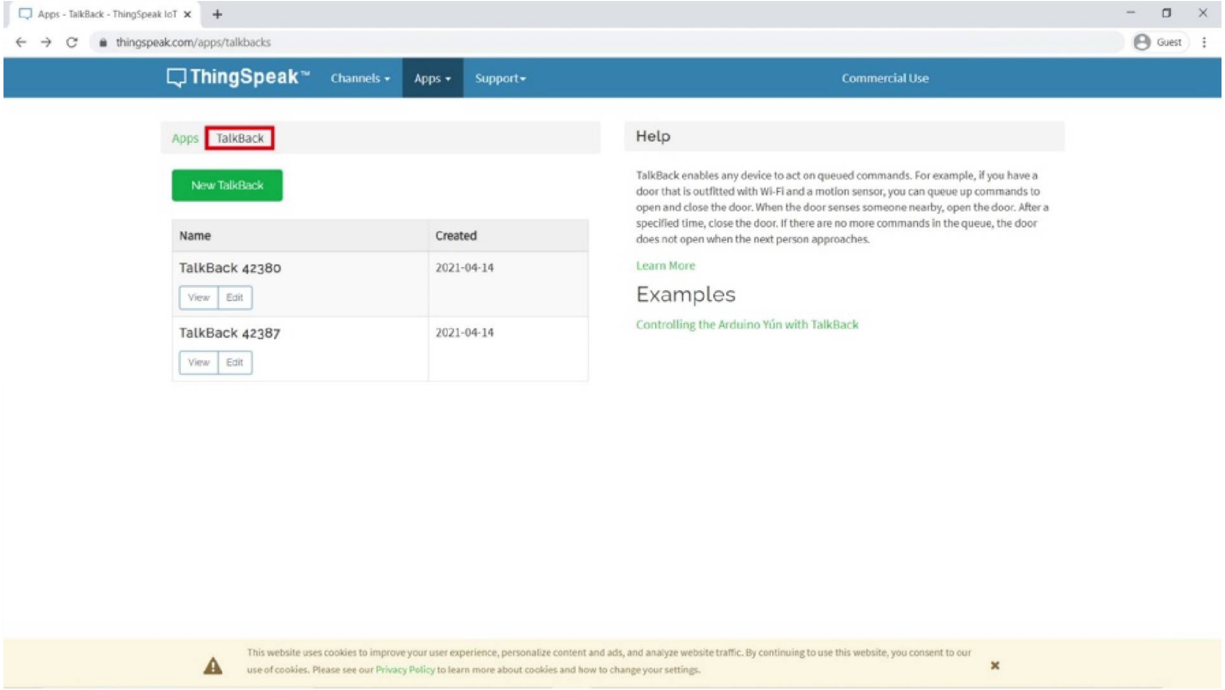
Note: You will lose your settings if you exit this window without saving.

Learn More

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

The time control application can be used to perform an action at a specific time or on a regular schedule. We can use this application in conjunction with the Thing Tweet app to send notifications accordingly.

The REACT application works hand in hand with thing tweet and Matlab analysis to perform certain actions, when channel data meets a certain condition. This totally explains the name react.



Name	Created
TalkBack 42380	2021-04-14
TalkBack 42387	2021-04-14

Help

TalkBack enables any device to act on queued commands. For example, if you have a door that is outfitted with Wi-Fi and a motion sensor, you can queue up commands to open and close the door. When the door senses someone nearby, open the door. After a specified time, close the door. If there are no more commands in the queue, the door does not open when the next person approaches.

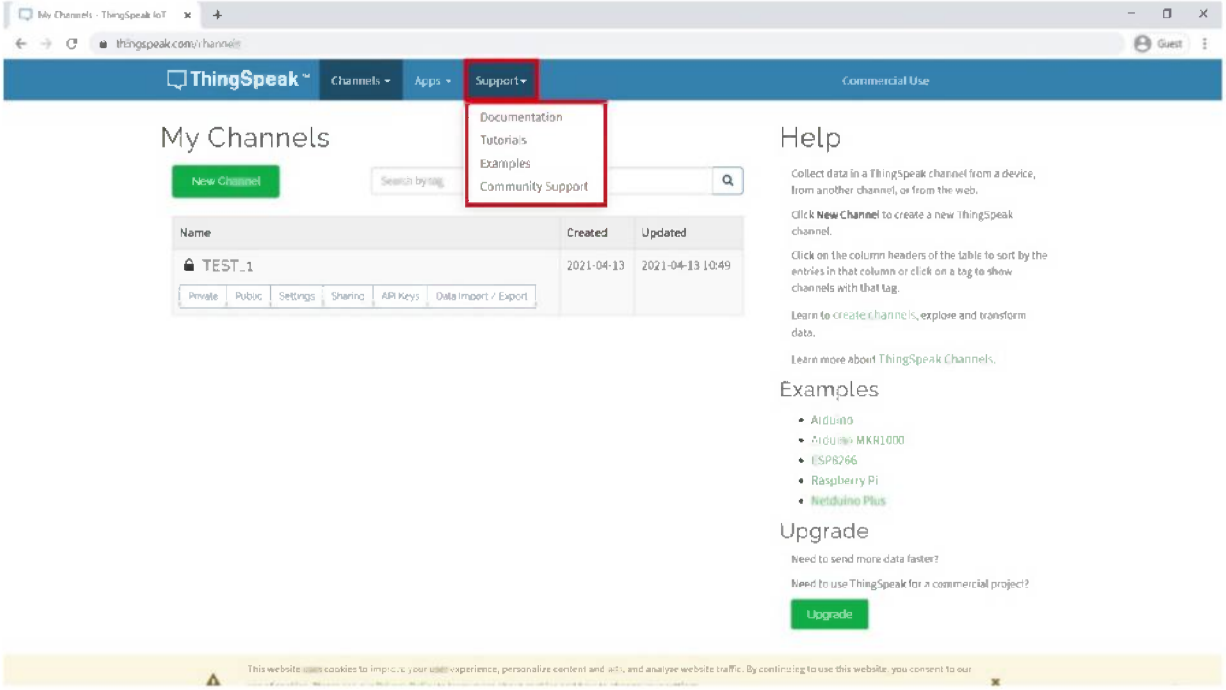
Learn More

Examples

Controlling the Arduino Yún with TalkBack

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

The talkback application provided by Thing Speak allows us to act on queued commands. This is similar to Okay Google or hey Siri in our smartphones. Thing Speak has a great support community.



Name	Created	Updated
TEST_1	2021-04-13	2021-04-13 10:49

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

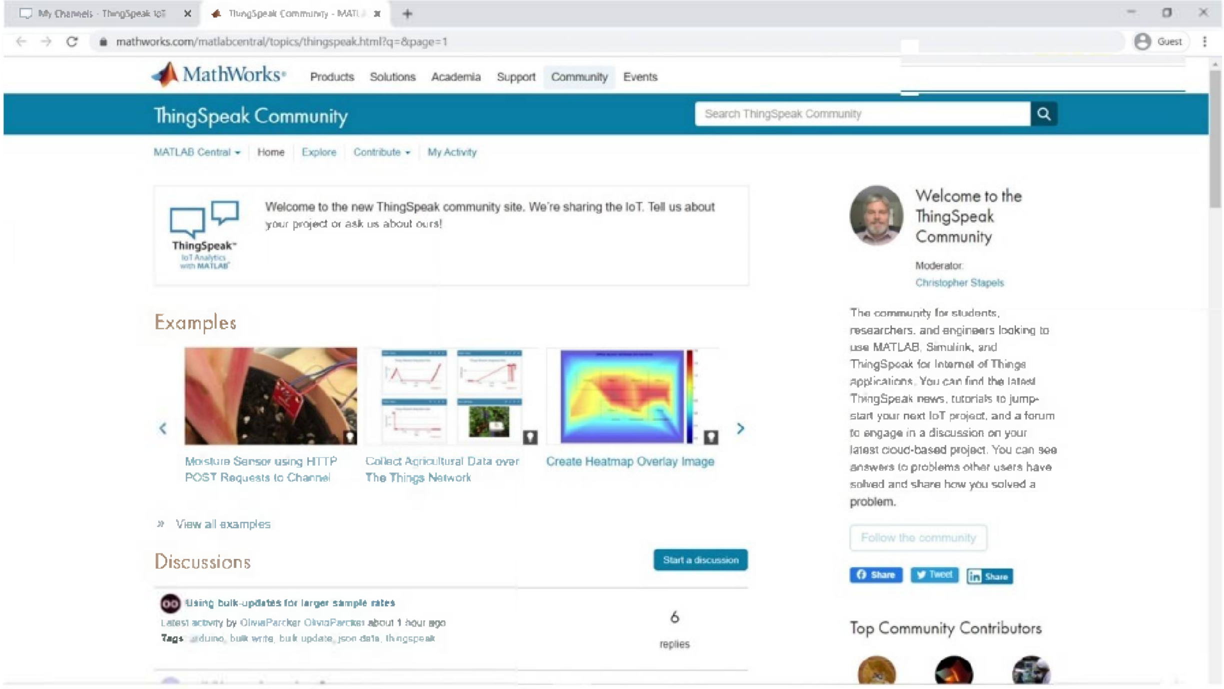
Need to send more data faster?

Need to use ThingSpeak for a commercial project?

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our

Thing Speak has provided us with documentation, tutorials and examples to get us started and get deep into Thing Speak. This helps us make full use of

Thing Speak features according to our preference, and develop a specific application.


My Channels - ThingSpeak IoT
ThingSpeak Community - MATL
mathworks.com/matlabcentral/topics/thingspeak.html?q=&page=1
Guest
MathWorks
Products
Solutions
Academia
Support
Community
Events
ThingSpeak Community
Search ThingSpeak Community
MATLAB Central
Home
Explore
Contribute
My Activity
ThingSpeak
IoT Analytics with MATLAB
Examples
Moisture Sensor using HTTP POST Requests to Channel
Collect Agricultural Data over The Things Network
Create Heatmap Overlay Image
View all examples
Discussions
Start a discussion
Follow the community
Share
Tweet
Share
Top Community Contributors

Welcome to the new ThingSpeak community site. We're sharing the IoT. Tell us about your project or ask us about ours!

Using bulk-updates for larger sample rates

Latest activity by OliviaParcker OliviaParcker about 1 hour ago

Tags: arduino, bulk write, bulk update, json data, thingspeak

6 replies

Welcome to the ThingSpeak Community

Moderator: Christopher Stapels

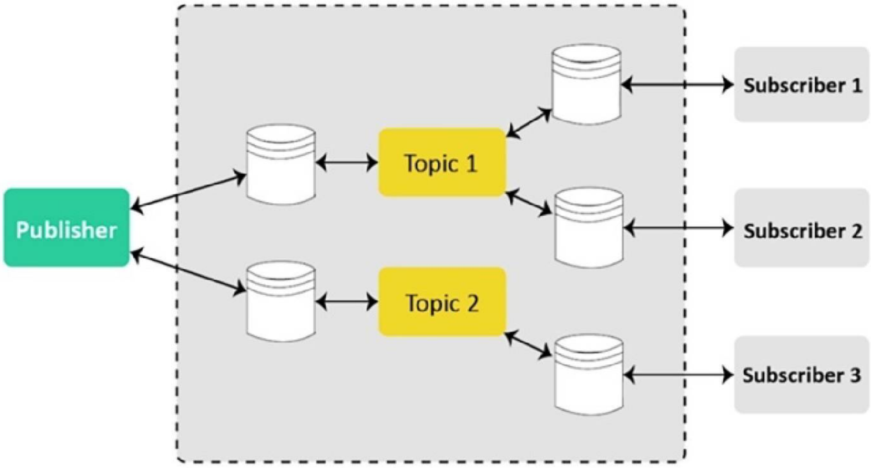
The community for students, researchers, and engineers looking to use MATLAB, Simulink, and ThingSpeak for Internet of Things applications. You can find the latest ThingSpeak news, tutorials to jump-start your next IoT project, and a forum to engage in a discussion on your latest cloud-based project. You can see answers to problems other users have solved and share how you solved a problem.

Moreover, Thing Speak has an amazing and active community. You can post your queries and also learn from other queries and answers here.

IN-DEPTH UNDERSTANDING OF THE THINGSPEAK API

we will cover the following topics, how MQTT works in Thing Speak. Overview of API Methods of the PubSub Client Library, installing PubSub client Library. You have already studied the MQTT protocol, how it works and we have also used it in Cayenne. But how does MQTT work in Thing Speak? As you know, MQTT is the most common protocol used in IoT systems to connect low level devices and sensors. MQTT is used to pass short messages to and from a broker. Around late 2016 Thing Speak introduced an MQTT broker so that devices can send MQTT messages to Thing Speak. These messages can include anything like the current temperature at your office collected by a sensor. Thing Speak then takes the message and stores its content to the desired Thing Speak channel. So why do we use MQTT? As we know in IoT applications, data from various sources needs to be collected and sent over the network.

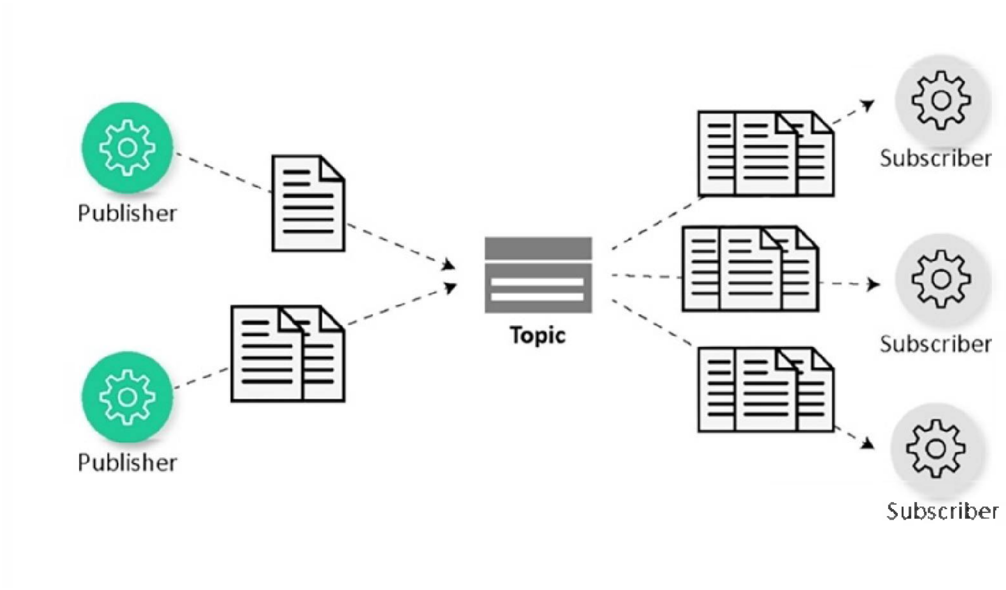
However, there is added complexity as IoT devices need remote monitoring and have limited network bandwidth as MQTT is a lightweight communication protocol and does not stress much over the device it is reliable to send data. MQTT is bidirectional and maintains stateful session awareness in comparison to the traditional request response model, MQTT uses the publish subscribe model for communication.



This helps the device to use the available bandwidth to its full potential.

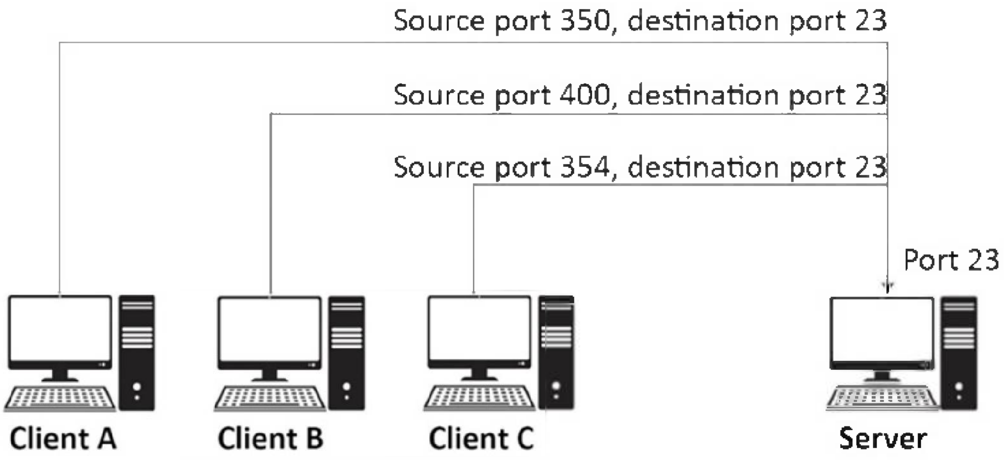
The light weightness and efficiency of MQTT make it possible to increase the amount of data being monitored or controlled significantly. All these factors make it a good fit and a go to option for IoT applications. So how does MQTT work?

The MQTT broker is the central point of communication and it is responsible for dispatching all the messages between senders and rightful receivers.

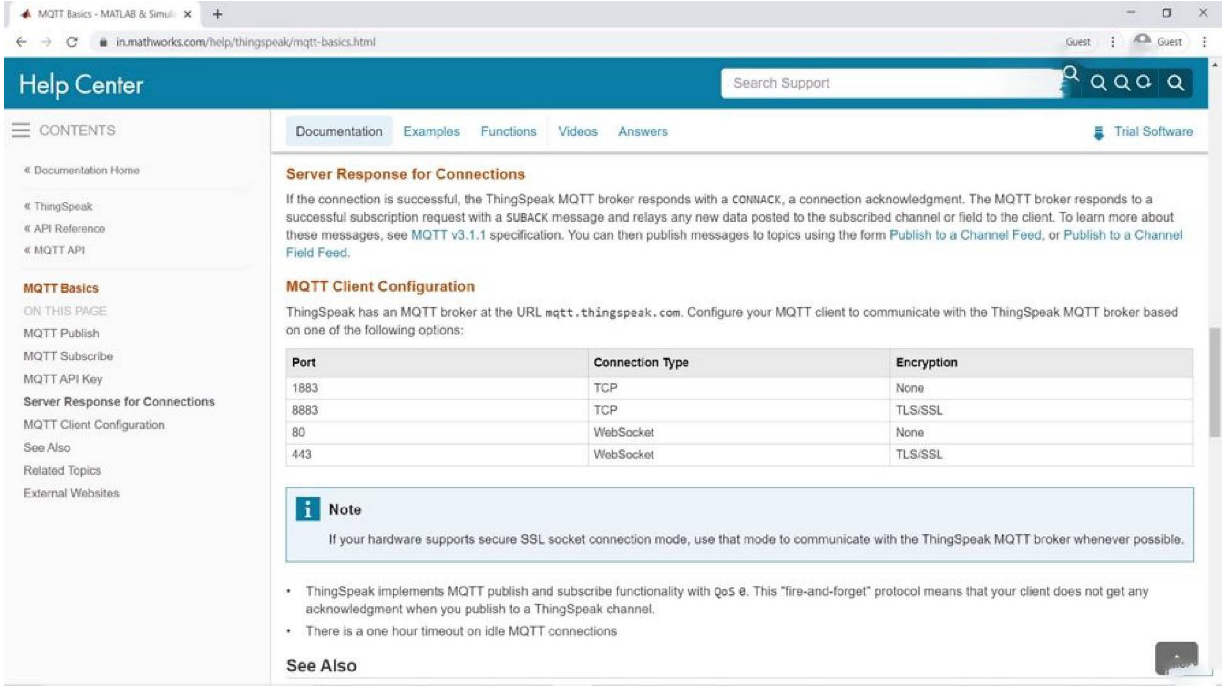


A client is any device that connects to the broker and can publish or subscribe to topics to access the information. A topic contains the routing information for the broker. Each client that wants to send messages publishes them to a certain topic, and each client that wants to receive the messages subscribes to a certain topic. The broker then delivers all the messages with the matching topic to the appropriate clients.

Thing Speak has an MQTT broker at at URL mqtt.Thing Speak.com at port 1883.


Source port 350, destination port 23
Source port 400, destination port 23
Source port 354, destination port 23
Port 23
Client A
Client B
Client C
Server

So what are ports? Ports provide a multiplexing service for multiple services or multiple communication sessions at one network address.



Server Response for Connections

If the connection is successful, the ThingSpeak MQTT broker responds with a `CONNACK`, a connection acknowledgment. The MQTT broker responds to a successful subscription request with a `SUBACK` message and relays any new data posted to the subscribed channel or field to the client. To learn more about these messages, see MQTT v3.1.1 specification. You can then publish messages to topics using the form Publish to a Channel Feed, or Publish to a Channel Field Feed.

MQTT Client Configuration

ThingSpeak has an MQTT broker at the URL `mqtt.thingspeak.com`. Configure your MQTT client to communicate with the ThingSpeak MQTT broker based on one of the following options:

Port	Connection Type	Encryption
1883	TCP	None
8883	TCP	TLS/SSL
80	WebSocket	None
443	WebSocket	TLS/SSL

Note

If your hardware supports secure SSL socket connection mode, use that mode to communicate with the ThingSpeak MQTT broker whenever possible.

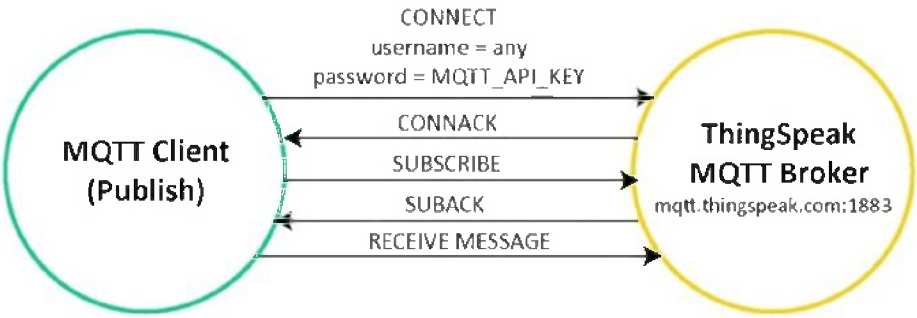
- ThingSpeak implements MQTT publish and subscribe functionality with `QoS 0`. This "fire-and-forget" protocol means that your client does not get any acknowledgment when you publish to a ThingSpeak channel.
- There is a one hour timeout on idle MQTT connections

See Also

To configure your MQTT client to communicate with Thing Speak MQTT broker. You can use one of the four options. They are port 1883 with the TCP Connection and no encryption, port 8883 with TCP Connection and

TLS SSL Encryption, Port 80 and 443 are also available with WebSocket Connection and no encryption and TLS/SSL connection respectively.

The client has to first connect to the Thing Speak broker to which the Thing Speak broker responds with a connection acknowledgement. The client can then publish messages to a topic and can subscribe to a topic. The client also has options to publish or subscribe to a channel feed or to a specific field in the channel feed. To publish to a channel feed a topic structure is predefined, which is as follows to publish to a particular channel and field, you need to make slight changes and add the field and field a number to the topic String.



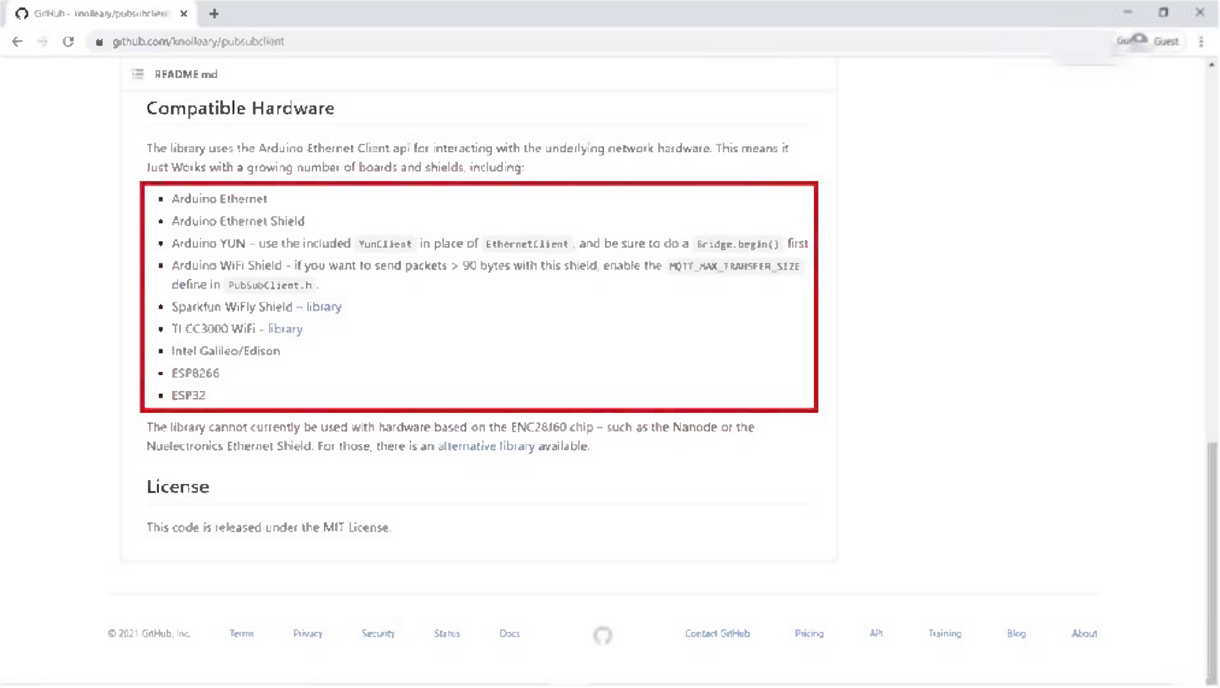
1. Subscribe to a channel feed
channels/<channelID>/subscribe/<format>/<apikey>

2. Subscribe to a private channel field
channels/<channelID>/subscribe/fields/field<fieldNumber>/<apikey>

3. Subscribe to all fields of a channel
channels/<channelID>/subscribe/fields/+/<apikey>

Similarly for subscribe operation, particular topic strings are predefined, which need to be followed. These topics strings specify the channel, field and API key of the device. API keys are necessary to publish or subscribe to channels. However, API keys are not required to subscribe to public channels.

To use MQTT with ESP32. We will be using the PubSub Client library. Using this library we can do simple publish subscribe messaging with the server that supports MQTT.



GitHub - knolleary/pubsubclient
github.com/knolleary/pubsubclient
Guest
README.md

Compatible Hardware

The library uses the Arduino Ethernet Client api for interacting with the underlying network hardware. This means it Just Works with a growing number of boards and shields, including:

- Arduino Ethernet
- Arduino Ethernet Shield
- Arduino YUN – use the included `YunClient` in place of `EthernetClient`, and be sure to do a `Bridge.begin()` first
- Arduino WiFi Shield - if you want to send packets > 90 bytes with this shield, enable the `MQTT_MAX_TRANSFER_SIZE` define in `PubSubClient.h`.
- Sparkfun WiFly Shield – library
- TI CC3000 WiFi - library
- Intel Galileo/Edison
- ESP8266
- ESP32

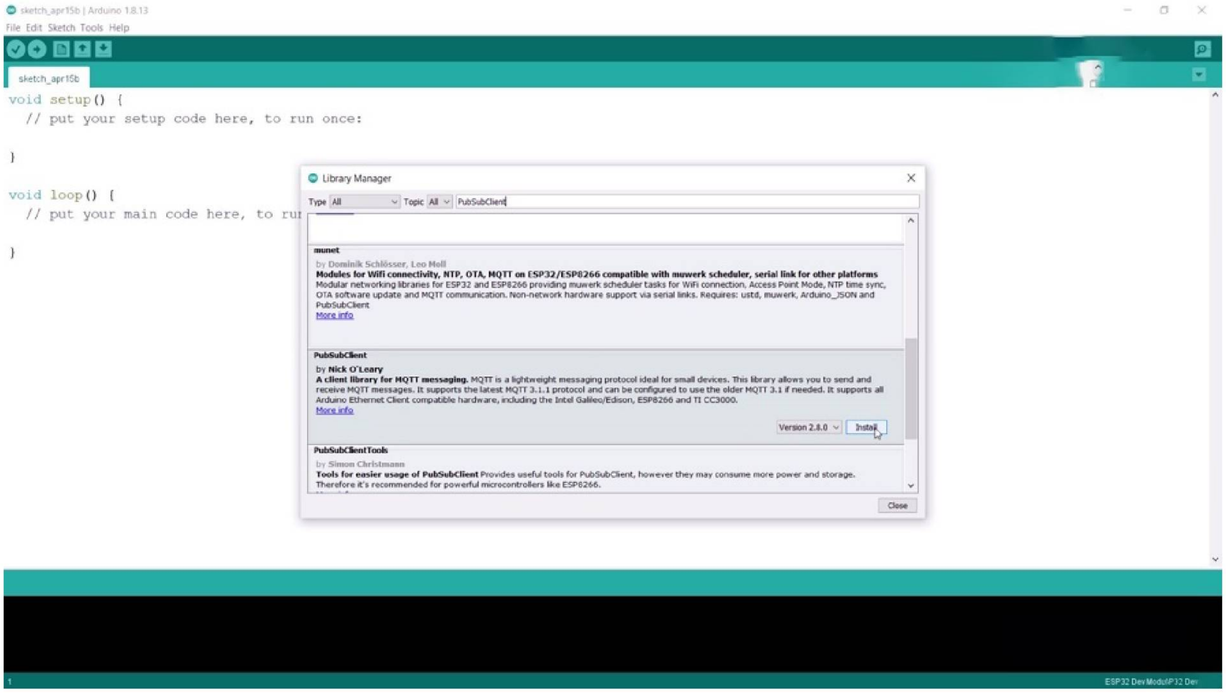
The library cannot currently be used with hardware based on the ENC28J60 chip – such as the Nanode or the Nuelectronics Ethernet Shield. For those, there is an alternative library available.

License

This code is released under the MIT License.

The library is compatible with the number of boards and shields, including Arduino, Intel Gallileo, ESP8266 and ESP32. The library consists of constructors and functions which are used in conjunction to program the device. According to need. Constructors are used for configuration and setting up the device. They initialize client instances. Various functions are also available which carry out operations to connect, disconnect, publish and subscribe to the MQTT. broker. The library provides some examples and source codes to get started with the library.

However, the library has some limitations. They are as follows. 1. it can only publish QoS 0 messages. It can subscribe at QoS 0 or QoS 1. 2. The maximum size, including the header is 256 bytes by default. 3. The keep alive interval is set to 15 seconds by default. 4. the client will use version 3.1.1 of MQTT by default. To know more about the library and its functions in detail.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run
}
```

PubSubClient

munet
by Dominik Schlösser, Leo Moll
Modules for Wifi connectivity, NTP, OTA, MQTT on ESP32/ESP8266 compatible with muwerk scheduler, serial link for other platforms Modular networking libraries for ESP32 and ESP8266 providing muwerk scheduler tasks for WiFi connection, Access Point Mode, NTP time sync, OTA software update and MQTT communication. Non-network hardware support via serial links. Requires: ustd, muwerk, Arduino_JSON and PubSubClient
More info

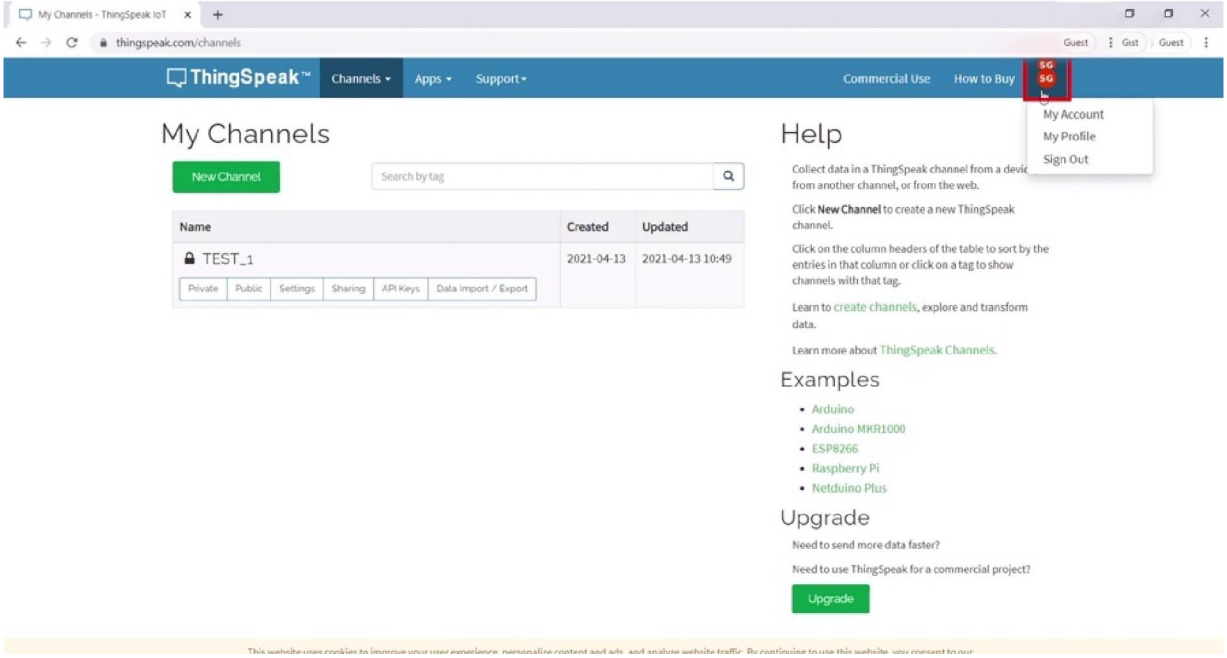
PubSubClient
by **Nick O'Leary**
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.
More info

PubSubClientTools
by Simon Christmann
Tools for easier usage of PubSubClient Provides useful tools for PubSubClient, however they may consume more power and storage. Therefore it's recommended for powerful microcontrollers like ESP8266.

Now let's install the PubSub Client Library in our Arduino IDE. To install the library, open your Arduino IDE, now go to tools and select manage libraries. The library manager will now appear on your screen. Search for PubSub Client. Now install the latest version of this library, we have successfully installed the PubSub Client Library and are ready to use it to program our ESP32.

we will cover the following topics, setting up the network, API key and MQTT configuration, example code overview, example code configurations, project implementation, troubleshooting and best practices. Until now, we have covered all the basics of the Thing Speak API. We have also had an overview of MQTT and the PubSUb Client Library. Let's put all of this to use and build a small project. We will be developing a project to detect motion using a PIR sensor. We will then send the data collected to Thing Speak using the MQTT protocol.

We will be connecting the ESP32 to the Internet and then using the MQTT protocol with the PubSub Client library. We will be using the port 1883 and the mqtt.Thing Speak.com server. These details are needed to connect to the correct MQTT broker. We will also need the MQTT password or the MQTT API key as specified in your Thing Speak account. To get the MQTT API key.



My Channels

Name	Created	Updated
TEST_1	2021-04-13	2021-04-13 10:49

Help

Collect data in a ThingSpeak channel from a devic from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

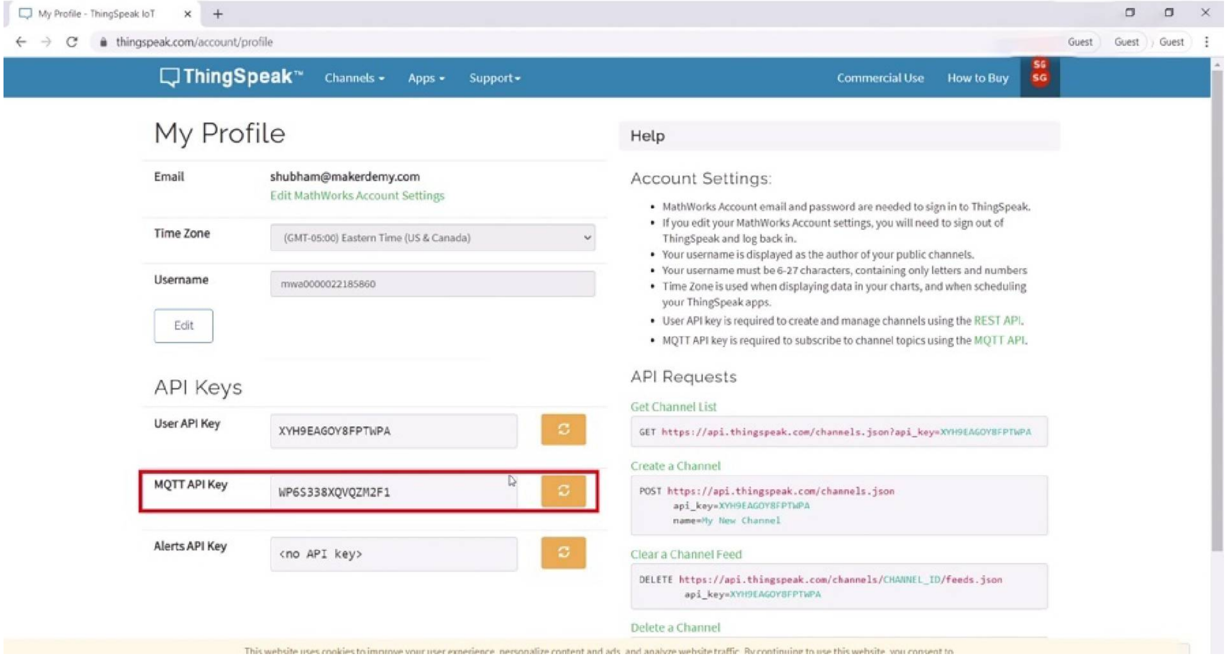
Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our

Click on your account icon on the top right corner then select my profile.



My Profile

Email shubham@makerdemy.com
Edit MathWorks Account Settings

Time Zone (GMT-05:00) Eastern Time (US & Canada)

Username mwa0000022185860

API Keys

User API Key XYH9EAGOY8FPTWPA

MQTT API Key WP6S338XQVQZM2F1

Alerts API Key <no API key>

Help

Account Settings:

- MathWorks Account email and password are needed to sign in to ThingSpeak.
- If you edit your MathWorks Account settings, you will need to sign out of ThingSpeak and log back in.
- Your username is displayed as the author of your public channels.
- Your username must be 6-27 characters, containing only letters and numbers
- Time Zone is used when displaying data in your charts, and when scheduling your ThingSpeak apps.
- User API key is required to create and manage channels using the REST API.
- MQTT API key is required to subscribe to channel topics using the MQTT API.

API Requests

Get Channel List

```
GET https://api.thingspeak.com/channels.json?api_key=XYH9EAGOY8FPTWPA
```

Create a Channel

```
POST https://api.thingspeak.com/channels.json
      api_key=XYH9EAGOY8FPTWPA
      name=My New Channel
```

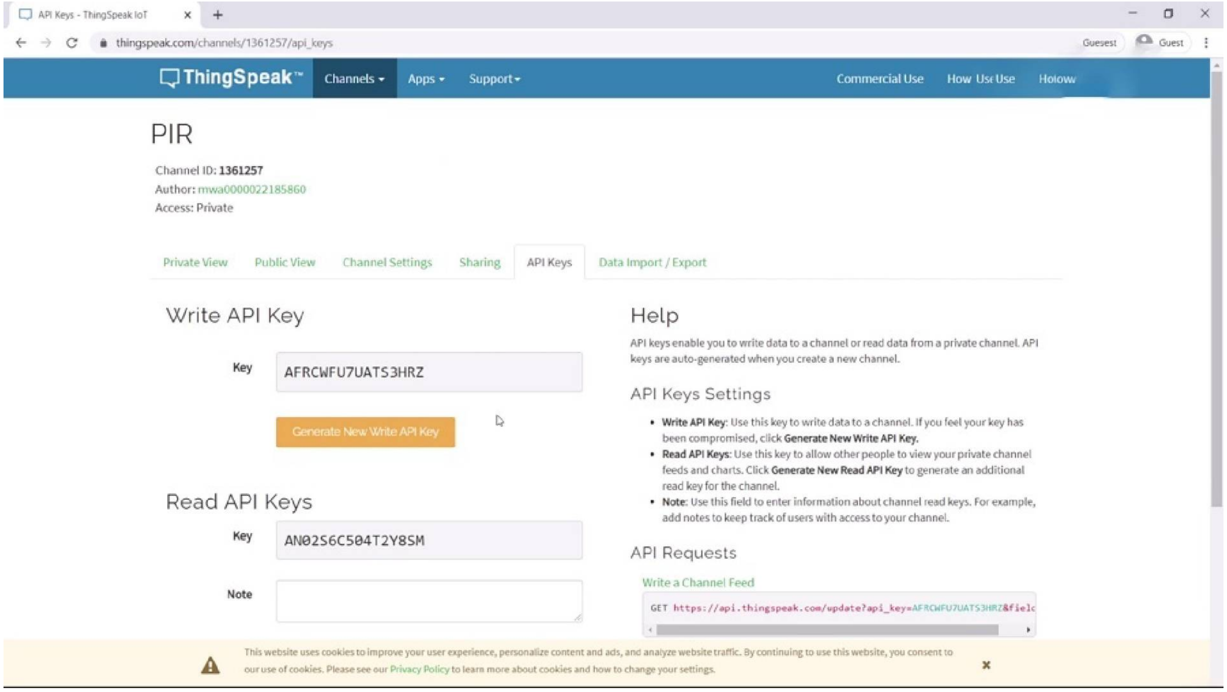
Clear a Channel Feed

```
DELETE https://api.thingspeak.com/channels/CHANNEL_ID/feeds.json
       api_key=XYH9EAGOY8FPTWPA
```

Delete a Channel

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to

Here you will find your MQTT API key. Note it down We will also need the Write API key for a channel to publish the data to a particular channel. To get the Write API. Create a new channel. Fill in the necessary details like the name and select 2 fields to publish. Now click on the Save Channel button. After you have created a new channel, go to API keys.



PIR

Channel ID: **1361257**
Author: mwa0000022185860
Access: Private

Write API Key

Key AFRCWFU7UATS3HRZ

Read API Keys

Key AN02S6C504T2Y8SM

Note

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

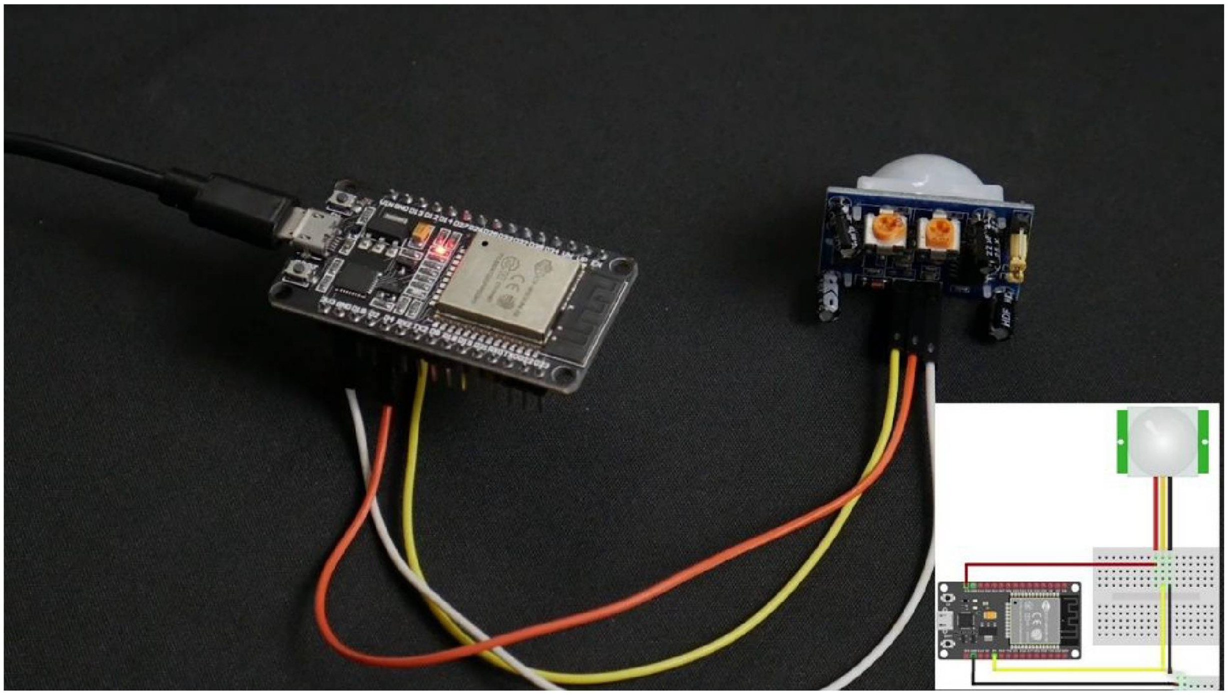
API Requests

Write a Channel Feed

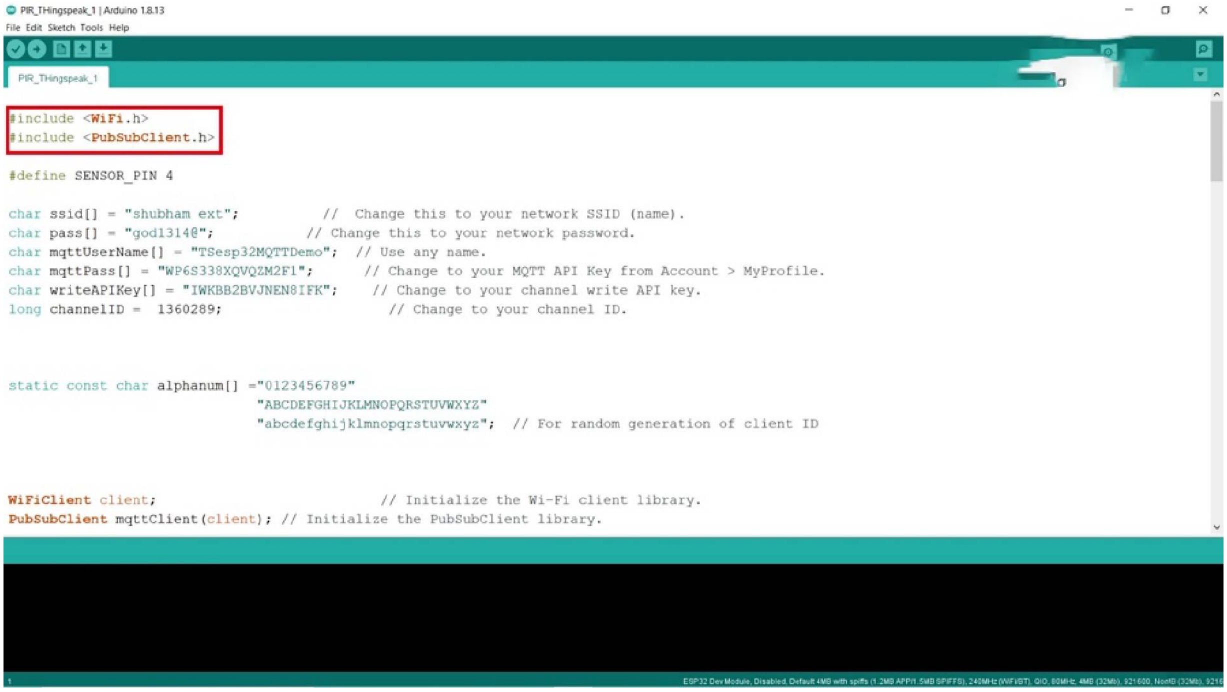
```
GET https://api.thingspeak.com/update?api_key=AFRCWFU7UATS3HRZ&fielc
```

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Here you will find your write API key. Note it down. We have now configured and have all the necessary credentials. Let's first check the hardware part for this project. As we are using the PIR sensor, it needs a 5V supply.

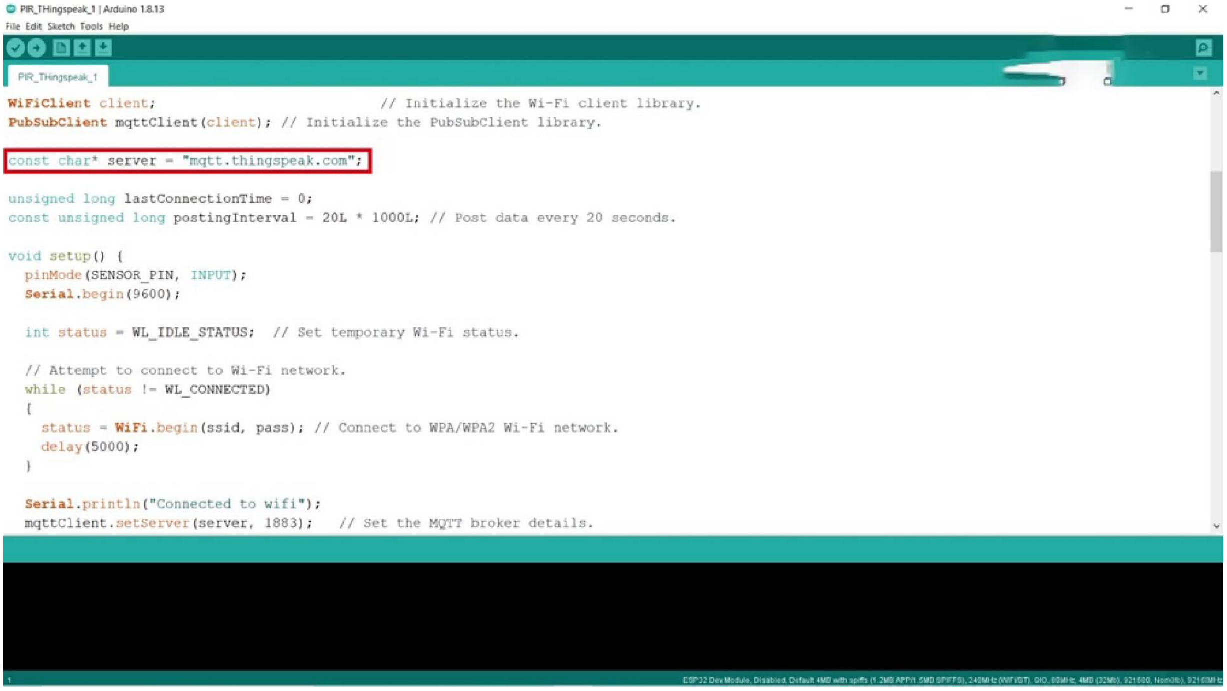


So we will connect the VCC of the PIR sensor to the VIN of the ESP32. We will then connect the ground of the PIR sensor to the ground of the ESP32. Now we have to connect the data out pin from the PIR sensor to GPIO 4 of the ESP32. You can connect it to any of the GPIO pins. But keep in mind to make the necessary changes in the code.


PIR_THingspeak_1 | Arduino 1.8.13
File Edit Sketch Tools Help
PIR_THingspeak_1
1
ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None (32Mb), 9216

```
#include <WiFi.h>
#include <PubSubClient.h>

#define SENSOR_PIN 4

char ssid[] = "shubham ext";          //  Change this to your network SSID (name).
char pass[] = "god1314@";           // Change this to your network password.
char mqttUserName[] = "TSesp32MQTTDemo";  // Use any name.
char mqttPass[] = "WP6S338XQVQZM2F1";      // Change to your MQTT API Key from Account > MyProfile.
char writeAPIKey[] = "IWKBB2BVJNEN8IFK";    // Change to your channel write API key.
long channelID =  1360289;                    // Change to your channel ID.


static const char alphanum[] ="0123456789"
                              "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                              "abcdefghijklmnopqrstuvwxyz";  // For random generation of client ID


WiFiClient client;                          // Initialize the Wi-Fi client library.
PubSubClient mqttClient(client); // Initialize the PubSubClient library.
```

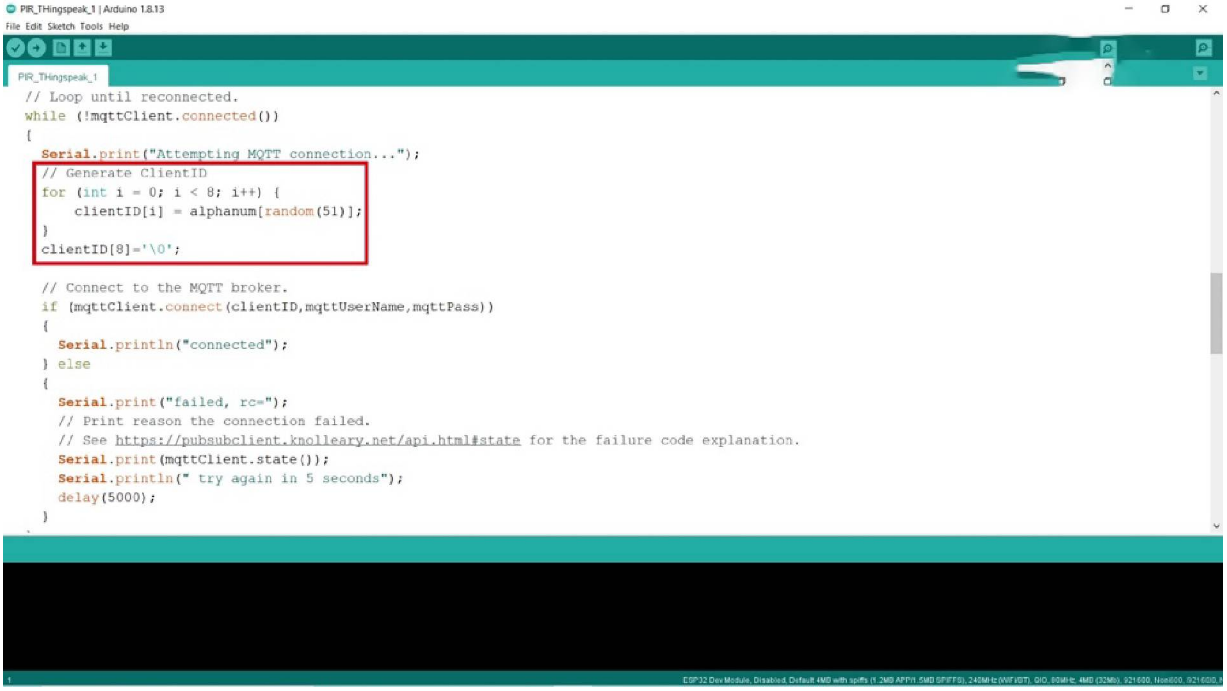
Now let's take the code. First, We will include the two libraries which are needed, the Wi-Fi library and the PubSub Client Library. Remember, in Cayenne we had installed a Cayenne library which was responsible for connecting to your Wi-Fi network. But in our case now we will use the Wi-Fi library to connect to our Wi-Fi. Next, we will define a variable sensor pin to GPIO 4 as we have connected the data out pin of the PIR sensor to GPIO four of the ESP32. We will now provide all the credentials necessary to connect to the Internet and MQTT broker. Here we have specified the Wi-Fi network SSID and password. Next, we have mentioned a user name. You can use any user name of your choice. Next, we have mentioned the MQTT API key you got earlier from your Thing Speak Accounts profile page. We will now mention the Write API key of the channel. The Channel ID, which you can find at the top, also needs to be mentioned here. We have given all the numbers and alphabets for random generation of client ID. This is needed to make the MQTT connection.


PIR_THingspeak_1 | Arduino 1.8.13
File Edit Sketch Tools Help
PIR_THingspeak_1

```
WiFiClient client;                          // Initialize the Wi-Fi client library.
PubSubClient mqttClient(client); // Initialize the PubSubClient library.

const char* server = "mqtt.thingspeak.com";

unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 20L * 1000L; // Post data every 20 seconds.

void setup() {
  pinMode(SENSOR_PIN, INPUT);
  Serial.begin(9600);

  int status = WL_IDLE_STATUS;  // Set temporary Wi-Fi status.

  // Attempt to connect to Wi-Fi network.
  while (status != WL_CONNECTED)
  {
    status = WiFi.begin(ssid, pass); // Connect to WPA/WPA2 Wi-Fi network.
    delay(5000);
  }

  Serial.println("Connected to wifi");
  mqttClient.setServer(server, 1883);   // Set the MQTT broker details.
```

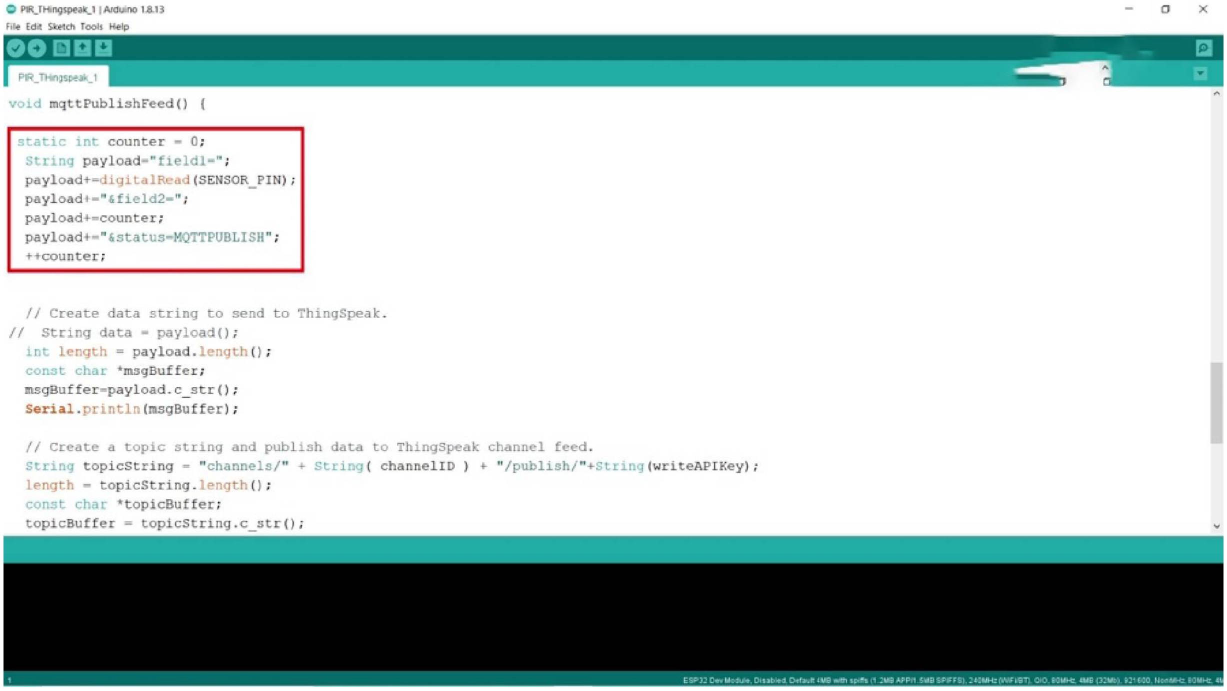
ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None(lb), 921600MHz

Here We have initialized the WiFi and PubSub Client library. Now we need to mention the server we need to connect to. In our case, it is the Thing Speak MQTT server. Here we have initialized the last connection time variable to zero and set the time to post data to our Thing Speak channel to twenty seconds. Now we need to set up all other parts and configure the Wi-Fi connection. First we set pin 4 as input and set the baud rate to 9600. Here we have connected to our wi fi using the SSID and password using the credentials given before. Next, we have provided the port to connect to the MQTT Broker.


PIR_THingspeak_1 | Arduino 1.8.13
File Edit Sketch Tools Help
PIR_THingspeak_1

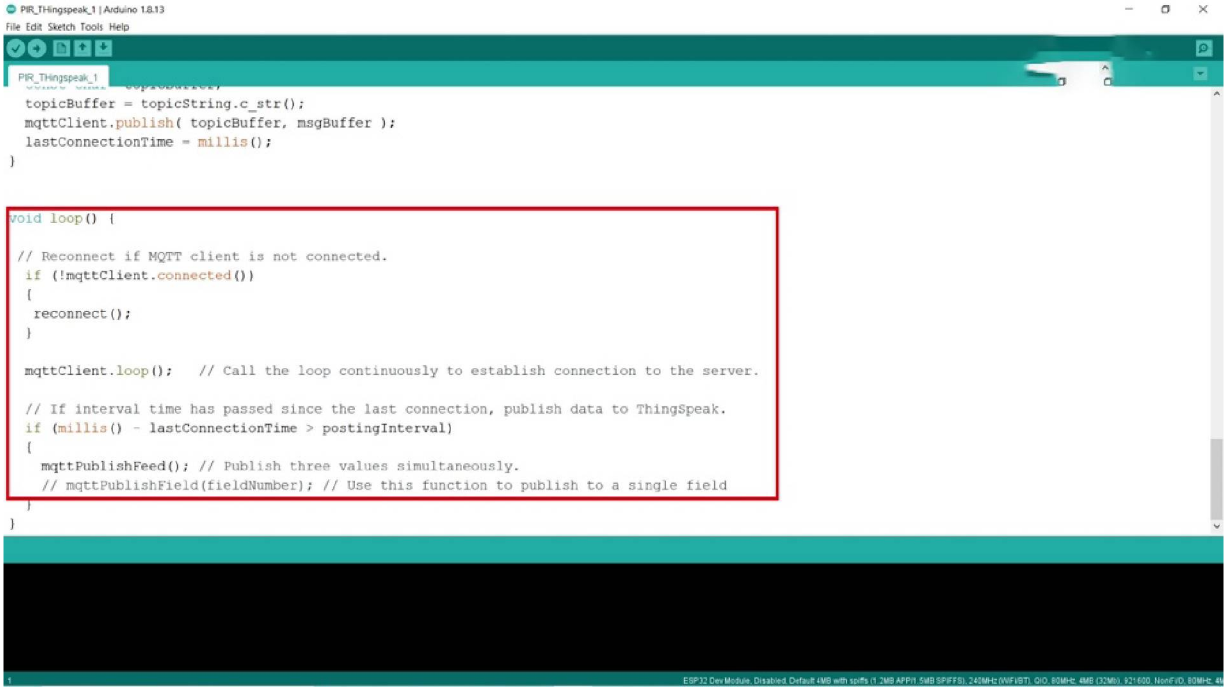
```
// Loop until reconnected.
while (!mqttClient.connected())
{
  Serial.print("Attempting MQTT connection...");
  // Generate ClientID
  for (int i = 0; i < 8; i++) {
      clientID[i] = alphanum[random(51)];
  }
  clientID[8]='\0';

  // Connect to the MQTT broker.
  if (mqttClient.connect(clientID,mqttUserName,mqttPass))
  {
    Serial.println("connected");
  } else
  {
    Serial.print("failed, rc=");
    // Print reason the connection failed.
    // See https://pubsubclient.knolleary.net/api.html#state for the failure code explanation.
    Serial.print(mqttClient.state());
    Serial.println(" try again in 5 seconds");
    delay(5000);
  }
```

Next we will connect to the MQTT Broker And for this we will generate random client ID every time to connect. We need to generate a random client idea every time, as if you use a generic client ID that someone else may also have the same client ID. If this happens, your device will not be able to connect to the broker. If any issue occurs during connection, we will also program the device to reconnect to the broker every five seconds.


PIR_THingspeak_1 | Arduino 1.8.13
File Edit Sketch Tools Help
PIR_THingspeak_1

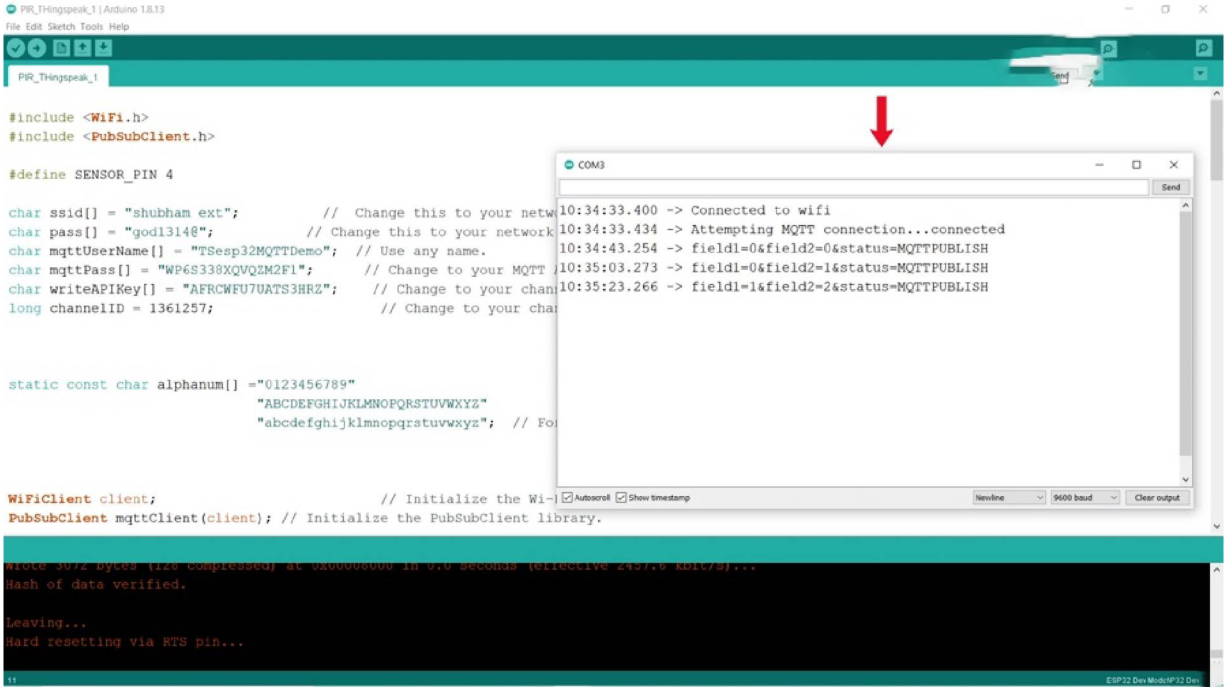
```
void mqttPublishFeed() {

 static int counter = 0;
  String payload="field1=";
  payload+=digitalRead(SENSOR_PIN);
  payload+="&field2=";
  payload+=counter;
  payload+="&status=MQTTPUBLISH";
  ++counter;


  // Create data string to send to ThingSpeak.
//  String data = payload();
  int length = payload.length();
  const char *msgBuffer;
  msgBuffer=payload.c_str();
  Serial.println(msgBuffer);

  // Create a topic string and publish data to ThingSpeak channel feed.
  String topicString = "channels/" + String( channelID ) + "/publish/"+String(writeAPIKey);
  length = topicString.length();
  const char *topicBuffer;
  topicBuffer = topicString.c_str();
```

We will now generate the payload stream, which will carry your data to the Thing Speak channel channel. Here We will read the data from the PIR sensor and set the counter for every time the data is collected. Here we will create a data string to send to Thing Speak. Here We will create a topic string of the data to send to Thing Speak. Remember in the last project where I mentioned about the topics string to publish, we have followed the same format. We will then use this topic string to publish the data, to Thing Speak.


PIR_THingspeak_1 | Arduino 1.8.13
File Edit Sketch Tools Help
PIR_THingspeak_1
ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None, 80MHz, 4M

```
  topicBuffer = topicString.c_str();
  mqttClient.publish( topicBuffer, msgBuffer );
  lastConnectionTime = millis();
}

void loop() {

 // Reconnect if MQTT client is not connected.
  if (!mqttClient.connected())
  {
   reconnect();
  }

  mqttClient.loop();   // Call the loop continuously to establish connection to the server.

  // If interval time has passed since the last connection, publish data to ThingSpeak.
  if (millis() - lastConnectionTime > postingInterval)
  {
    mqttPublishFeed(); // Publish three values simultaneously.
    // mqttPublishField(fieldNumber); // Use this function to publish to a single field
  }
}
```

Here in the void loop. We have used functions to continuously call the loops to connect to the server and simultaneously publish values. Now save and upload the code.

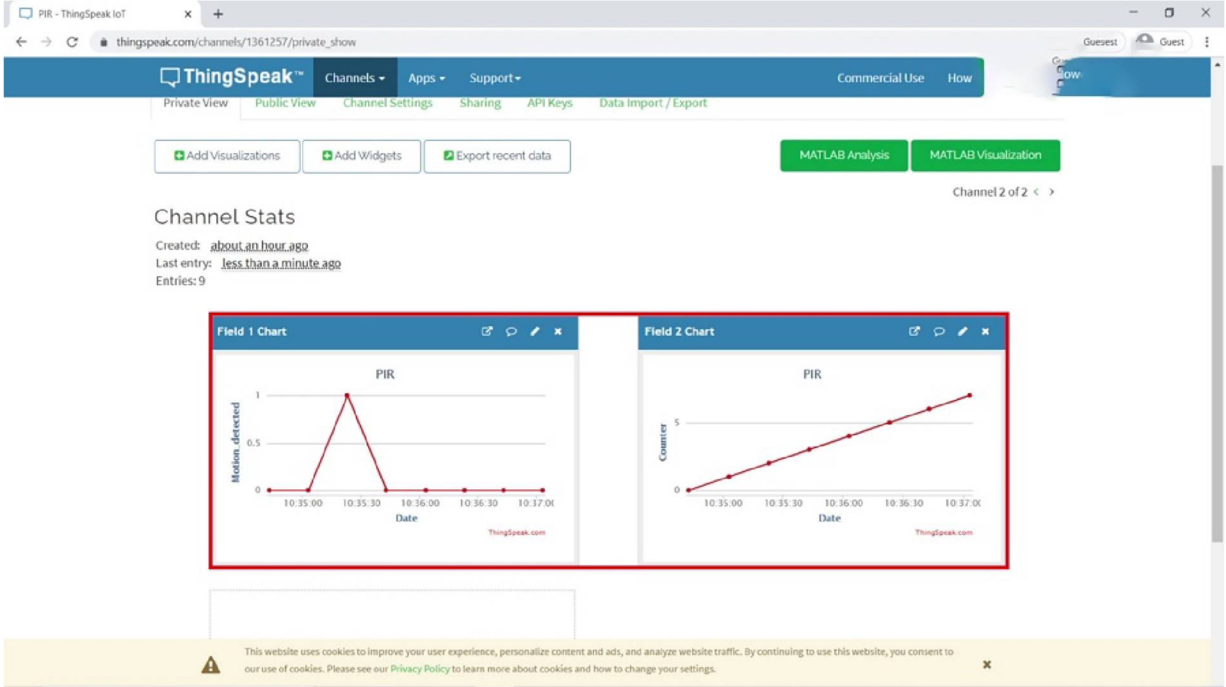

PIR_THingspeak_1 | Arduino 1.8.13
File Edit Sketch Tools Help
PIR_THingspeak_1
COM3
Send
☑ Autoscroll ☑ Show timestamp
Newline
9600 baud
Clear output
ESP32 Dev Module, ESP32 Dev

```
#include <WiFi.h>
#include <PubSubClient.h>

#define SENSOR_PIN 4

char ssid[] = "shubham ext";          //  Change this to your netw
char pass[] = "god1314@";           // Change this to your network
char mqttUserName[] = "TSesp32MQTTDemo";  // Use any name.
char mqttPass[] = "WP6S338XQVQZM2F1";      // Change to your MQTT A
char writeAPIKey[] = "AFRCWFU7UATS3HRZ";    // Change to your chan
long channelID = 1361257;                    // Change to your cha


static const char alphanum[] ="0123456789"
                              "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                              "abcdefghijklmnopqrstuvwxyz";  // Fo


WiFiClient client;                           // Initialize the Wi-
PubSubClient mqttClient(client); // Initialize the PubSubClient library.
```

```
10:34:33.400 -> Connected to wifi
10:34:33.434 -> Attempting MQTT connection...connected
10:34:43.254 -> field1=0&field2=0&status=MQTTPUBLISH
10:35:03.273 -> field1=0&field2=1&status=MQTTPUBLISH
10:35:23.266 -> field1=1&field2=2&status=MQTTPUBLISH
```

```
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 2457.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

You can open your serial monitor to check the data which is being published.



Channel Stats

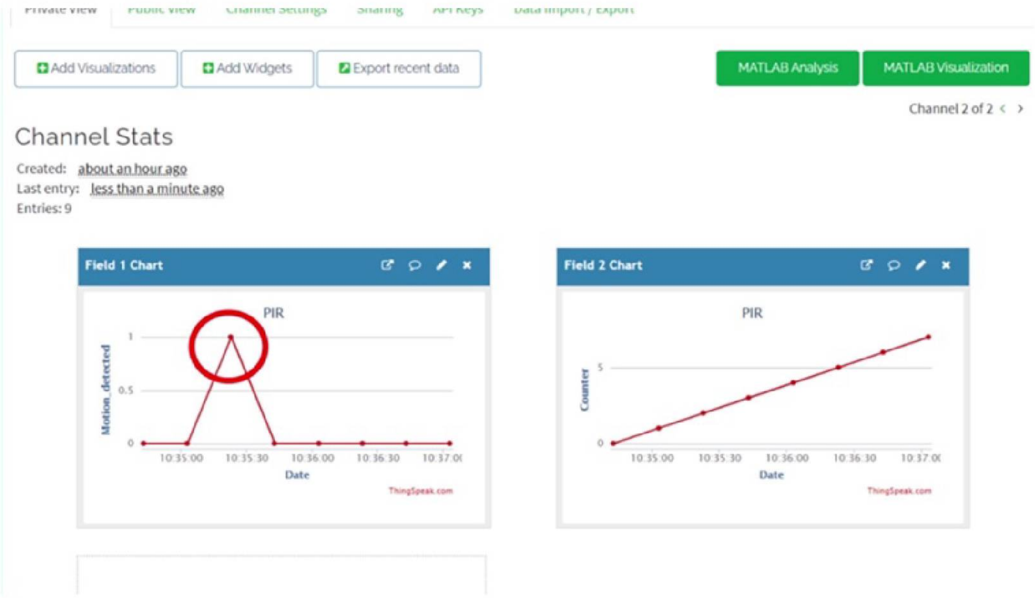
Created: about an hour ago
Last entry: less than a minute ago
Entries: 9

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Now, check your Thing Speak account. You will see your Channel one Publishing the values one or zero one denotes that the motion is detected while zero denotes that motion is not detected. Your Channel two will show the count for every entry as you know IoT is a tough domain because of the involvement of both hardware and software parts. You should consider some basic troubleshooting methods and practices so that you do not have problems implementing projects. First of all, you should always install the latest version of libraries you want to use. This is mandatory as the library publisher fixes bugs and issues that may be present in the previous library version. Most of the time, the latest library versions are better than the previous ones. Now, do you remember that we introduced a counter in our project. This is done to have a track of how many times the data sent and if our system is correctly working with Thing Speak. You can comment out the counterpart. Even then, the system will do the task assigned to it. Moreover, you should always use proper jumper Wires and breadboards to make connections. As many times there are issues with loose connections. You should also check your connections properly and connect the devices and sensors to the proper pins. Also check your sensor beforehand. It may so happen that your sensor is broken and will not output any data even when a proper power supply is given. Keeping some of these points in mind, you will be able to develop an IoT system with ease.

OVERVIEW AND INTEGRATION OF IFTTT

we will cover the following topics. What is IFTTT and why do we need it with Thing Speak, setting up an IFTTT account, integrating IFTTT into example code implementation In the previous project, we send our sensor data to Thing Speak.



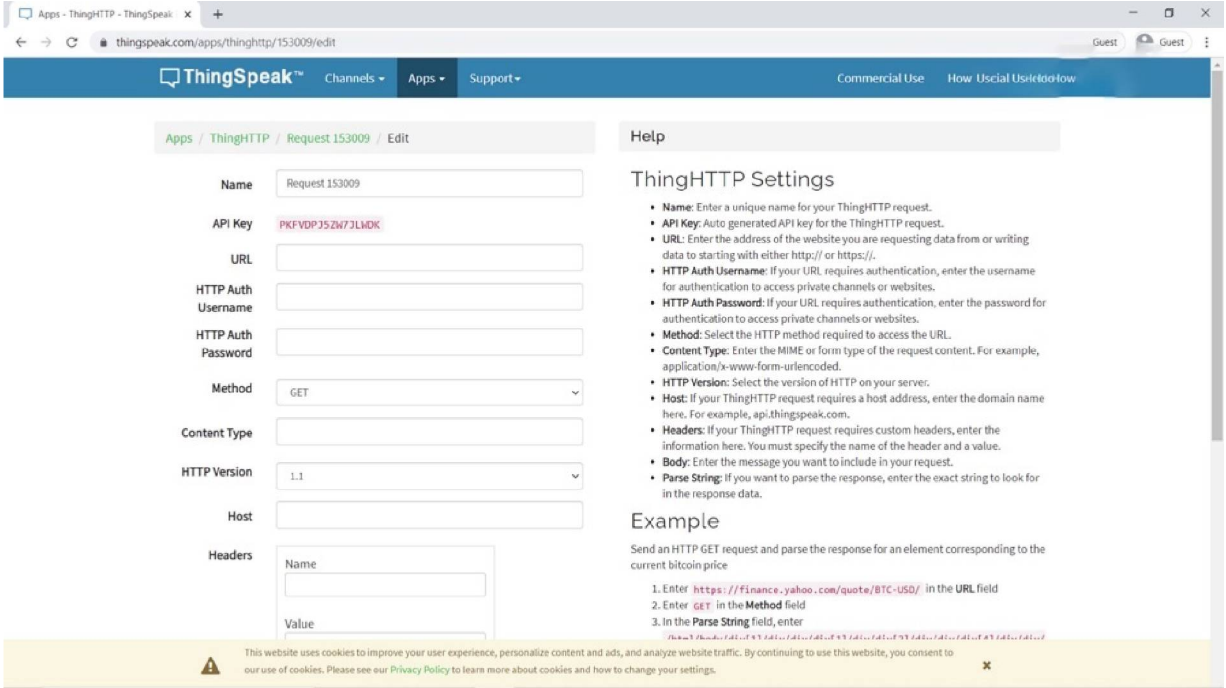
Channel Stats

Created: about an hour ago

Last entry: less than a minute ago

Entries: 9

We can clearly see that the graph for motion detection shoots to one whenever the sensor detects motion. However, it is not feasible to keep an eye on the graph 24/7. How can we automate the system? Yes, we can surely automat it and send alerts to us whenever required. Remember in Cayenne we use the inbuilt trigger notification feature Similarly, we can automate Thing Speak to send an email notification every time motion is detected. This is where IFTTT comes into action, so what is IFTTT? IFTTT has its name derived from the programming conditional statement If this then that. IFTTT was found in 2010 and launched in 2011. The company provides a software platform that connects apps, devices and services from different developers in order to trigger one or more automation. There are various apps that can be used to update Google sheets, create reminders and send notifications according to your need. According to IFTTT at the moment. There are more than 90 million applet connections. So why do we need IFTTT? In Cayenne we use the inbuilt trigger and notification function in order to send updates.


Apps - ThingHTTP - ThingSpeak
thingspeak.com/apps/thinghttp/153009/edit
Guest
ThingSpeak™
Channels ▾
Apps ▾
Support ▾
Commercial Use
How Uscial UsHoloHow
Apps / ThingHTTP / Request 153009 / Edit

Name	Request 153009
API Key	PKFVDPJ5ZW7JLWDK
URL	
HTTP Auth Username	
HTTP Auth Password	
Method	GET
Content Type	
HTTP Version	1.1
Host	
Headers	Name Value

Help

ThingHTTP Settings

- **Name**: Enter a unique name for your ThingHTTP request.
- **API Key**: Auto generated API key for the ThingHTTP request.
- **URL**: Enter the address of the website you are requesting data from or writing data to starting with either http:// or https://.
- **HTTP Auth Username**: If your URL requires authentication, enter the username for authentication to access private channels or websites.
- **HTTP Auth Password**: If your URL requires authentication, enter the password for authentication to access private channels or websites.
- **Method**: Select the HTTP method required to access the URL.
- **Content Type**: Enter the MIME or form type of the request content. For example, application/x-www-form-urlencoded.
- **HTTP Version**: Select the version of HTTP on your server.
- **Host**: If your ThingHTTP request requires a host address, enter the domain name here. For example, api.thingspeak.com.
- **Headers**: If your ThingHTTP request requires custom headers, enter the information here. You must specify the name of the header and a value.
- **Body**: Enter the message you want to include in your request.
- **Parse String**: If you want to parse the response, enter the exact string to look for in the response data.

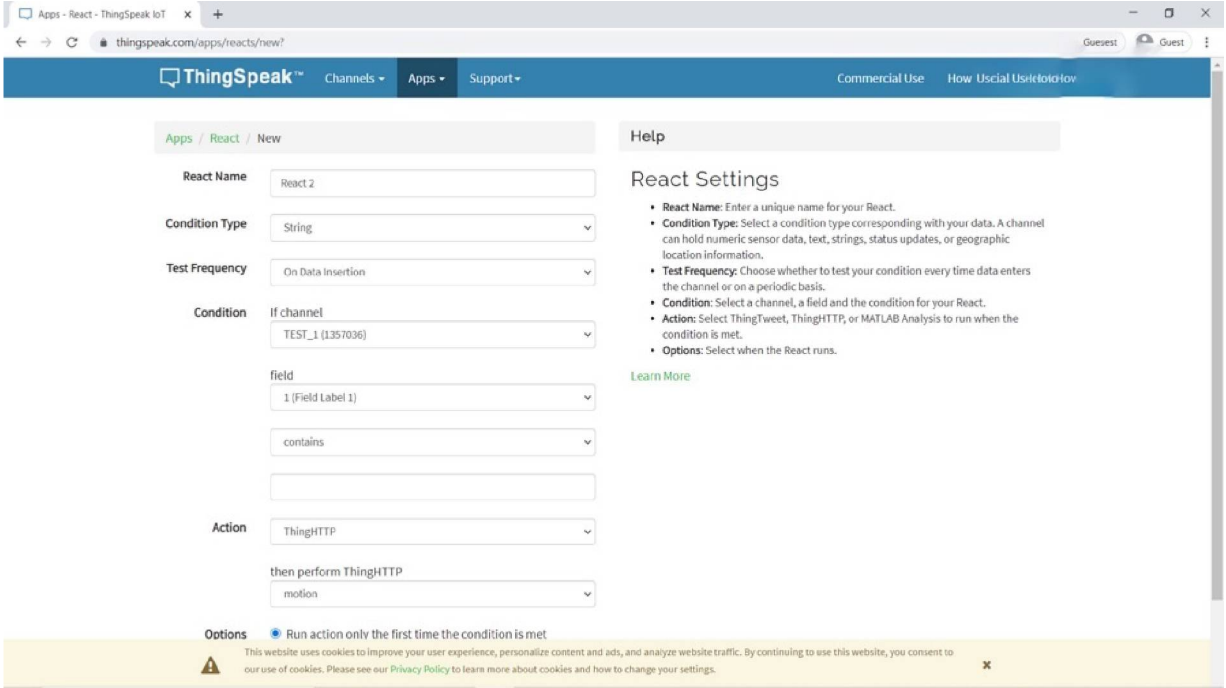
Example

Send an HTTP GET request and parse the response for an element corresponding to the current bitcoin price

1. Enter `https://finance.yahoo.com/quote/BTC-USD/` in the **URL** field
2. Enter `GET` in the **Method** field
3. In the **Parse String** field, enter

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Thing Speak does not have any direct feature to send notifications on a trigger, but it has other apps like ThingHTTP and React, which can be used in conjunction with IFTTT to automate our systems.



React Name	React 2
Condition Type	String
Test Frequency	On Data Insertion
Condition	If channel TEST_1 (1357036) field 1 (Field Label 1) contains
Action	ThingHTTP then perform ThingHTTP motion
Options	● Run action only the first time the condition is met

Help

React Settings

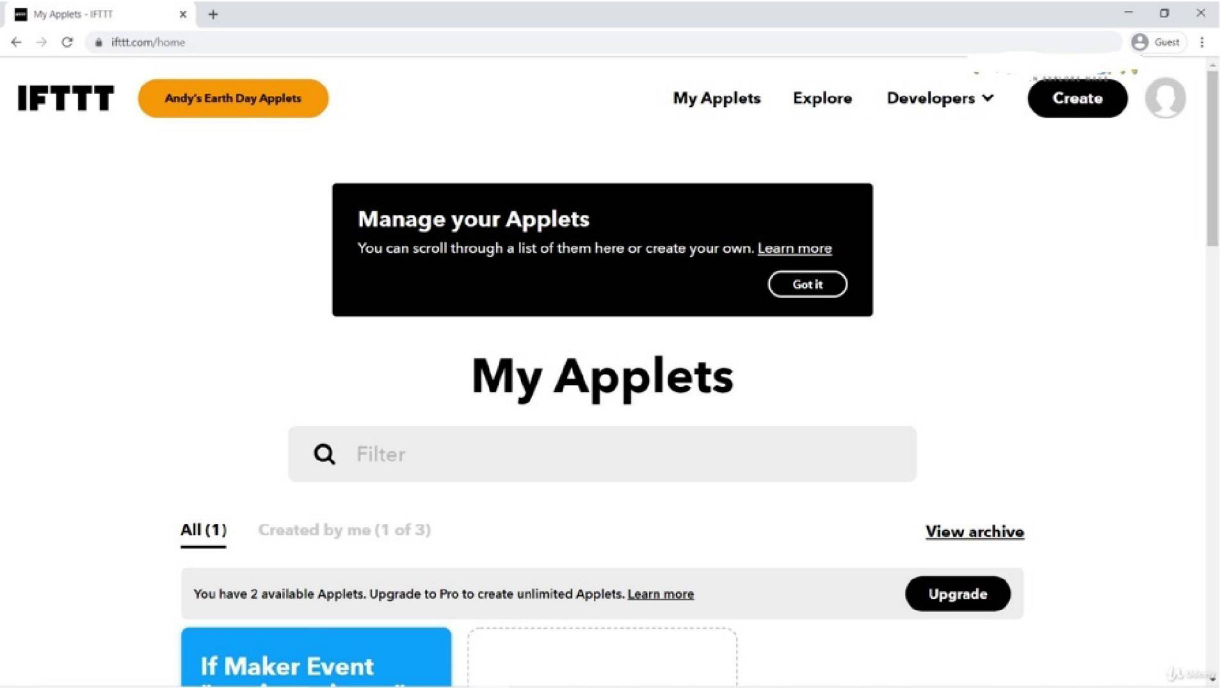
- **React Name**: Enter a unique name for your React.
- **Condition Type:** Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- **Test Frequency:** Choose whether to test your condition every time data enters the channel or on a periodic basis.
- **Condition**: Select a channel, a field and the condition for your React.
- **Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- **Options**: Select when the React runs.

Learn More

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

IFTTT is available as an application for both Android and iOS. This makes it easy to use and we can create instant applets to automate our systems.

Let's get started with IFTTT and set up your account. first. Go to the IFTTT website. Please find the link to the website in the resources of this project. Now click on the get started option on the top right corner. You can either sign up with your Apple, Google or Facebook ID. You can even sign up manually by clicking here. It will then ask you for your email ID and a password to be created. Once your account is created, log into your account.



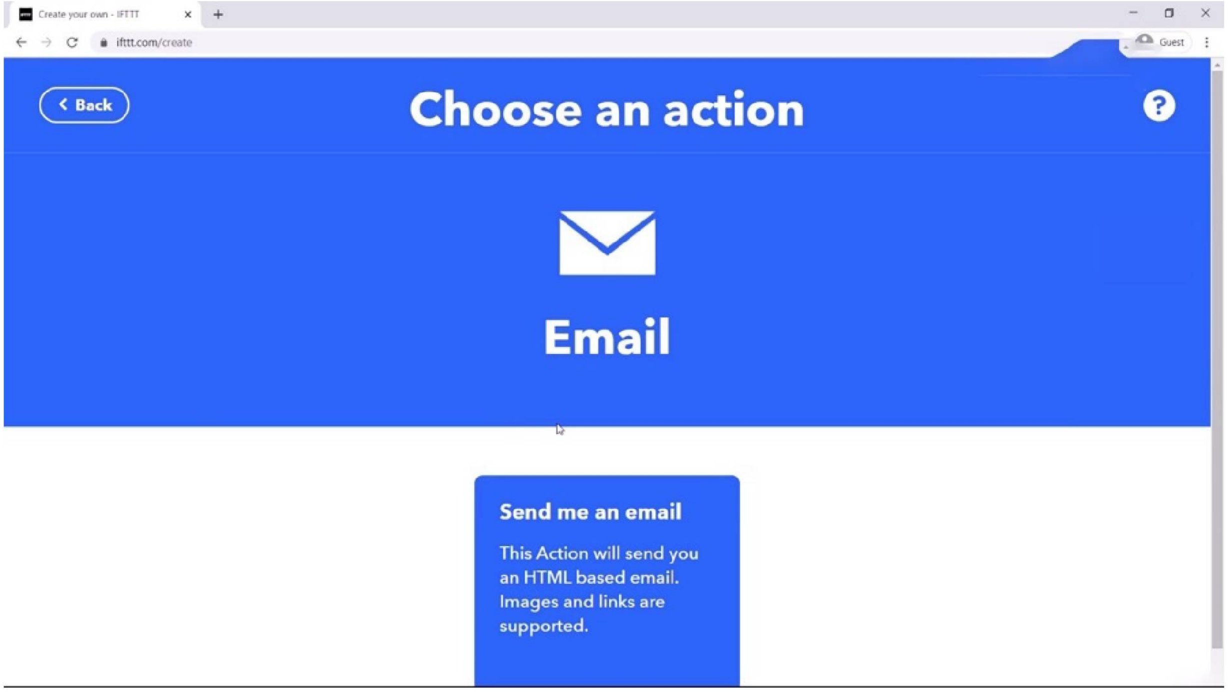
Manage your Applets

You can scroll through a list of them here or create your own. Learn more

My Applets

You have 2 available Applets. Upgrade to Pro to create unlimited Applets. Learn more

You will arrive at the homepage. Now, let's integrate IFTTT with things be to send trigger notifications whenever the PIR sensor detects motion. Now click on Create at the top right corner you will be directed to a new page that displays if this then that. Here we have to set our triggers and notifications. Click on this now. Search for Web hooks, click on Web hooks and select the receiver web request, complete the trigger fields and enter a name for the event. Now click on the create trigger option. You have now set that trigger.



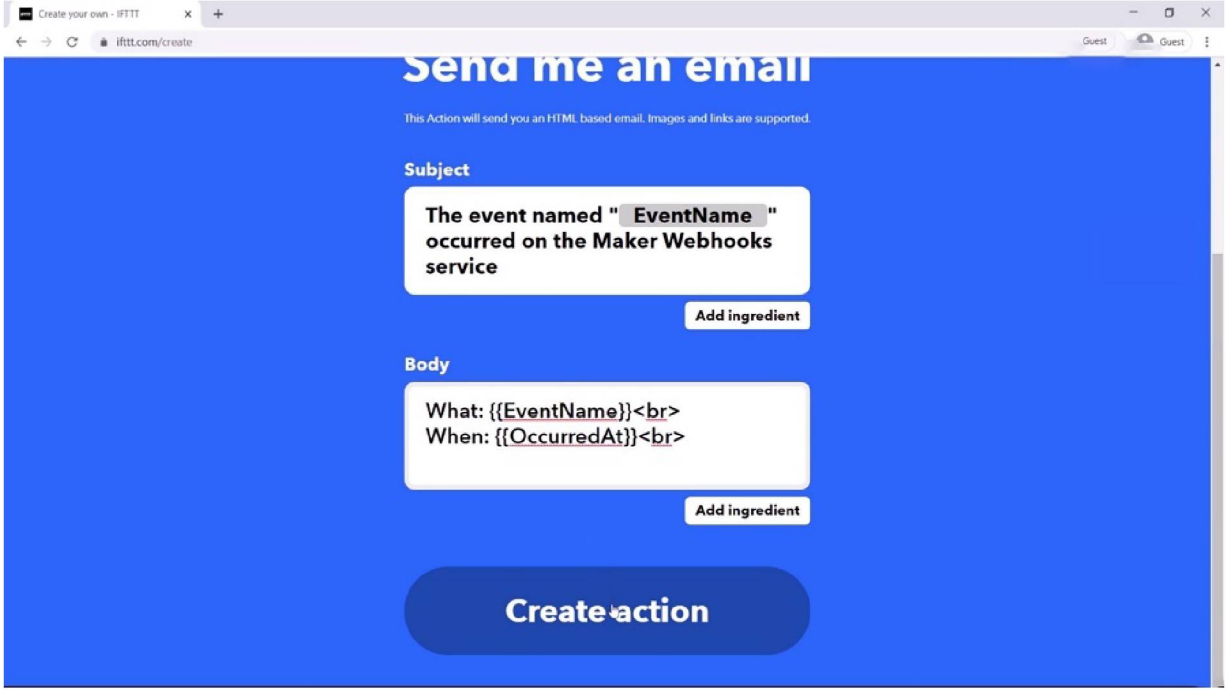
Choose an action

Email

Send me an email

This Action will send you an HTML based email. Images and links are supported.

Now we need to set the resulting action, go for notifications and e-mail option. You will have to enter the e-mail you wish to send the email to.



Send me an email

This Action will send you an HTML based email. Images and links are supported.

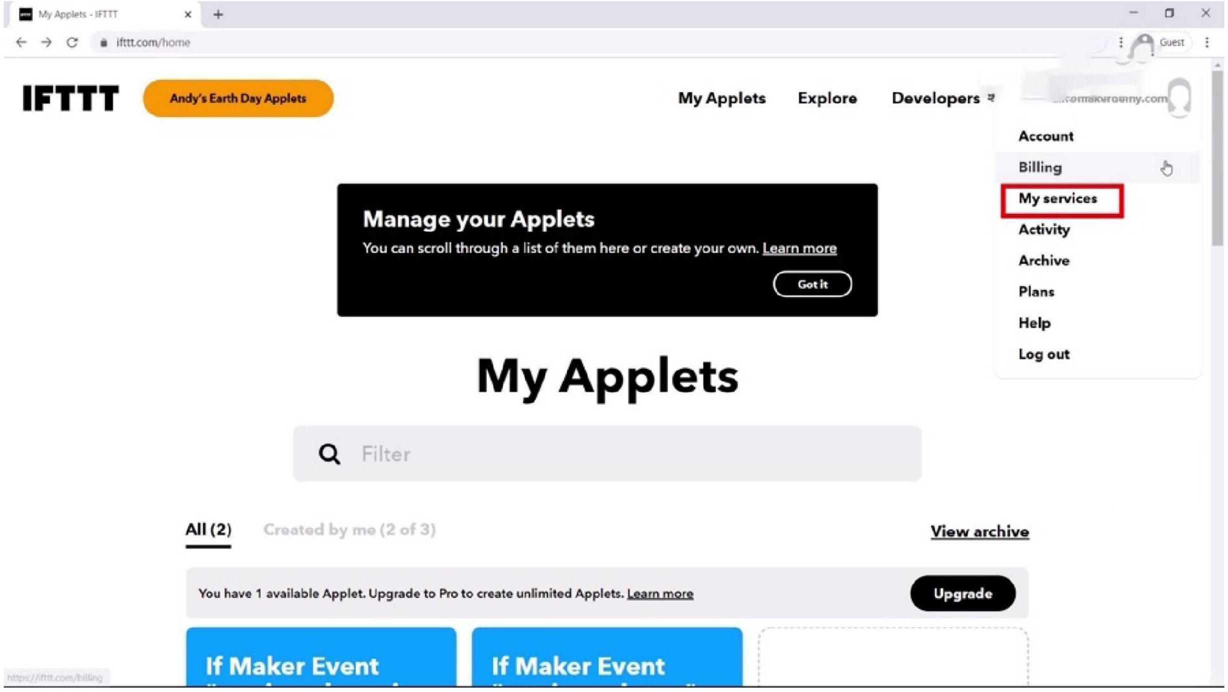
Subject

The event named " EventName " occurred on the Maker Webhooks service

Body

```
What: {{EventName}}<br>
When: {{OccurredAt}}<br>
```

Now click on Create Action. Now, your tablet is ready to be used.



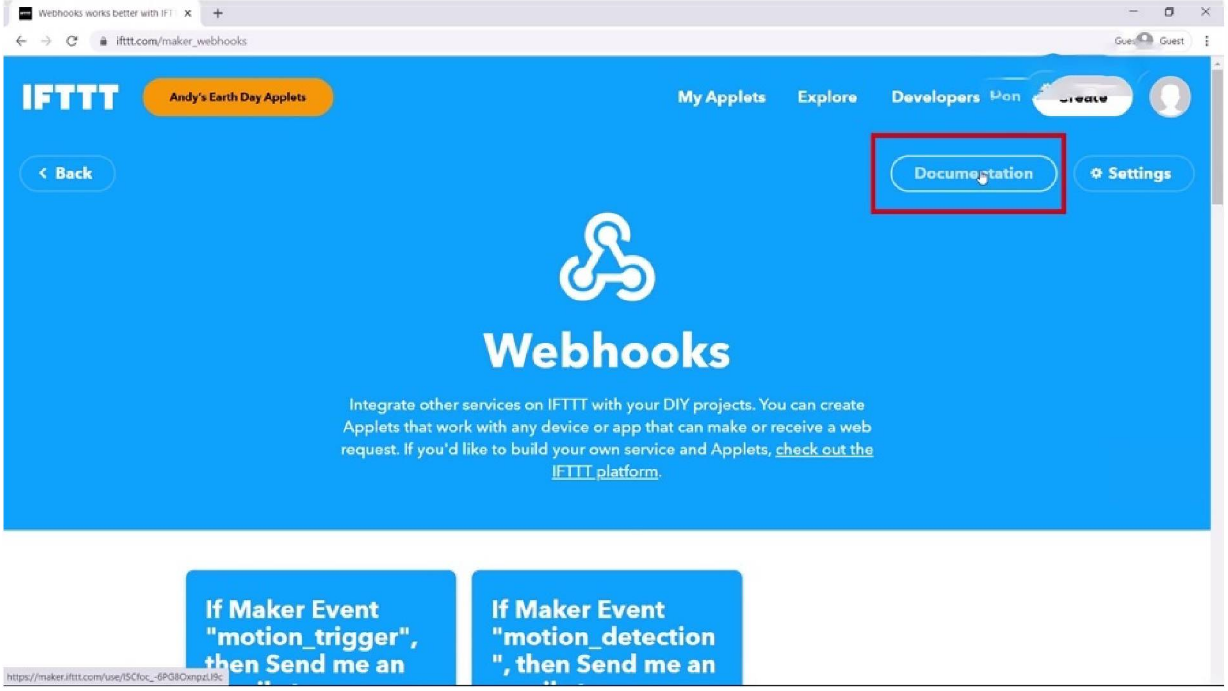
Manage your Applets

You can scroll through a list of them here or create your own. Learn more

My Applets

You have 1 available Applet. Upgrade to Pro to create unlimited Applets. Learn more

Now Go to my services from your profile. And click on Webhooks.

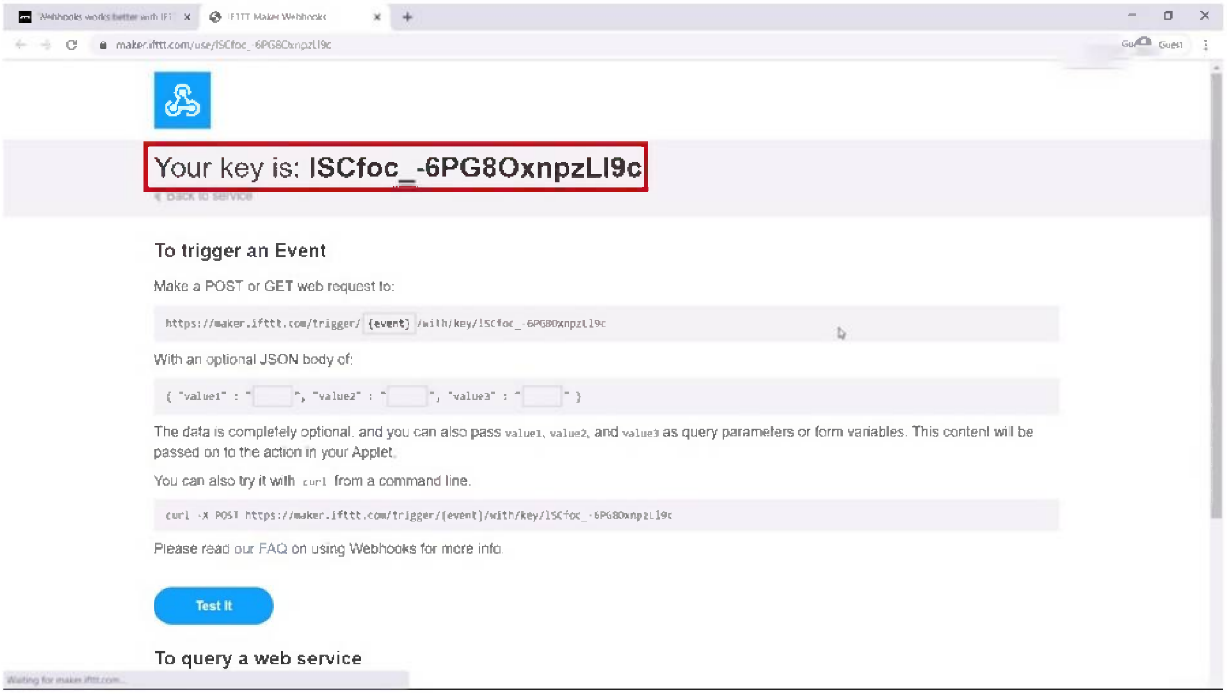


Webhooks works better with IFT
ifttt.com/maker_webhooks
Guest
IFTTT
Andy's Earth Day Applets
My Applets
Explore
Developers
Create
Back
Documentation
Settings
If Maker Event "motion_trigger", then Send me an
If Maker Event "motion_detection", then Send me an
https://maker.ifttt.com/use/lSCfoc_-6PG8OxnpzLl9c

Webhooks

Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform.

And click on documentation.



Webhooks works better with IFT | IFTTT Maker Webhooks

maker.ifttt.com/use/lSCfoc_-6PG8OxnpzLl9c

Your key is: lSCfoc_-6PG8OxnpzLl9c

◂ Back to service

To trigger an Event

Make a POST or GET web request to:

```
https://maker.ifttt.com/trigger/ {event} /with/key/lSCfoc_-6PG8OxnpzLl9c
```

With an optional JSON body of:

```
{ "value1" : "    ", "value2" : "    ", "value3" : "    " }
```

The data is completely optional, and you can also pass value1, value2, and value3 as query parameters or form variables. This content will be passed on to the action in your Applet.

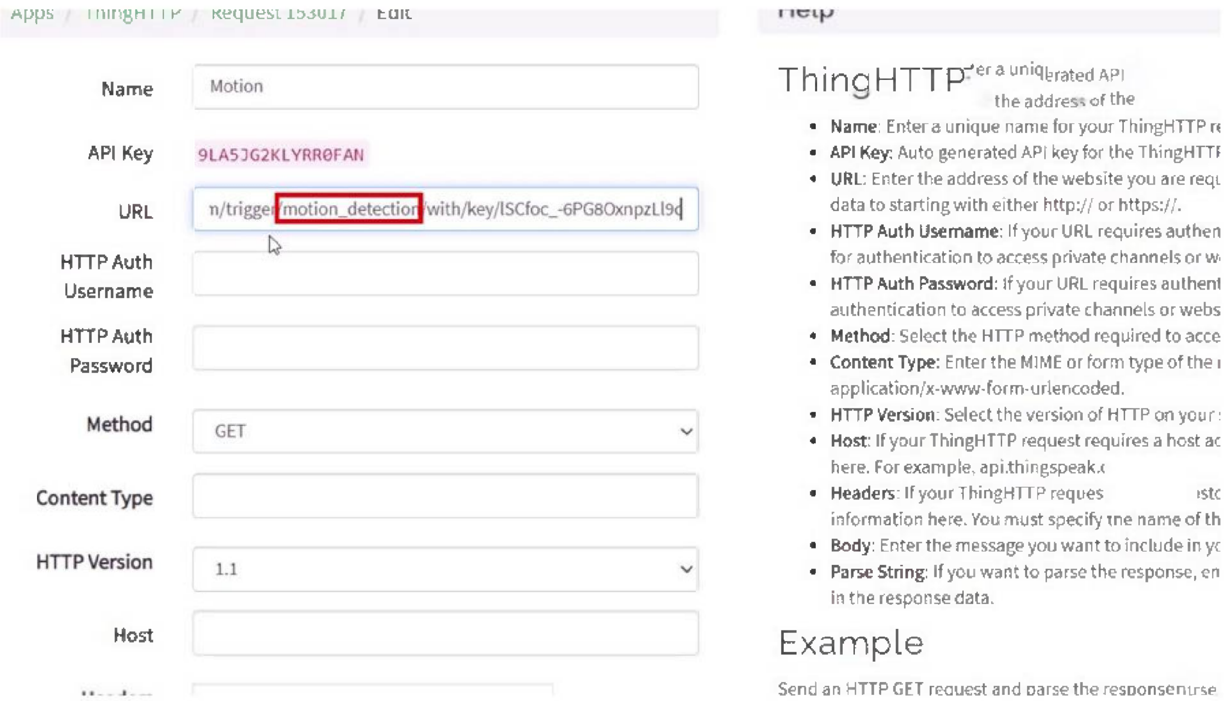
You can also try it with curl from a command line.

```
curl -X POST https://maker.ifttt.com/trigger/{event}/with/key/lSCfoc_-6PG8OxnpzLl9c
```

Please read our FAQ on using Webhooks for more info.

To query a web service

Here you can find your URL with your key to post or get a Web request, now you have all the necessary credentials to use IFTTT with Thing Speak. Now, go to the Thing Speak web page, click on apps and go to the ThingHTTP. Here enter a suitable name and copy and paste the URL from web hooks documentation here.



Apps / ThingHTTP / Request 153017 / Edit

Name	Motion
API Key	9LA5JG2KLYRR0FAN
URL	n/trigger/motion_detection/with/key/lSCfoc_-6PG8OxnpzLl9c
HTTP Auth Username	
HTTP Auth Password	
Method	GET
Content Type	
HTTP Version	1.1
Host	

Help

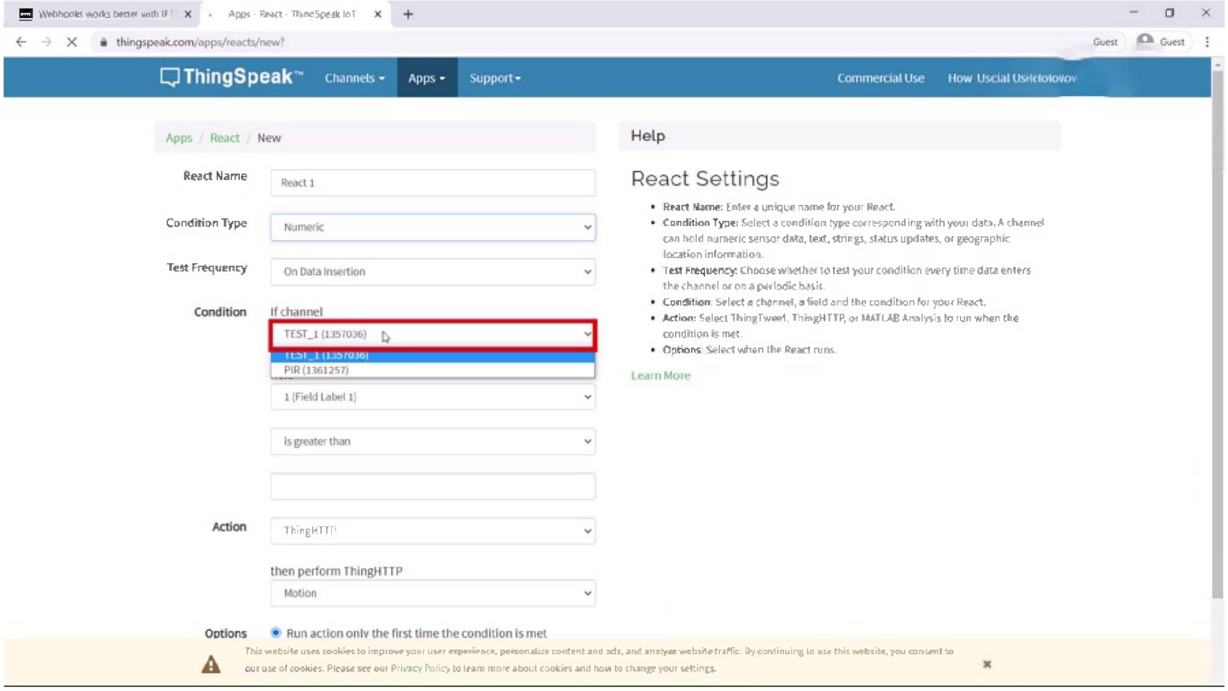
ThingHTTP

- **Name**: Enter a unique name for your ThingHTTP re
- **API Key**: Auto generated API key for the ThingHTTP
- **URL**: Enter the address of the website you are requ data to starting with either http:// or https://.
- **HTTP Auth Username**: If your URL requires authen for authentication to access private channels or w
- **HTTP Auth Password**: If your URL requires authent authentication to access private channels or webs
- **Method**: Select the HTTP method required to acce
- **Content Type**: Enter the MIME or form type of the application/x-www-form-urlencoded.
- **HTTP Version**: Select the version of HTTP on your
- **Host**: If your ThingHTTP request requires a host ac here. For example, api.thingspeak.
- **Headers**: If your ThingHTTP reques information here. You must specify the name of th
- **Body**: Enter the message you want to include in yo
- **Parse String**: If you want to parse the response, en in the response data.

Example

Send an HTTP GET request and parse the response

Do not forget to add your event name in the URL.



React Name: React 1

Condition Type: Numeric

Test Frequency: On Data Insertion

Condition: If channel TEST_1 (1357036)

TEST_1 (1357036)
PIR (1361257)

1 (Field Label 1)

is greater than

Action: ThingHTTP

then perform ThingHTTP: Motion

Options: ● Run action only the first time the condition is met

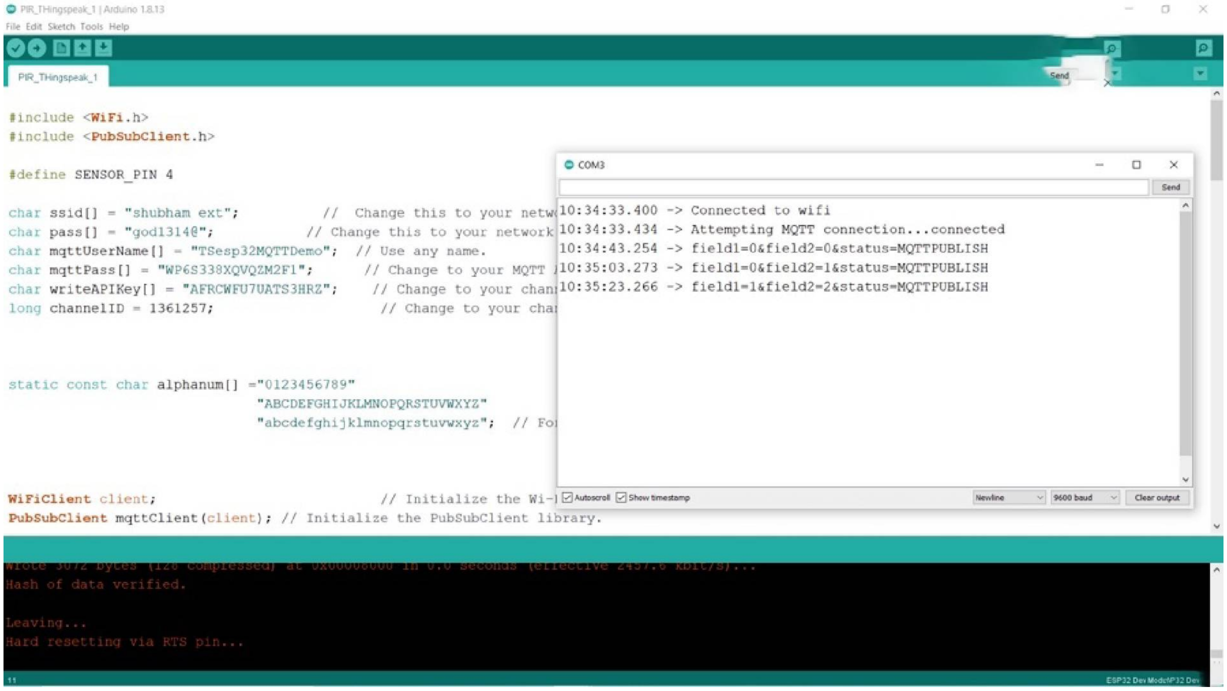
React Settings

- React Name: Enter a unique name for your React.
- Condition Type: Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- Test Frequency: Choose whether to test your condition every time data enters the channel or on a periodic basis.
- Condition: Select a channel, a field and the condition for your React.
- Action: Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- Options: Select when the React runs.

Learn More

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Now, again, go to Apps and use the React app, click on the new react, give an appropriate name and change condition typed to numeric. Now select the channel where we are publishing the sensor data and select the appropriate field. Now select greater then and put the value as zero. This means that whenever motion is detected and the value is published, it will trigger our applet. Now select Thing HTTP and the action to be performed and click on save react. We are now done with all the setup for automating our system.



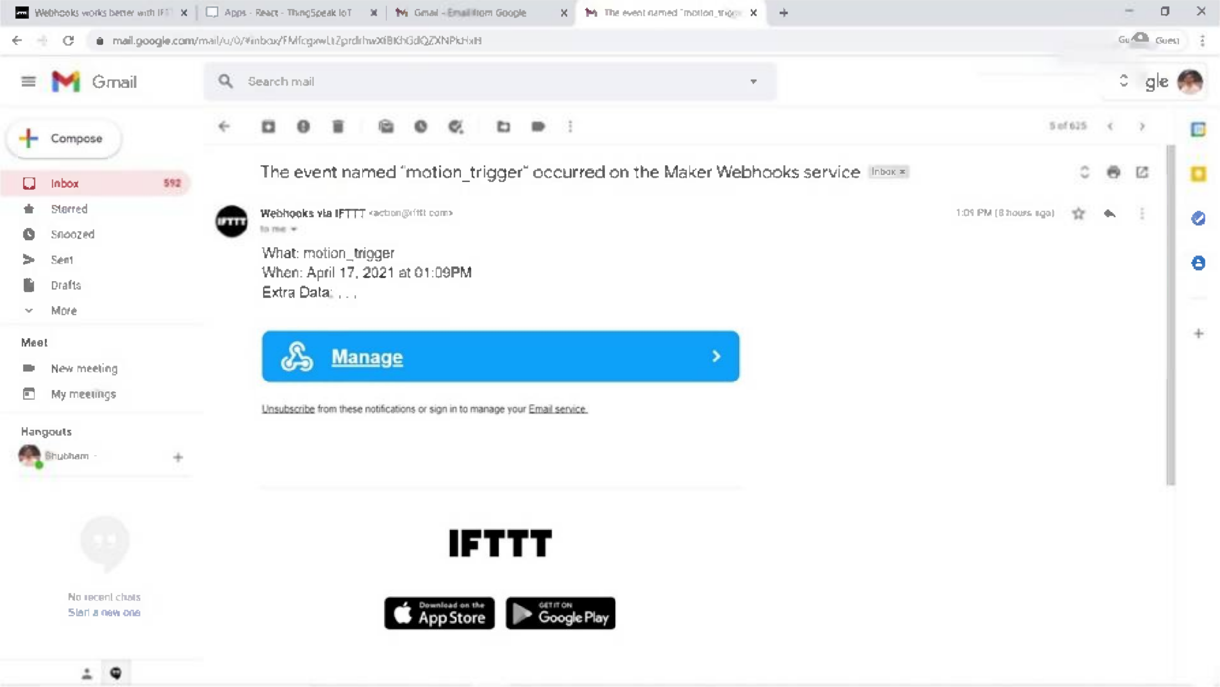
PIR_THingspeak_1 | Arduino 1.8.13
File Edit Sketch Tools Help
PIR_THingspeak_1
COM3
Send
☑ Autoscroll ☑ Show timestamp
Newline
9600 baud
Clear output
11
ESP32 Dev Module, ESP32 Dev

```
#include <WiFi.h>
#include <PubSubClient.h>

#define SENSOR_PIN 4

char ssid[] = "shubham ext";          //  Change this to your netw
char pass[] = "god1314@";           // Change this to your network
char mqttUserName[] = "TSesp32MQTTDemo";  // Use any name.
char mqttPass[] = "WP6S338XQVQZM2F1";      // Change to your MQTT 
char writeAPIKey[] = "AFRCWFU7UATS3HRZ";    // Change to your chan
long channelID = 1361257;                    // Change to your cha



static const char alphanum[] ="0123456789"
                                "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                                "abcdefghijklmnopqrstuvwxyz";  // Fo



WiFiClient client;                           // Initialize the Wi-
PubSubClient mqttClient(client); // Initialize the PubSubClient library.
```

```
10:34:33.400 -> Connected to wifi
10:34:33.434 -> Attempting MQTT connection...connected
10:34:43.254 -> field1=0&field2=0&status=MQTTPUBLISH
10:35:03.273 -> field1=0&field2=1&status=MQTTPUBLISH
10:35:23.266 -> field1=1&field2=2&status=MQTTPUBLISH
```

```
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 2457.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Here you need to make the same connections with your PIR sensor and ESP32 and run the same code which we did in the last project. This will publish the data. Additionally, whenever motion is detected and the value one is sent to Thing Speak, the Thing Speak app react will send this information to ThingHTTP, which will pass this to IFTTT through the URL we have already mentioned.



Webhooks works better with IFTTT
Apps - React - ThingSpeak IoT
Gmail - Email from Google
The event named "motion_trigg
mail.google.com/mail/u/0/#inbox/FMfcgxwLtZprdrhwXlBKhGdQZXNPkHxH
Guest
Gmail
Search mail
Compose
Inbox 592
Starred
Snoozed
Sent
Drafts
More
Meet
New meeting
My meetings
Hangouts
Shubham
No recent chats
Start a new one
5 of 625
Inbox
Download on the App Store
GET IT ON Google Play

The event named "motion_trigger" occurred on the Maker Webhooks service

Webhooks via IFTTT <action@ifttt.com>
to me
1:09 PM (8 hours ago)

What: motion_trigger
When: April 17, 2021 at 01:09PM
Extra Data: , , ,

Manage

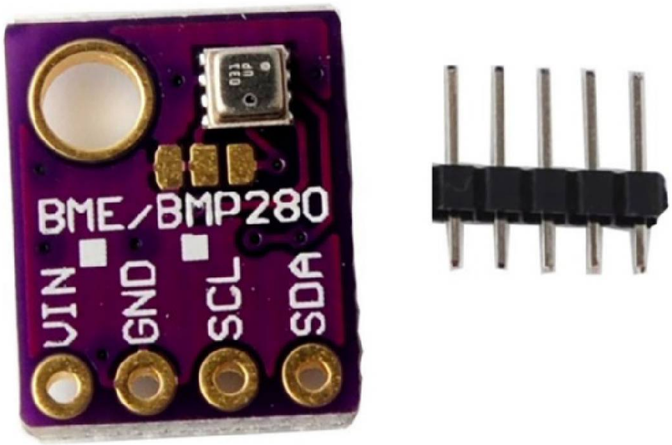
Unsubscribe from these notifications or sign in to manage your Email service.

IFTTT

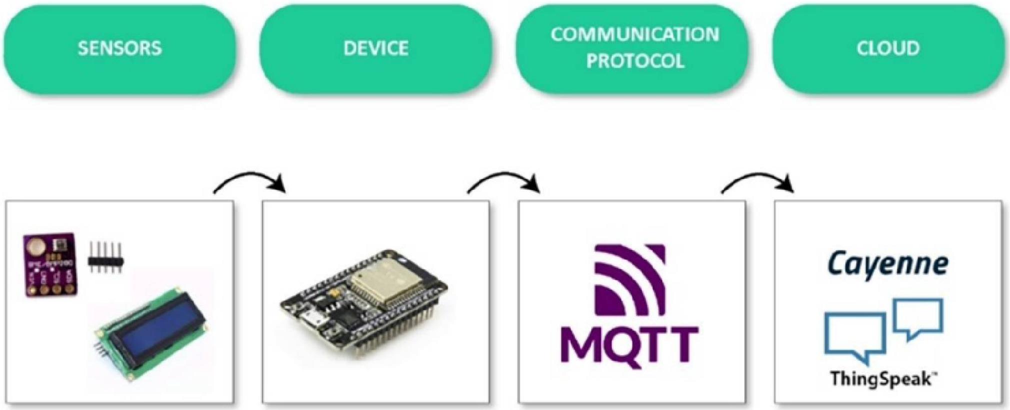
IFTTT will then send an email notification to us mentioning about motion being detected.

CAPSTONE PROJECT 2 WEATHER STATION

we will cover the following topics IoT architecture comparison, configuring devices, interfacing the sensor and actuator. Now that we have worked with Thing Speak and IFTTT, you must have an idea of developing projects using the Thing Speak API. You may have also noticed the difference in building projects with Cayenne and with Thing Speak. In general You may have noticed that the architecture is overall the same and the only difference is in the API and how it functions. Let's compare our weather station project.



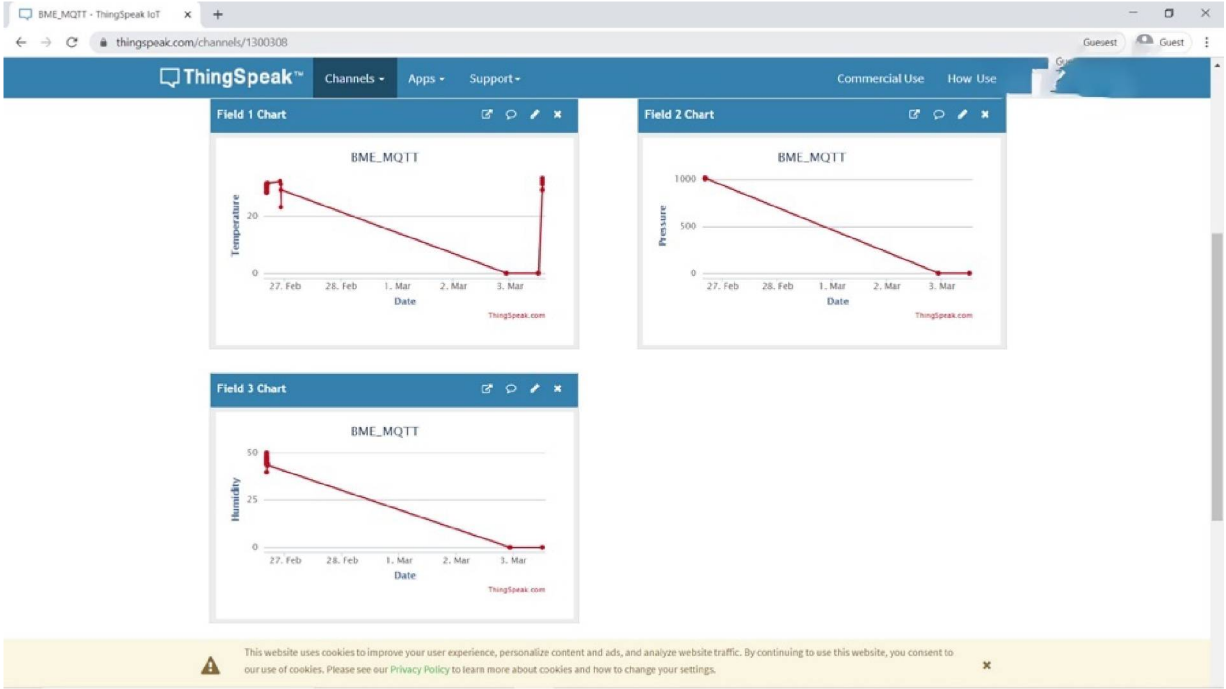
We have the BME280 sensor which measures temperature, humidity and pressure. Moreover, as you know, we are also able to calculate and measure altitude.



We have interfaced this sensor with the ESP32. The ESP32 is responsible for connecting to the Internet and sending all the data measured by the

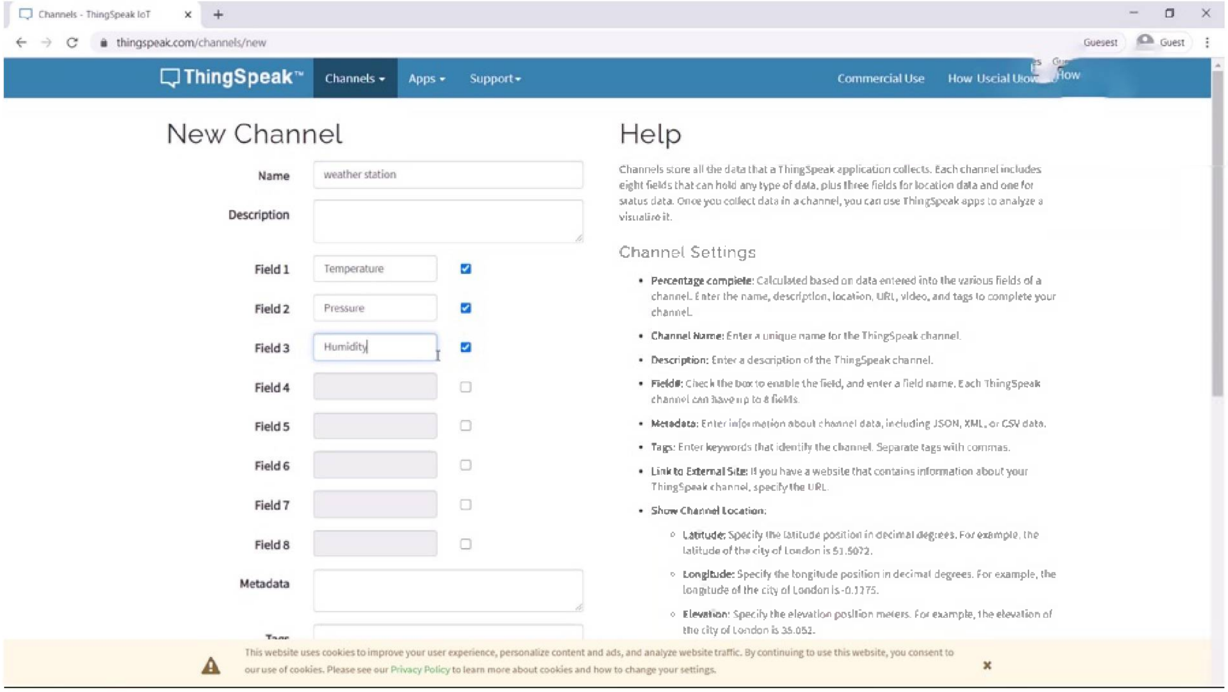
BME280 sensor, using the MQTT protocol to the respective cloud either Cayenne or Thing Speak. We then read the data from the cloud using another ESP32 and display notification on an LCD.

While Cayenne is easy to use, the Drag and Drop feature especially can be used for prototype building. However, due to some limitations on interfacing and the nature of Cayenne, it is suitable only to build beginner level projects.



This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Thing Speak is more complex and more suitable for those who can write code. Due to this, Thing Speak can even be used for high end applications. Other apps such as React, Thing Tweet and ThingHTTP and their integration with mathworks bring in additional features that can be used to develop high end IoT applications. Now let's get started with building the weather station, using Thing Speak. First, we will configure both the ESP32s and keep all the credentials needed in hand. First, we will create a new channel where we will publish the data measured by the BME280 sensor.



thingspeak.com/channels/new

New Channel

Name: weather station

Description:

Field 1: Temperature ☑

Field 2: Pressure ☑

Field 3: Humidity ☑

Field 4: ☐

Field 5: ☐

Field 6: ☐

Field 7: ☐

Field 8: ☐

Metadata:

Help

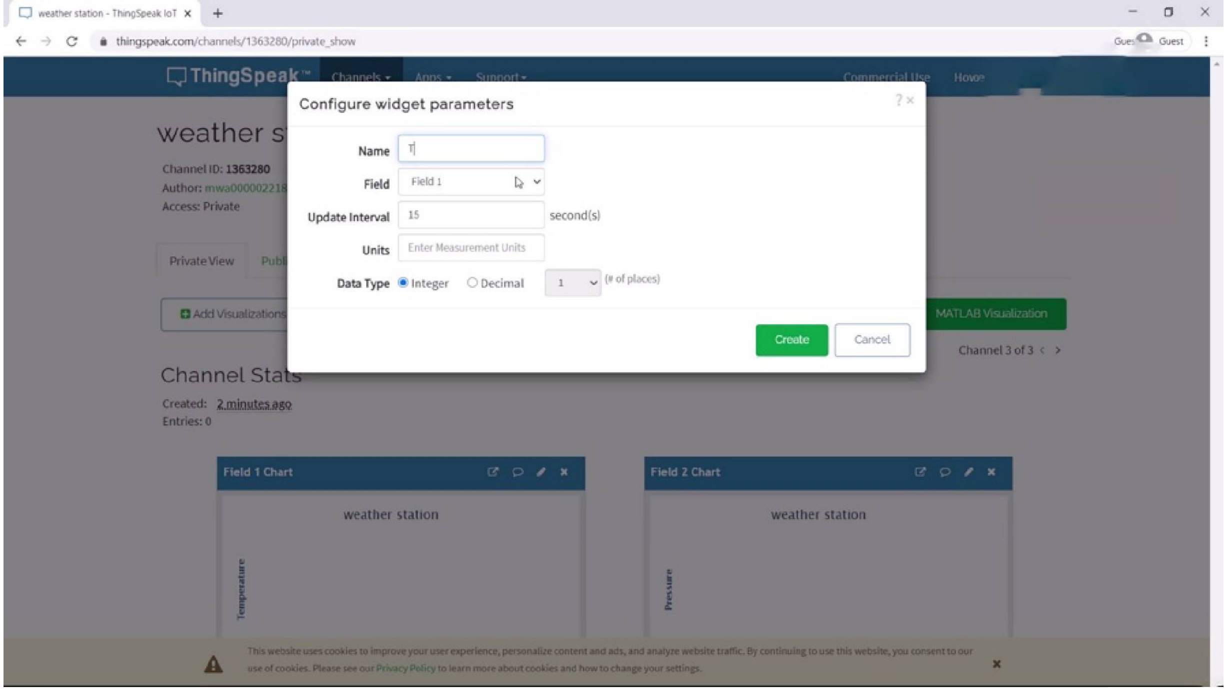
Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze a visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

To create a new channel. Click on new channel, give an appropriate name. Now select three fields, which will be temperature, pressure and humidity respectively.



thingspeak.com/channels/1363280/private_show

weather s

Channel ID: 1363280
Author: mwa000002218
Access: Private

Configure widget parameters

Name: T

Field: Field 1

Update Interval: 15 second(s)

Units: Enter Measurement Units

Data Type: ● Integer ○ Decimal 1 (# of places)

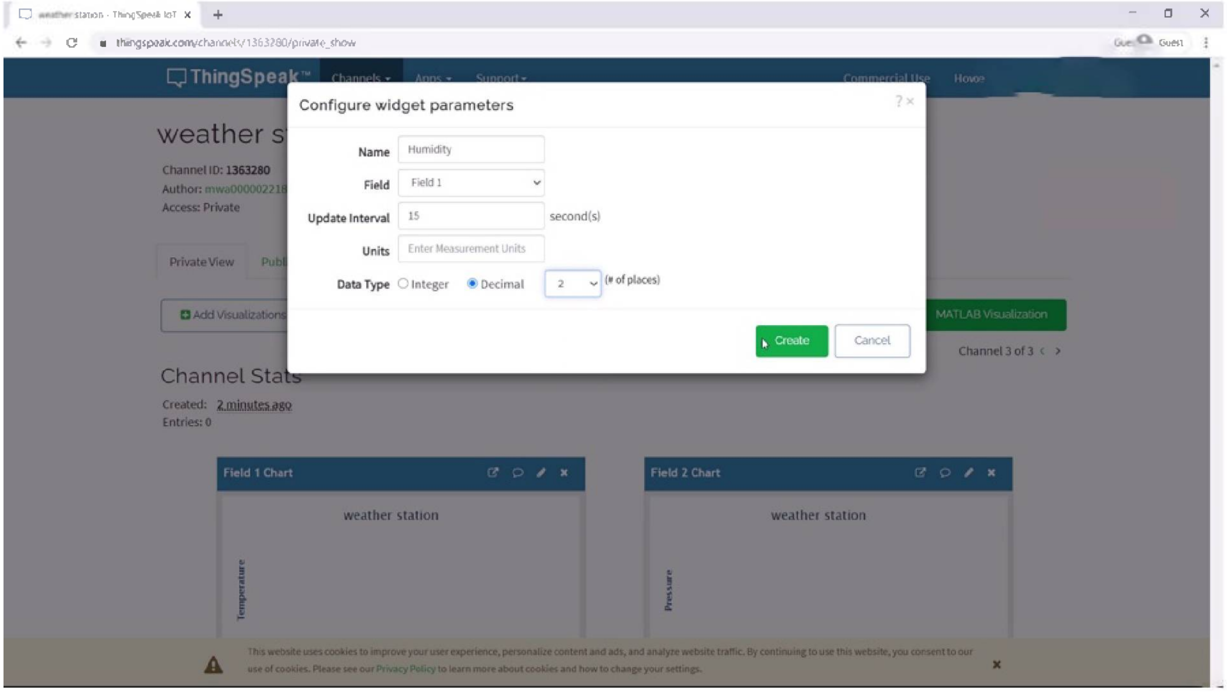
Create Cancel

Channel Stats

Created: 2 minutes ago
Entries: 0

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

We will additionally add to improve the display of our channel.


Configure widget parameters

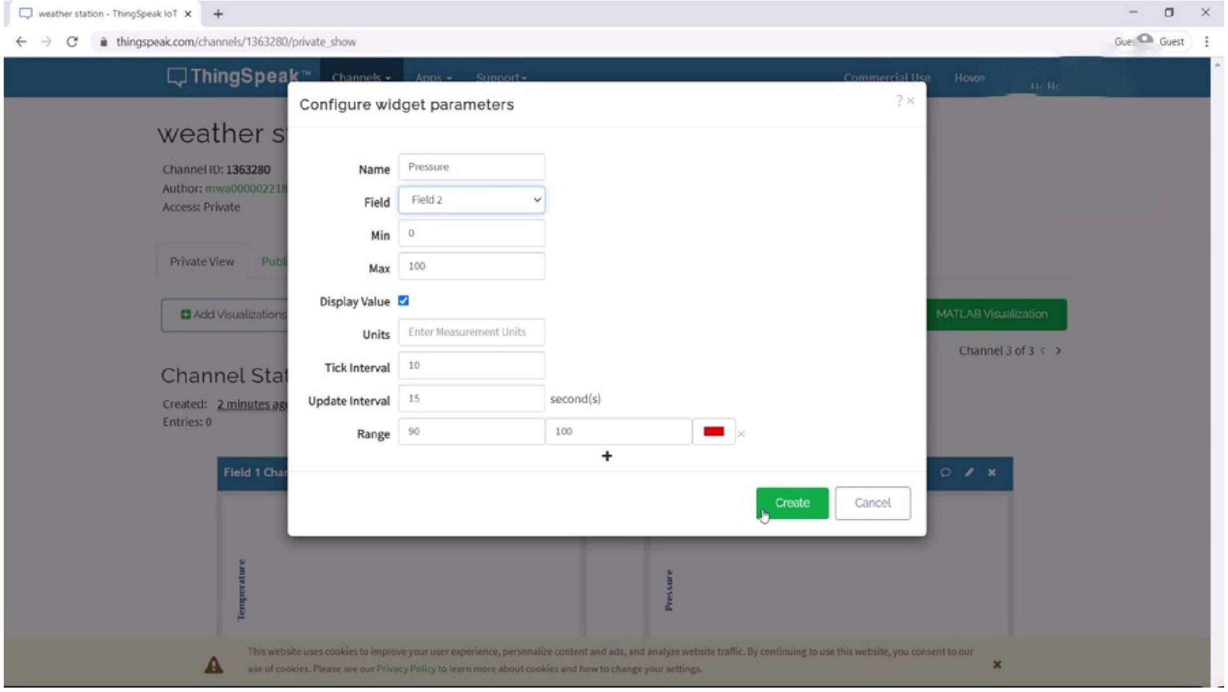
Name: Humidity

Field: Field 1

Update Interval: 15 second(s)

Units: Enter Measurement Units

Data Type: ○ Integer ● Decimal 2 (# of places)


Configure widget parameters

Name: Pressure

Field: Field 2

Min: 0

Max: 100

Display Value: ☑

Units: Enter Measurement Units

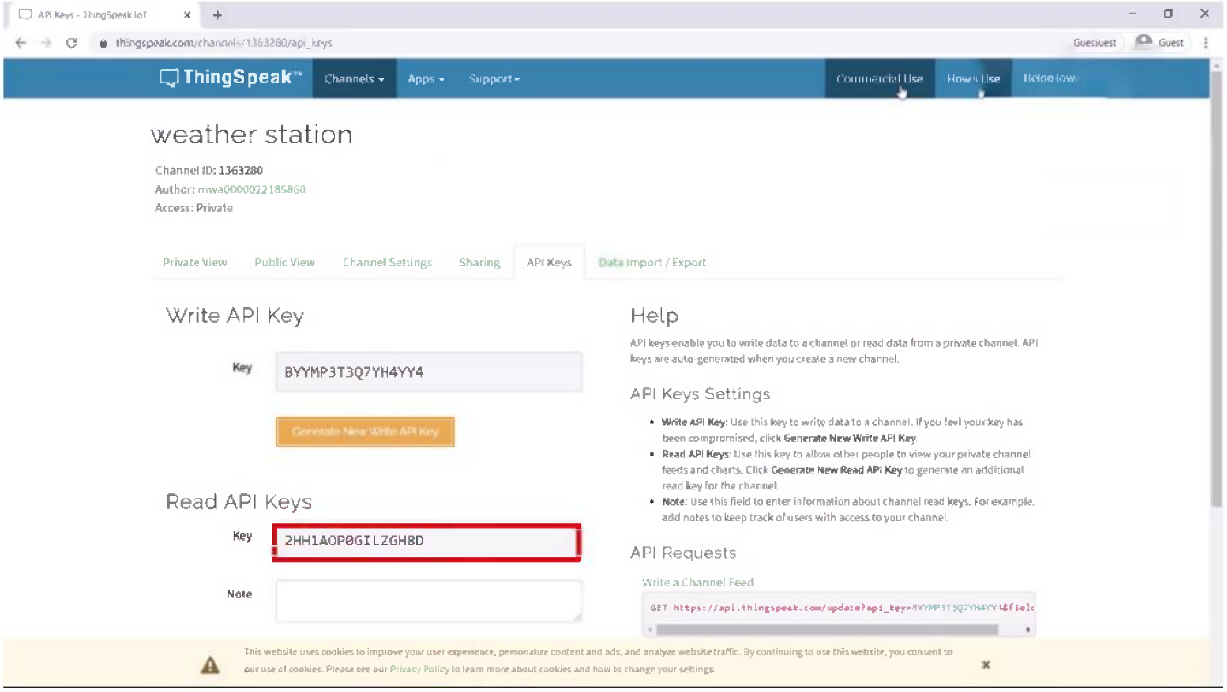
Tick Interval: 10

Update Interval: 15 second(s)

Range: 90 100 🟥

weather station - ThingSpeak IoT
thingspeak.com/channels/1363280/private_show
ThingSpeak™
Channels
Apps
Support
Commercial Use
How to
weather s
Channel ID: 1363280
Author: mwa0000022l8
Access: Private
Private View
Publ
Add Visualizations
MATLAB Visualization
Channel 3 of 3
Channel Stat
Created: 2 minutes ag
Entries: 0
Field 1 Char
Temperature
Pressure
+
Create
Cancel
This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

We will add the numeric display and the Gauge display Widgets. Now go to API keys and note down both the read and write API keys.



weather station

Channel ID: 1363280
Author: mwa0000022185860
Access: Private

Write API Key

Key: BYYMP3T3Q7YH4YY4

Read API Keys

Key: 2HH1AOP0GILZGH8D

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

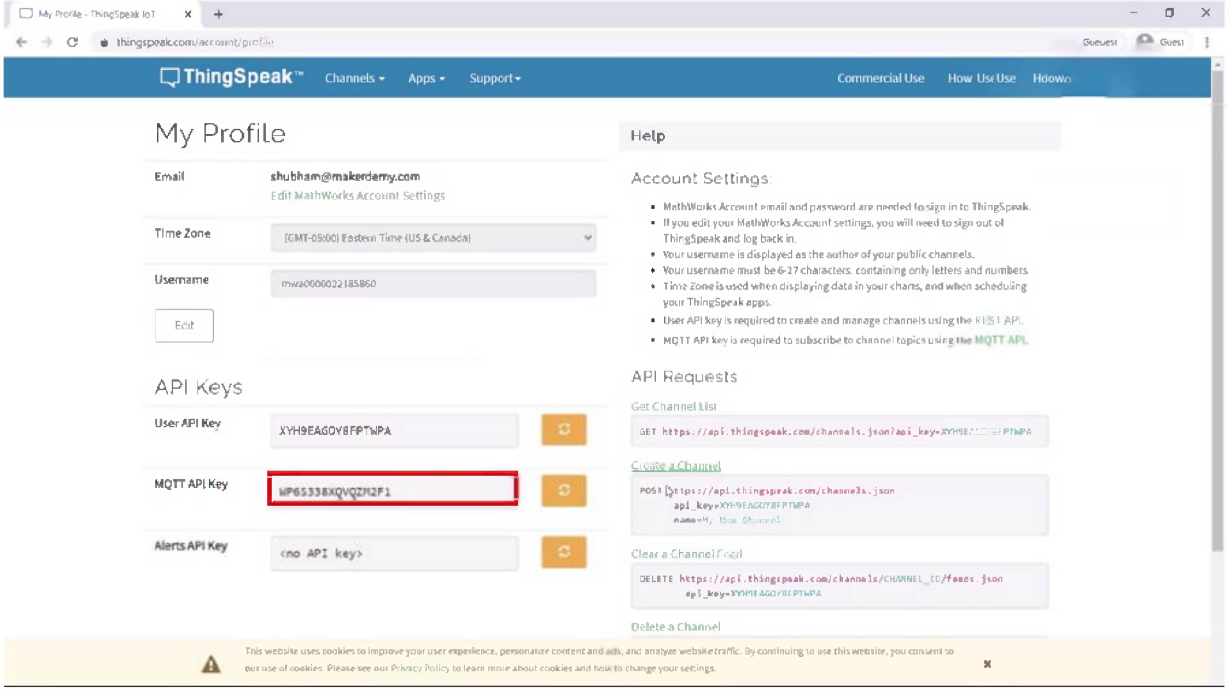
API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=BYYMP3T3Q7YH4YY4&field
```

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

You will need the right API key to publish data to the channel while you need the read API key to subscribe and get data from the channel.



My Profile

Email: shubham@makerdemy.com
Edit MathWorks Account Settings

Time Zone: (GMT-05:00) Eastern Time (US & Canada)

Username: mwa0000022185860

API Keys

User API Key: XYH9EAGOY8FPTWPA

MQTT API Key: WP6533BXQVQZM2F1

Alerts API Key: <no API key>

Help

Account Settings:

- MathWorks Account email and password are needed to sign in to ThingSpeak.
- If you edit your MathWorks Account settings, you will need to sign out of ThingSpeak and log back in.
- Your username is displayed as the author of your public channels.
- Your username must be 6-27 characters, containing only letters and numbers.
- Time Zone is used when displaying data in your charts, and when scheduling your ThingSpeak apps.
- User API key is required to create and manage channels using the REST API.
- MQTT API key is required to subscribe to channel topics using the MQTT API.

API Requests

Get Channel List

```
GET https://api.thingspeak.com/channels.json?api_key=XYH9EAGOY8FPTWPA
```

Create a Channel

```
POST https://api.thingspeak.com/channels.json
     api_key=XYH9EAGOY8FPTWPA
     name=My New Channel
```

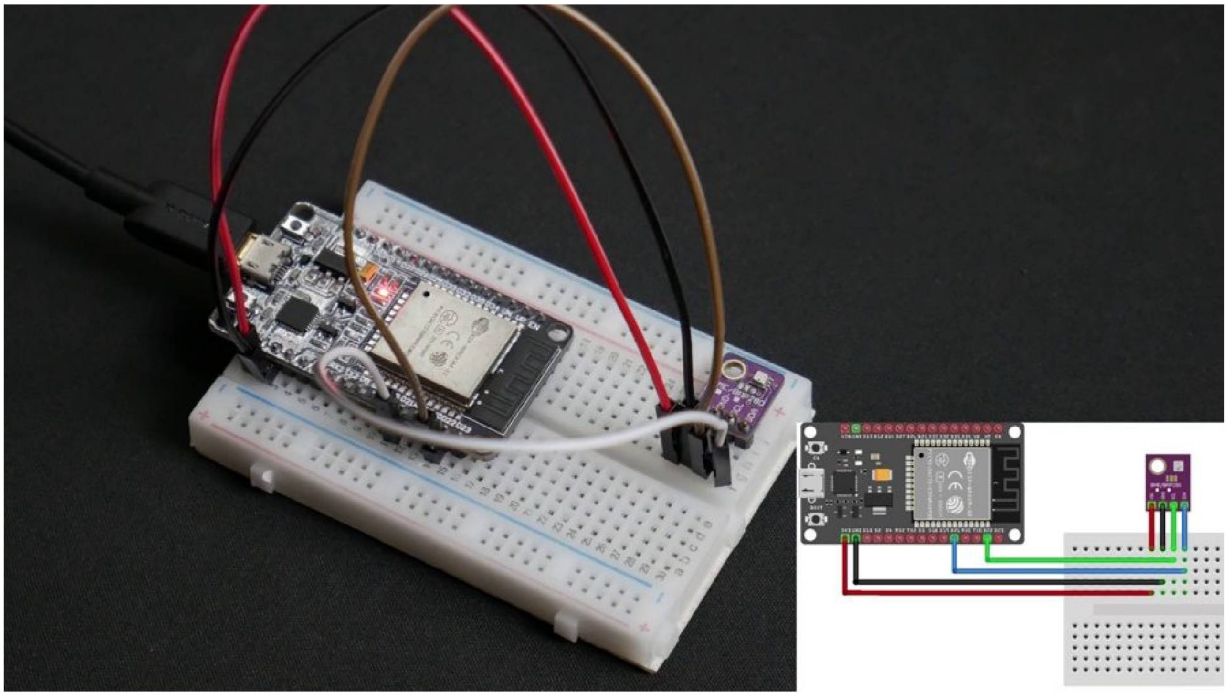
Clear a Channel Feed

```
DELETE https://api.thingspeak.com/channels/CHANNEL_ID/feeds.json
       api_key=XYH9EAGOY8FPTWPA
```

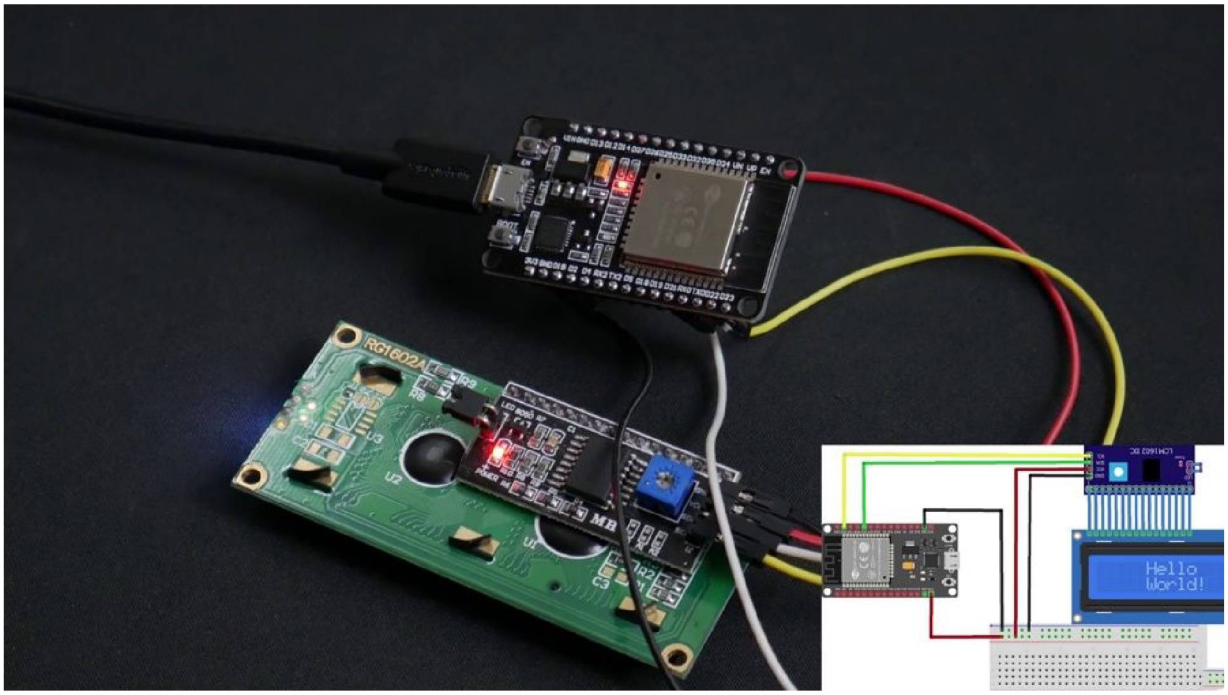
Delete a Channel

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

You will also need thee MQTT API key from the My profile option here. Now we will make connections and complete the hardware part of the project, first we will connect the BME280 Sensor to one of the ESP32.

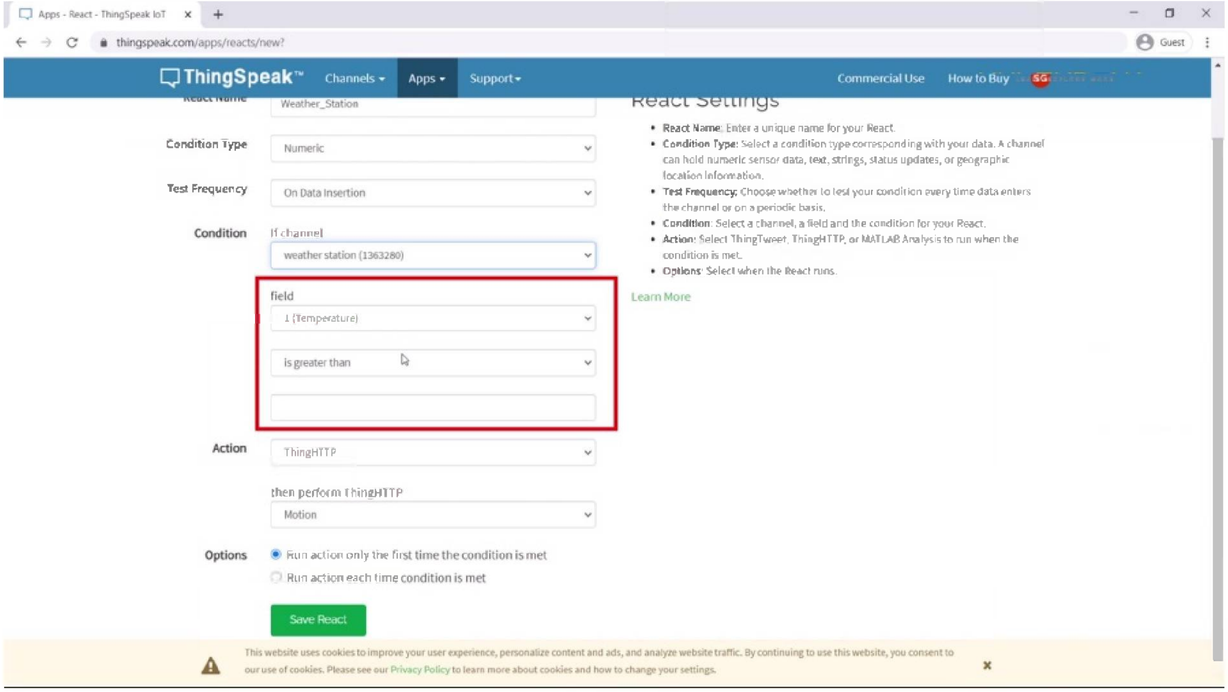


Connect the Vin of the BME280 sensor to the 3.3V pin. of the ESP32. Now connect the ground of the sensor to the ground of the ESP32. Now connect the serial clock or SCL pin of the sensor to GPIO 22 of the ESP32. This pin also acts as the SCL pin for ESP32. We will now connect the SDA or serial data transfer pin of the sensor to GPO 21 of the ESP32. You are now ready with your hardware for the publish side of the weather station.



Now we need to connect the LCD with the second ESP32, which will receive data from Thing Speak and display notifications accordingly. First, we will connect the Vin of the I2C module of the LCD to the Vin of the ESP32. Remember, we need a 5V supply for LCD and a 3.3V supply for the BME280. Now connect the ground pin of both the actuator and the ESP32. As we are using the I2C module, we will connect the Serial data pin of the module to GPIO21 of the ESP32 and we will connect the Serial clock pin of the module to GPA 22 of the ESP32. We are now done with the connections for the subscribe side of the weather station. Now our hardware is ready and we need to make the necessary software changes to run our weather station successfully.

we will cover the following topics. Configuring Thing Speak and IFTTT, code configuration and explanation, implementing trigger notification for the 16x2 LCD, end to end IoT project implementation. We have taken care of all the hardware parts and configured our devices to implement the end to end project. Now let's check the software part. First we will configure our IFTTT and Thing Speak according to our needs. First, let's check. IFTTT. Go to your IFTTT account and create a new applet, you will now arrive at the if this then that page. We have to now select the trigger and action for sending notifications. Now let's select the input action. Click on the word this. Enter Webhooks in the search field and select the web hooks card. After you select web hooks as the trigger, click on receive a Web request card to continue. Complete the trigger fields. Enter an event name and then click on Create Trigger. Now to select the resulting action, click on the word that. Now search from email and select the send me an email card. Now click on Create Action and then continue. Click on Finish to create the applet. Now, go to my services from here. Click on Web hooks and then documentation. You will need the URL from the documentation. Now let's configure Thing Speak, go to your Thing Speak account and then go to ThingHTTP from here. Here enter a suitable name and copy and paste the URL from webhook documentation here. Do not forget to add your exact event name in the URL.



React Name: Weather_Station

Condition Type: Numeric

Test Frequency: On Data Insertion

Condition: If channel weather station (1363280)

field 1 (Temperature)

is greater than

Action: ThingHTTP

then perform ThingHTTP Motion

Options:

● Run action only the first time the condition is met

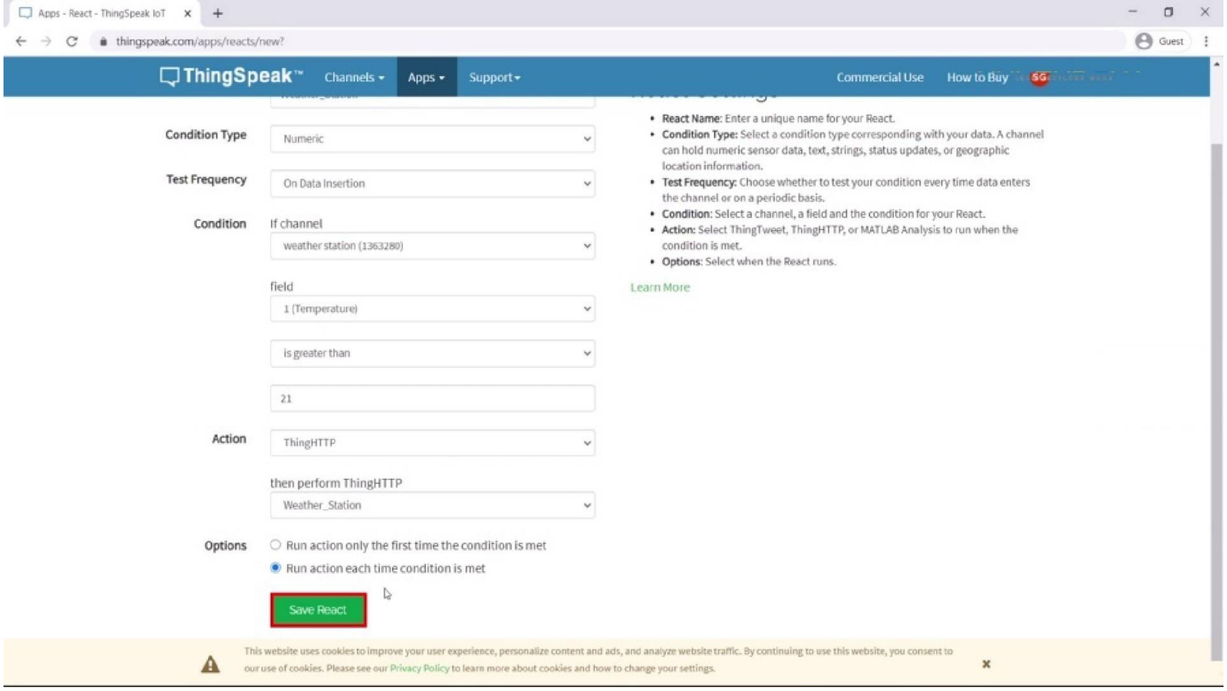
○ Run action each time condition is met

React Settings

- React Name: Enter a unique name for your React.
- Condition Type: Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- Test Frequency: Choose whether to test your condition every time data enters the channel or on a periodic basis.
- Condition: Select a channel, a field and the condition for your React.
- Action: Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- Options: Select when the React runs.

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Now again Go to apps and use the react app. Click on new react, give an appropriate name and change condition type to numeric. Now select the channel where we are publishing sensorZ data and select the appropriate field. Now select the greater than option and put in the value. According to you, this means that whenever the temperature value goes above the threshold value, it will trigger your applet.



Condition Type: Numeric

Test Frequency: On Data Insertion

Condition: If channel weather station (1363280)

field 1 (Temperature)

is greater than

21

Action: ThingHTTP

then perform ThingHTTP Weather_Station

Options:

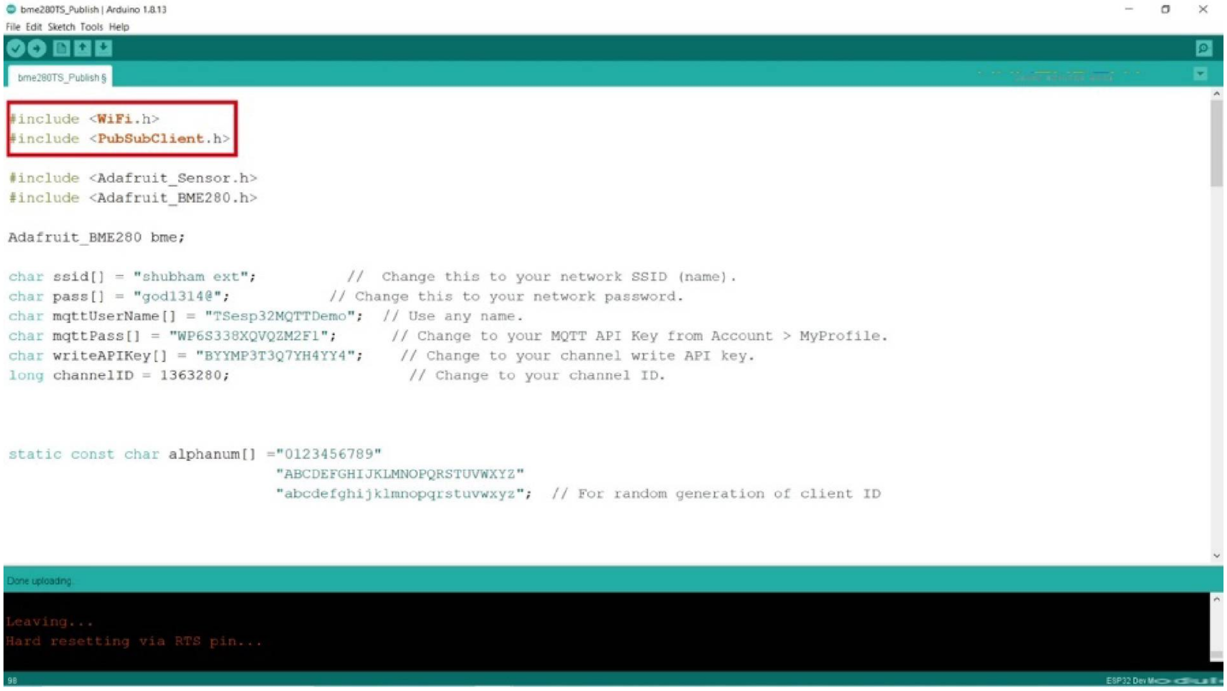
○ Run action only the first time the condition is met

● Run action each time condition is met

- React Name: Enter a unique name for your React.
- Condition Type: Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- Test Frequency: Choose whether to test your condition every time data enters the channel or on a periodic basis.
- Condition: Select a channel, a field and the condition for your React.
- Action: Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- Options: Select when the React runs.

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Now select the ThingHTTP and the action to be performed and click on, save react. We are now done with all the setup for automating our system.

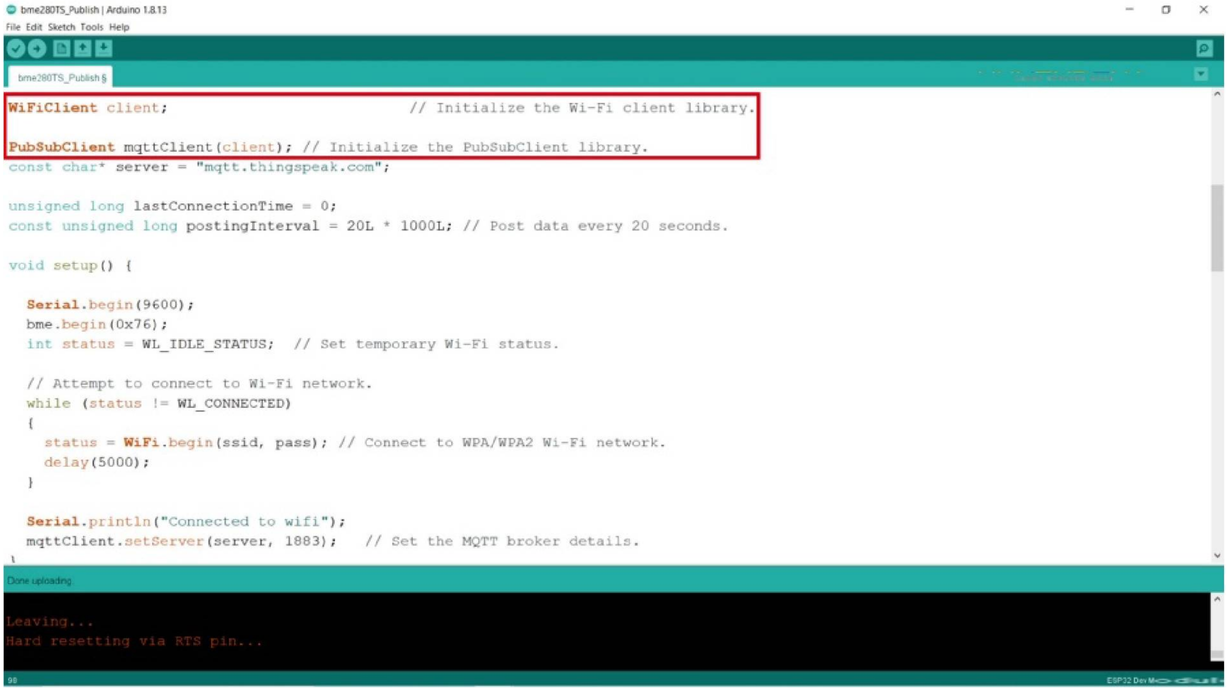


bme280TS_Publish | Arduino 1.8.13
File Edit Sketch Tools Help
bme280TS_Publish §

```
#include <WiFi.h>
#include <PubSubClient.h>

#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme;

char ssid[] = "shubham ext";          //  Change this to your network SSID (name).
char pass[] = "god1314@";           // Change this to your network password.
char mqttUserName[] = "TSesp32MQTTDemo";  // Use any name.
char mqttPass[] = "WP6S338XQVQZM2F1";      // Change to your MQTT API Key from Account > MyProfile.
char writeAPIKey[] = "BYYMP3T3Q7YH4YY4";    // Change to your channel write API key.
long channelID = 1363280;                  // Change to your channel ID.


static const char alphanum[] ="0123456789"
                              "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                              "abcdefghijklmnopqrstuvwxyz";  // For random generation of client ID
```

```
Leaving...
Hard resetting via RTS pin...
```

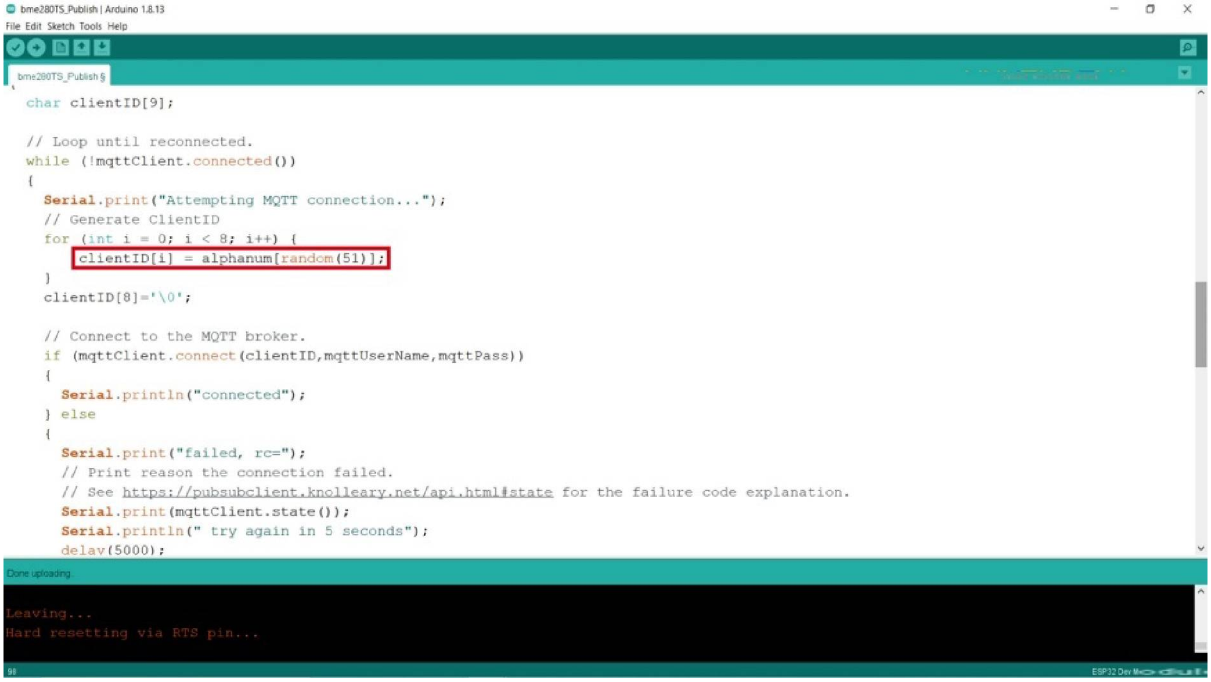
Now let's take the code for our project. First, we will check the publish side of our project. Here We have included two libraries, the Wi-Fi and the PubSub Client Library. They are responsible for connecting to the Internet and to send data to Thing Speak using MQTT. In the next step, we include the Adafruit libraries for the BME280 sensors.


bme280TS_Publish | Arduino 1.8.13
File Edit Sketch Tools Help
bme280TS_Publish §
Done uploading
98
ESP32 Dev Module

```
WiFiClient client;                          // Initialize the Wi-Fi client library.

PubSubClient mqttClient(client); // Initialize the PubSubClient library.
const char* server = "mqtt.thingspeak.com";

unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 20L * 1000L; // Post data every 20 seconds.

void setup() {

  Serial.begin(9600);
  bme.begin(0x76);
  int status = WL_IDLE_STATUS;  // Set temporary Wi-Fi status.

  // Attempt to connect to Wi-Fi network.
  while (status != WL_CONNECTED)
  {
    status = WiFi.begin(ssid, pass); // Connect to WPA/WPA2 Wi-Fi network.
    delay(5000);
  }

  Serial.println("Connected to wifi");
  mqttClient.setServer(server, 1883);   // Set the MQTT broker details.
}
```

```
Leaving...
Hard resetting via RTS pin...
```

Here we have created an instance for the sensor. Now we will provide all the necessary credentials to connect to Thing Speak. First We have provided Wi-Fi SSID and password. Next, we have mentioned the MQTT API key , Write API Key and channel ID. This is needed for the random generation of the client ID. Here We have initialized both the Wi-Fi and PubSub client library. Now we will mention the server to which connection needs to be established. Here We have set our posting interval to twenty seconds. Here We have set the baud rate to 9600 and initialized the sensor with appropriate I2C address.

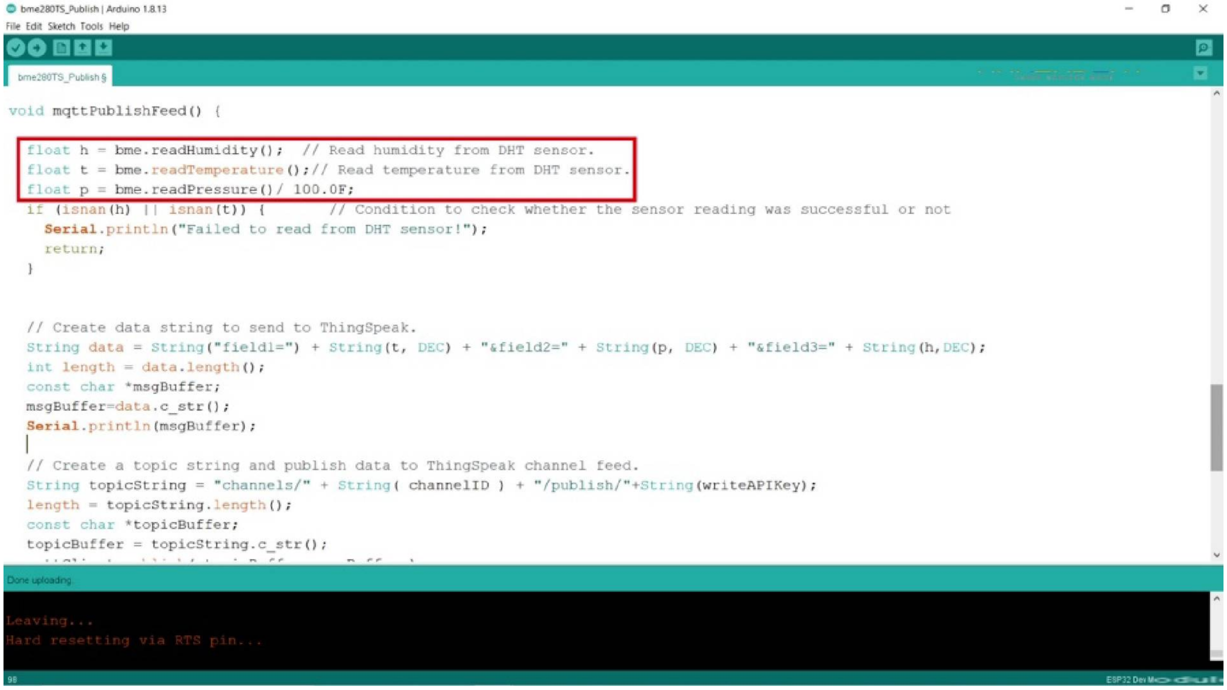

bme280TS_Publish | Arduino 1.8.13
File Edit Sketch Tools Help
bme280TS_Publish §

```
  char clientID[9];

  // Loop until reconnected.
  while (!mqttClient.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Generate ClientID
    for (int i = 0; i < 8; i++) {
        clientID[i] = alphanum[random(51)];
    }
    clientID[8]='\0';

    // Connect to the MQTT broker.
    if (mqttClient.connect(clientID,mqttUserName,mqttPass))
    {
      Serial.println("connected");
    } else
    {
      Serial.print("failed, rc=");
      // Print reason the connection failed.
      // See https://pubsubclient.knolleary.net/api.html#state for the failure code explanation.
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
```

Done uploading.

```
Leaving...
Hard resetting via RTS pin...
```

Here we will generate a new client ID each time to connect to the server. We have also written code to reconnect to the server every five seconds in case any issue occurs.

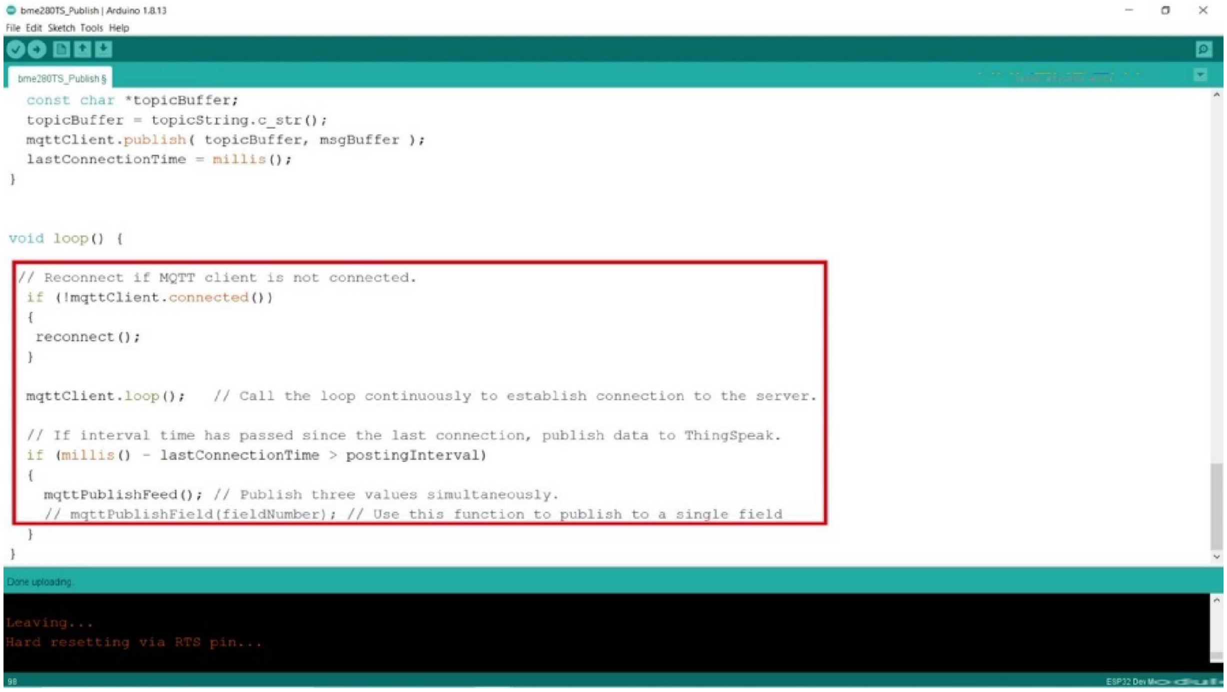

bme280TS_Publish | Arduino 1.8.13
File Edit Sketch Tools Help
bme280TS_Publish §

```
void mqttPublishFeed() {

  float h = bme.readHumidity();  // Read humidity from DHT sensor.
  float t = bme.readTemperature();// Read temperature from DHT sensor.
  float p = bme.readPressure()/ 100.0F;
  if (isnan(h) || isnan(t)) {       // Condition to check whether the sensor reading was successful or not
    Serial.println("Failed to read from DHT sensor!");
    return;
  }


  // Create data string to send to ThingSpeak.
  String data = String("field1=") + String(t, DEC) + "&field2=" + String(p, DEC) + "&field3=" + String(h,DEC);
  int length = data.length();
  const char *msgBuffer;
  msgBuffer=data.c_str();
  Serial.println(msgBuffer);

  // Create a topic string and publish data to ThingSpeak channel feed.
  String topicString = "channels/" + String( channelID ) + "/publish/"+String(writeAPIKey);
  length = topicString.length();
  const char *topicBuffer;
  topicBuffer = topicString.c_str();
```

Done uploading.

```
Leaving...
Hard resetting via RTS pin...
```

Here we are reading the sensor data. We are using float values to get accurate measurements from the sensor. Here we are creating a data string to send to Thing Speak. Here We have created a topic string and published data to the Thing Speak channel feed. We are using the format provided by Mathworks to create the topic string.

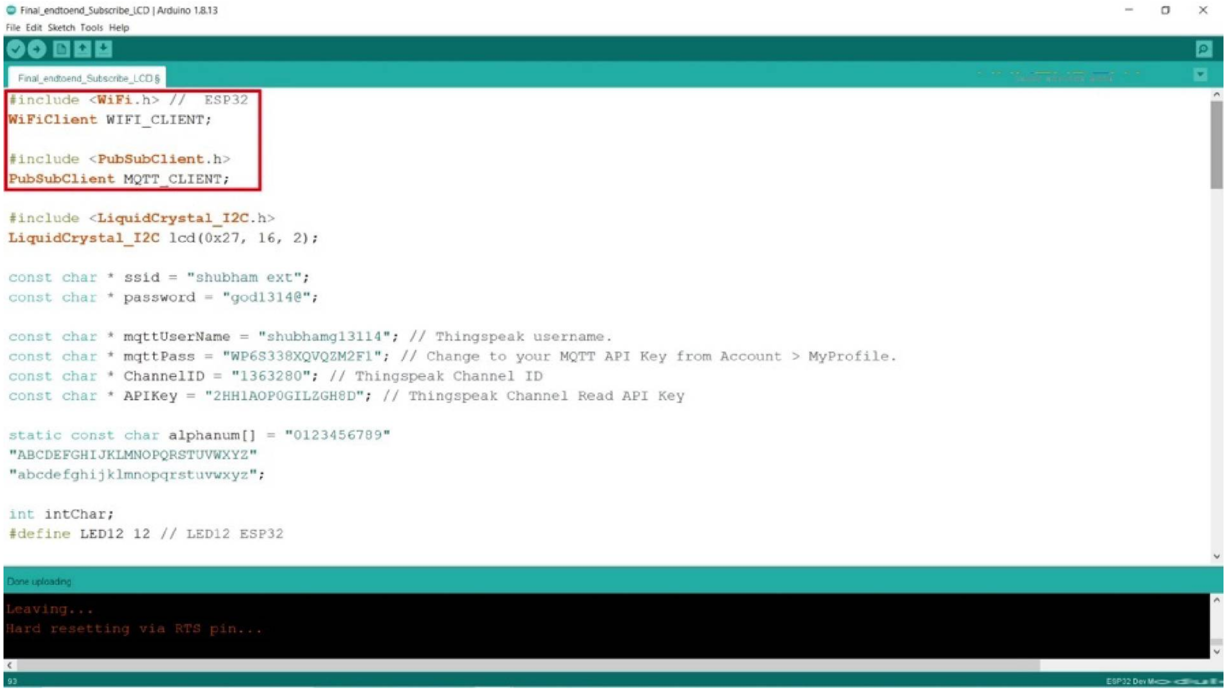

bme280TS_Publish | Arduino 1.8.13
File Edit Sketch Tools Help
bme280TS_Publish §

```
  const char *topicBuffer;
  topicBuffer = topicString.c_str();
  mqttClient.publish( topicBuffer, msgBuffer );
  lastConnectionTime = millis();
}


void loop() {

// Reconnect if MQTT client is not connected.
  if (!mqttClient.connected())
  {
   reconnect();
  }

  mqttClient.loop();   // Call the loop continuously to establish connection to the server.

  // If interval time has passed since the last connection, publish data to ThingSpeak.
  if (millis() - lastConnectionTime > postingInterval)
  {
    mqttPublishFeed(); // Publish three values simultaneously.
    // mqttPublishField(fieldNumber); // Use this function to publish to a single field
  }
}
```

Done uploading.

```
Leaving...
Hard resetting via RTS pin...
```

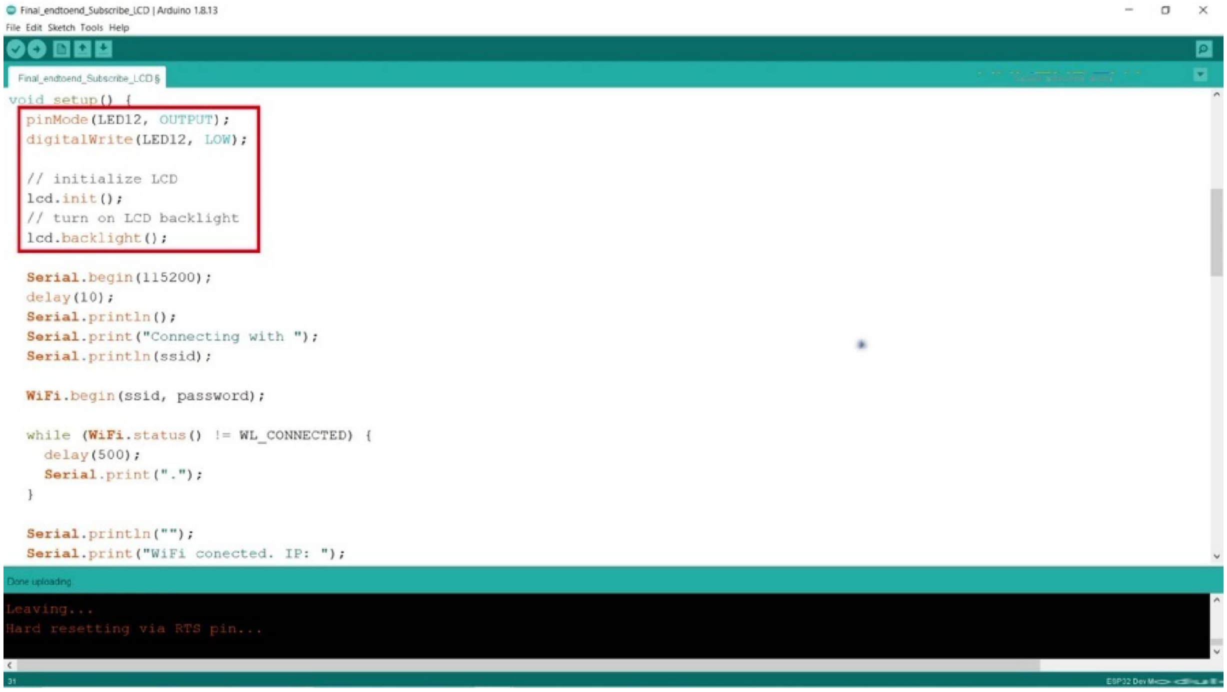
Here we call the loops to continuously establish connection to the server and publish value simultaneously to the server.


Final_endtoend_Subscribe_LCD | Arduino 1.8.13
File Edit Sketch Tools Help
Final_endtoend_Subscribe_LCD §
Done uploading
93
ESP32 Dev Module

```
#include <WiFi.h> //  ESP32
WiFiClient WIFI_CLIENT;

#include <PubSubClient.h>
PubSubClient MQTT_CLIENT;

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

const char * ssid = "shubham ext";
const char * password = "god1314@";

const char * mqttUserName = "shubhamg13114"; // Thingspeak username.
const char * mqttPass = "WP6S338XQVQZM2F1"; // Change to your MQTT API Key from Account > MyProfile.
const char * ChannelID = "1363280"; // Thingspeak Channel ID
const char * APIKey = "2HH1AOP0GILZGH8D"; // Thingspeak Channel Read API Key

static const char alphanum[] = "0123456789"
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
"abcdefghijklmnopqrstuvwxyz";

int intChar;
#define LED12 12 // LED12 ESP32
```

```
Leaving...
Hard resetting via RTS pin...
```

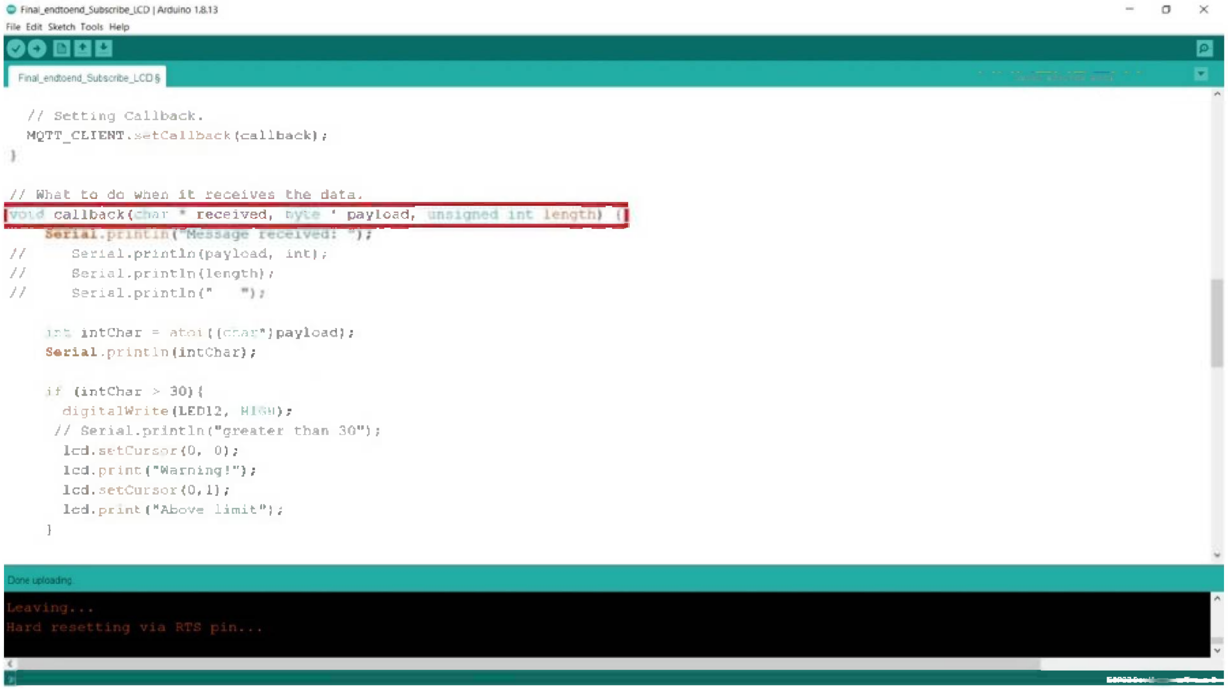
Now let's check the code for the subscribe operation of our project. Here We have included the wi fi and pub sub client library needed to connect to the Internet and to use MQTT. We have also initialized these Libraries here. Here we have included the I2C LCD library. Here We have specified the I2C address and the number of columns and rows of our LCD. Here We have specified the wi fi SSID and password. Now we need to mention the necessary MQTT details, remember that you need to specify the read API key this time as we are subscribing to the channel and reading data from it. Here we have defined an LED at pin 12 of the ESP32. This LED can be used for debugging. It is okay even if you do not use the LED.


Final_endtoend_Subscribe_LCD | Arduino 1.8.13
File Edit Sketch Tools Help
Final_endtoend_Subscribe_LCD §
Done uploading

```
void setup() {
  pinMode(LED12, OUTPUT);
  digitalWrite(LED12, LOW);

  // initialize LCD
  lcd.init();
  // turn on LCD backlight
  lcd.backlight();

  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.print("Connecting with ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.print("WiFi conected. IP: ");
```

```
Leaving...
Hard resetting via RTS pin...
```

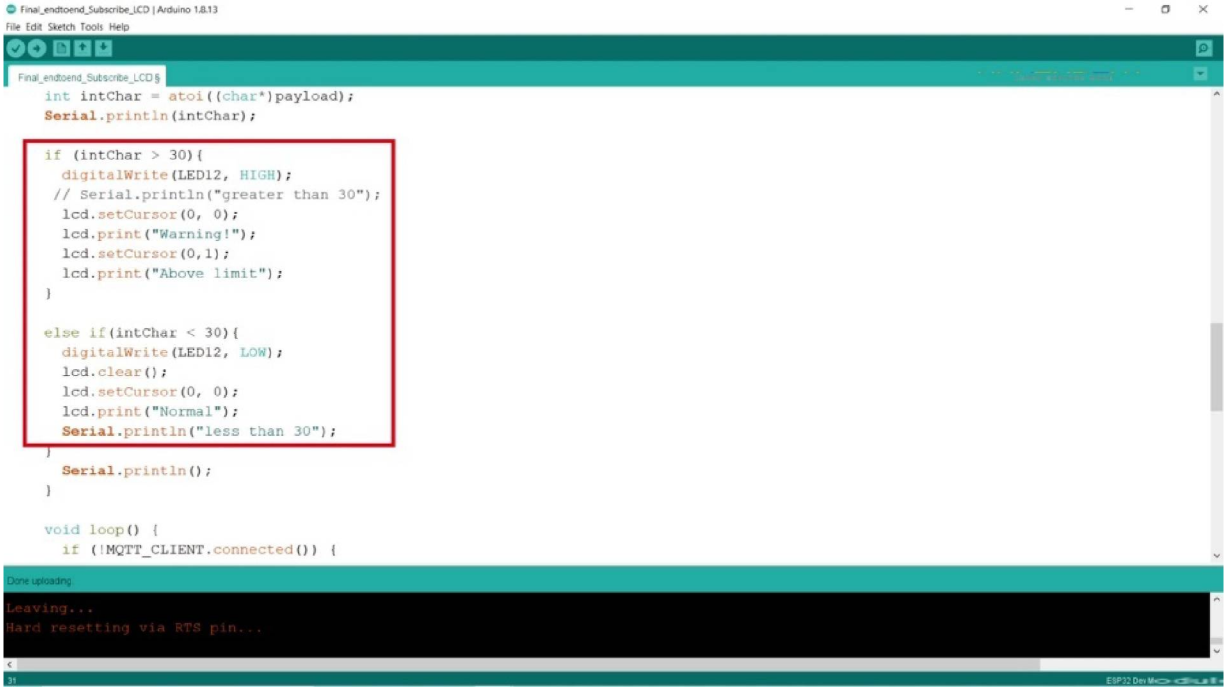
Now we will set up the LED and initialize the LCD. Here We have set the bar rate to 115200. Now we will connect to the wi fi using the credentials mentioned before.


Final_endtoend_Subscribe_LCD | Arduino 1.8.13
File Edit Sketch Tools Help
Final_endtoend_Subscribe_LCD §
Done uploading

```
  // Setting Callback.
  MQTT_CLIENT.setCallback(callback);
}

// What to do when it receives the data.
void callback(char * received, byte * payload, unsigned int length) {
    Serial.println("Message received: ");
//      Serial.println(payload, int);
//      Serial.println(length);
//      Serial.println("    ");

    int intChar = atoi((char*)payload);
    Serial.println(intChar);

    if (intChar > 30){
      digitalWrite(LED12, HIGH);
     // Serial.println("greater than 30");
      lcd.setCursor(0, 0);
      lcd.print("Warning!");
      lcd.setCursor(0,1);
      lcd.print("Above limit");
    }
```

```
Leaving...
Hard resetting via RTS pin...
```

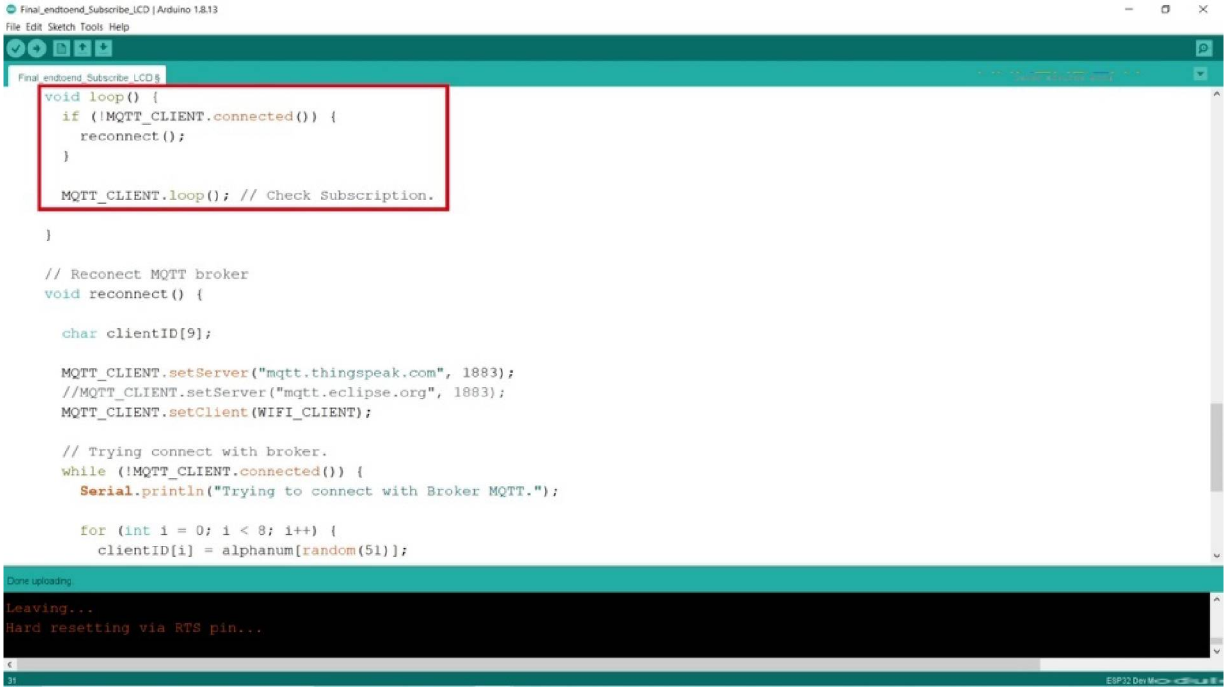
Here we have set a callback. This function is responsible for actions. when the data is received here. We have converted the character input string to integer values.


Final_endtoend_Subscribe_LCD | Arduino 1.8.13
File Edit Sketch Tools Help
Final_endtoend_Subscribe_LCD §
Done uploading.
31
ESP32 Dev Module

```
  int intChar = atoi((char*)payload);
  Serial.println(intChar);

  if (intChar > 30){
    digitalWrite(LED12, HIGH);
   // Serial.println("greater than 30");
    lcd.setCursor(0, 0);
    lcd.print("Warning!");
    lcd.setCursor(0,1);
    lcd.print("Above limit");
  }

  else if(intChar < 30){
    digitalWrite(LED12, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Normal");
    Serial.println("less than 30");
  }
    Serial.println();
  }

  void loop() {
    if (!MQTT_CLIENT.connected()) {
```

```
Leaving...
Hard resetting via RTS pin...
```

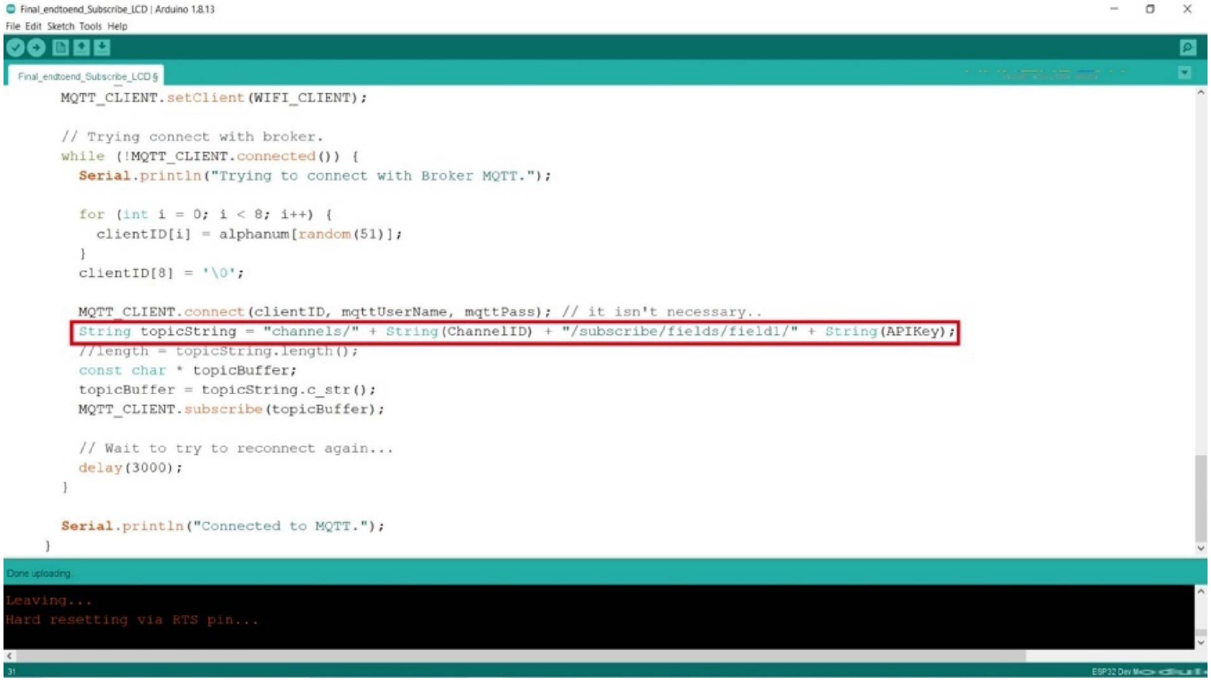
Now we will set the action so as to display the text warning and above limit whenever the temperature is above the specified value. If the temperature is below the specified value, we will print the word normal.


Final_endtoend_Subscribe_LCD | Arduino 1.8.13
File Edit Sketch Tools Help
Final_endtoend_Subscribe_LCD §
Done uploading.
31
ESP32 Dev Module

```
  void loop() {
    if (!MQTT_CLIENT.connected()) {
      reconnect();
    }

    MQTT_CLIENT.loop(); // Check Subscription.

  }

  // Reconect MQTT broker
  void reconnect() {

    char clientID[9];

    MQTT_CLIENT.setServer("mqtt.thingspeak.com", 1883);
    //MQTT_CLIENT.setServer("mqtt.eclipse.org", 1883);
    MQTT_CLIENT.setClient(WIFI_CLIENT);

    // Trying connect with broker.
    while (!MQTT_CLIENT.connected()) {
      Serial.println("Trying to connect with Broker MQTT.");

      for (int i = 0; i < 8; i++) {
        clientID[i] = alphanum[random(51)];
```

```
Leaving...
Hard resetting via RTS pin...
```

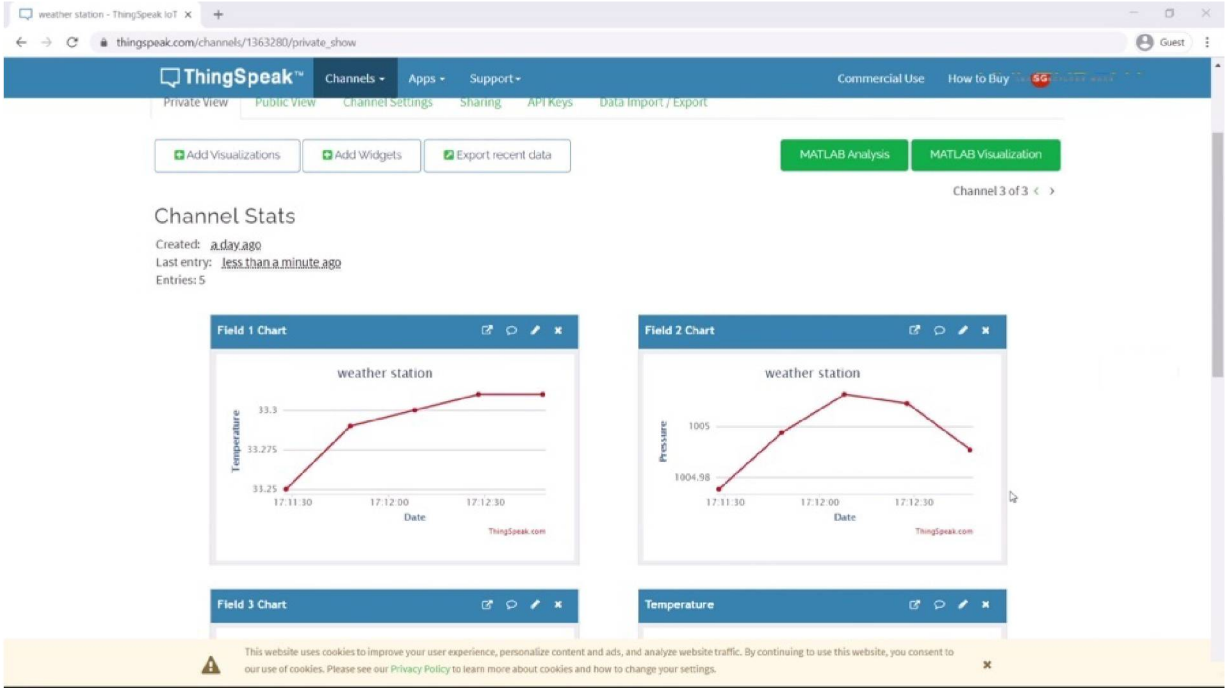
The void loop is responsible for the connection to the Internet and to check subscriptions. Here we have written a function to reconnect to the broker in case of any issues.


Final_endtoend_Subscribe_LCD | Arduino 1.8.13
File Edit Sketch Tools Help
Final_endtoend_Subscribe_LCD §
Done uploading.
31
ESP32 Dev Module

```
    MQTT_CLIENT.setClient(WIFI_CLIENT);

    // Trying connect with broker.
    while (!MQTT_CLIENT.connected()) {
      Serial.println("Trying to connect with Broker MQTT.");

      for (int i = 0; i < 8; i++) {
        clientID[i] = alphanum[random(51)];
      }
      clientID[8] = '\0';

      MQTT_CLIENT.connect(clientID, mqttUserName, mqttPass); // it isn't necessary..
      String topicString = "channels/" + String(ChannelID) + "/subscribe/fields/field1/" + String(APIKey);
      //length = topicString.length();
      const char * topicBuffer;
      topicBuffer = topicString.c_str();
      MQTT_CLIENT.subscribe(topicBuffer);

      // Wait to try to reconnect again...
      delay(3000);
    }

    Serial.println("Connected to MQTT.");
  }
```

```
Leaving...
Hard resetting via RTS pin...
```

Here the topic string is mentioned as specified by Thing Speak in order to subscribe to the appropriate channel.

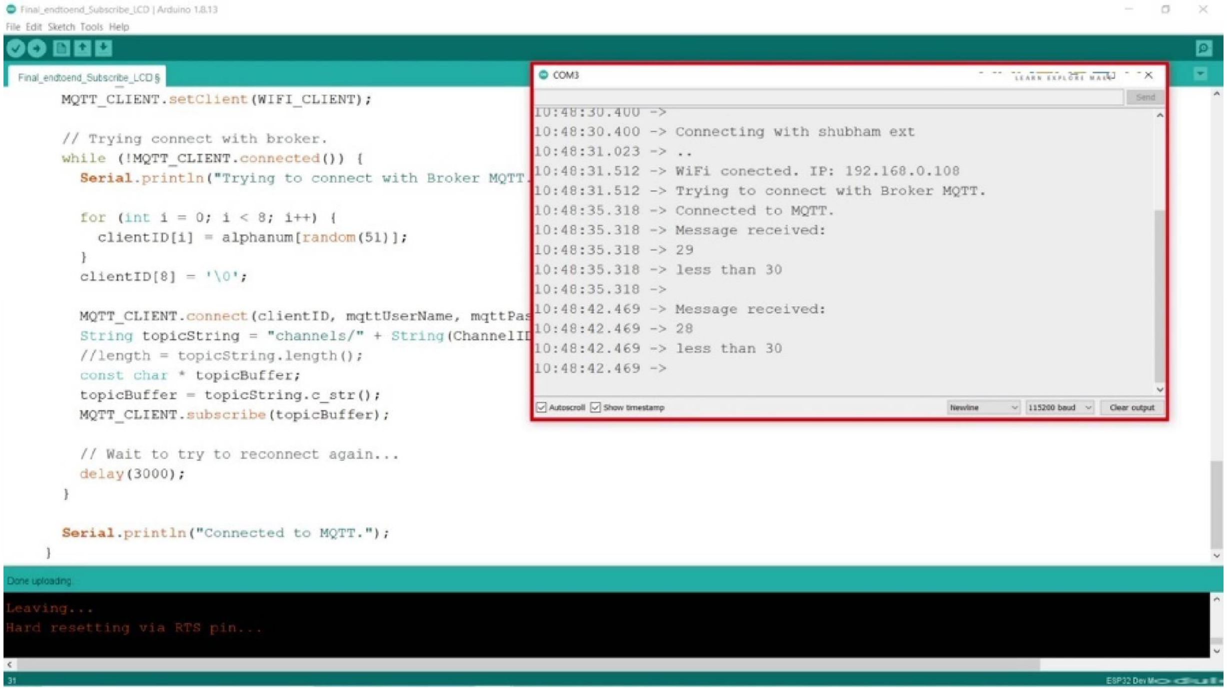


Channel Stats

Created: a day ago
Last entry: less than a minute ago
Entries: 5

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Now save and upload both the codes to the respective ESP32s. You can now see that the measured values by the BME280 sensor are published on the Thing Speak. Channel.

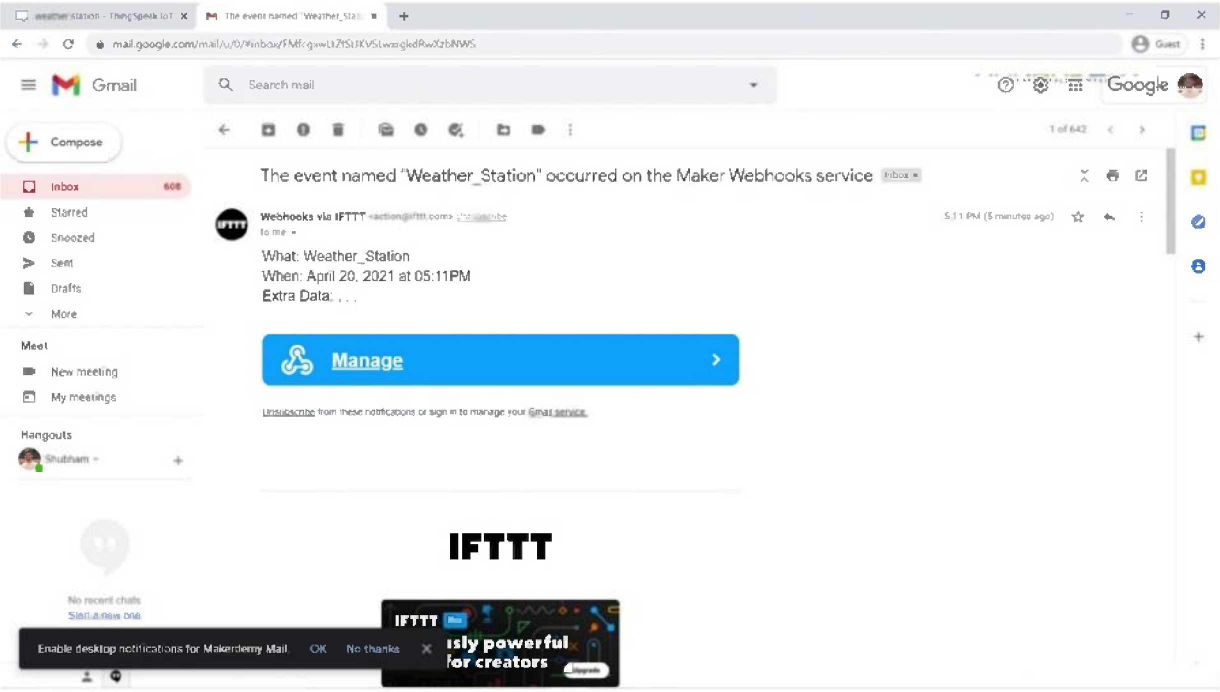

Final_endtoend_Subscribe_LCD | Arduino 1.8.13
File Edit Sketch Tools Help
Final_endtoend_Subscribe_LCD §
COM3
Send
Autoscroll
Show timestamp
Newline
115200 baud
Clear output
ESP32 Dev Module

```
    MQTT_CLIENT.setClient(WIFI_CLIENT);

    // Trying connect with broker.
    while (!MQTT_CLIENT.connected()) {
      Serial.println("Trying to connect with Broker MQTT.
      
      for (int i = 0; i < 8; i++) {
        clientID[i] = alphanum[random(51)];
      }
      clientID[8] = '\0';

      MQTT_CLIENT.connect(clientID, mqttUserName, mqttPas
      String topicString = "channels/" + String(ChannelII
      //length = topicString.length();
      const char * topicBuffer;
      topicBuffer = topicString.c_str();
      MQTT_CLIENT.subscribe(topicBuffer);

      // Wait to try to reconnect again...
      delay(3000);
    }

    Serial.println("Connected to MQTT.");
  }
```

```
10:48:30.400 ->
10:48:30.400 -> Connecting with shubham ext
10:48:31.023 -> ..
10:48:31.512 -> WiFi conected. IP: 192.168.0.108
10:48:31.512 -> Trying to connect with Broker MQTT.
10:48:35.318 -> Connected to MQTT.
10:48:35.318 -> Message received:
10:48:35.318 -> 29
10:48:35.318 -> less than 30
10:48:35.318 ->
10:48:42.469 -> Message received:
10:48:42.469 -> 28
10:48:42.469 -> less than 30
10:48:42.469 ->
```

```
Done uploading.
Leaving...
Hard resetting via RTS pin...
```

You can also check the subscribe side of the ESP32 and the serial monitor.



The event named "Weather_Station" occurred on the Maker Webhooks service

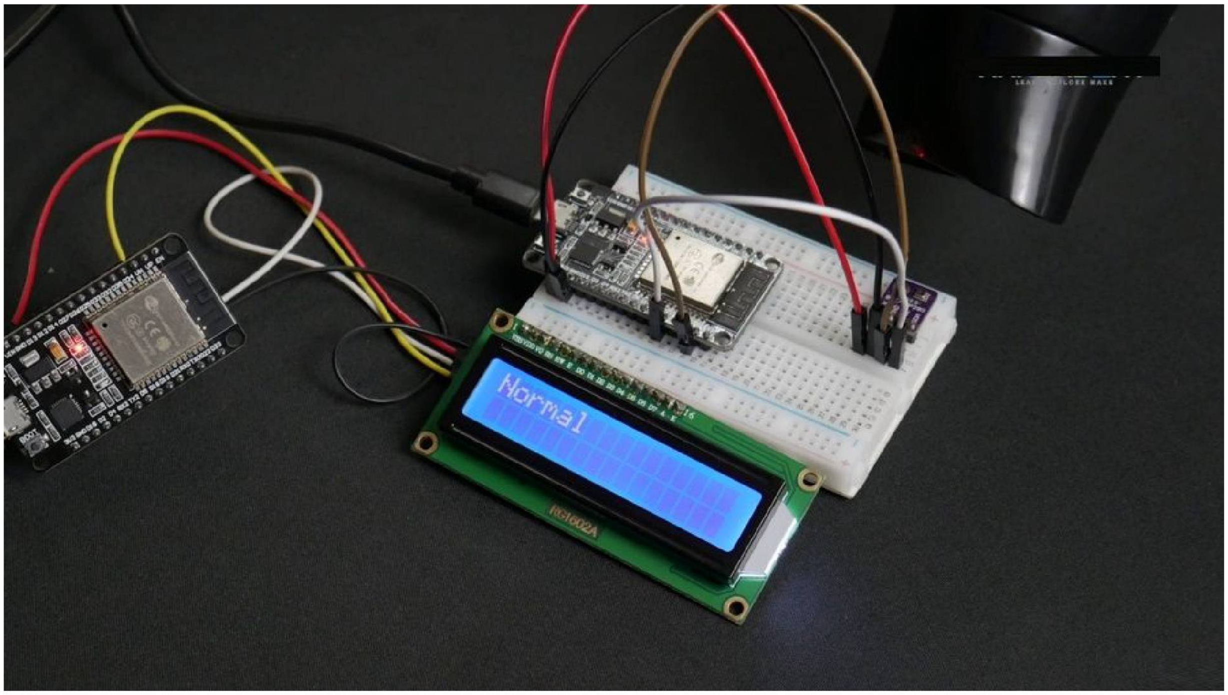
Webhooks via IFTTT <action@ifttt.com> Unsubscribe
to me
5:11 PM (5 minutes ago)

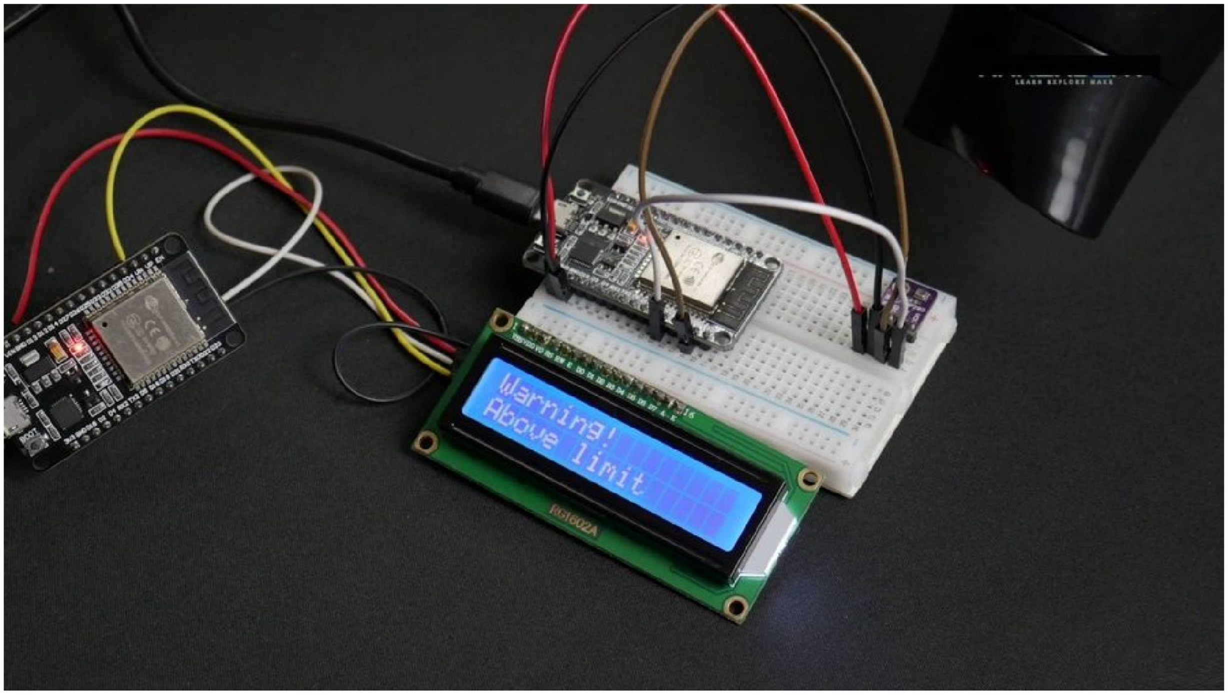
What: Weather_Station
When: April 20, 2021 at 05:11PM
Extra Data: , , ,

Manage

Unsubscribe from these notifications or sign in to manage your Gmail service.

It will display the received value. If the received value is more than the specified value You will receive an email stating the same.





The LCD will also display the appropriate message according to the values received.

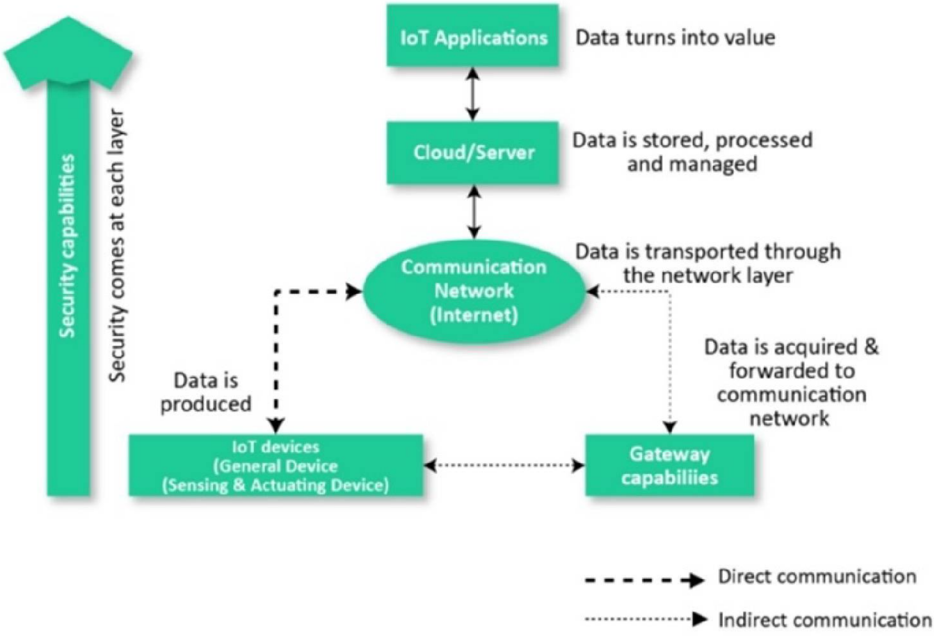
CONCLUSION

Though we have briefly covered all the phases in developing an IoT system. As this domain is vast, there are endless possibilities to work on.

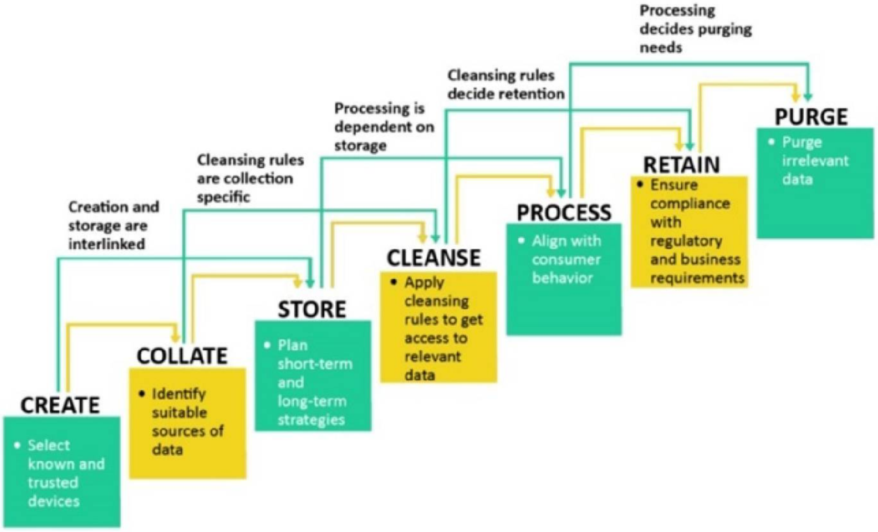
Since we have come to the end of the project, let's have a recap of all the concepts learned until now. The project began with getting to know about the microcontroller we used in this project. Not only did we learn about the features of the ESP32, we also had a discussion about what factors to consider and how to select the appropriate microcontroller. Next, we had an overview of the hardware specifications of the ESP32. We also brushed up on the Pin out diagram for the development board. We concluded that ESP32 is one of the best suitable boards for IoT development due to its features.

Then we set up our device and the development environment and got a basic overview of using the ESP32 and the Arduino IDE in conjunction.

In the next section we learned about IoT in depth. We started right from why IoT Systems are needed and went on to learn about the various parts of an IoT system.

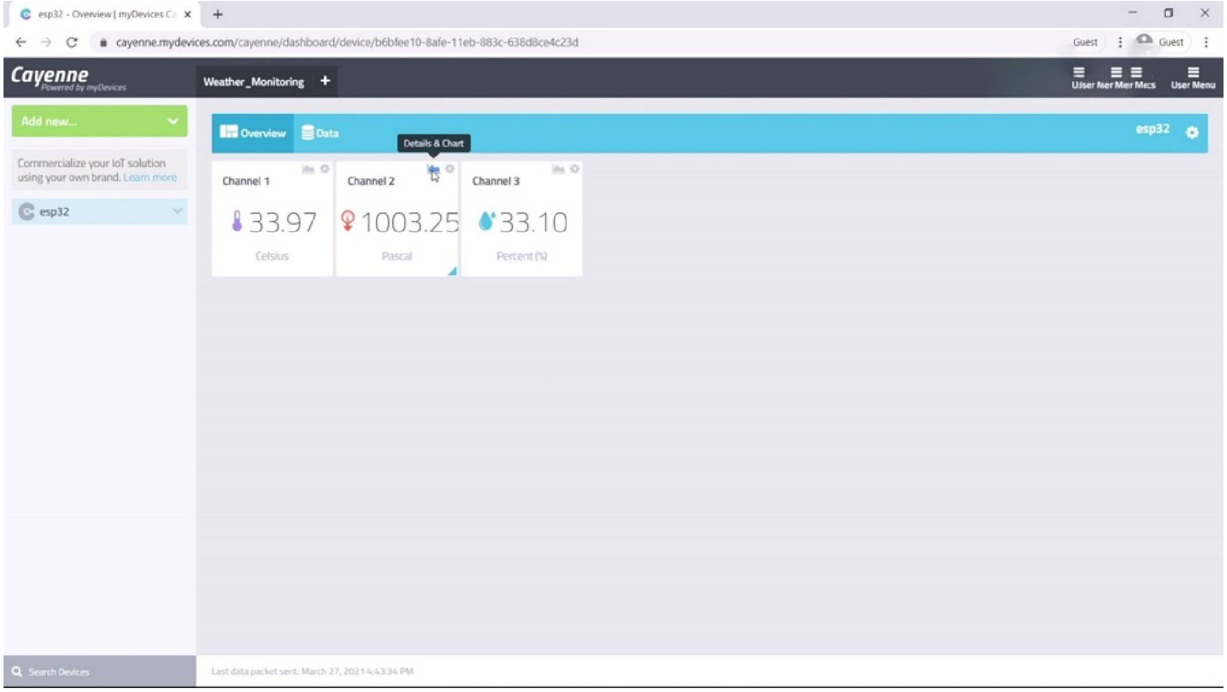

Security capabilities
Security comes at each layer
IoT Applications
Data turns into value
Cloud/Server
Data is stored, processed and managed
Communication Network (Internet)
Data is transported through the network layer
Data is produced
Data is acquired & forwarded to communication network
IoT devices (General Device (Sensing & Actuating Device)
Gateway capabiliies
Direct communication
Indirect communication

The basic architecture of an IoT system includes IoT devices, gateways, communication networks, cloud or a server and IoT applications. We also had an overview of how the data flows in an IoT system.

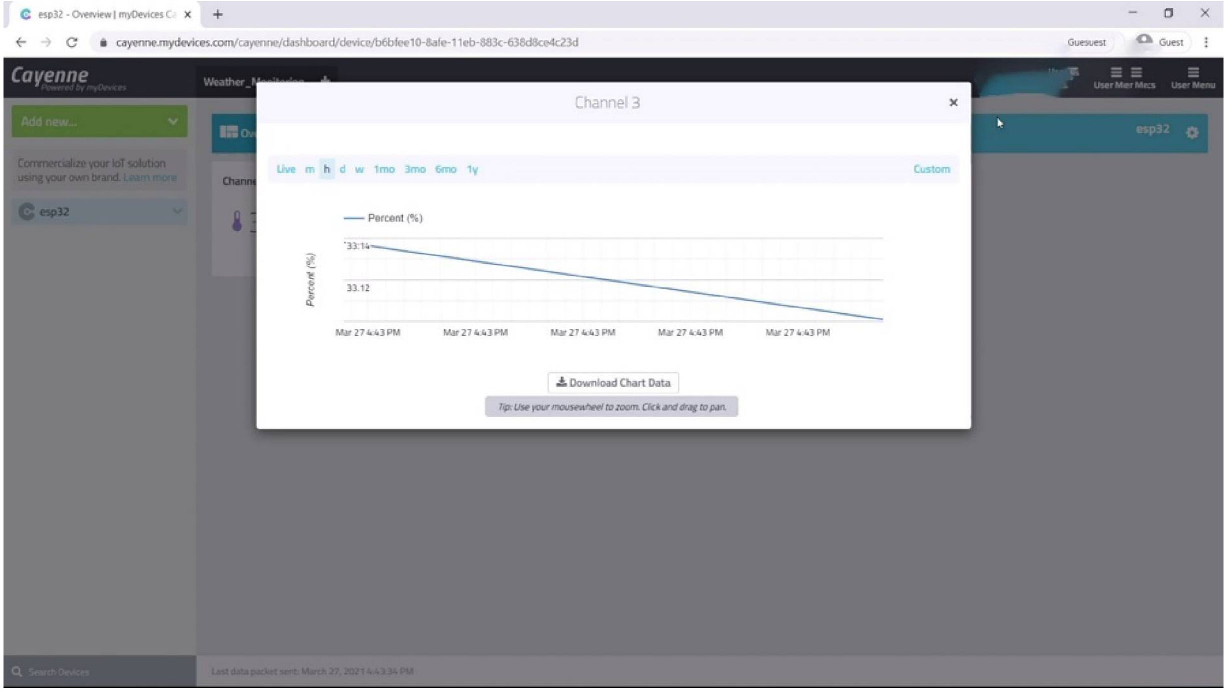

Processing decides purging needs
Cleansing rules decide retention
Processing is dependent on storage
Cleansing rules are collection specific
Creation and storage are interlinked
PURGE
• Purge irrelevant data
RETAIN
• Ensure compliance with regulatory and business requirements
PROCESS
• Align with consumer behavior
CLEANSE
• Apply cleansing rules to get access to relevant data
STORE
• Plan short-term and long-term strategies
COLLATE
• Identify suitable sources of data
CREATE
• Select known and trusted devices

We then looked over to the management side of IoT. We covered topics like how device management takes place, how data is handled and what to do after collecting the data from the IoT device. Some of the important concepts, such as data analytics and data visualization and the integration of an IoT system, were discussed in the following section. Some of the important developments in the IoT domain have been the addition of different communication protocols for data transfer. We also discussed the security aspects of IoT and its integration with artificial intelligence.

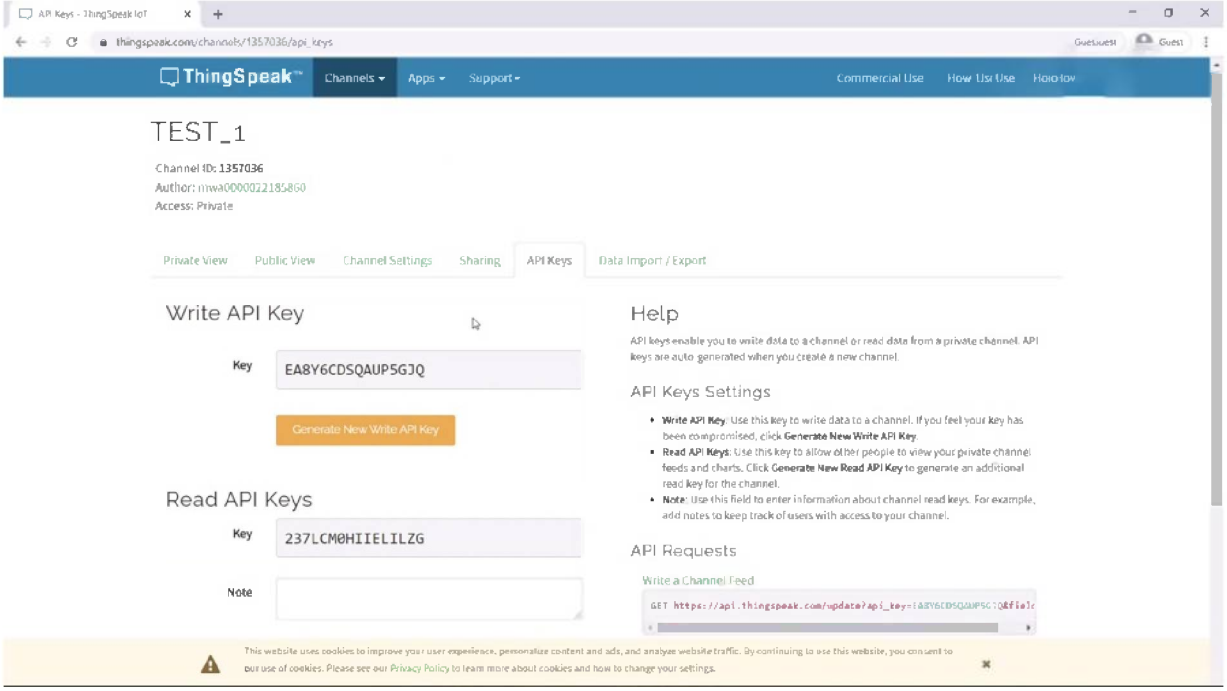
We then briefly discussed the newer technologies that have developed and spearheaded the growth of IoT cloud computing and edge computing. In the next section, we got an overview of Cayenne and set up our Cayenne Account.


esp32 - Overview | myDevices C
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d
Guest
Cayenne
Powered by myDevices
Weather_Monitoring
User Menu
Add new...
Commercialize your IoT solution using your own brand. Learn more
esp32
Overview
Data
Details & Chart
esp32
Channel 1
33.97
Celsius
Channel 2
1003.25
Pascal
Channel 3
33.10
Percent (%)
Search Devices
Last data packet sent: March 27, 2021 4:43:34 PM

Cayenne is one of the easiest project builders to develop a proof of concept and end to end projects. The drag and drop feature of Cayenne makes it easy, even for a beginner to develop into an IoT projects.


esp32 - Overview | myDevices C
cayenne.mydevices.com/cayenne/dashboard/device/b6bfee10-8afe-11eb-883c-638d8ce4c23d
Cayenne
Channel 3
Live
m
h
d
w
1mo
3mo
6mo
1y
Custom
Percent (%)
33.14
33.12
Percent (%)
Mar 27 4:43 PM
Mar 27 4:43 PM
Mar 27 4:43 PM
Mar 27 4:43 PM
Mar 27 4:43 PM
Download Chart Data
Tip: Use your mousewheel to zoom. Click and drag to pan.
esp32
Last data packet sent: March 27, 2021 4:43:34 PM

We then went on to develop a weather station with Cayenne The BME280 sensor can measure physical quantities such as temperature,pressure and humidity. As the BME280 is precise, in measuring the quantities. Therefore, we can also measure altitude. We then explored various features of Cayenne, including the trigger notification and event scheduling feature of Cayenne. Then we interfaced the 16x2 LCD to the ESP32. Also, do you remember the issue while we implemented the end to end weather station project? How we had to work around the issue?



TEST_1

Channel ID: 1357036
Author: mwa0000022185860
Access: Private

Write API Key

Key: EA8Y6CDSQAUP5GJQ

Read API Keys

Key: 237LCM0HIIELILZG

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

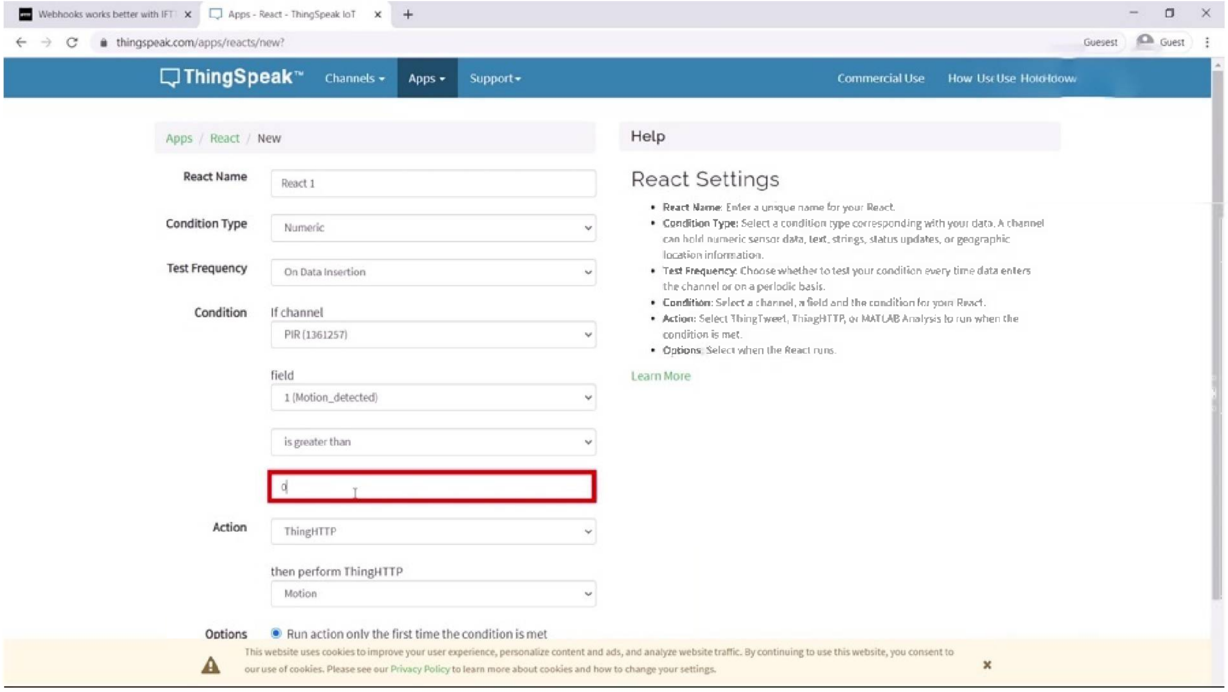
API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=EA8Y6CDSQAUP5GJQ&field
```

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

In the next section we checked upon Thing Speak, which is another IoT platform. We also discussed the pros and cons of using both Thing Speak and Cayenne. The different API keys and channel IDs can be used to read and write data to a particular channel.


Webhooks works better with IFT
Apps - React - ThingSpeak IoT
thingspeak.com/apps/reacts/new?
ThingSpeak™
Channels ▾
Apps ▾
Support ▾
Commercial Use
How Us: Use HoldHow
Apps / React / New

React Name	React 1
Condition Type	Numeric
Test Frequency	On Data Insertion
Condition	If channel PIR (1361257) field 1 (Motion_detected) is greater than 0
Action	ThingHTTP then perform ThingHTTP Motion
Options	● Run action only the first time the condition is met

Help

React Settings

- **React Name**: Enter a unique name for your React.
- **Condition Type:** Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- **Test Frequency:** Choose whether to test your condition every time data enters the channel or on a periodic basis.
- **Condition:** Select a channel, a field and the condition for your React.
- **Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- **Options:** Select when the React runs.

Learn More

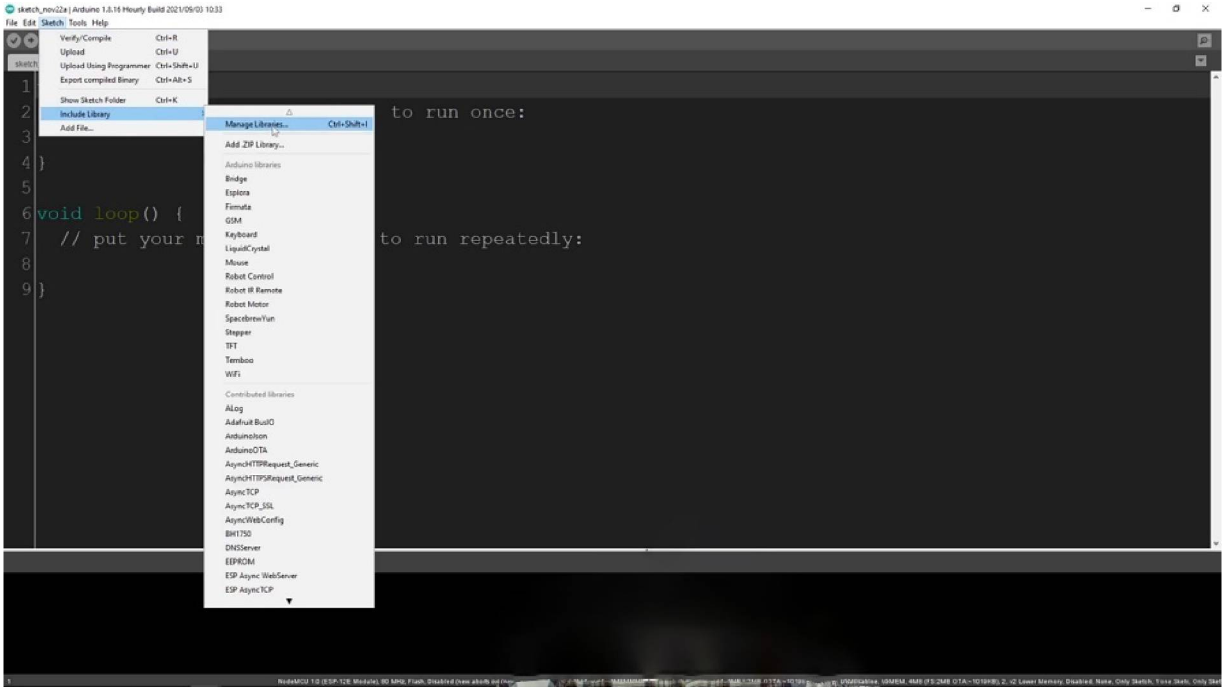
This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Additionally, apps from Thing Speak like the ThingHTTP and react can be used in conjunction with IFTTT to send notifications to your email. You can also use other apps from IFTTT to create a customized a trigger action plan for your IoT system. We then developed a weather station using the BME280 sensor and things speak this time. We did not have the use a workaround to make our system work. After completing projects in both Cayenne and Thing Speak, you may have noticed a difference in using the IoT platforms. Then we learned about the important concepts of IoT, such as the architecture data flow in an IoT system and the different technologies that can be introduced in an IoT system. Next, we moved on to get that introduction to the Drag and Drop Project builder called Cayenne. We then implemented the capstone project, which is a weather station using Cayenne and with all the concepts learned before. Finally we implemented the second capstone project using the BME280 and Thing Speak. This helped us to learn about the difference between the two IoT platforms.

V

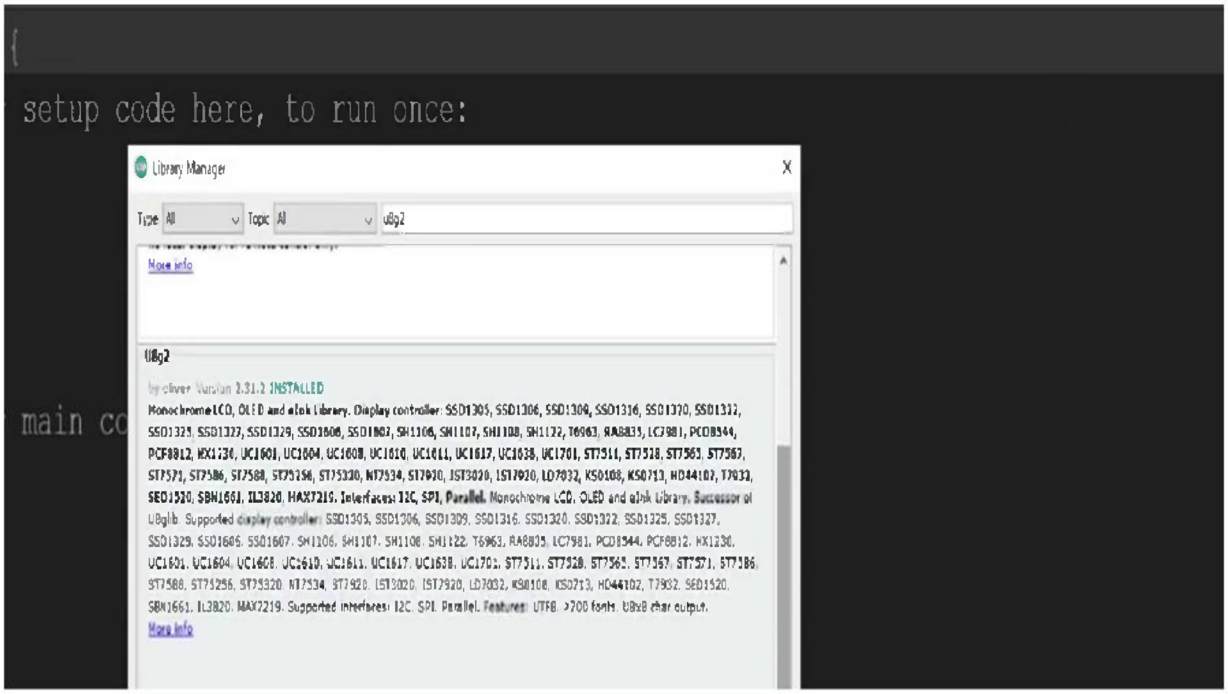
FIRST GRAPHICS TEST

we want to make an example and references was two examples from the libraries. Therefore, I opened new sketch and now we are downloading the necessary library.



```
                       to run once:

}

void loop() {
  // put your m                  to run repeatedly:

}
```

Therefore, we go to sketch and we are going to include library ménage libraries. Let's take a little while. Because this open window is very, very slow. But when it has loaded, we have in you 80 to. You see, it's very, very slow. I was in the typing.



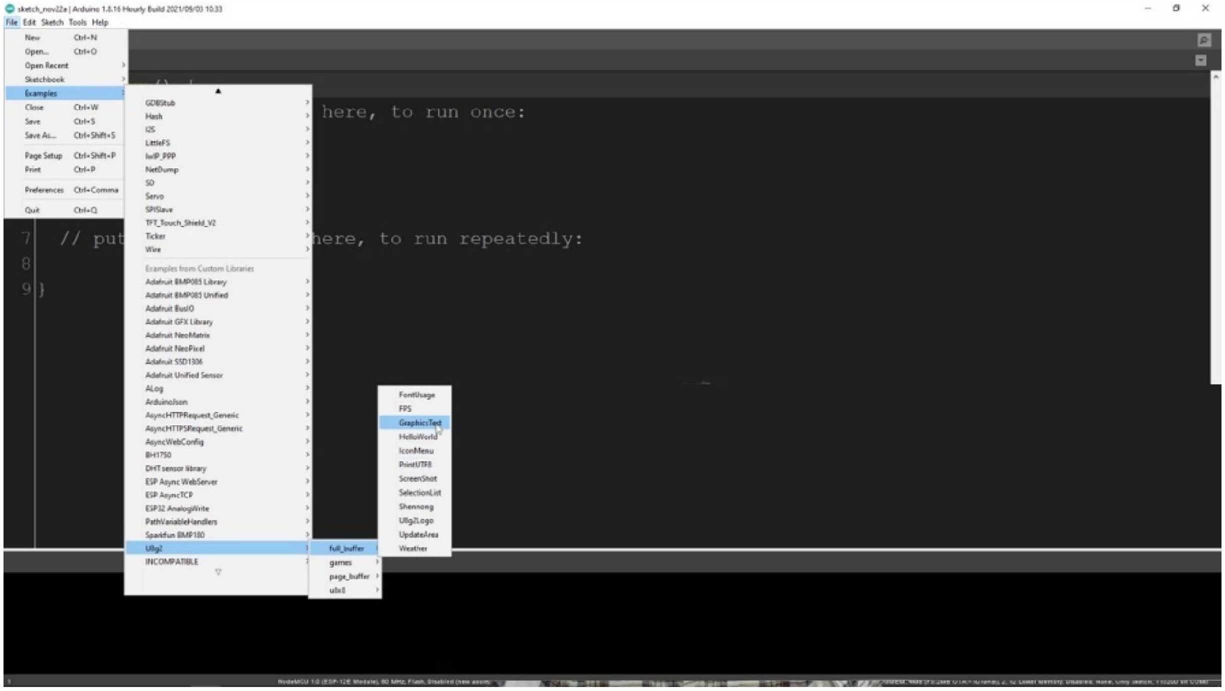
```
{
 setup code here, to run once:

 main co
```

U8g2

by oliver Version 2.31.2 INSTALLED

Monochrome LCD, OLED and eInk Library. Display controller: SSD1305, SSD1306, SSD1309, SSD1316, SSD1320, SSD1322, SSD1325, SSD1327, SSD1329, SSD1606, SSD1607, SH1106, SH1107, SH1108, SH1122, T6963, RA8835, LC7981, PCD8544, PCF8812, HX1230, UC1601, UC1604, UC1608, UC1610, UC1611, UC1617, UC1638, UC1701, ST7511, ST7528, ST7565, ST7567, ST7571, ST7586, ST7588, ST75256, ST75320, NT7534, ST7920, IST3020, IST7920, LD7032, KS0108, KS0713, HD44102, T7932, SED1520, SBN1661, IL3820, MAX7219. Interfaces: I2C, SPI, Parallel. Monochrome LCD, OLED and eInk library. Successor of U8glib. Supported display controller: SSD1305, SSD1306, SSD1309, SSD1316, SSD1320, SSD1322, SSD1325, SSD1327, SSD1329, SSD1606, SSD1607, SH1106, SH1107, SH1108, SH1122, T6963, RA8835, LC7981, PCD8544, PCF8812, HX1230, UC1601, UC1604, UC1608, UC1610, UC1611, UC1617, UC1638, UC1701, ST7511, ST7528, ST7565, ST7567, ST7571, ST7586, ST7588, ST75256, ST75320, NT7534, ST7920, IST3020, IST7920, LD7032, KS0108, KS0713, HD44102, T7932, SED1520, SBN1661, IL3820, MAX7219. Supported interfaces: I2C, SPI, Parallel. Features: UTF8, >700 fonts, U8x8 char output.

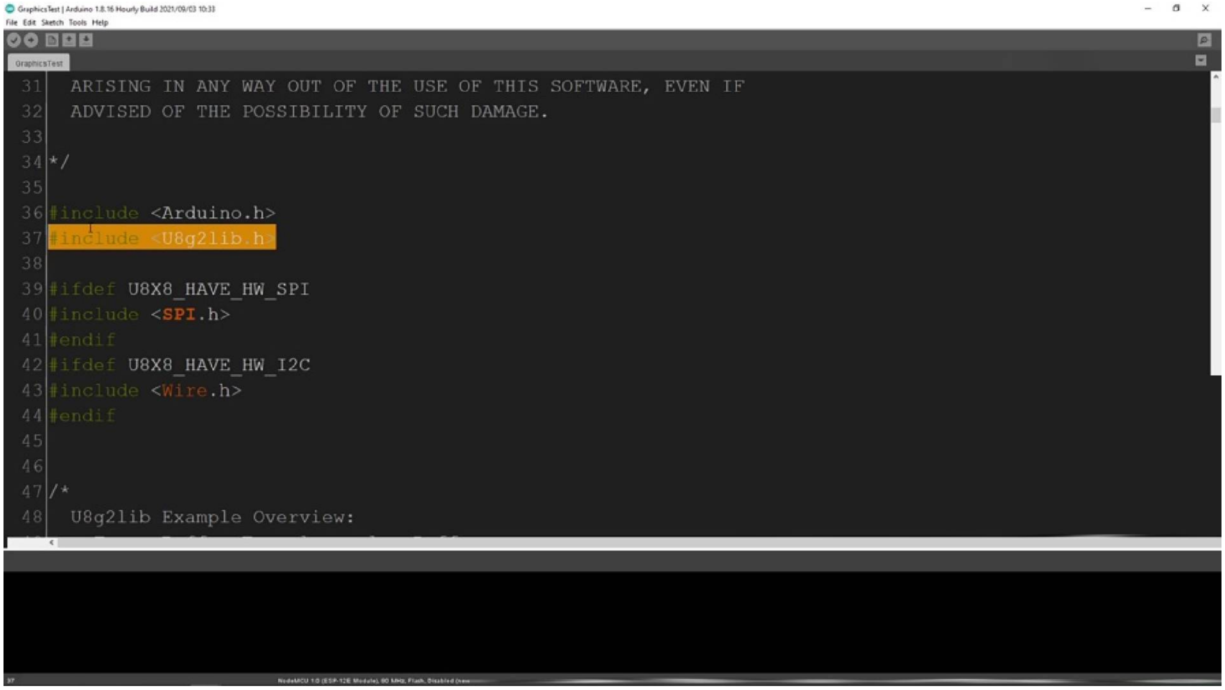
Then we are searching. Hit enter. And here's the desired library you ate a too. And by Oliver, this is what we want. Click on install. Maybe you have also installed some more referenced libraries. So just click OK. And then when you have installed the whole library, there should be an undergrad file example.



sketch_nov22a | Arduino 1.8.16 Hourly Build 2021/09/03 10:33

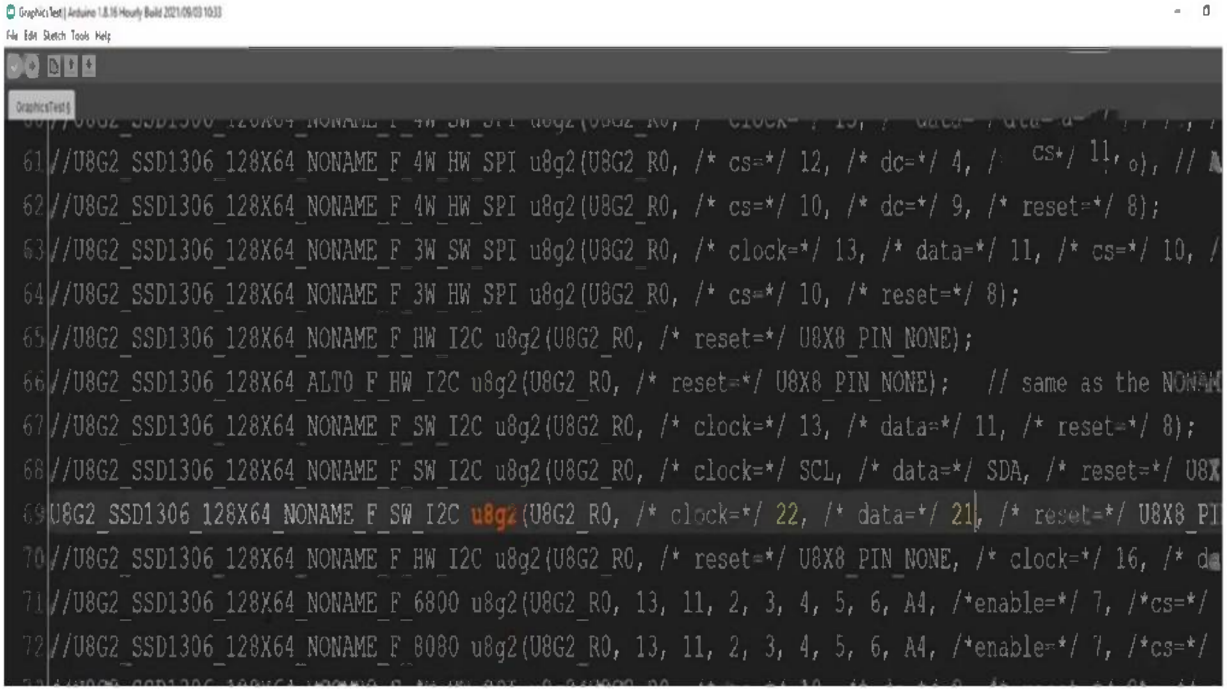
```
 here, to run once:

7   // put          here, to run repeatedly:
8
9 }
```

When you. Go a little bit down in the menu, then you should have a new and trim you eight h two. Then we go to full buffer. And also we go to graphic artists file example. You hate to flip off a graphic artist.


GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest
NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, Disabled (new

```
  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/

#include <Arduino.h>
#include <U8g2lib.h>

#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>
#endif
#ifdef U8X8_HAVE_HW_I2C
#include <Wire.h>
#endif


/*
  U8g2lib Example Overview:
```

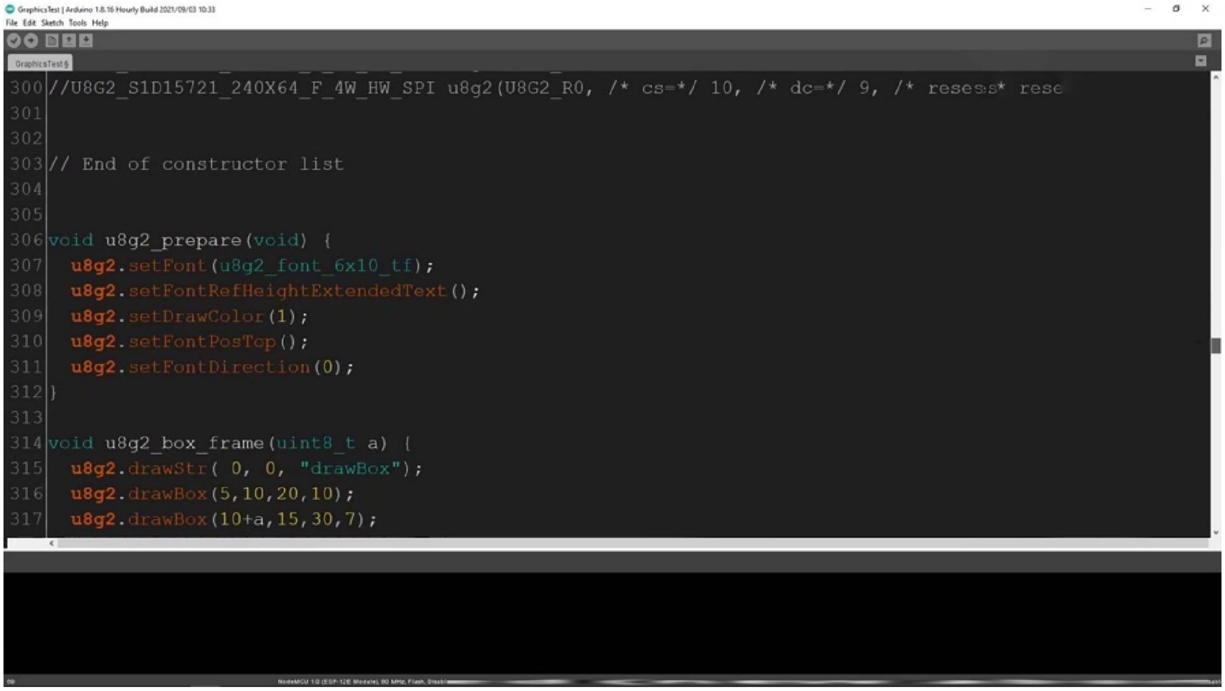
A new sketch is opened, and what you now can see is that we only need one library, this library and this library is the only library would we use for making the whole graphic, the whole bitmap and string showing on the display? So it's growing a little bit down and then we see a lot of constructions to start the right and proper only display. I'm using on one 28s multiply. 64. I have to search to white one.



GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest §

```
//U8G2_SSD1306_128X64_NONAME_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 12, /* dc=*/ 4, /  cs*/ 11, o), //
//U8G2_SSD1306_128X64_NONAME_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
//U8G2_SSD1306_128X64_NONAME_F_3W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /
//U8G2_SSD1306_128X64_NONAME_F_3W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* reset=*/ 8);
//U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);
//U8G2_SSD1306_128X64_ALT0_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);   // same as the NONAM
//U8G2_SSD1306_128X64_NONAME_F_SW_I2C u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* reset=*/ 8);
//U8G2_SSD1306_128X64_NONAME_F_SW_I2C u8g2(U8G2_R0, /* clock=*/ SCL, /* data=*/ SDA, /* reset=*/ U8X
U8G2_SSD1306_128X64_NONAME_F_SW_I2C u8g2(U8G2_R0, /* clock=*/ 22, /* data=*/ 21, /* reset=*/ U8X8_PI
//U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE, /* clock=*/ 16, /* da
//U8G2_SSD1306_128X64_NONAME_F_6800 u8g2(U8G2_R0, 13, 11, 2, 3, 4, 5, 6, A4, /*enable=*/ 7, /*cs=*/
//U8G2_SSD1306_128X64_NONAME_F_8080 u8g2(U8G2_R0, 13, 11, 2, 3, 4, 5, 6, A4, /*enable=*/ 7, /*cs=*/
```

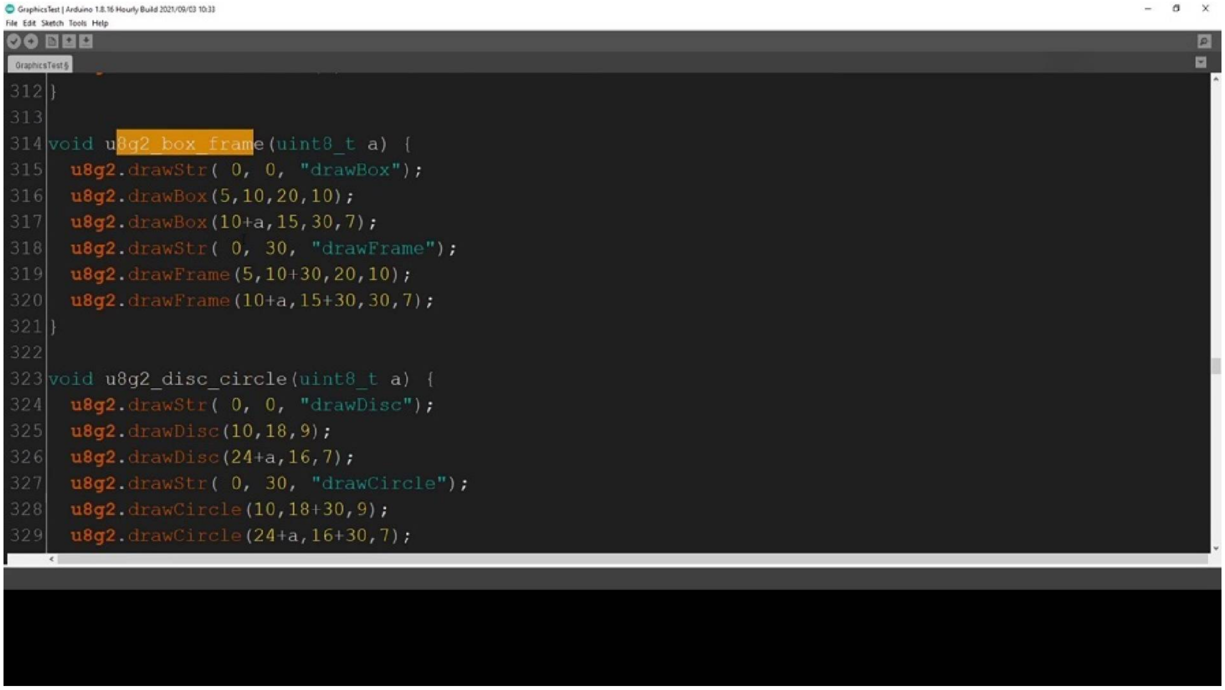
So and here we have the proper one. It's cards known M A, C e to see him, and then we are switching gears. The clock was twenty two, I think, yes, 20 to him and where the data was twenty one. And this is our constructor.



GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest §
NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, Disabl

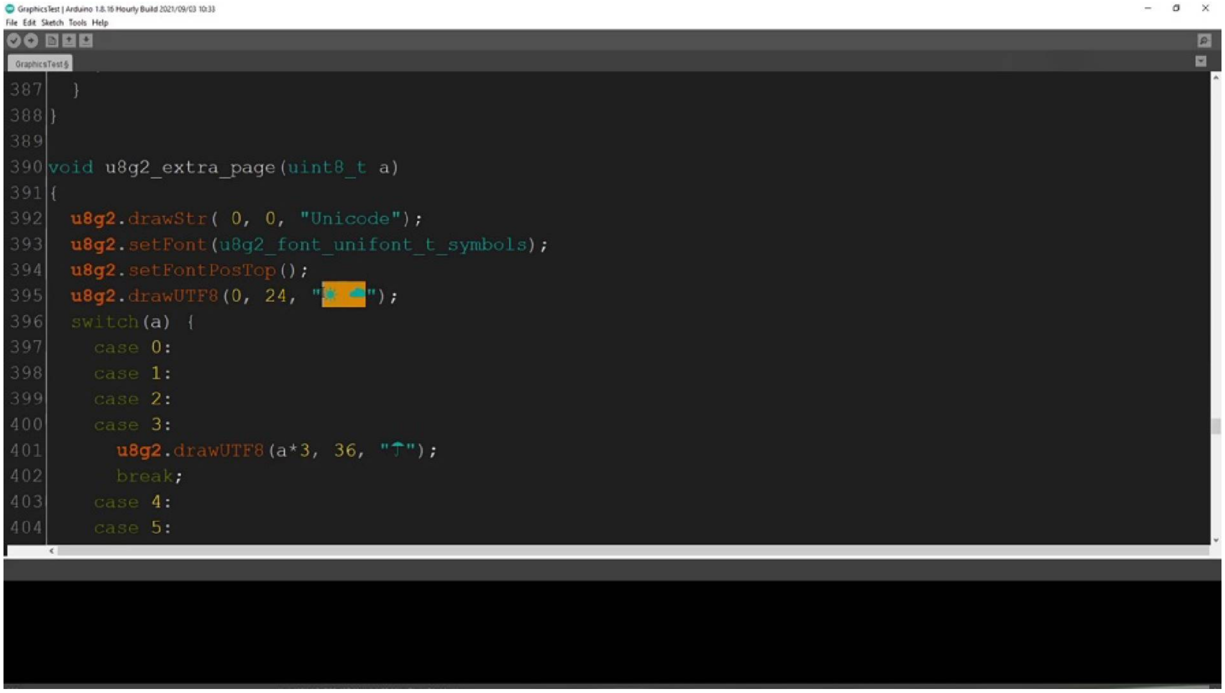
```
//U8G2_S1D15721_240X64_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reses s* rese


// End of constructor list


void u8g2_prepare(void) {
  u8g2.setFont(u8g2_font_6x10_tf);
  u8g2.setFontRefHeightExtendedText();
  u8g2.setDrawColor(1);
  u8g2.setFontPosTop();
  u8g2.setFontDirection(0);
}

void u8g2_box_frame(uint8_t a) {
  u8g2.drawStr( 0, 0, "drawBox");
  u8g2.drawBox(5,10,20,10);
  u8g2.drawBox(10+a,15,30,7);
```

And that's it, that's all what we have to set up for the first graphic artists, and when I scroll a little bit down, you can see here a lot of the constructors, yeah, a lot of function, which we are using afterwards in our project, for example, how you can draw a frame, how you're making disc circles and other frame, how you use a string.


GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest §

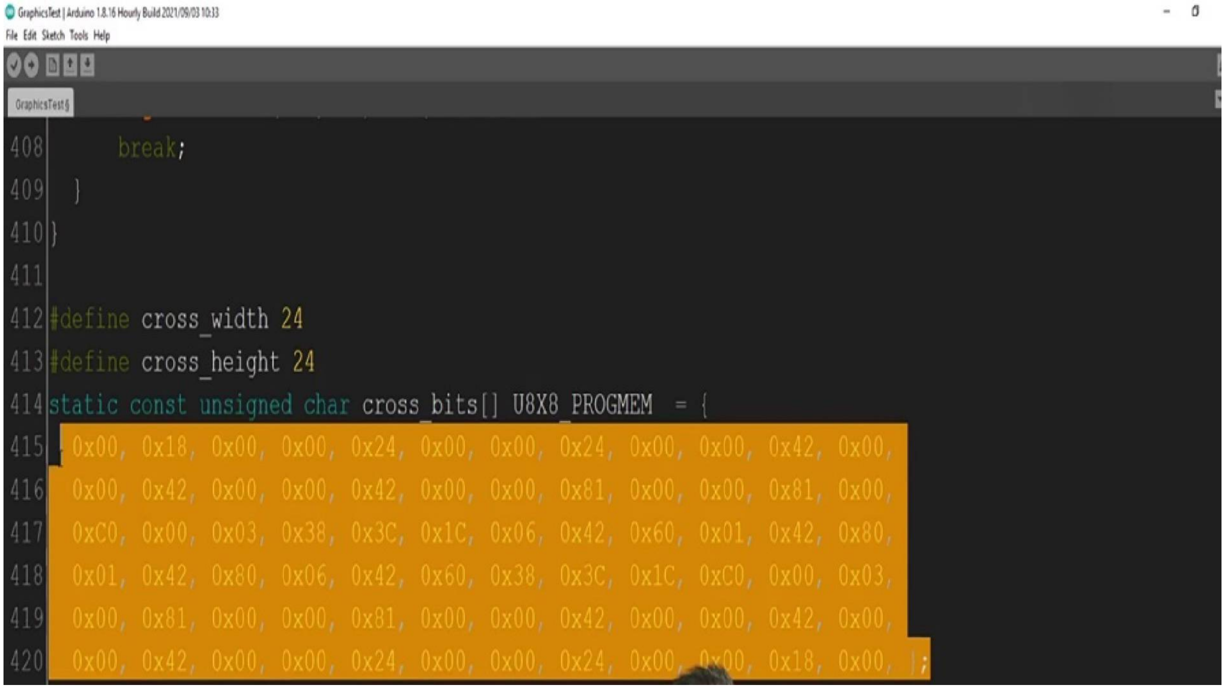
```
}

void u8g2_box_frame(uint8_t a) {
  u8g2.drawStr( 0, 0, "drawBox");
  u8g2.drawBox(5,10,20,10);
  u8g2.drawBox(10+a,15,30,7);
  u8g2.drawStr( 0, 30, "drawFrame");
  u8g2.drawFrame(5,10+30,20,10);
  u8g2.drawFrame(10+a,15+30,30,7);
}

void u8g2_disc_circle(uint8_t a) {
  u8g2.drawStr( 0, 0, "drawDisc");
  u8g2.drawDisc(10,18,9);
  u8g2.drawDisc(24+a,16,7);
  u8g2.drawStr( 0, 30, "drawCircle");
  u8g2.drawCircle(10,18+30,9);
  u8g2.drawCircle(24+a,16+30,7);
```

That's of course what we are using. You can also make the direction, change the direction, the line thickness. You can draw triangles. What's that ASCII symbols?


GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest §

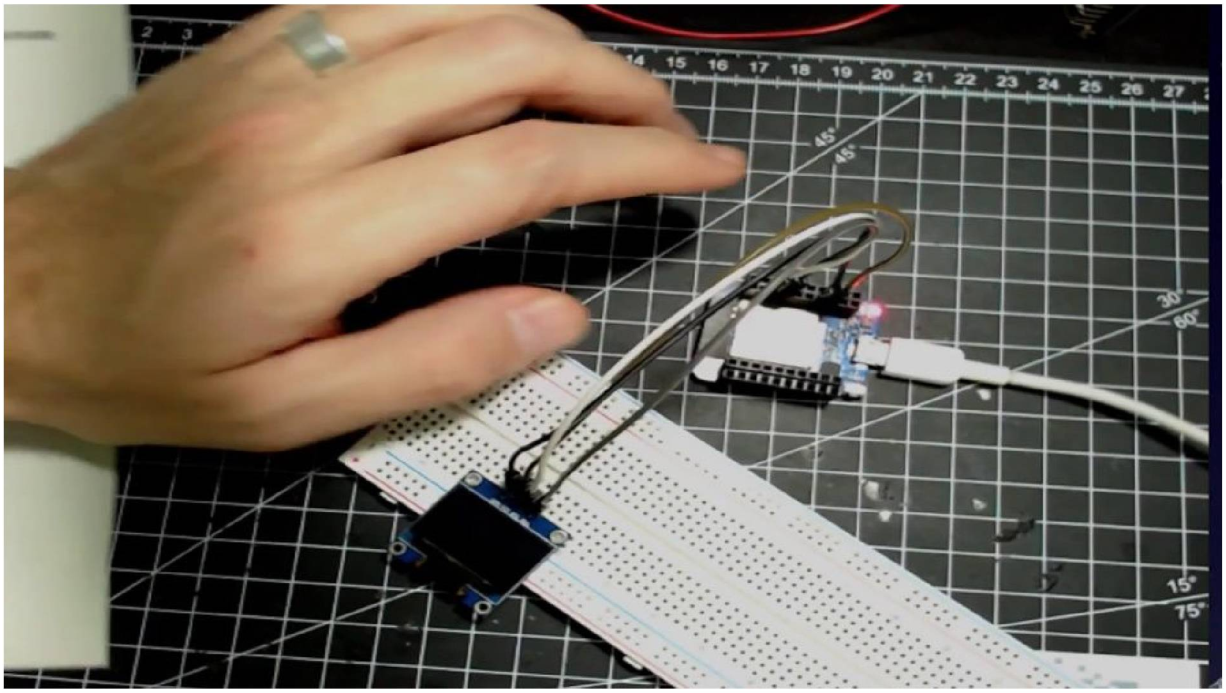
```
  }
}

void u8g2_extra_page(uint8_t a)
{
  u8g2.drawStr( 0, 0, "Unicode");
  u8g2.setFont(u8g2_font_unifont_t_symbols);
  u8g2.setFontPosTop();
  u8g2.drawUTF8(0, 24, "☀ ☁");
  switch(a) {
    case 0:
    case 1:
    case 2:
    case 3:
      u8g2.drawUTF8(a*3, 36, "☂");
      break;
    case 4:
    case 5:
```

Also, using X-ray pages, then you can use also and that's what you are, of project using some bitmap files or in this case, it's expenses for showing some drawings.

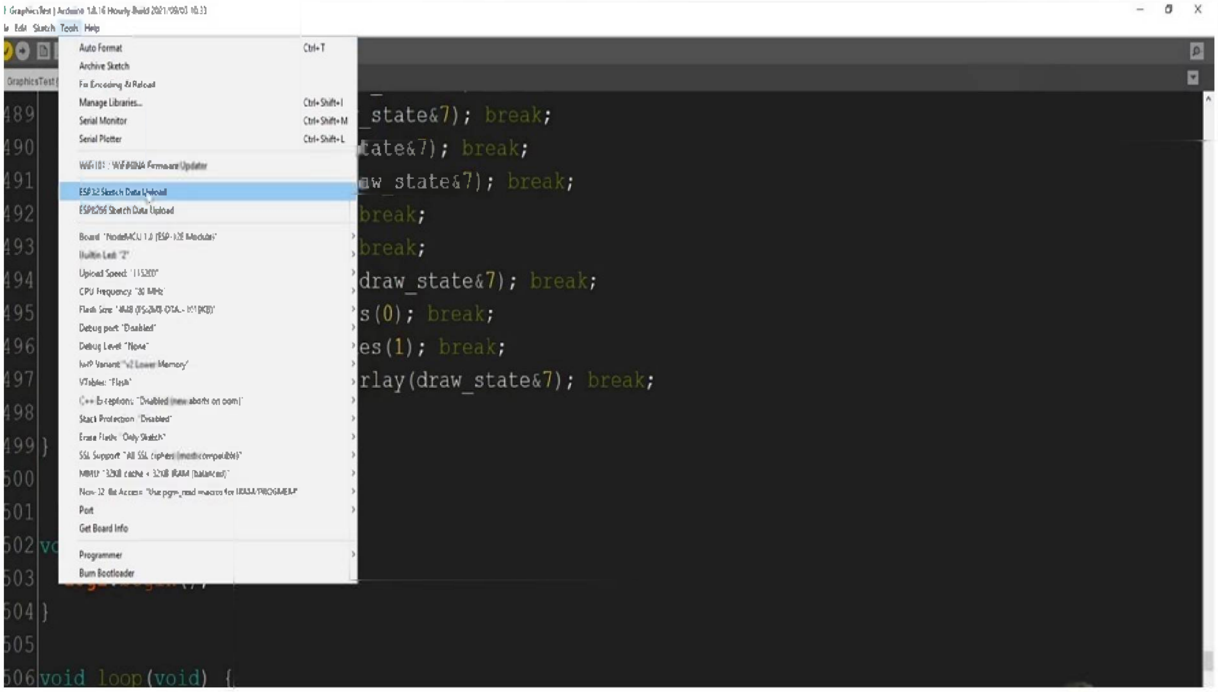

GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest §

```
      break;
  }
}

#define cross_width 24
#define cross_height 24
static const unsigned char cross_bits[] U8X8_PROGMEM  = {
  0x00, 0x18, 0x00, 0x00, 0x24, 0x00, 0x00, 0x24, 0x00, 0x00, 0x42, 0x00,
  0x00, 0x42, 0x00, 0x00, 0x42, 0x00, 0x00, 0x81, 0x00, 0x00, 0x81, 0x00,
  0xC0, 0x00, 0x03, 0x38, 0x3C, 0x1C, 0x06, 0x42, 0x60, 0x01, 0x42, 0x80,
  0x01, 0x42, 0x80, 0x06, 0x42, 0x60, 0x38, 0x3C, 0x1C, 0xC0, 0x00, 0x03,
  0x00, 0x81, 0x00, 0x00, 0x81, 0x00, 0x00, 0x42, 0x00, 0x00, 0x42, 0x00,
  0x00, 0x42, 0x00, 0x00, 0x24, 0x00, 0x00, 0x24, 0x00, 0x00, 0x18, 0x00, };
```

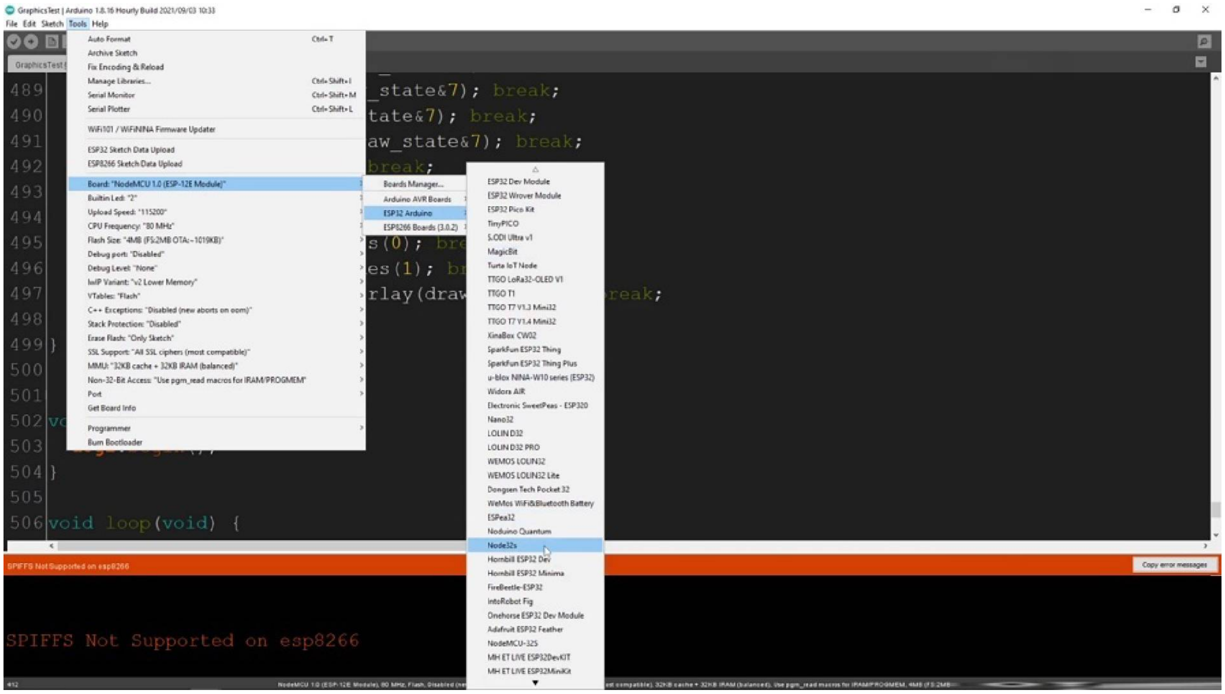
And also, we can overlay drawing and text. This is also what we are using later bitmap overlay, of course, scoring a little bit more down that he gets to know what's there.



OK, and that's the drawing function. OK. Let us compile the first graphic tests and connects our. Just be sorted through to the computer. While the host country is compiling. Switching back. Don't forget to.


GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP32 Sketch Data Upload
ESP8266 Sketch Data Upload
Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"
Port
Get Board Info
Programmer
Burn Bootloader

```
_state&7); break;
tate&7); break;
aw_state&7); break;
break;
break;
draw_state&7); break;
s(0); break;
es(1); break;
rlay(draw_state&7); break;

}


vo

}

void loop(void) {
```



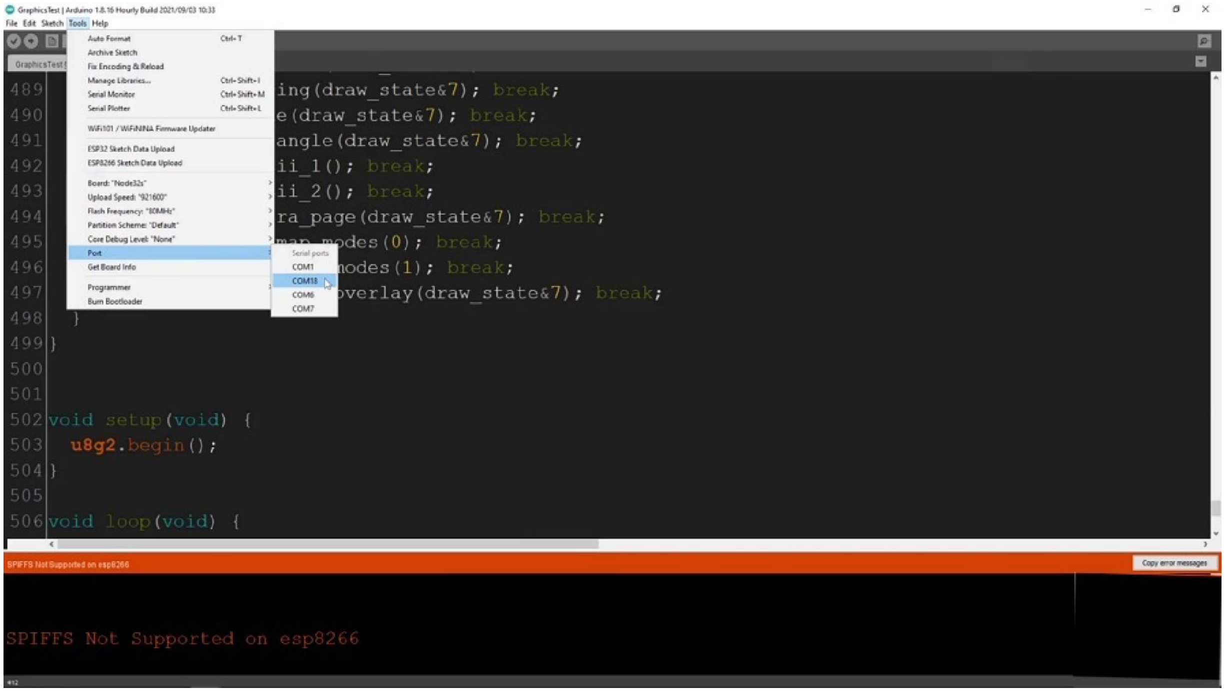
Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"

```
_state&7); break;
tate&7); break;
aw_state&7); break;
break;
s(0); bre
es(1); br
rlay(draw          reak;
}
vo

}

void loop(void) {
```

```
SPIFFS Not Supported on esp8266
```

```
SPIFFS Not Supported on esp8266
```

Set up to ride the rides boards, I'm using to search too. And also, don't forget to use the prop comports.


GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP32 Sketch Data Upload
ESP8266 Sketch Data Upload
Port
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM1
COM18
COM6
COM7
Copy error messages

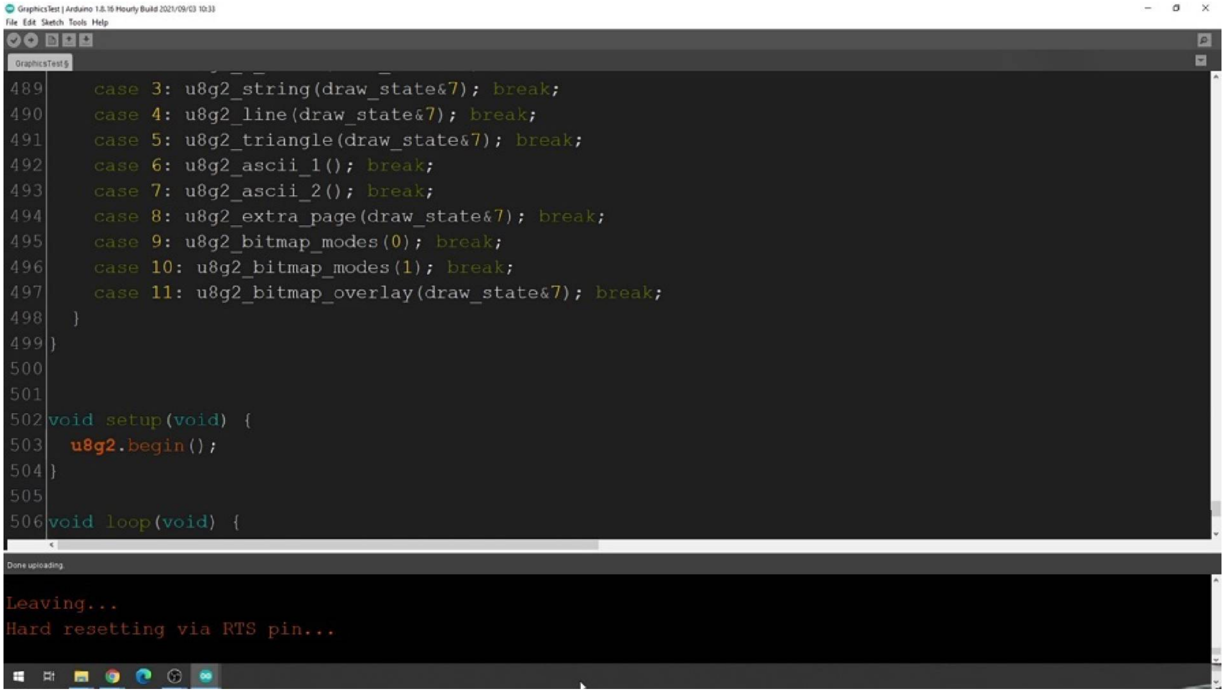
Board: "Node32s"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Partition Scheme: "Default"
Core Debug Level: "None"

```
ing(draw_state&7); break;
e(draw_state&7); break;
angle(draw_state&7); break;
ii_1(); break;
ii_2(); break;
ra_page(draw_state&7); break;
map_modes(0); break;
  modes(1); break;
  overlay(draw_state&7); break;
  }
}


void setup(void) {
  u8g2.begin();
}

void loop(void) {
```

```
SPIFFS Not Supported on esp8266
```

```
SPIFFS Not Supported on esp8266
```

So the comparison was successful, and then I'm uploading the sketch to my ECP search tool.



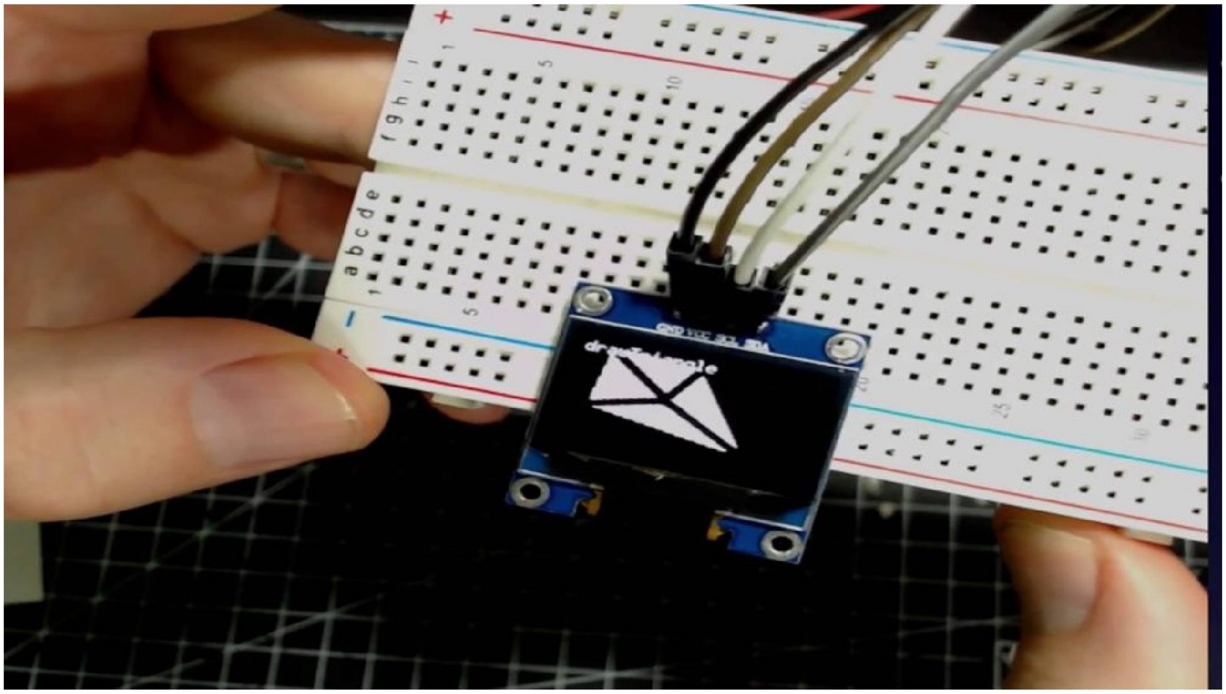
GraphicsTest | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
GraphicsTest §

```
      case 3: u8g2_string(draw_state&7); break;
      case 4: u8g2_line(draw_state&7); break;
      case 5: u8g2_triangle(draw_state&7); break;
      case 6: u8g2_ascii_1(); break;
      case 7: u8g2_ascii_2(); break;
      case 8: u8g2_extra_page(draw_state&7); break;
      case 9: u8g2_bitmap_modes(0); break;
      case 10: u8g2_bitmap_modes(1); break;
      case 11: u8g2_bitmap_overlay(draw_state&7); break;
  }
}


void setup(void) {
  u8g2.begin();
}

void loop(void) {
```

Done uploading.

```
Leaving...
Hard resetting via RTS pin...
```

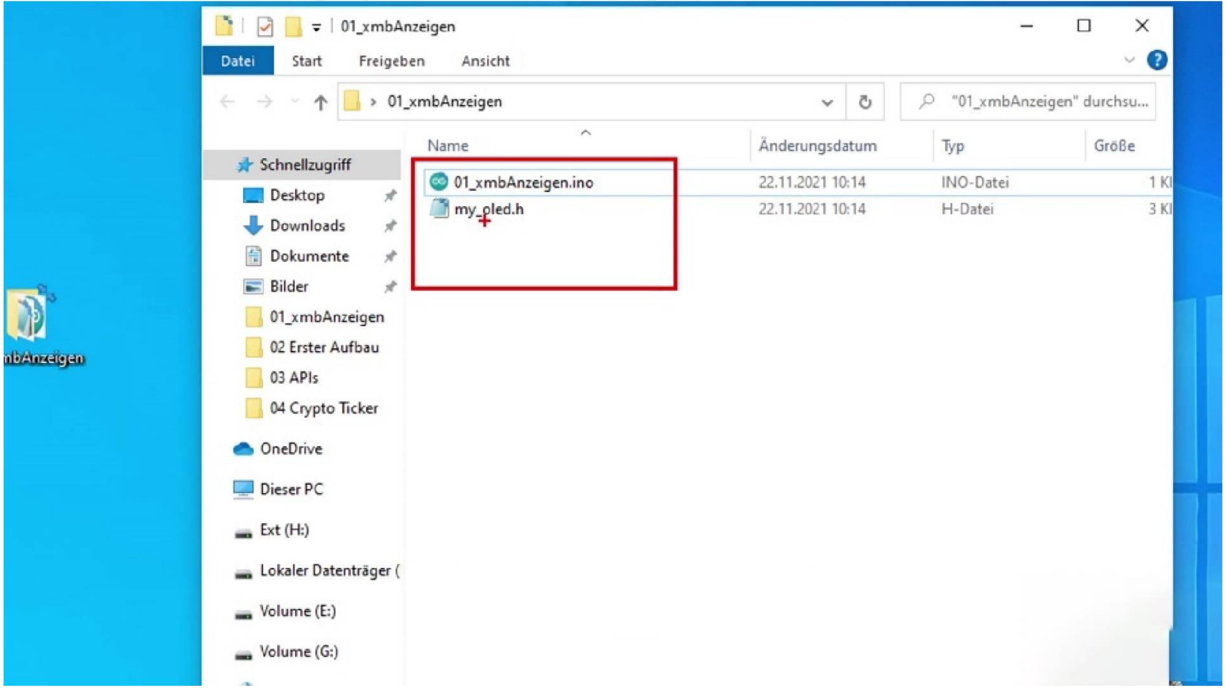
It's finished uploading, I'm resetting it change. To the tabletop, and now what we see is the graphic test.



So we see different lines, we see triangles also some texts. How many rows you can display, some unique codes. And this is what we are, of course using is the pigment overlay. So some expand photos, you also can animate the whole thing. And then it started all over again. So when you can see this test example, you have white everything in the right way and we're ready to go and make some code for ourselves.

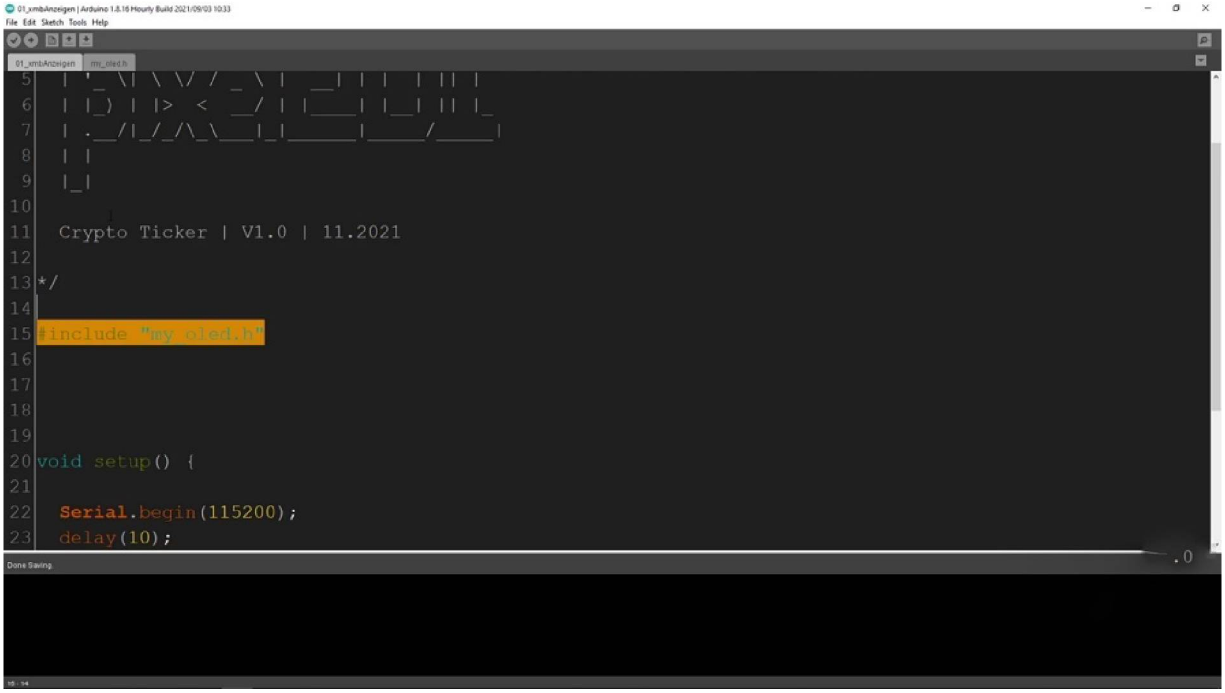
SHOW XBM FILES

we are going to convert cheap to expensive so that we've got the structure. And we want to display it on our old display.



Name	Änderungsdatum	Typ	Größe
01_xmbAnzeigen.ino	22.11.2021 10:14	INO-Datei	1 KI
my_oled.h	22.11.2021 10:14	H-Datei	3 KI

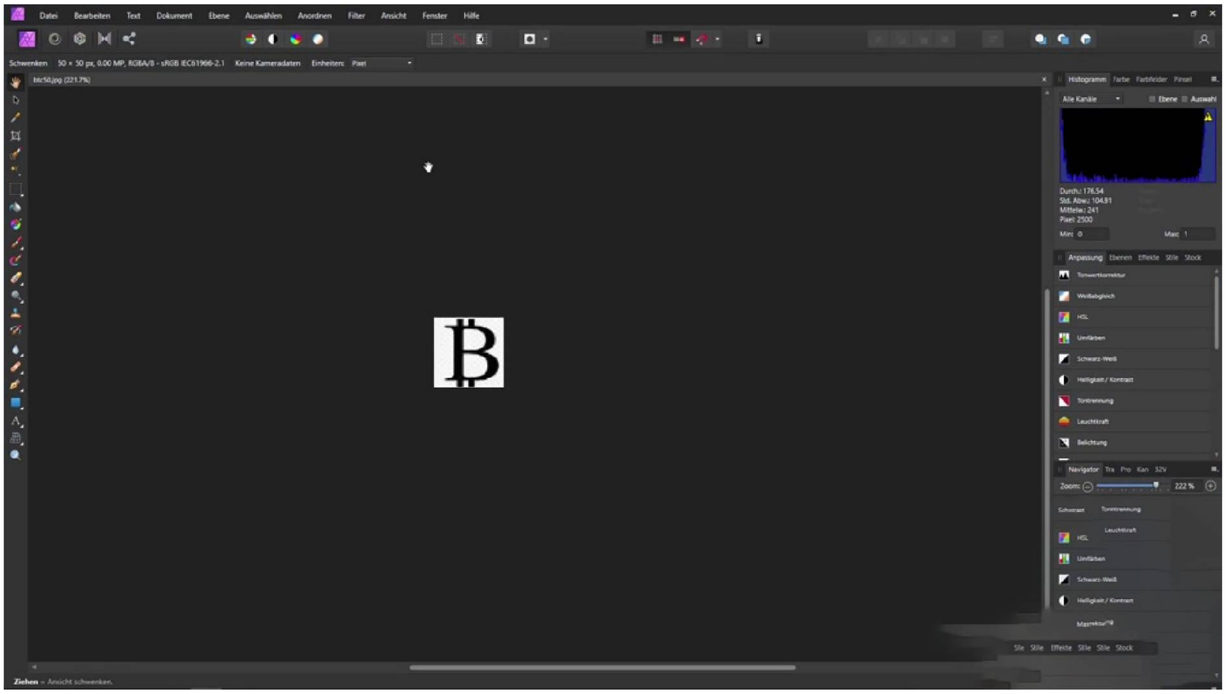
First of all, I created a new sketch and also a new photo. And in this folder I've also implemented insert a new file. It's called my old age. And when you open the main file, also distort, each file will be opened.



01_xmbAnzeigen | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
01_xmbAnzeigen
my_oled.h
Done Saving.

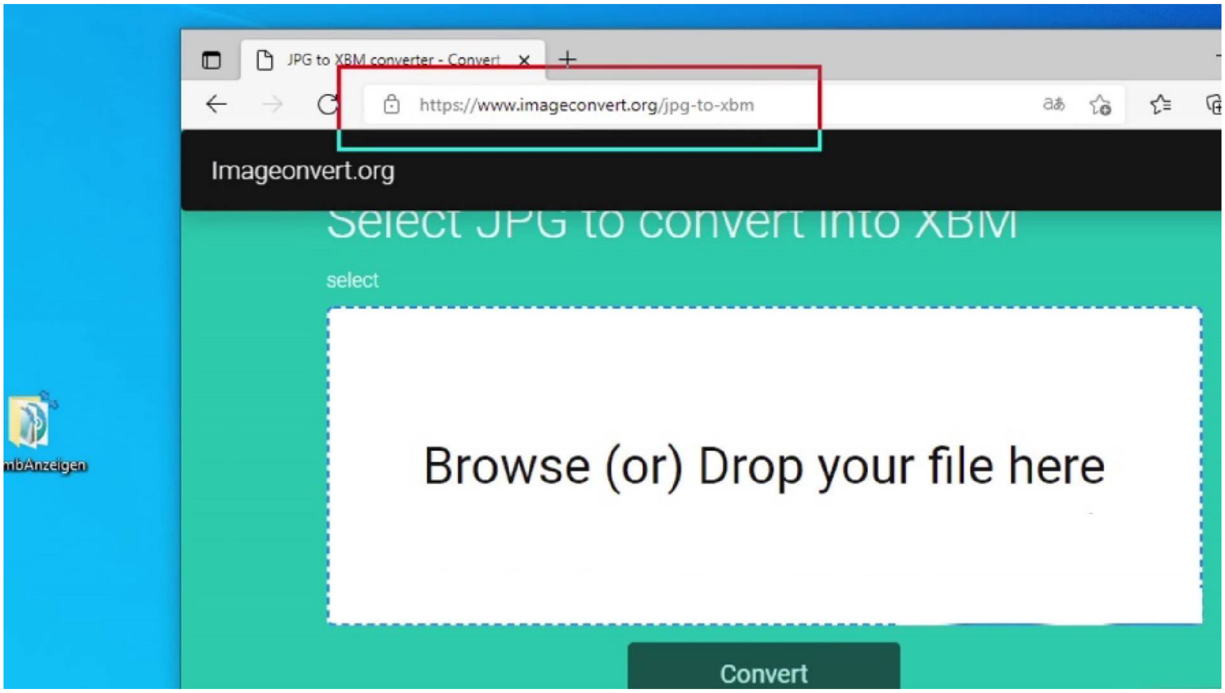
```
 | '_ \| \ \/ / _ \ |  __| | |  | || |
 | |_) | |>  <  __/ | |____| |__| || |_
 | .__/|_/_/\_\___|_|______|_____/_____|
 | |
 |_|

 Crypto Ticker | V1.0 | 11.2021

*/

#include "my_oled.h"



void setup() {

  Serial.begin(115200);
  delay(10);
```

And in this my old age, I'm pasting all the graphic data insights and in the main data in the main exam and saying data I make can include statement with my all on the edge. And as the name says, we including the WHO code inside this file. So this gives us some way to structure the whole code a little bit more in the proper way. This is to host coach Justin Silver begin and then delay. That's all.

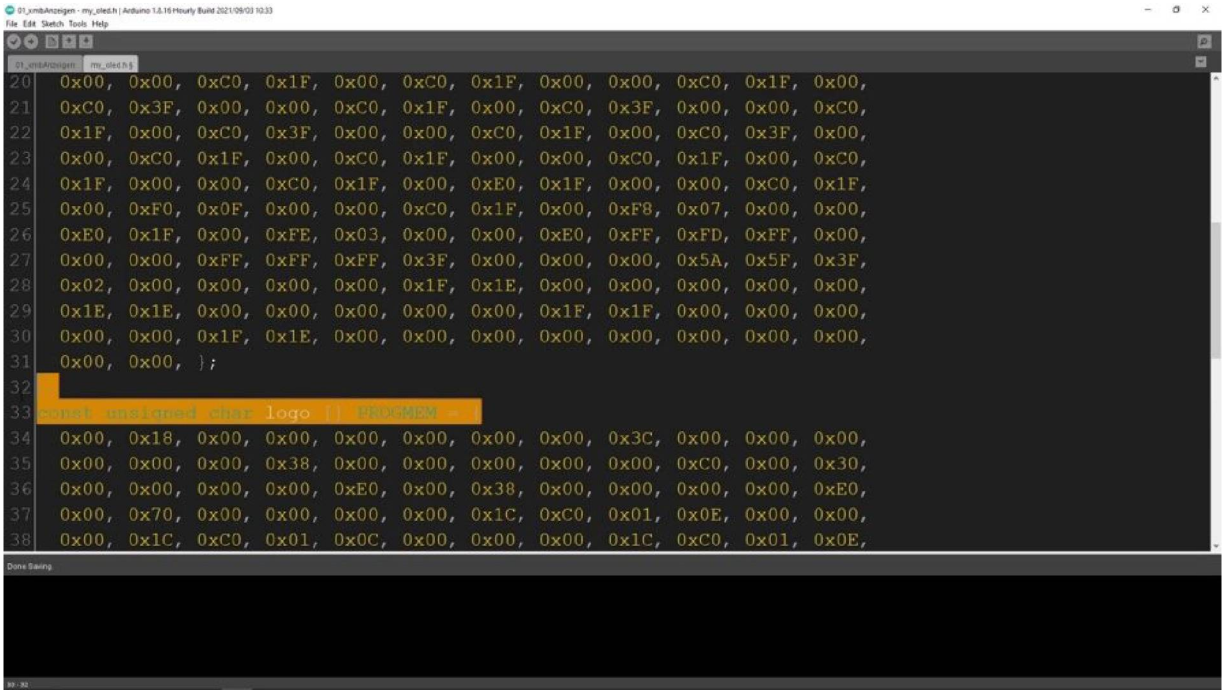


Datei Bearbeiten Text Dokument Ebene Auswählen Anordnen Filter Ansicht Fenster Hilfe
Schwenken 50 × 50 px, 0,00 MP, RGBA/8 - sRGB IEC61966-2.1 Keine Kameradaten Einheiten: Pixel
btc50.jpg (221.7%)
Histogramm Farbe Farbfelder Pinsel
Anpassung Ebenen Effekte Stile Stock
Navigator
Zoom: 222 %
Ziehen = Ansicht schwenken.

So the first thing what we are going to do now is we are open, a graphic program where you can edit some pixel graphics and I'm made as I made in your document with 50, with 50 in the higher. And now you can insert any kind of pixel graphic as you want. I use and bitcoin being in and. Be aware of the contrast in the foreground and in the background, because our display is very simple, and so I only have one color display. I think there are also two color display in this size, but here I only use a black foreground and in white background safeties two in JPEG.

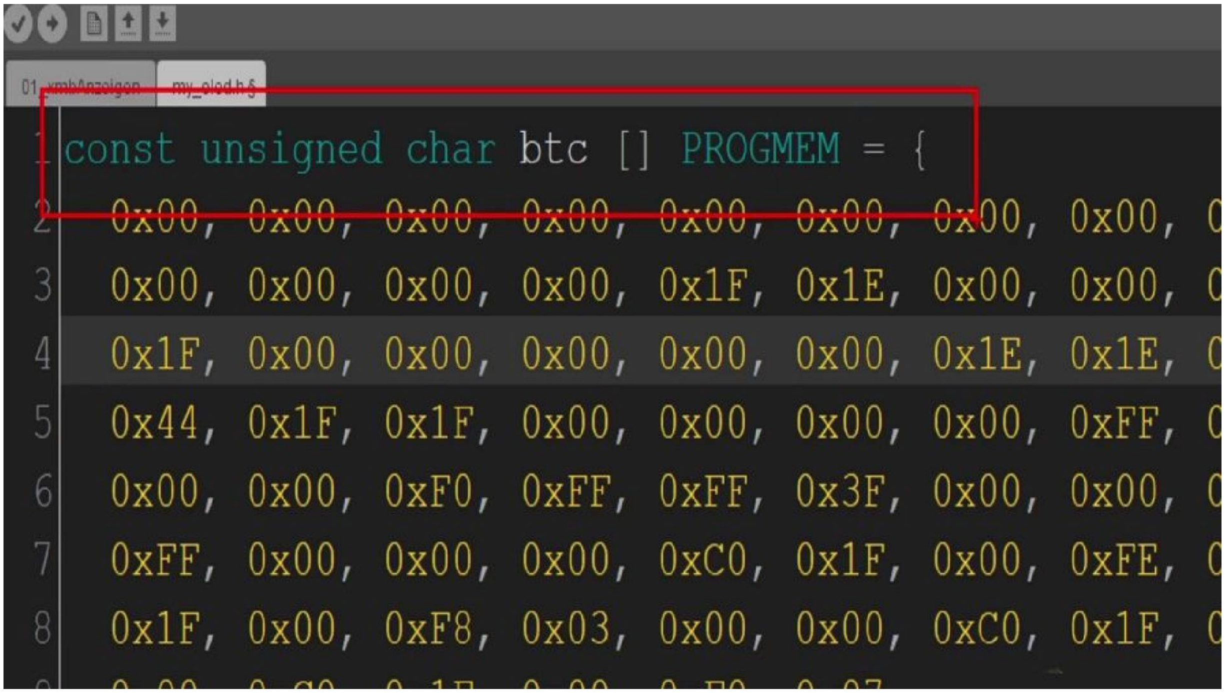

JPG to XBM converter - Convert
https://www.imageconvert.org/jpg-to-xbm
Imageonvert.org
Select JPG to convert into XBM
select
Browse (or) Drop your file here
Convert
mbAnzeigen

And then we are open. For example, this online convert image converted or cheap, cheap to expand, it's become very real at the end of 2021, but any other convert will do it the same, then trick and drop the chip. Click on Convert and I think it's already downloaded and you got here on file. So I'm not using to expand file. I click on Edit with no bid, for example, and there is what we want. We copy. The defiant switch back to our Arduino and.


01_xmbAnzeigen - my_oled.h | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
01_xmbAnzeigen
my_oled.h §
Done Saving

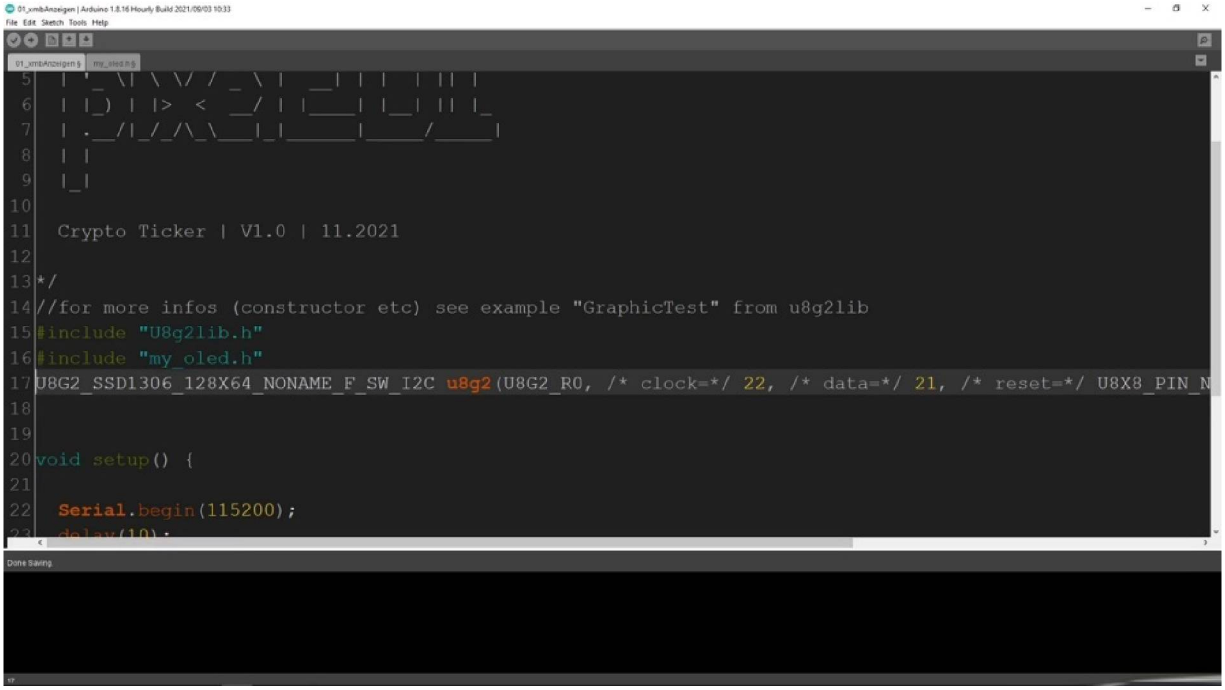
```
  0x00, 0x00, 0xC0, 0x1F, 0x00, 0xC0, 0x1F, 0x00, 0x00, 0xC0, 0x1F, 0x00,
  0xC0, 0x3F, 0x00, 0x00, 0xC0, 0x1F, 0x00, 0xC0, 0x3F, 0x00, 0x00, 0xC0,
  0x1F, 0x00, 0xC0, 0x3F, 0x00, 0x00, 0xC0, 0x1F, 0x00, 0xC0, 0x3F, 0x00,
  0x00, 0xC0, 0x1F, 0x00, 0xC0, 0x1F, 0x00, 0x00, 0xC0, 0x1F, 0x00, 0xC0,
  0x1F, 0x00, 0x00, 0xC0, 0x1F, 0x00, 0xE0, 0x1F, 0x00, 0x00, 0xC0, 0x1F,
  0x00, 0xF0, 0x0F, 0x00, 0x00, 0xC0, 0x1F, 0x00, 0xF8, 0x07, 0x00, 0x00,
  0xE0, 0x1F, 0x00, 0xFE, 0x03, 0x00, 0x00, 0xE0, 0xFF, 0xFD, 0xFF, 0x00,
  0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x3F, 0x00, 0x00, 0x00, 0x5A, 0x5F, 0x3F,
  0x02, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x1E, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x1F, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x1F, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, };

const unsigned char logo [] PROGMEM = {
  0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x30,
  0x00, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0xE0,
  0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x1C, 0xC0, 0x01, 0x0E, 0x00, 0x00,
  0x00, 0x1C, 0xC0, 0x01, 0x0C, 0x00, 0x00, 0x00, 0x1C, 0xC0, 0x01, 0x0E,
```

Paced the whole information from the new local insights to my older age, but I'm now doing is I'm hoping to.



```
const unsigned char btc [] PROGMEM = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x1F, 0x1E, 0x00, 0x00,
    0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x1E,
    0x44, 0x1F, 0x1F, 0x00, 0x00, 0x00, 0x00, 0xFF,
    0x00, 0x00, 0xF0, 0xFF, 0xFF, 0x3F, 0x00, 0x00,
    0xFF, 0x00, 0x00, 0x00, 0xC0, 0x1F, 0x00, 0xFE,
    0x1F, 0x00, 0xF8, 0x03, 0x00, 0x00, 0xC0, 0x1F,
```

The first thing because we need a concert unsigned Char, and we are calling this an PTZ and also the Brookman is important to use. So always use this structure that we got with to espouse. So due to no errors when you are displaying this kind of information or this kind of graphics. So this was the first step converting the JPEG to an experiment, including into a separate file.

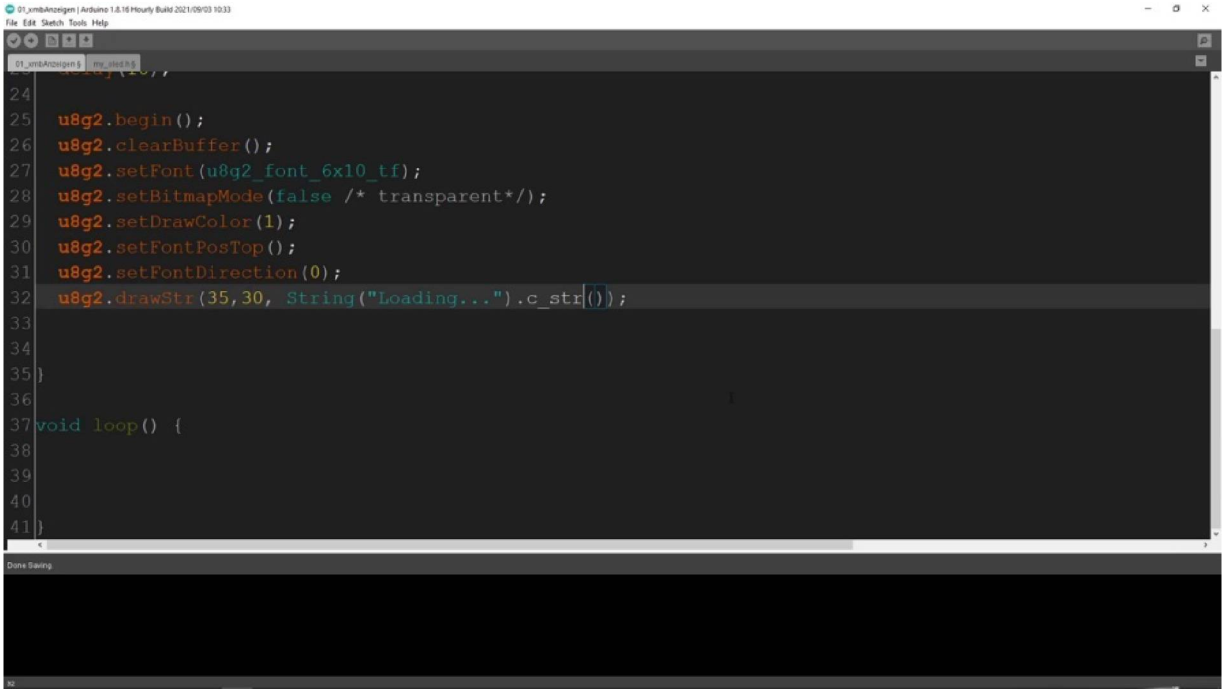


01_xmbAnzeigen | Arduino 1.8.16 Hourly Build 2021/09/03 10:33

```
  | '_ \| \ \/ / _ \ |  __| |  | || |
  | |_) | |>  <  __/ | |____| |__| || |_
  | .__/|_/_/\_\___|_|______|_____/_____|
  | |
  |_|

  Crypto Ticker | V1.0 | 11.2021

*/
//for more infos (constructor etc) see example "GraphicTest" from u8g2lib
#include "U8g2lib.h"
#include "my_oled.h"
U8G2_SSD1306_128X64_NONAME_F_SW_I2C u8g2(U8G2_R0, /* clock=*/ 22, /* data=*/ 21, /* reset=*/ U8X8_PIN_N


void setup() {

  Serial.begin(115200);
  delay(10);
```

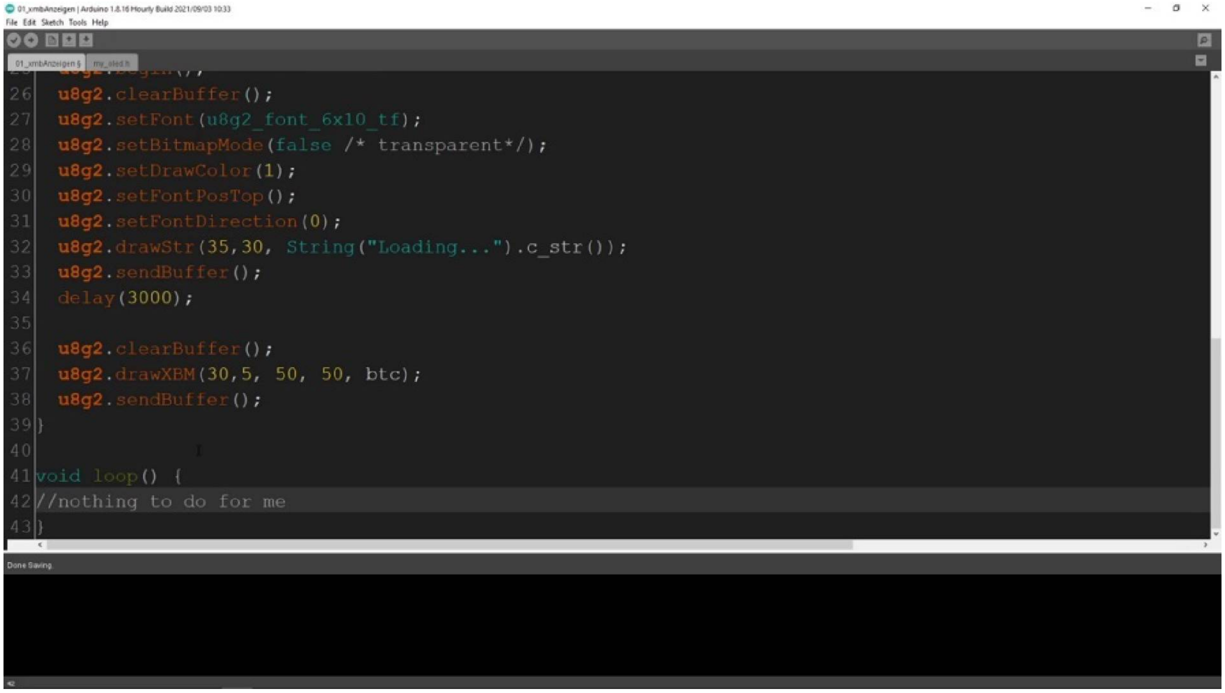
Done Saving.

Now we're switching back to our main file and we are starting with including our library u et je to dot h. And also the constructor what we used in the graphic tests just copy and paste to construct a line. And that's it for the whole part, which was the library.


01_xmbAnzeigen | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
01_xmbAnzeigen §
my_oled.h §
Done Saving

```
  delay(10);

  u8g2.begin();
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_6x10_tf);
  u8g2.setBitmapMode(false /* transparent*/);
  u8g2.setDrawColor(1);
  u8g2.setFontPosTop();
  u8g2.setFontDirection(0);
  u8g2.drawStr(35,30, String("Loading...").c_str());


}

void loop() {



}
```

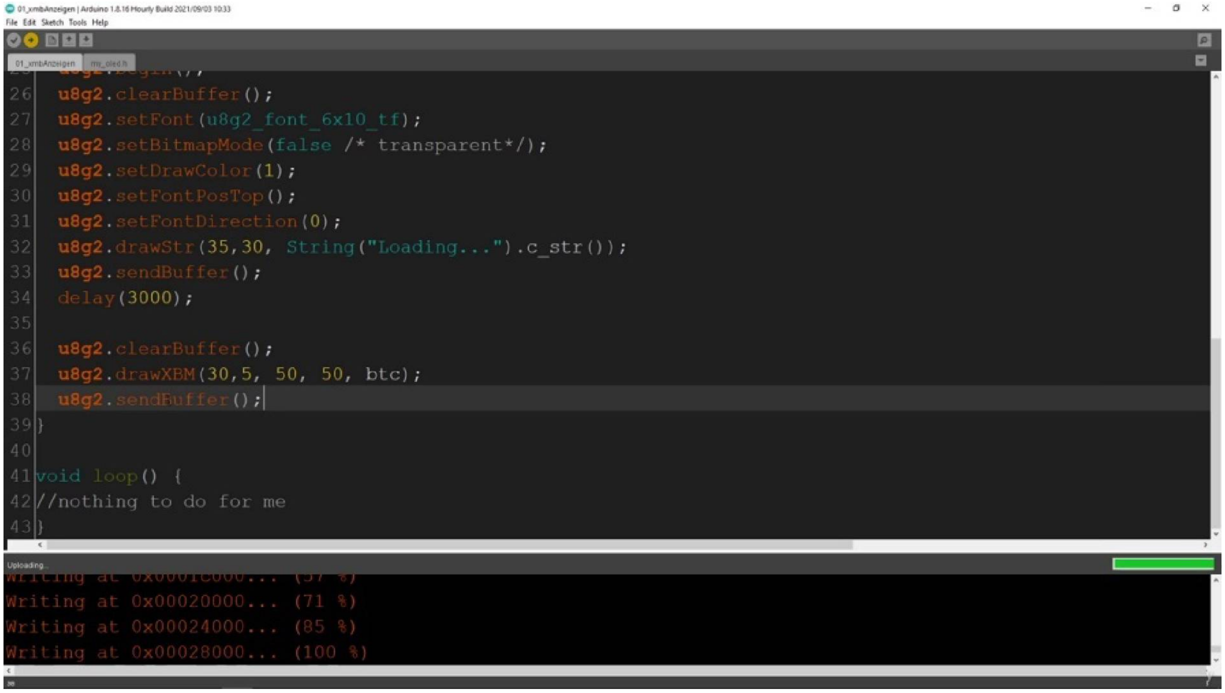
And then we are starting to. In the in the set up with our approach to showing, it's at the expense, first of all, we are making a begin of collaboration and then we are clearing out all of the existing content on it with clear buffer and then we are sitting in front because the first thing what we all want to do is now we want to write something on the display. We are setting the bits mode, the bitmap mode on false. That means that we have an transparent background so that we can also make an overlay of this when we are already had so. This one I've used from the example, from the graphic tests, and now we are once we are one to do some strings, you eighty two point two raw. SDR and then we are setting up to expand the way access to exclusive 128 pixels in my case and the y axis have 64 pixels, so I'm using thirty five to thirty am kind of in the middle of the of the display. And then I want to show unloading text, but it's not very easy because we are have to convert this one to a string and then give them in character, so we have to convert it to an C string.


01_xmbAnzeigen | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
01_xmbAnzeigen §
my_oled.h

```
  u8g2.begin();
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_6x10_tf);
  u8g2.setBitmapMode(false /* transparent*/);
  u8g2.setDrawColor(1);
  u8g2.setFontPosTop();
  u8g2.setFontDirection(0);
  u8g2.drawStr(35,30, String("Loading...").c_str());
  u8g2.sendBuffer();
  delay(3000);

  u8g2.clearBuffer();
  u8g2.drawXBM(30,5, 50, 50, btc);
  u8g2.sendBuffer();
}

void loop() {
//nothing to do for me
}
```

Done Saving.

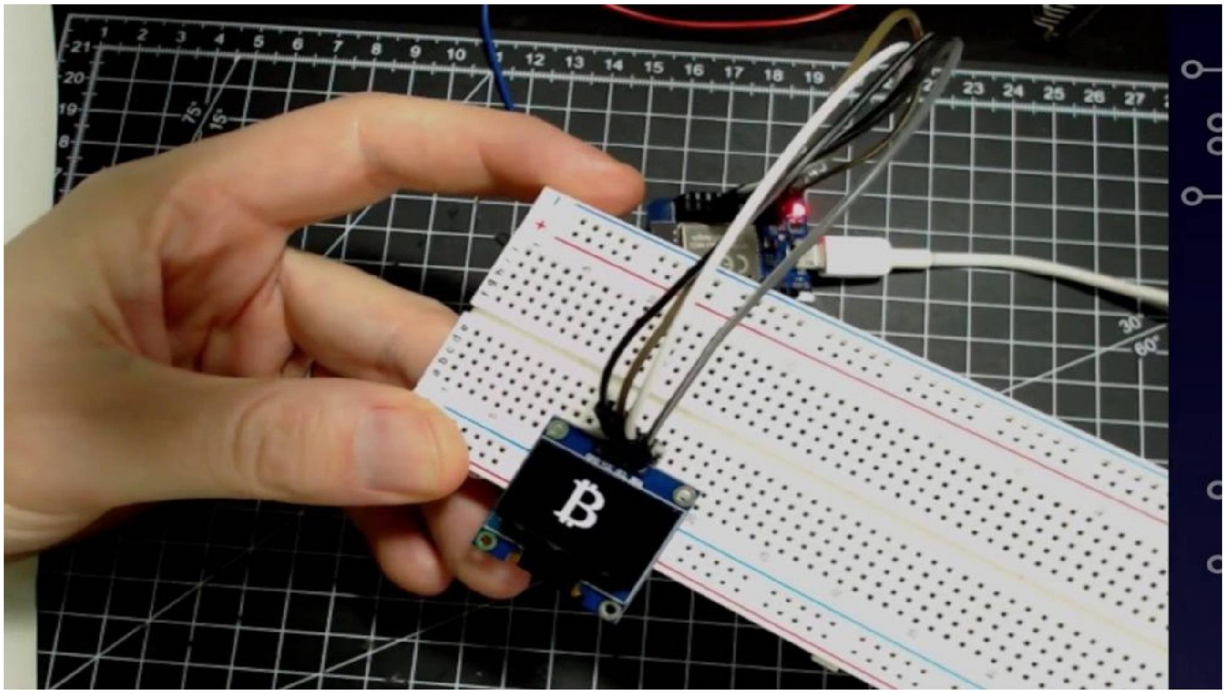
So then the last part, the last points to show it on the display is sent buffer with this statement. We are actually converting all these statements before and sending to the display. Then we're making a delay because we are in the testing phase from maybe two or three seconds and now we are clearing the whole. Disappear once again, because now we want to display our new generated BTC file. Therefore, we're switching back and now we are using you to point to raw extreme. X x y x is then the size of the image, I have set it up to a 50 to 50. And the name of the. Of the file and this is called Pizzey. OK. Then. So last month we send it sent buffer loop. Nothing to do for me, because this is just the first test, if we have it converted in the right way, then take a look in the compiler.


01_xmbAnzeigen | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
01_xmbAnzeigen
ms_oled.h
Uploading...

```
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_6x10_tf);
  u8g2.setBitmapMode(false /* transparent*/);
  u8g2.setDrawColor(1);
  u8g2.setFontPosTop();
  u8g2.setFontDirection(0);
  u8g2.drawStr(35,30, String("Loading...").c_str());
  u8g2.sendBuffer();
  delay(3000);

  u8g2.clearBuffer();
  u8g2.drawXBM(30,5, 50, 50, btc);
  u8g2.sendBuffer();
}

void loop() {
//nothing to do for me
}
```

```
Writing at 0x00020000... (71 %)
Writing at 0x00024000... (85 %)
Writing at 0x00028000... (100 %)
```

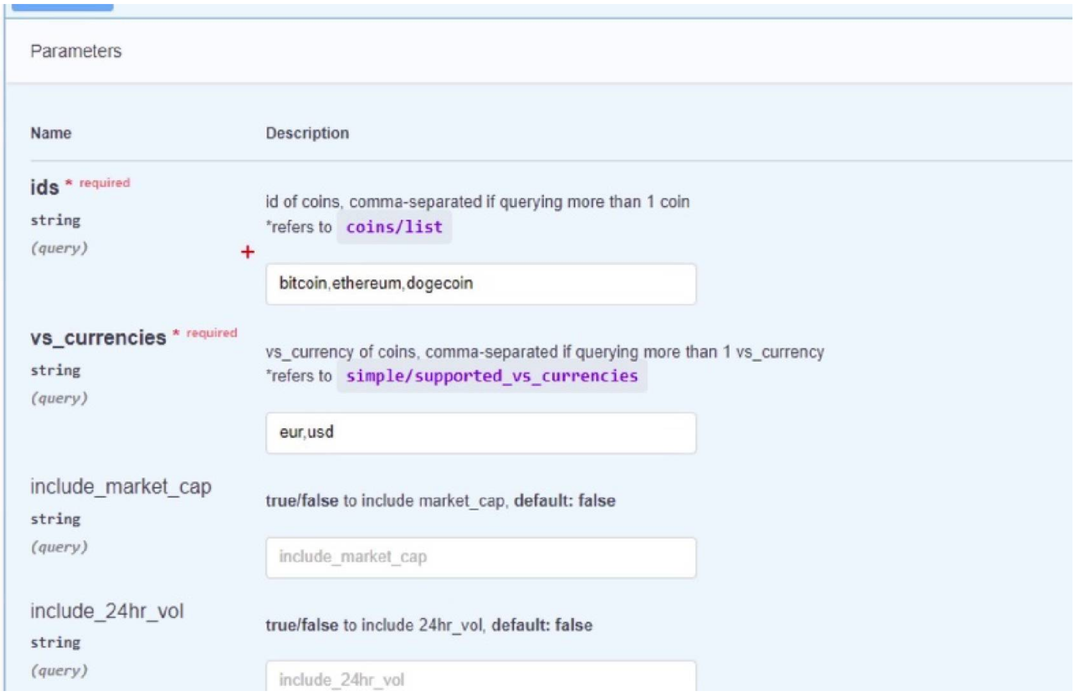
If we misspelled something looks good, then use be connected. Upload the sketch.



Then we're switching to camera. There we go, we have our loading statement. And now the generated extreme fund is showing once again, I'm. Restarts, the new here is the loading. And then the bitcoin is showing up, and we defined it to certain six and five on the y axis. And as you can see, it's that simple to showing some a bitmap Pfizer face on an old display with our library.

MORE CRYPTOS

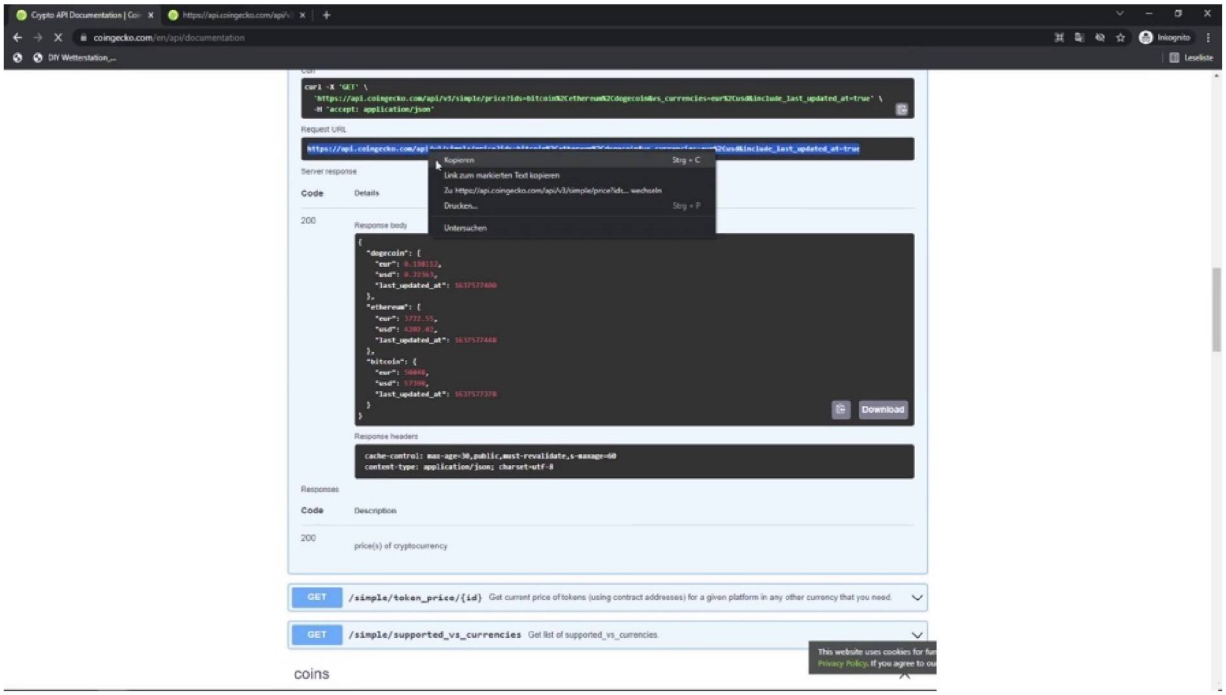
we're adapting to CoinGecko API so that we fetch more cryptos and descriptors, we are going to try to display on the alert with ensemble and also list current price. So let's start with adapting the CoinGecko.



Parameters

Name	Description
ids * required string *(query)*	id of coins, comma-separated if querying more than 1 coin *refers to `coins/list` bitcoin,ethereum,dogecoin
vs_currencies * required string *(query)*	vs_currency of coins, comma-separated if querying more than 1 vs_currency *refers to `simple/supported_vs_currencies` eur,usd
include_market_cap string *(query)*	**true/false** to include market_cap, **default: false** include_market_cap
include_24hr_vol string *(query)*	**true/false** to include 24hr_vol, **default: false** include_24hr_vol

You are therefore switching back to the CoinGecko side. And what I'm now adapting is the ideas. We are also making Ethereum and Dogecoin into the ideas comma separated without any space euros and USD, and also to include last updated is true click and executes and then we can



```
curl -X 'GET' \
  'https://api.coingecko.com/api/v3/simple/price?ids=bitcoin%2Cethereum%2Cdogecoin&vs_currencies=eur%2Cusd&include_last_updated_at=true' \
  -H 'accept: application/json'
```

Request URL

```
https://api.coingecko.com/api                    2Cusd&include_last_updated_at=true
```

Server response

Code Details

200

Response body

```
{
  "dogecoin": {
    "eur": 0.198112,
    "usd": 0.22363,
    "last_updated_at": 1637577400
  },
  "ethereum": {
    "eur": 3722.55,
    "usd": 4202.02,
    "last_updated_at": 1637577448
  },
  "bitcoin": {
    "eur": 50048,
    "usd": 57398,
    "last_updated_at": 1637577378
  }
}
```

Response headers

```
cache-control: max-age=30,public,must-revalidate,s-maxage=60
content-type: application/json; charset-utf-8
```

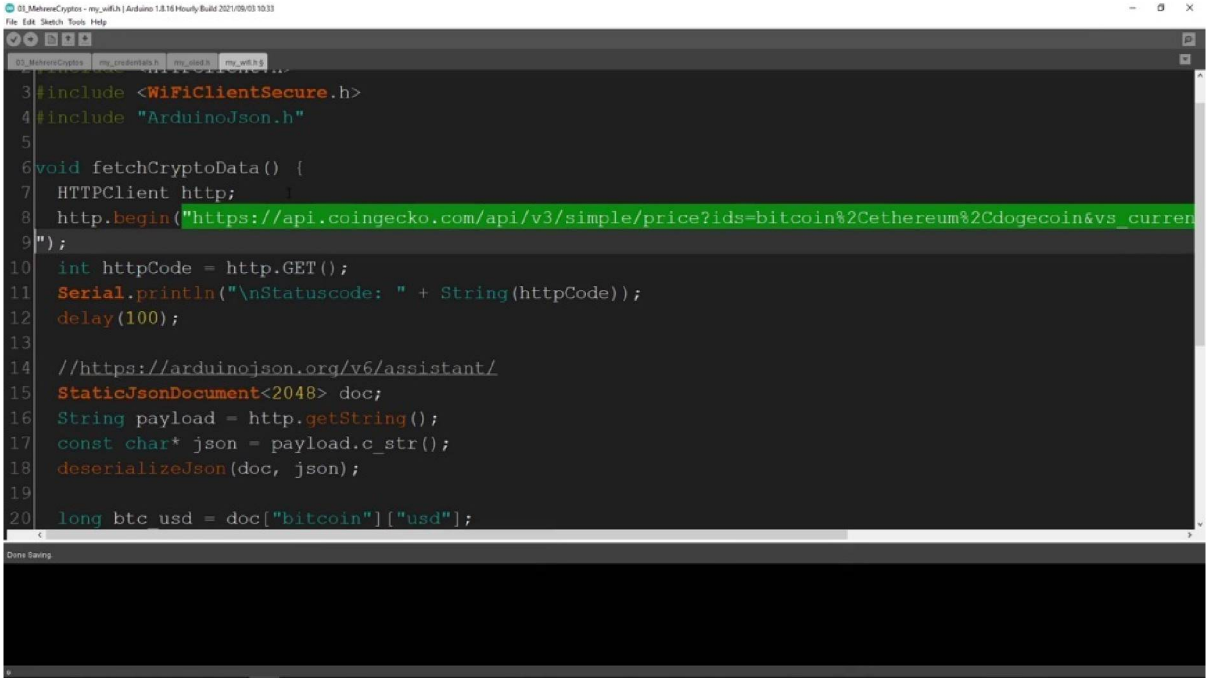
Responses

Code Description

200 price(s) of cryptocurrency

Crypto API Documentation | Coi
coingecko.com/en/api/documentation
Kopieren Strg + C
Link zum markierten Text kopieren
Zu https://api.coingecko.com/api/v3/simple/price?ids... wechseln
Drucken... Strg + P
Untersuchen
Download
GET /simple/token_price/{id} Get current price of tokens (using contract addresses) for a given platform in any other currency that you need.
GET /simple/supported_vs_currencies Get list of supported_vs_currencies.
coins

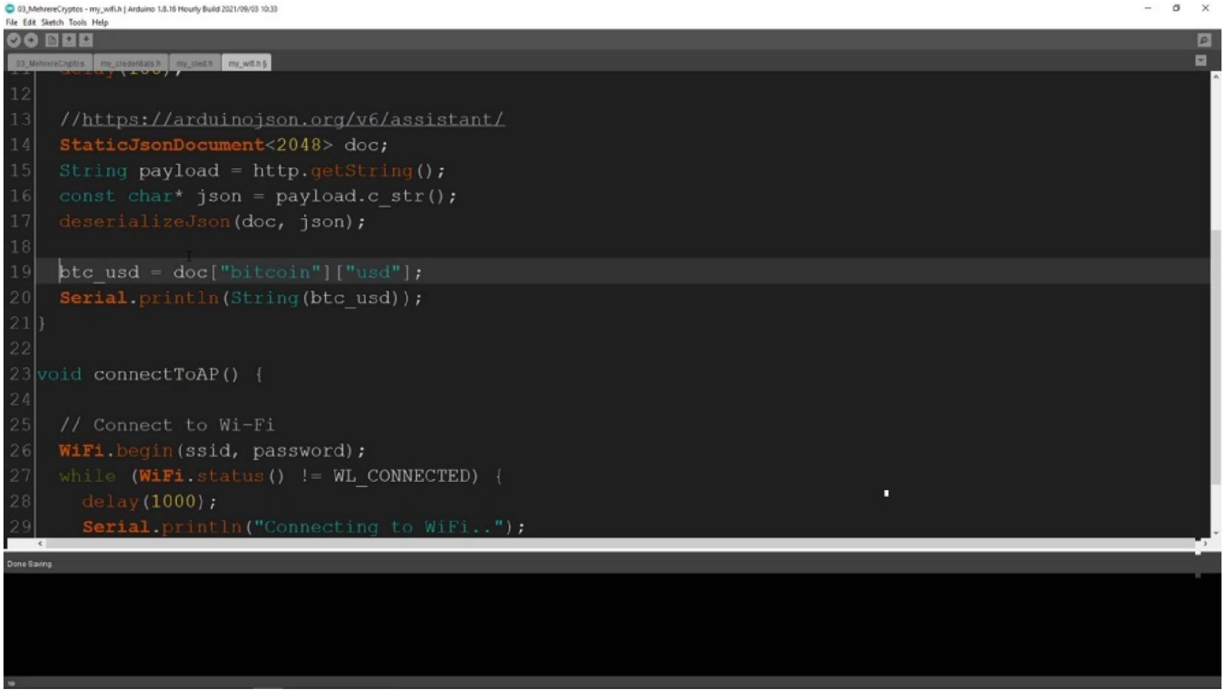
see the new Jason three objects with you were used to and last updated elements, each of them. Looks very good to me. We are copying the request you are switching back to the Arduino and replacing the euro with.



```
#include <WiFiClientSecure.h>
#include "ArduinoJson.h"

void fetchCryptoData() {
  HTTPClient http;
  http.begin("https://api.coingecko.com/api/v3/simple/price?ids=bitcoin%2Cethereum%2Cdogecoin&vs_curren
");
  int httpCode = http.GET();
  Serial.println("\nStatuscode: " + String(httpCode));
  delay(100);

  //https://arduinojson.org/v6/assistant/
  StaticJsonDocument<2048> doc;
  String payload = http.getString();
  const char* json = payload.c_str();
  deserializeJson(doc, json);

  long btc_usd = doc["bitcoin"]["usd"];
```

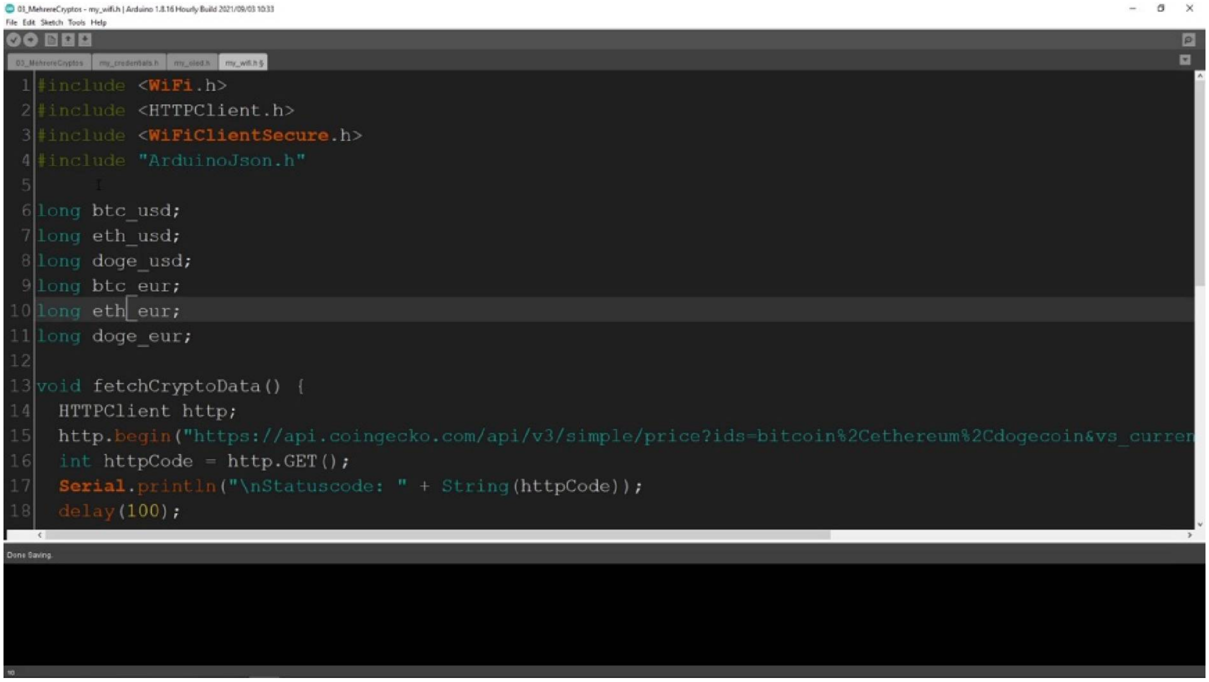
01_MehrereCryptos - my_wifi.h | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
Done Saving.

The new one, this was the first step and the next step we can see here that we have starts the Bitcoin USD price in a local variable.


03_MehreneCryptos - my_wifi.h | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
03_MehreneCryptos
my_credentials.h
my_oled.h
my_wifi.h §
Done Saving.

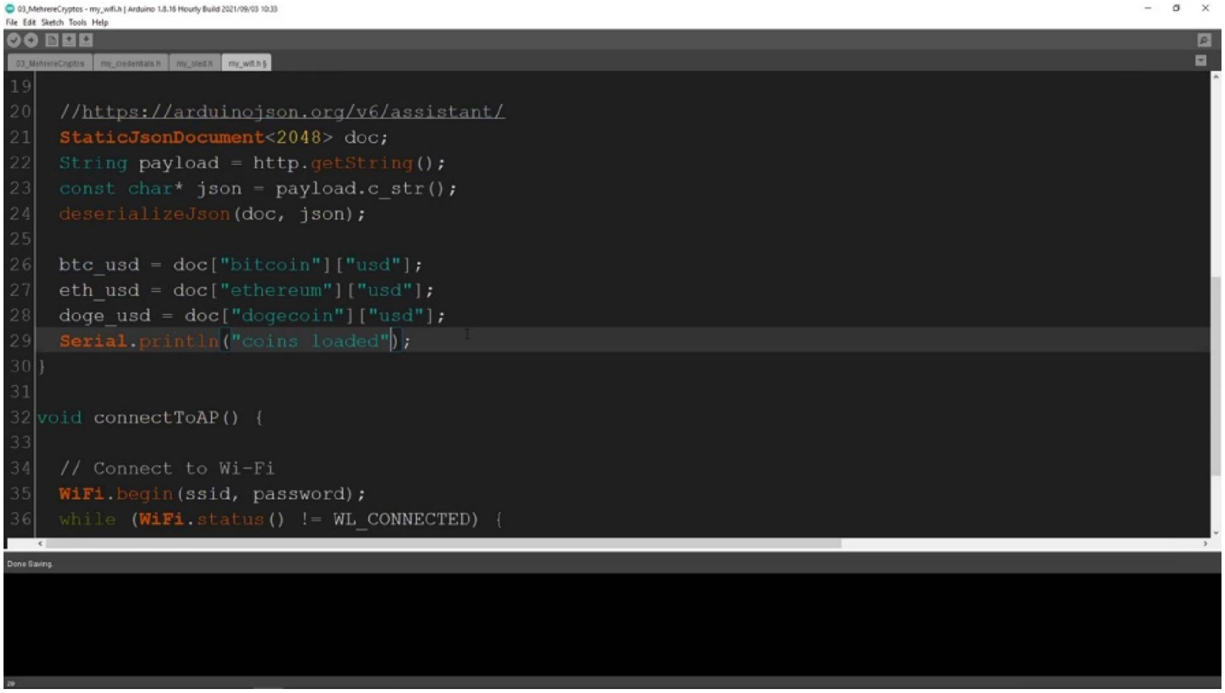
```
  delay(100);

  //https://arduinojson.org/v6/assistant/
  StaticJsonDocument<2048> doc;
  String payload = http.getString();
  const char* json = payload.c_str();
  deserializeJson(doc, json);

  btc_usd = doc["bitcoin"]["usd"];
  Serial.println(String(btc_usd));
}

void connectToAP() {

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
```

That's not good for our next purpose. Therefore, we are going to the top of my wife team and making some global very. That means that we have access to these variables outside the function. Bitcoin used to, Ethereum used to and also use steam.


01_MehreneCryptos - my_wifi.h | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
03_MehreneCryptos
my_credentials.h
my_oled.h
my_wifi.h §
Done Saving.

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiClientSecure.h>
#include "ArduinoJson.h"

long btc_usd;
long eth_usd;
long doge_usd;
long btc_eur;
long eth_eur;
long doge_eur;

void fetchCryptoData() {
  HTTPClient http;
  http.begin("https://api.coingecko.com/api/v3/simple/price?ids=bitcoin%2Cethereum%2Cdogecoin&vs_curren
  int httpCode = http.GET();
  Serial.println("\nStatuscode: " + String(httpCode));
  delay(100);
```

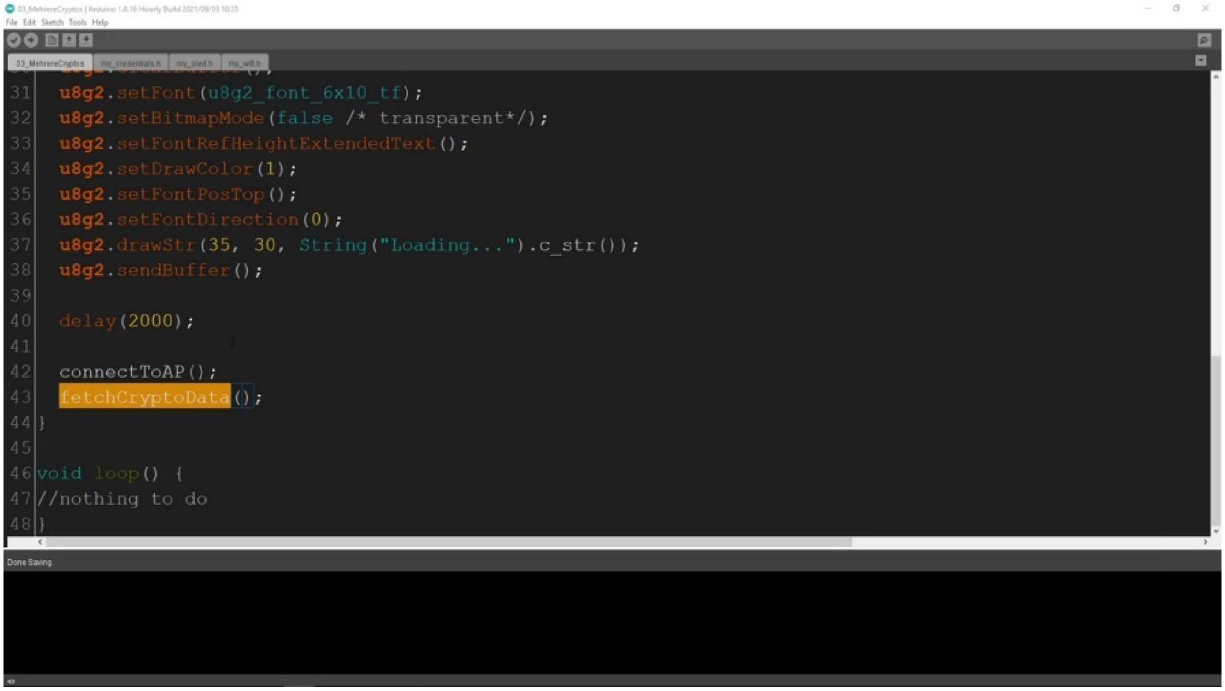
And we are making the same this year because maybe you want to also visualize the euro's. And Long and Deutsch Euros, but not.


03_MehreneCryptos - my_wifi.h | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
03_MehreneCryptos
my_credentials.h
my_oled.h
my_wifi.h §

```
  //https://arduinojson.org/v6/assistant/
  StaticJsonDocument<2048> doc;
  String payload = http.getString();
  const char* json = payload.c_str();
  deserializeJson(doc, json);

  btc_usd = doc["bitcoin"]["usd"];
  eth_usd = doc["ethereum"]["usd"];
  doge_usd = doc["dogecoin"]["usd"];
  Serial.println("coins loaded");
}

void connectToAP() {

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

Done Saving.

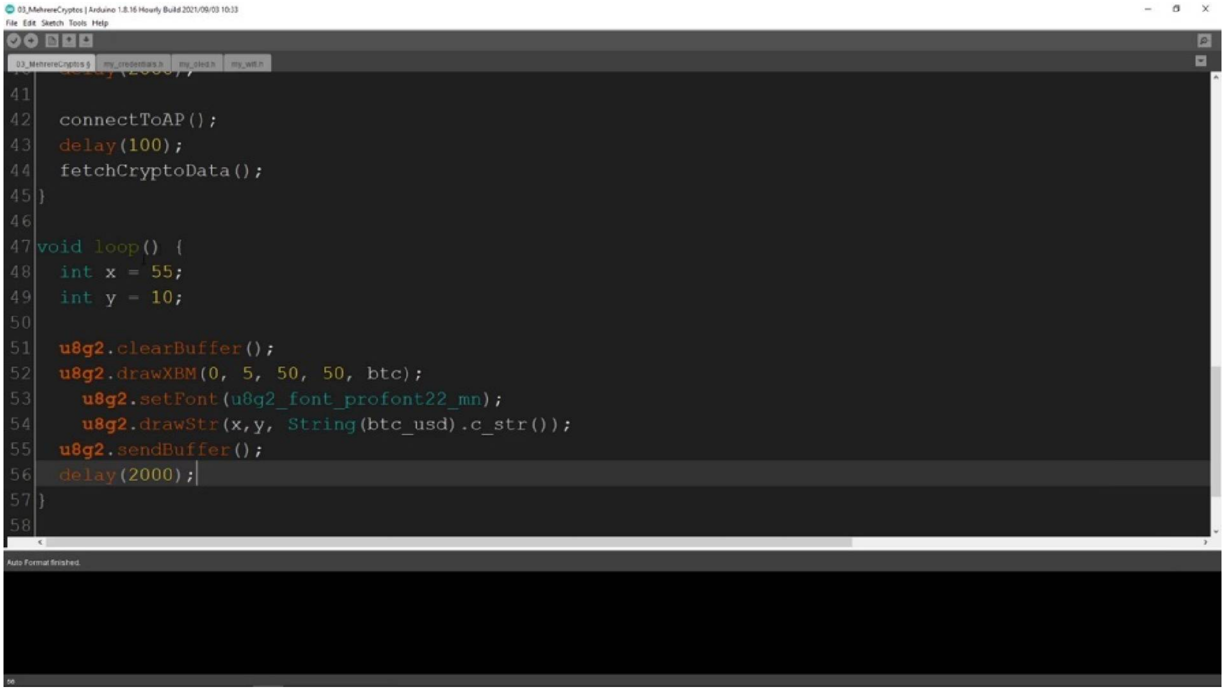
So then going back to our group to date on Ether, I'm used to is DOC. Um. Let me throw on, I was misspell it. USDA and the same with Dutch. I'm not sure. Greetings to Mr. Elon Musk. Dogecoin, you, Steve. And we are printing out. Coins loaded, so so far, so good, we just adopted the new rule, making some.


03_MehrereCryptos | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
03_MehrereCryptos
my_credentials.h
my_oled.h
my_wifi.h

```
  u8g2.setFont(u8g2_font_6x10_tf);
  u8g2.setBitmapMode(false /* transparent*/);
  u8g2.setFontRefHeightExtendedText();
  u8g2.setDrawColor(1);
  u8g2.setFontPosTop();
  u8g2.setFontDirection(0);
  u8g2.drawStr(35, 30, String("Loading...").c_str());
  u8g2.sendBuffer();

  delay(2000);

  connectToAP();
  fetchCryptoData();
}

void loop() {
//nothing to do
}
```

Done Saving.

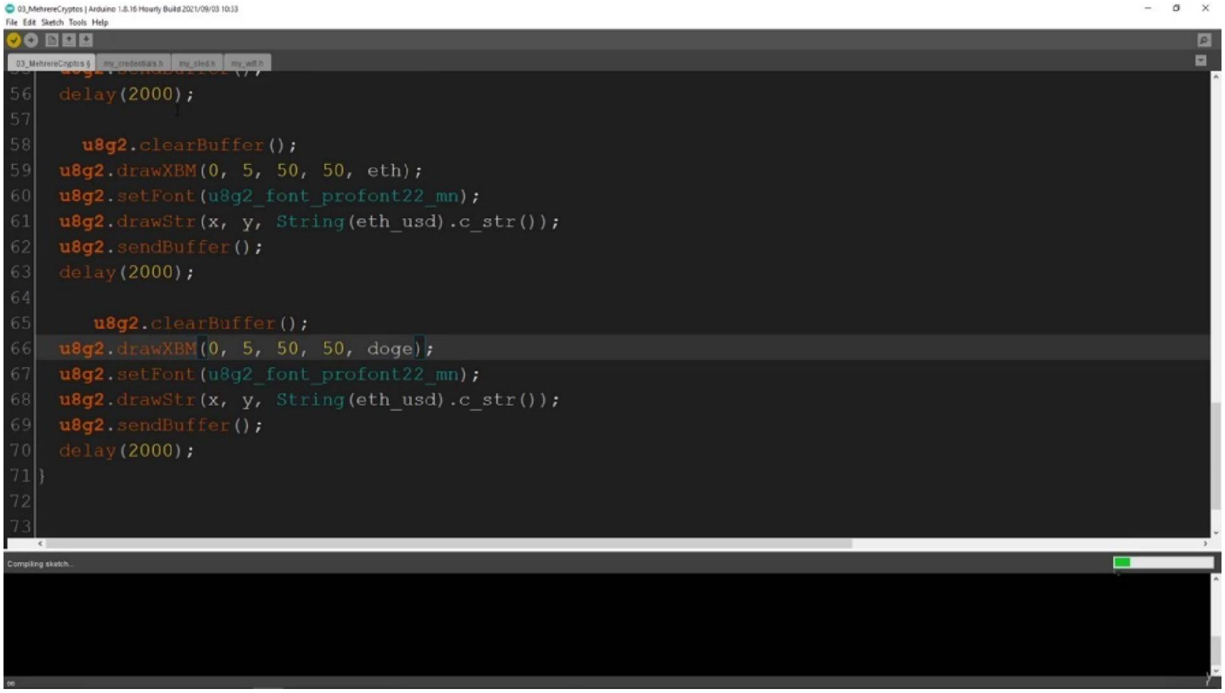
Global variables going back to our main data and therefore we're making some quick and dirty. Visualization just for debugging purpose.


03_MehrereCryptos | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
03_MehrereCryptos §
my_credentials.h
my_oled.h
my_wifi.h

```
  delay(2000);

  connectToAP();
  delay(100);
  fetchCryptoData();
}

void loop() {
  int x = 55;
  int y = 10;

  u8g2.clearBuffer();
  u8g2.drawXBM(0, 5, 50, 50, btc);
    u8g2.setFont(u8g2_font_profont22_mn);
    u8g2.drawStr(x,y, String(btc_usd).c_str());
  u8g2.sendBuffer();
  delay(2000);
}

```

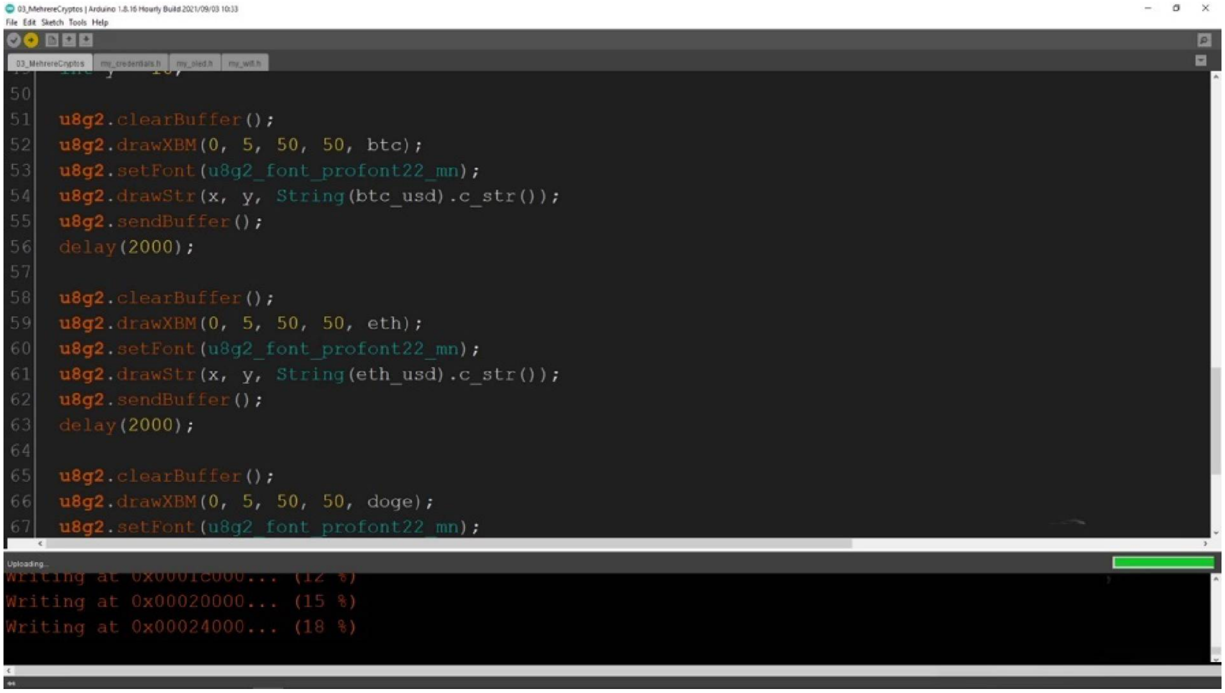
Auto Format finished.

So I'm making a small delay here, and then we are going to the loop. First of all, we are setting up some global awesome local barriers as interchange for our visualization epsilon starting point soon. And then we starting with clearing the buffer and then we are throwing our first expen file. It's called or it's on the exposition zero y five. It's for the size 550 and as we did before. It's in my wallet. It's called BTC. Then we're setting the funds. It's in smaller funds. And now we can draw on string a G to dot draw string. We are sitting in the very X and Y, and then we are posting the BTC USD price. But as we did before, we have to convert it to a string because now it's and long and not a string. We cannot pass the string. It's needed as and character. OK. Then we have to send it. To send Buffon and for testing purposes, we made him delay from 2000.


03_MehrereCryptos | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
03_MehrereCryptos §
my_credentials.h
my_oled.h
my_wifi.h
Compiling sketch...

```
  u8g2.sendBuffer();
  delay(2000);

    u8g2.clearBuffer();
  u8g2.drawXBM(0, 5, 50, 50, eth);
  u8g2.setFont(u8g2_font_profont22_mn);
  u8g2.drawStr(x, y, String(eth_usd).c_str());
  u8g2.sendBuffer();
  delay(2000);

     u8g2.clearBuffer();
  u8g2.drawXBM(0, 5, 50, 50, doge);
  u8g2.setFont(u8g2_font_profont22_mn);
  u8g2.drawStr(x, y, String(eth_usd).c_str());
  u8g2.sendBuffer();
  delay(2000);
}
```

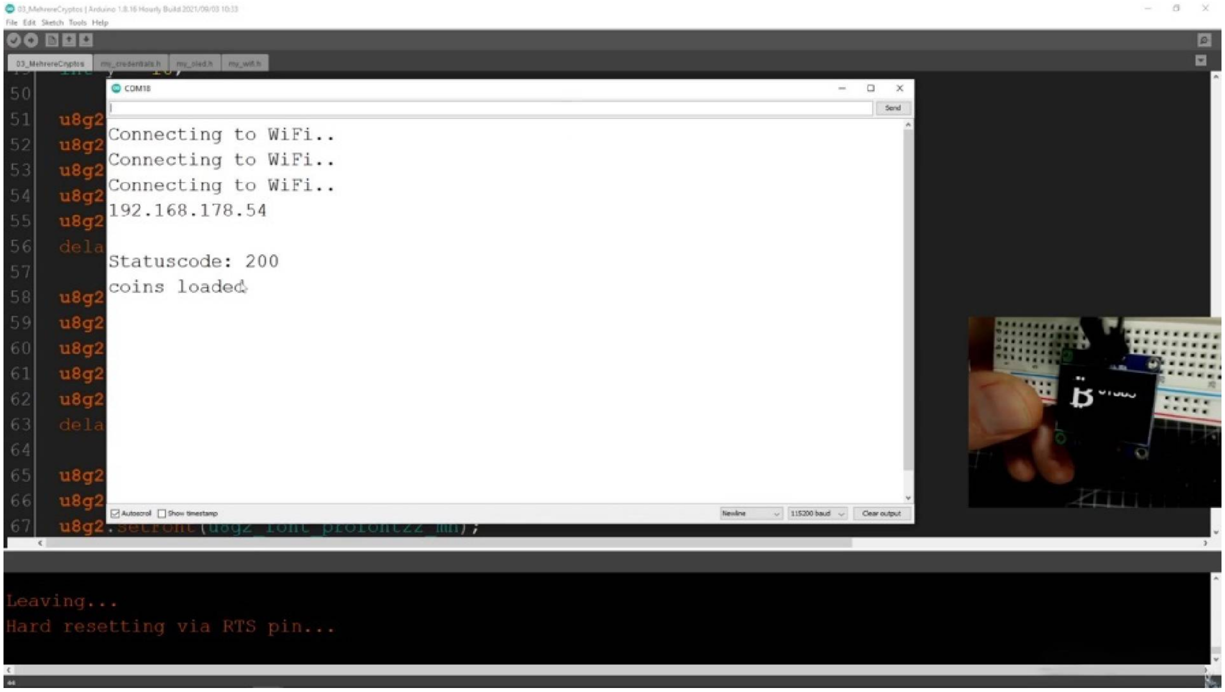
That means two seconds out of a match. Just give a look. If we have compiled everything right? And then, as I said, for testing purposes, the next one item, I've already converted a picture from Ethereum. And then I sitting in each, huh? And the third one was two seconds. DeLay Dutch. If I called it Dutch. Let's see. Kate Snow, BTC, Dutch. OK, so and then also centered around making Italy, what we've done here is just visualizing the data, the information we're making. We don't make any new requests. So this is just visualization. We are fetching the data in the first set up and then we are only making this stuff.


03_MehrereCryptos | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
03_MehrereCryptos
my_credentials.h
my_oled.h
my_wifi.h
Uploading...

```
  u8g2.clearBuffer();
  u8g2.drawXBM(0, 5, 50, 50, btc);
  u8g2.setFont(u8g2_font_profont22_mn);
  u8g2.drawStr(x, y, String(btc_usd).c_str());
  u8g2.sendBuffer();
  delay(2000);

  u8g2.clearBuffer();
  u8g2.drawXBM(0, 5, 50, 50, eth);
  u8g2.setFont(u8g2_font_profont22_mn);
  u8g2.drawStr(x, y, String(eth_usd).c_str());
  u8g2.sendBuffer();
  delay(2000);

  u8g2.clearBuffer();
  u8g2.drawXBM(0, 5, 50, 50, doge);
  u8g2.setFont(u8g2_font_profont22_mn);
```

```
Writing at 0x0001c000... (12 %)
Writing at 0x00020000... (15 %)
Writing at 0x00024000... (18 %)
```

OK? Composes the compiler says no errors. Let us upload the file.

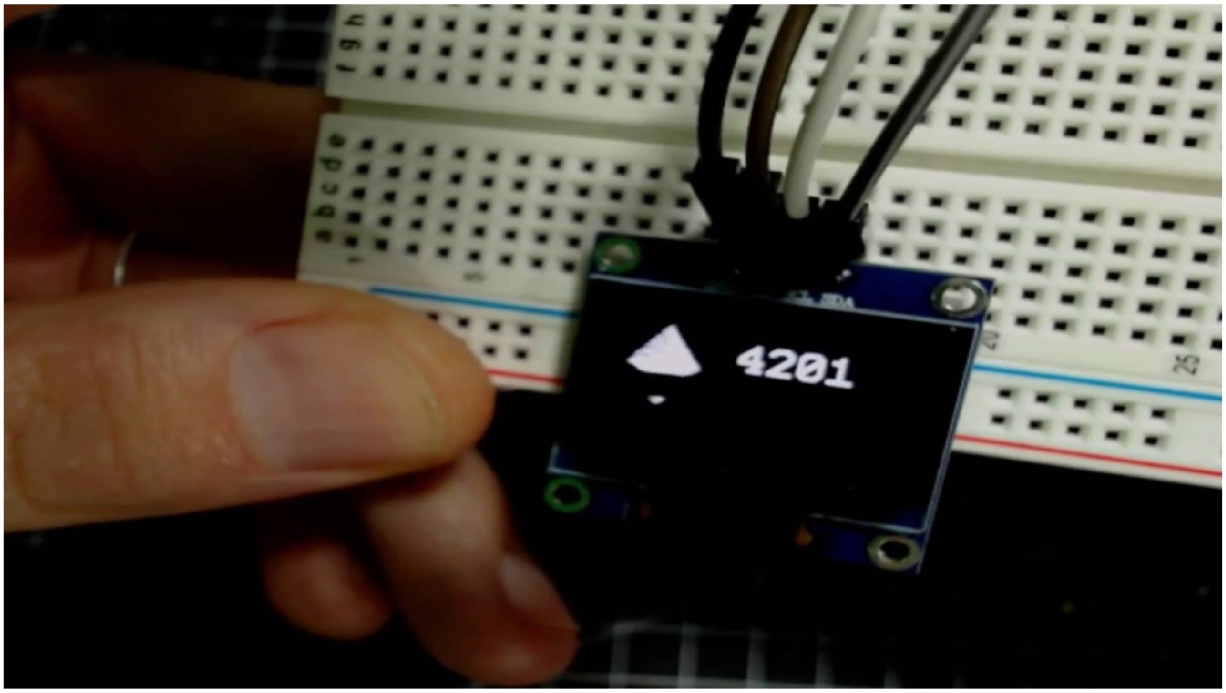


```
Connecting to WiFi..
Connecting to WiFi..
Connecting to WiFi..
192.168.178.54

Statuscode: 200
coins loaded
```

```
Leaving...
Hard resetting via RTS pin...
```

Turning on the other camera, and it's also interesting, though, is this ceremony time switching on the ceremony time then connecting to WiFi?

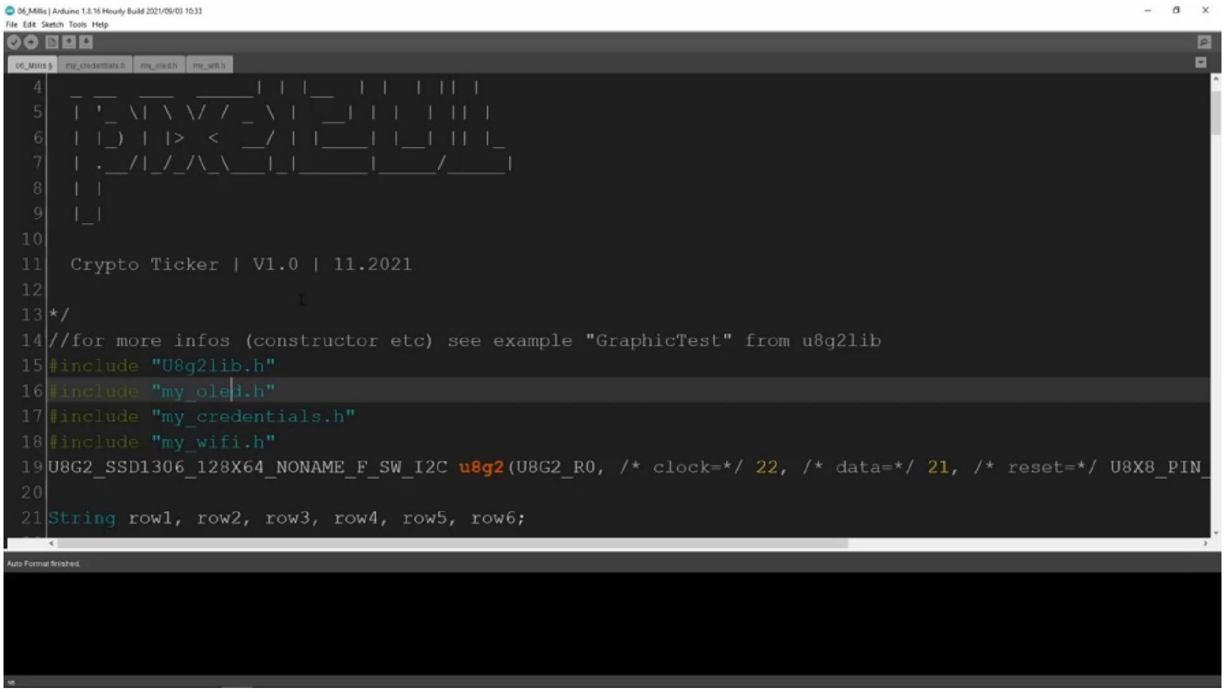
Coins are loaded. And now we are seeing here, Ethereum. Bitcoin. Dogecoin. Very nice.



So in our loop, we are displaying no. And XM and be fine and also and string this the current oh, there are those colonies with the price of Ethereum. Here's the mistake Dogecoin. And then we also have an incorrect dot com price because it should somewhere, you know, 20 cents, I think.

RELOAD DATA

I'm sure that you're familiar with the concept of blinking without delays. So in a general speaking, the delays are not really good for your program's structure in court because when you see delays, the whole program stands still until the delay time is over.

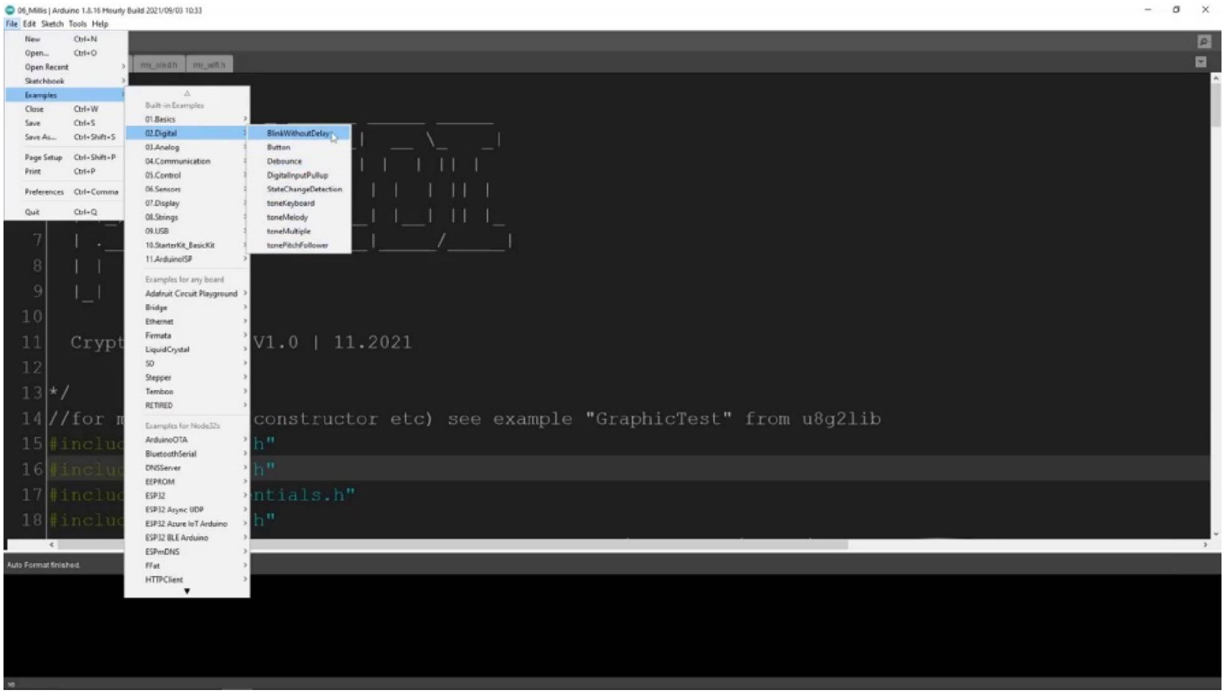


06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
06_Millis §
my_credentials.h
my_oled.h
my_wifi.h

```
  _ __  ___  _____| | |__  | |  | || |
 | '_ \| \ \/ / _ \ |  __| | |  | || |
 | |_) | |>  <  __/ | |____| |__| || |_
 | .__/|_/_/\_\___|_|______|_____/_____|
 | |
 |_|

 Crypto Ticker | V1.0 | 11.2021

*/
//for more infos (constructor etc) see example "GraphicTest" from u8g2lib
#include "U8g2lib.h"
#include "my_oled.h"
#include "my_credentials.h"
#include "my_wifi.h"
U8G2_SSD1306_128X64_NONAME_F_SW_I2C u8g2(U8G2_R0, /* clock=*/ 22, /* data=*/ 21, /* reset=*/ U8X8_PIN_

String row1, row2, row3, row4, row5, row6;
```

Auto Format finished.

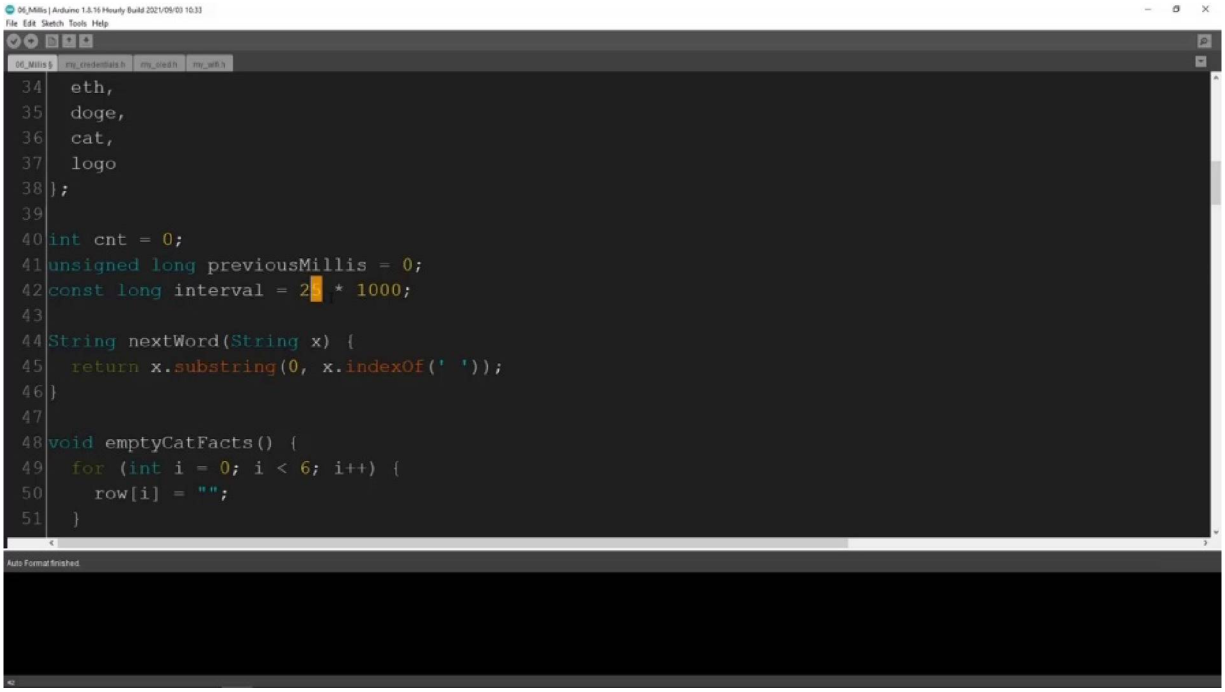
That means you input, for example, a five delay of five seconds of delay. The program standstill one second to three four five and then execute to the rest. When we are using the concept of blink without delays when you're working with families. The loop is running blue, blue, blue, blue and each loop is checked. If the condition millis are rated in the interval and when the condition is true, then it executes to the code and afterwards it can also execute auto parts of the court.



```
  Crypt                V1.0 | 11.2021

*/
//for m                constructor etc) see example "GraphicTest" from u8g2lib
#incluc                h"
#incluc                h"
#incluc                ntials.h"
#incluc                h"
```

Auto Format finished.

So when you are new to the concept of blink without malice, go to examples to digital and go to blink without delay. For example.

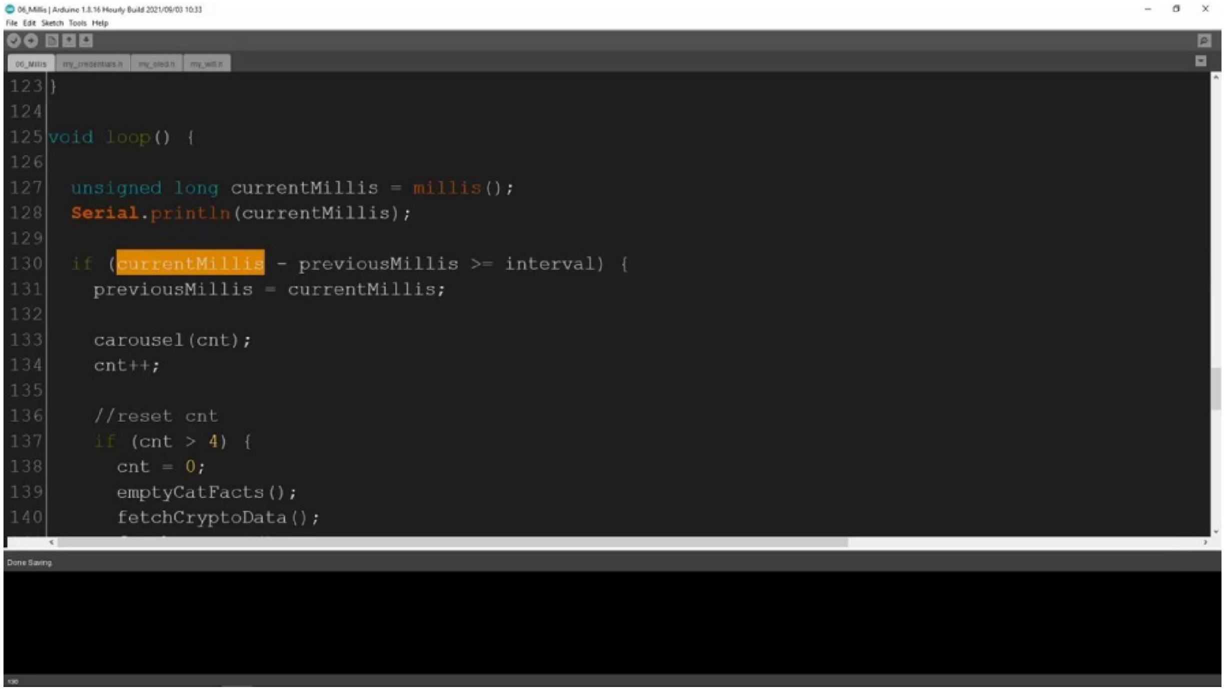

06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
06_Millis §
my_credentials.h
my_oled.h
my_wifi.h

```
  eth,
  doge,
  cat,
  logo
};

int cnt = 0;
unsigned long previousMillis = 0;
const long interval = 25 * 1000;

String nextWord(String x) {
  return x.substring(0, x.indexOf(' '));
}

void emptyCatFacts() {
  for (int i = 0; i < 6; i++) {
    row[i] = "";
  }
```

Auto Format finished.

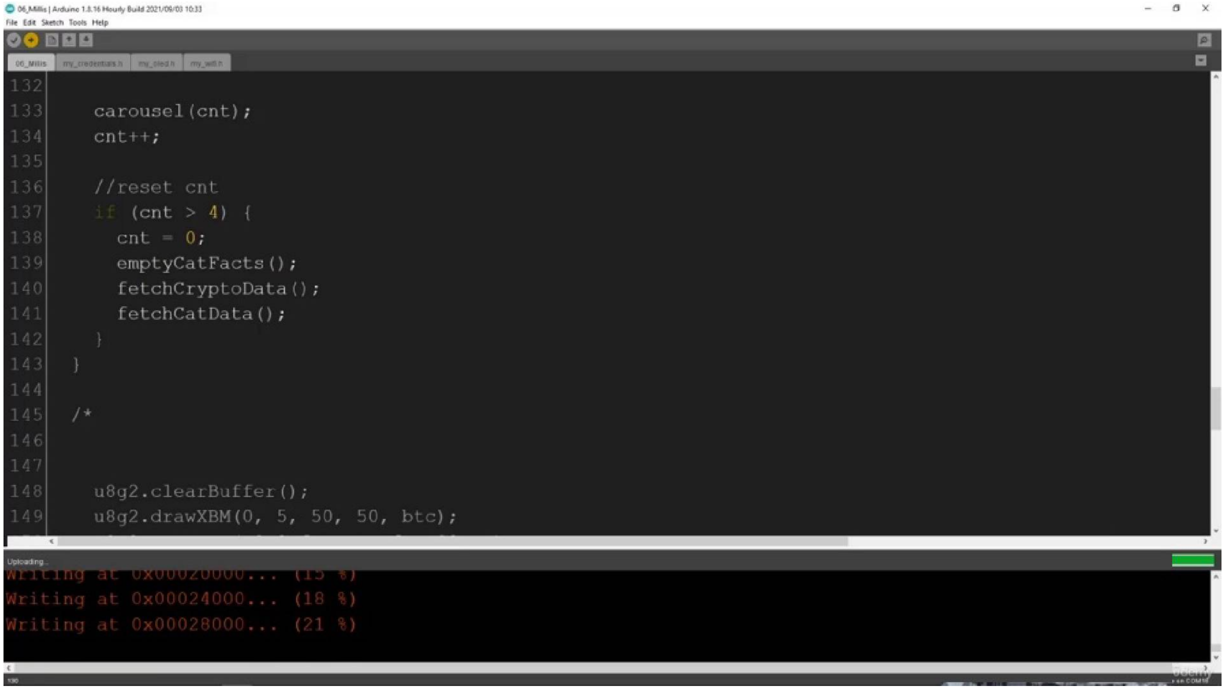
We are now implementing the blink without delay in our concepts, in our program codes, and therefore we need to declare eight new variables.

Previous models and the interview the previous minister has at the very first beginning zero and the interval is set to 25 seconds.


06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
06_Millis
my_credentials.h
my_oled.h
my_wifi.h
Done Saving.

```
}

void loop() {

  unsigned long currentMillis = millis();
  Serial.println(currentMillis);

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    carousel(cnt);
    cnt++;

    //reset cnt
    if (cnt > 4) {
      cnt = 0;
      emptyCatFacts();
      fetchCryptoData();
```

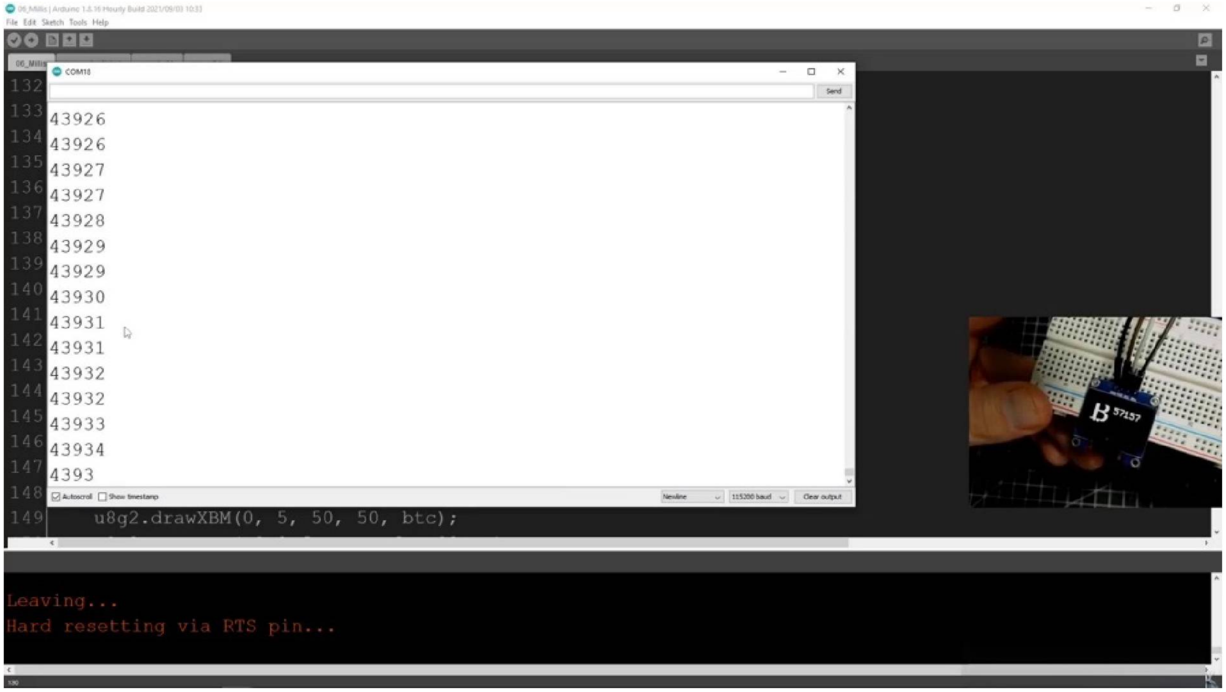
Then jumping through, jumping to the loop. Now we are getting rid of the. Candy. Because now we're going to using the new concepts. So what we're going to do is we are first declaring in the loop a certain area that is called current and. And this release is set to run time. So that means how long is the program running? So we are printing for a deeper purpose. The current release out of the ceremony time and now we are making an if statement. If the current release, minus two previous movies created, then the interview. Then. Make our main logic. So let us go through this code once more. We we're not finished, we have to set back the current the previous meetings, so. Tick, tock, tick. OK, we are starting to program the current malaise the previous Mills is set to zero at the interval sets to 25. We have the first, second to second, third, fourth, fifth, second. So for example, a fifth second five minus zero is not greater than 25. It's a false set of code is not executed. The loop goes foot on the gas foot. On the 10th 12th, the concert took a 13 seconds hit. When we are now into 25 seconds, twenty five point one minus sidra is greater than the interval set to 25 seco nds. Yes, it's true. It's now our main logical code would be executed, but the previous release is set to the current release. So the next loop we are now at the twenty seventh, second twenty seven minus 25, which we know on the set. Here are two seconds and two seconds are not. That then sets the two the twenty five seconds to the interval. We are thirty five seconds. Thirty five minus twenty five from the previous Milly's is not great to the interval. You see, this is a very simple way that we can. Yeah, making delays without delay statements, but we are working with the M.E. statement so.


06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
06_Millis
my_credentials.h
my_oled.h
my_wifi.h
Uploading...

```

    carousel(cnt);
    cnt++;

    //reset cnt
    if (cnt > 4) {
      cnt = 0;
      emptyCatFacts();
      fetchCryptoData();
      fetchCatData();
    }
  }

  /*


    u8g2.clearBuffer();
    u8g2.drawXBM(0, 5, 50, 50, btc);
```

```
Writing at 0x00020000... (15 %)
Writing at 0x00024000... (18 %)
Writing at 0x00028000... (21 %)
```

Let's compile it and check if we have everything written correctly. This looks good. We are uploading the new sketch to our Arduino. Then turn of the other camera, and so the display is exactly the same, only the logic behind will be a little bit different.



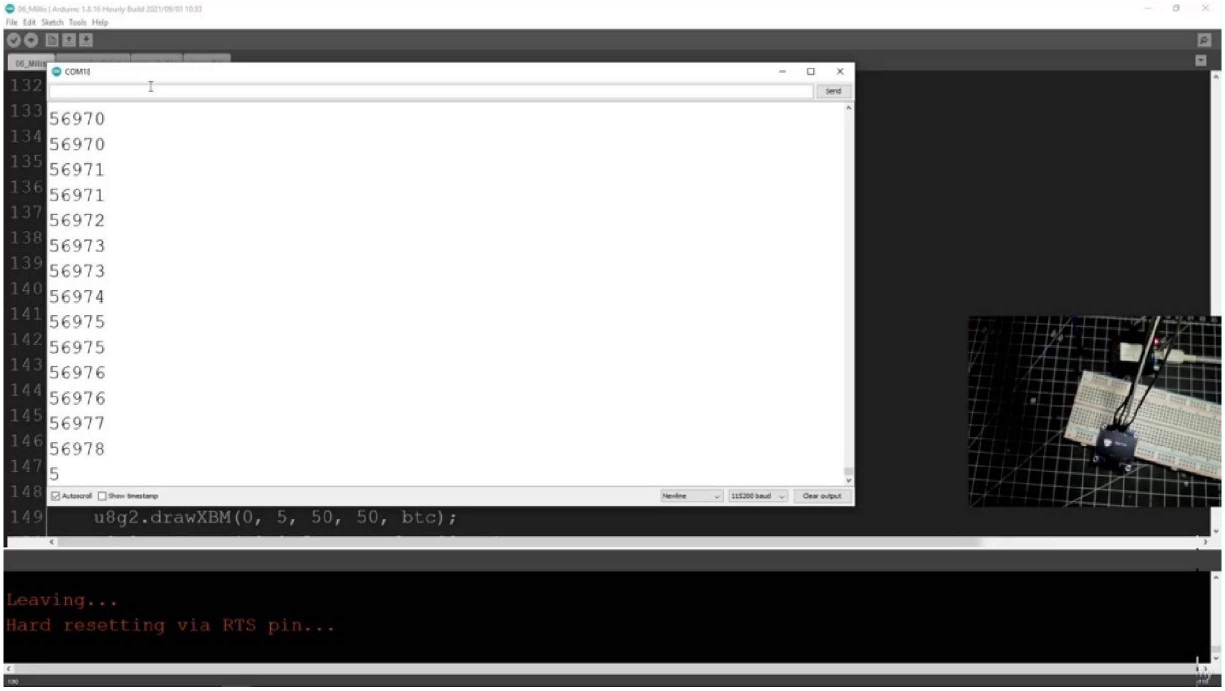
06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
06_Millis
COM10
Send
☑ Autoscroll ☐ Show timestamp
Newline
115200 baud
Clear output
57157

```
43926
43926
43927
43927
43928
43929
43929
43930
43931
43931
43932
43932
43933
43934
4393
```

```
    u8g2.drawXBM(0, 5, 50, 50, btc);
```

```
Leaving...
Hard resetting via RTS pin...
```

So we are switching on the super monitor. Now you can see connected to WiFi. Cranes are loaded, cats are loaded, and now the mills should be printed out, and at 25, you should see that we jump right into the bitcoin. We have 20 seconds. So every entry is one loop duck. No, we have done bitcoin. With fifty seven thousand one hundred fifty seven west us. And in the next 15 or 10 seconds, it should jump to the Ether Etherium.



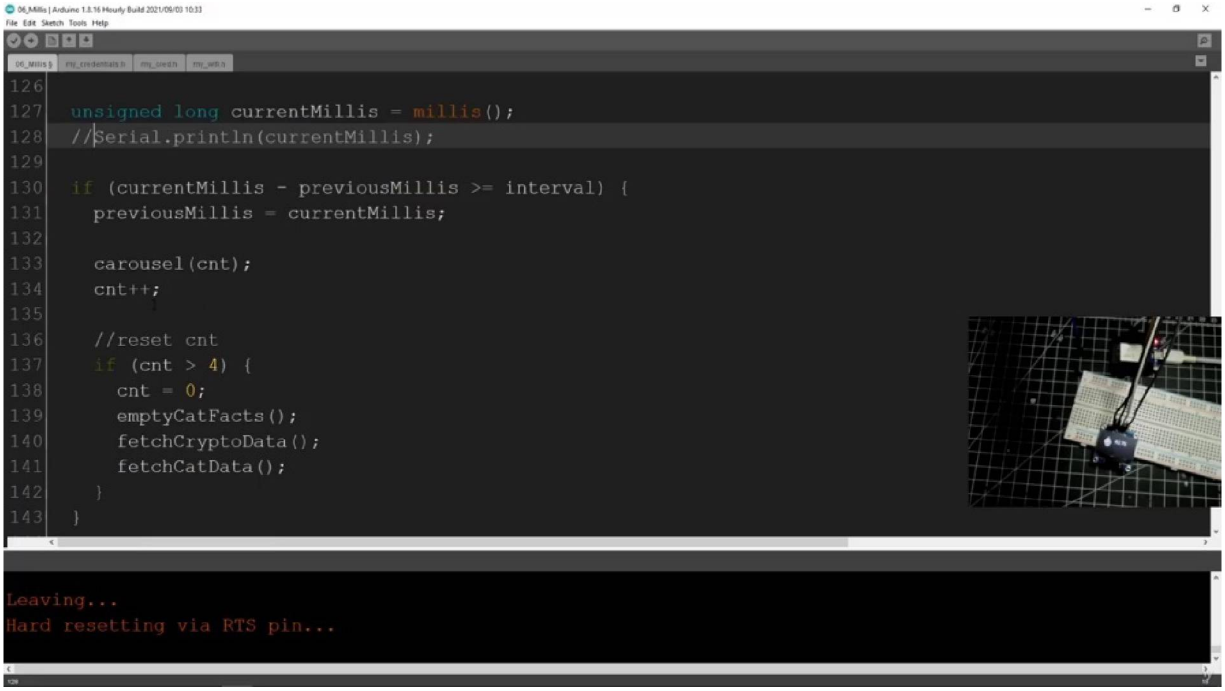
06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
06_Millis
COM10
Send
☑ Autoscroll ☐ Show timestamp
Newline
115200 baud
Clear output

```
56970
56970
56971
56971
56972
56973
56973
56974
56975
56975
56976
56976
56977
56978
5
```

```
    u8g2.drawXBM(0, 5, 50, 50, btc);
```

```
Leaving...
Hard resetting via RTS pin...
```

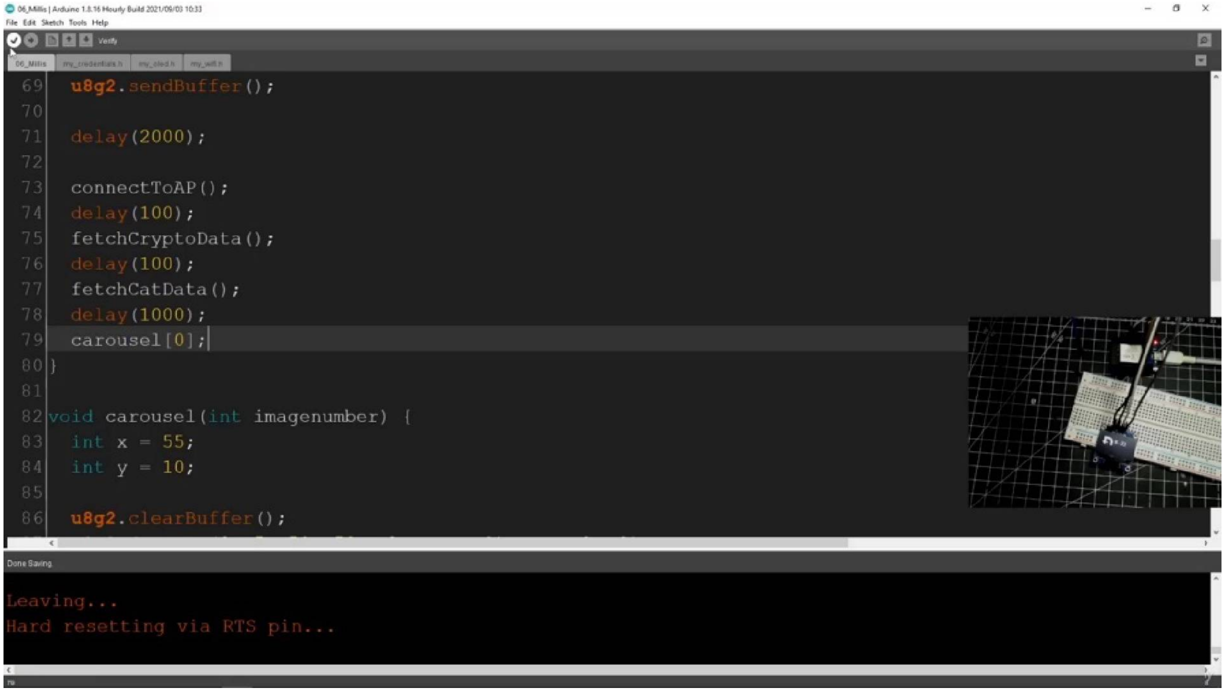
So. Forty 70, 80. 50 seconds and it switched to the next one. As you can see him, our main logic implementation works very well.


06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
06_Millis §
my_credentials.h
my_oled.h
my_wifi.h

```

  unsigned long currentMillis = millis();
  //Serial.println(currentMillis);

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    carousel(cnt);
    cnt++;

    //reset cnt
    if (cnt > 4) {
      cnt = 0;
      emptyCatFacts();
      fetchCryptoData();
      fetchCatData();
    }
  }
```

```
Leaving...
Hard resetting via RTS pin...
```

And now for further purpose, I would say we are I'm convinced this current mill is setting and also that we see at the very first beginnings


06_Millis | Arduino 1.8.16 Hourly Build 2021/09/03 10:33
File Edit Sketch Tools Help
Verify
06_Millis
my_credentials.h
my_oled.h
my_wifi.h

```
  u8g2.sendBuffer();

  delay(2000);

  connectToAP();
  delay(100);
  fetchCryptoData();
  delay(100);
  fetchCatData();
  delay(1000);
  carousel[0];
}

void carousel(int imagenumber) {
  int x = 55;
  int y = 10;

  u8g2.clearBuffer();
```

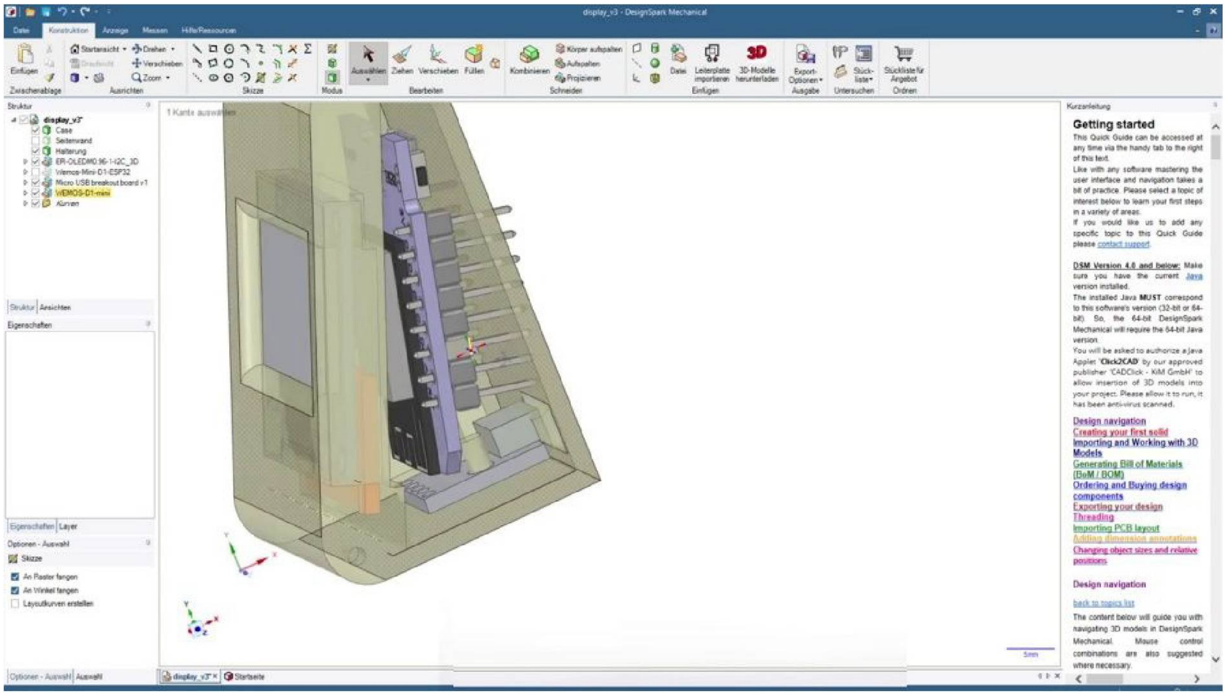
Done Saving

```
Leaving...
Hard resetting via RTS pin...
```

such data I would recommend to delay one thousand and sell. Zero. That we start with bitcoin.

SUGGESTION FOR A 3D PRINTED CASE

I would like to give you an idea and how you can implement or build your own 3-D case for the scripted Typekit project. So I've made a small case, and I will guide you through the main idea of this case. First of all, there is and proper fit for the all display. And you can see there is a little deepening. So that's the only display fits perfectly in, and there's also here one more parts to small plates, which fits perfectly into into this deepening. So that's the all it can't be fallout of the housing. There is also on some parts on the backside. I have found breakout boards and Michael was people. It's because it's I think it's it's it's mix and really pretty case when the USB port is in the middle, not just somewhere above or on underneath the topic on the case. And also to use certitude as you can see it.



Getting started

This Quick Guide can be accessed at any time via the handy tab to the right of this text.

Like with any software mastering the user interface and navigation takes a bit of practice. Please select a topic of interest below to learn your first steps in a variety of areas.

If you would like us to add any specific topic to this Quick Guide please contact support.

DSM Version 4.0 and below: Make sure you have the current Java version installed.

The installed Java **MUST** correspond to this software's version (32-bit or 64-bit). So, the 64-bit DesignSpark Mechanical will require the 64-bit Java version.

You will be asked to authorize a Java Applet '**Click2CAD**' by our approved publisher 'CADClick - KiM GmbH' to allow insertion of 3D models into your project. Please allow it to run, it has been anti-virus scanned.

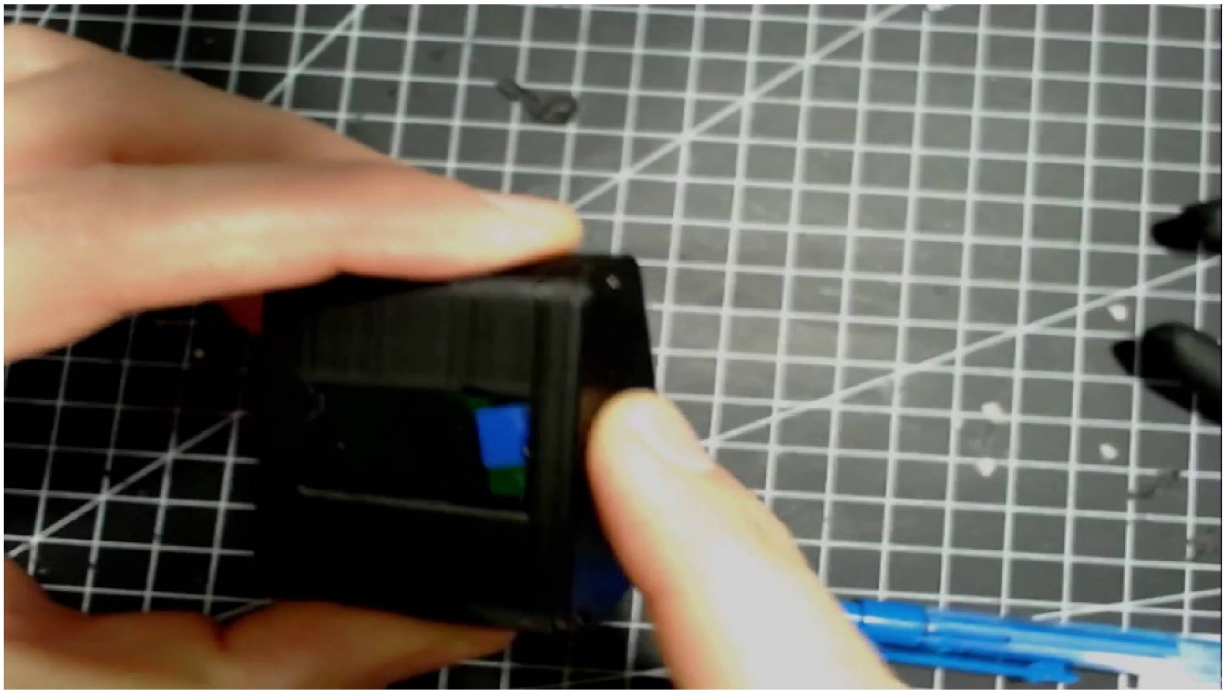
Design navigation
Creating your first solid
Importing and Working with 3D Models
Generating Bill of Materials (BoM / BOM)
Ordering and Buying design components
Exporting your design
Threading
Importing PCB layout
Adding dimension annotations
Changing object sizes and relative positions

Design navigation

back to topics list

The content below will guide you with navigating 3D models in DesignSpark Mechanical. Mouse control combinations are also suggested where necessary.

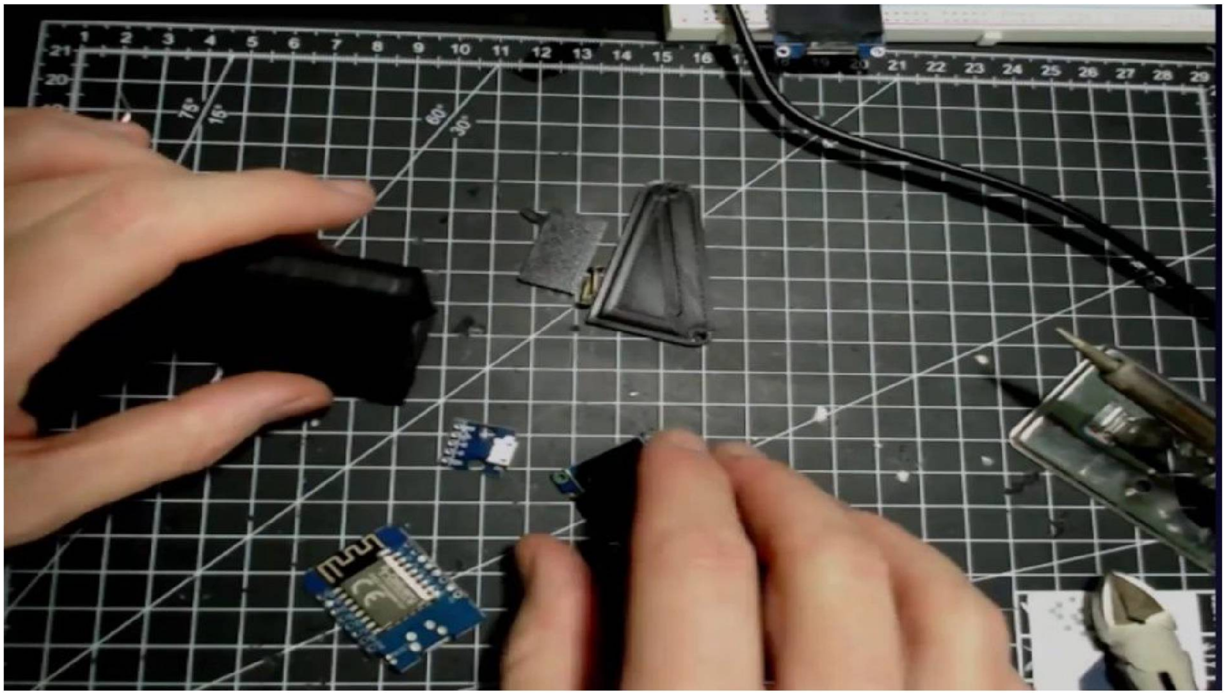
And also the USP 80 to sixty six has enough room in this case. Of course, this shouldn't be soldered in the final product, but you get the idea. So let's take a closer look on the finished product.



So here is to all display displaying the site wall and you can see it's. Perfectly fits into there is a small plates. And I get it out. Which holds the display. Also, you can see the micro USB port. The breakout port and decide where it's a little uplifting pilot so that it's it's really it perfectly fits on to the case and you can screw the sidewall with these tiny screws with two centimeters in length. Also perfectly fits into these tiny holes. And then you have a good looking case which looks very nicely in small, perfectly to your desk. So that's a short summary on how I would give you an idea how to. You can implement, replicate and switch 3D printed case for your trip to tinker projects.

ASSEMBLY

So this is one of the last means of design project, and now I'm soldiering altogether.



If she departs from the case and also the components, so I'm going from the Micro Use Be Breakup Board, from the music scene to the arena, on the grounds to the older, you know, and from the Arduino, do you want me? I am connecting the four pings, as we did before to the whole display. Then I'm all together on the side. All are connected to the main case and then we have a finished product.

So, yeah, let's do camera one for a while so that you get a good overview of how I'm sitting all together and then I'm looking forward to seeing maybe in another class.

THE END