Chapter

Smart Home Monitoring System Using ESP32 Microcontrollers

Marek Babiuch and Jiri Postulka

Abstract

This chapter deals with the implementation of our own monitoring system with home security. The system is designed using IoT modules and uses ESP32 microcontrollers. The chapter describes the design of the system, its hardware components, software implementation, security solutions, communication, the collecting and monitoring of processed data, as well as the quantification of costs for the production and deployment of this system. The proposed system secures a house by detecting an intruder in the building, triggering an alarm and capturing it all with camera images, and then sending data to the owner's smart mobile phone. The secondary task of the system is to collect data from sensors for monitoring the temperature of an object and presenting it via a web server.

Keywords: ESP32, microcontroller, MQTT, IoT, monitoring systems

1. Introduction

The main idea was to use the ESP32 chip, which we have installed with a camera module to monitor a house, as well as to monitor the temperature of individual rooms such as the corridor and boiler room. The proposed system contains PIR sensors and temperature sensors in the monitored rooms and connected camera modules to the ESP32 microcontroller. The data collected from the sensors is sent wirelessly to the control unit, which is the Raspberry PI Zero. The other created HW modules are an input panel with a touch screen and input panel with an ESP32 Wroom board and a membrane keyboard. It is used to unlock and lock the house to activate the motion sensor and cameras. The control unit is extended using the GSM module IoT-GA5-B. It ensures the sending of messages to the mobile phone of the homeowner, which inform him/her about the security status of the house. Other HW modules are touch screens showing the current status and temperature in the rooms and control LEDs, recording the status of the entire system. The main aspects of the whole system are the following:

- Motion detection using PIR sensors
- The capturing of a camera image
- The monitoring of the physical quantities of the household such as temperature, humidity, the possibility of extension to other monitored physical quantities

- The storage and monitoring of measured data
- Access to data via a web server
- Responsive applications for mobile devices
- Wireless communication, MQTT communication, System security

GSM communication with the system (at the request of the owner)

In the individual chapters of this article, we will gradually describe the selection of hardware components, the software implementation of the entire system, the design of system security, the installation and testing of the monitoring system and the financial aspects of implementation with the possibility of expansion.

2. Background and related works

Shortly after its introduction, the ESP32 microcontroller became fully integrated into industrial automation, mainly into the deployment of embedded systems and various IoT tasks. Its great advantage is undoubtedly its price, circuit structure, the possibility of connecting peripherals and IoT modules and other sensors, as well as having excellent support for creating applications. The ESP32 chip is well implemented as a web server, using wireless Wi-Fi communication, Bluetooth and mostly the MQTT communication standard at the level of exchanging messages with the surroundings. It often works with another suitable microcomputer such as the Raspberry Pi. In [1] we have described in detail the creation of an architecture for an embedded system based on the ESP32 chip and the processing of a monitoring task using Dual-core for its operation, while one core takes care of obtaining and processing data from the sensors, the other core solves communication issues with the surrounding devices. The suitability of using multiple cores is also discussed [2] in the field of deployment of ESP32 in the field of machine learning and neural networks. Currently, ESP32 is implemented in a wide range of IoT industries and applications in the areas are listed in this chapter with a specific example of use.

The ESP32 chip is suitable for implementation in applications of various monitoring and security tasks. The common element of the implemented applications is its high-quality, low-cost solution in multiple areas, such as the Solar Water Pumping System in agriculture [3] or various types of monitoring systems in farming [4, 5]. Other suitable deployments can also be found in the field of monitoring air quality systems [6], monitoring LPG leakage [7] or waste management systems [8]. Article [9] describes the use of ESP32 as a web server for a real-time photovoltaic monitoring system. With the help of other suitable peripherals such as a camera, linear actuators and drivers, we can use ESP32 to implement a position control system [10] or a Smart surveillance system [11]. The ESP32 microcontroller is also used to perform Device-Free Passive (DfP) tasks such as detection, localization and the tracking of human entities [12], or utilized using GPS as an indoor positioning system [13] or security system [14]. SCADA systems have also begun to focus on IoT in recent years [15]. SCADA architectures have evolved over the years from monolithic (stand-alone) through distributed and networked architectures to the latest Internet of Things (IoT) architecture. Article [16] presents the design and implementation of the Open Source SCADA system by using a local server IoT platform as Master Terminal Units for handling data processing and human-machine interactions and an ESP32 micro-controller as the Remote Terminal Unit for receiving, processing and sending the remote data from the field instrumentation devices.

Article [17] describes the design of a Smart home IoT system and notes the trade-off between scalability, security and data collection efficiency for the Internet-of-Things sensor networks. The article [18] discusses security risks in detail and states that IoT systems carry risks in terms of security and privacy. Without eliminating these risks, the IoT requirements are not adequately fulfilled. The article describes the vulnerability of the IoT system at individual layers of the network system and proposes a solution in the form of a security model. Article [19] presents an environment monitoring system based on ESP32 together with a wireless sensor network. The result is not only its low cost but also its low power system, which is today a sought-after benefit when deploying a monitoring system. An integral part of IoT today is the area of iHealth and health monitoring using smart embedded systems. This area is evidenced by the use of ESP32 for monitoring heart rate [20], an image recognition system for blind people in article [21], or a smart saline monitoring system in intravenous therapy [22]. Article [23] describes the possibilities of implementing a control system with the help of human gestures on wearable devices. ESP32 chips are implemented not only in iHealth monitoring systems, but now smart systems are also being developed for the comfort and convenience of mental health and well-being. The article [24] documents a case study of the creation of an intelligent lighting system according to the detection of human emotions. It uses, among other things, necessary hardware components, the predecessor of the ESP32 chip, namely ESP8266, and the already mentioned MQTT communication standard, which is the most common communication standard. It is possible to use the encrypted MQTTs protocol with SSL/TLS certificates to secure MQTT communication, as described [25]. Other security features of communication such as the Algorithm of Advanced Encryption Standard (AES), implemented on ESP32 with a LoRa module to secure wireless communication are described in [26]. Lora communication techniques and the used Zigbee communication standard in the field of home automation are dealt with in [27, 28]. IoT architecture, in general, is described with its requirements and paradigms in [29] and the integration of IoT and cloud computing in terms of the configuration of embedded systems is described in a research article [30]. In the field of embedded systems and monitoring, ESP32 development variants with touch screens can be used, either integrated into the ESP32 Wrover board or connected externally [31]. A comparative analysis of ESP32 and other modules with the ESP32 recommendation for the IoT area is discussed [32]. We agree with his conclusions, as the authors express the idea that the microcontroller operating system FreeRTOS is open source software providing great support for real-time applications. Thus, it is expected that ESP32 will play a major role in the design of future IoT systems and embedded projects. The performance evaluation mentioned in the article [33] - Enabling ESP32-based IoT Applications in Building Automation Systems confirms the suitability of using ESP32 in low-cost applications with an industrial-grade performance. The results of this article are essential for developing cheap but nevertheless reliable industrial solutions based on SoC.

3. Hardware modules

The system will therefore include the Raspberry Pi as a control unit. The set will also include three camera modules equipped with a PIR and a temperature sensor. The ESP32 equipped with a display and a membrane keypad for entering the boiler room and a touch screen also controlled by the ESP32 at the main entrance will be used to lock the house. The set also includes one ESP32 microprocessor equipped with two temperature sensors, the same as the camera module, so we will not describe this device in the next chapter. Some basic components are shown in **Figure 1**.

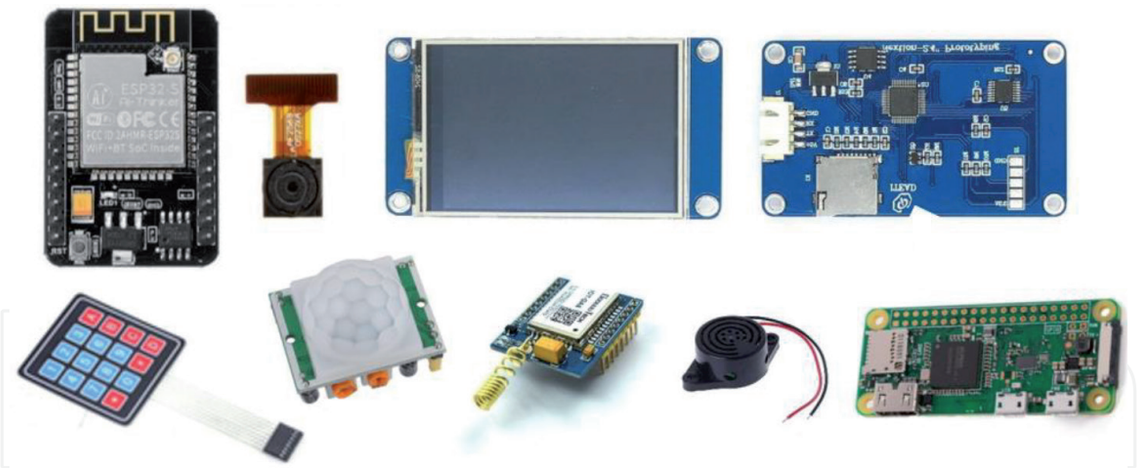

Figure 1.
Some hardware components.

3.1 Input panel with membrane keyboard

The system contains the following elements:

- ESP-WROOM-32

ESP-WROOM-32 belongs among Microcontroller units (MCU), which are fundamentally computer board platforms with CPU, memory, busses and built-in peripherals indispensable to read connected sensors or drive actuators [34].

- Monochrome character 16x2 (16 characters per line and 2 lines)

The LCD display is connected via a I2C bus. It is a serial bus that divides the connected device into a master or slave category. One wire is used to transmit the clock signal (SCL - synchronous clock) and is the data channel (SDA - synchronous data).

- Matrix membrane 4x4 keyboard for single-board computers

The keyboard contains characters (1 ÷ 9, A ÷ D and special characters # and *), and these are connected to 8 pins. The buttons in the individual rows and columns are always connected to a common wire. After pressing the button, the row and the column are always connected at a given point. The manufacturer guarantees a service life of 100 million presses.

- Alarm

The last component in this set is an alarm whose operating voltage is in the range of 3-24 V and intensity 95 dB. Its function is to trigger a loud tone in the event of an intruder entering the house, which should alert you that there is an intrusion in your home.

3.2 Input panel with touch screen

The main input will be the same ESP32 model as the previous set. As for the display unit, a 2.8” USART touch screen from NEXTION will be used. The most significant advantage of these displays is the graphic editor, in which you can quickly and without problems define the graphical and touch environment and perform simulations. This editor saves a lot of time when developing applications. The

resolution of this display is 320x240 together with 65 thousand colors, and adjustable brightness will make the perfect choice for all kinds of applications. The biggest advantage is the communication via USART, through which the created graphical environment can be uploaded directly to the display, in which a 4 MB flash memory is integrated. The second variant is using a micro SD card, for which this display has a slot. The USART interface is also used to communicate with the microprocessor, through which it receives variables and sends a click response to the element. Any other display can be used to implement the system. Our goal was not to reduce the price of components to the lowest level but to develop a monitoring system in a comfortable, user-friendly environment, so we chose this display [35]. Custom-made input panels and boards with the help of a 3D printer are shown in **Figure 2**.

3.3 Camera modules

The composition contains the following elements:

- Ai-Thinker ESP32-CAM. Again, this is a Wi-Fi + BT MCU module, expanded by 520 KB SRAM, including external 4MPSRAM. It also supports OV2640 and OV7670 cameras and SD cards. ESP32-CAM can be widely used in various IoT applications. It is suitable for home intelligent devices, industrial wireless control, wireless monitoring and other IoT applications.

- RGB LED module. It is a type of RGB LED with a common anode, i.e. the individual ledges are switched by connecting to the ground. Its function will be to give information about the current status of the camera module (whether it is connected to a Wi-Fi network, or whether there was an error loading the camera or other error messages) indications in different colors.

- Temperature sensor DALLAS DS18B20, This sensor allows you to measure the temperature in a range of −55 to +125 degrees Celsius, while in the range of −10 to +85 degrees Celsius it has a guaranteed accuracy of ±0.5° C. It is available in a TO-92 package, which is similar in size to ordinary transistors, it is also available in a waterproof variant, where the sensor is sealed in a stainless steel stick. OneWire bus is used for communication, which uses only one communication pin. This sensor also supports the so-called parasitic mode, where only two wires are needed to connect the sensor to the microcontroller.

- PIR module: There are many variants of motion sensors designed for applications similar to this one. Again, we chose from two variants. The first of them

Figure 2.
Input board in boiler room.

Figure 3.
ESP32 camera modules in the corridors and living room.

is called HC - SR501, whose supply voltage is in the range of 4.5 to 20 V and the output logic is 0 / 3.3 V. The size of the sensor is 32x24mm, so it is a larger sensor but allows you to set the sensitivity of the sensor and timing using two potentiometers. Also, its detection distance is more than sufficient; the manufacturer states a distance of up to 7 meters with a sensing angle of 120°. The second variant is a smaller module called AM312. The operating voltage is in the range of 2.7 to 12 V, while the output logic is the same as for the previous module. The dimensions of the plate are only 10x8mm. However, the detection distance is also smaller, approximately in the range of 3–5 m at a scanning angle of 110°. **Figure 3** shows the mounted camera modules in the individual rooms.

3.4 Control unit

Even though the Raspberry Pi is already running with the fourth generation of the development board, the economical Pi Zero variant is sufficient for our system. Also, Zero W has built-in Wi-Fi and Bluetooth. Zero is built with a single-core ARMv6 processor clocked at 1 GHz. It also has 512 MB of RAM, audiovisual output via Mini-HDMI, a classic micro-USB 2.0 connector and power supply via micro-USB. It also has a 40-pin GPIO interface. The control unit is then further expanded by a GSM module called IOT-GA6-B. It is an ideal solution for IoT devices that communicate via a serial line at the TTL level, and it supports voice calls, SMS, GPRS data transfer and standard AT commands. Of course, it would be possible to connect this module to the ESP32 microprocessor. Since it would be necessary to forward all messages from the control unit to the input panel and then to the terminal, this implementation would still be too complicated. For this reason, we decided to omit one intermediary and connect the device directly to the control unit. The architecture of our IoT system is therefore centralized, for our smart system, the central point is the control unit, which is also an MQTT broker that communicates with ESP32 modules and input panels. The conditions under which it would be appropriate to decentralize the IoT system describes [36], among other things, a detailed comparison of the decentralized IoT architecture approach with different approaches.

4. Application software

This chapter describes the individual services that have been used in this system. There are also descriptions of source code fragments that are uploaded to individual

devices. Thanks to this outline, it is possible to understand better the way the whole assembly works. **Figure 4** shows the resulting software application, which is described in this subchapter.

The operating system is the latest version of Raspbian called Buster and runs on Debian Linux version 4.16. It is an operating system designed directly by the manufacturer of this board, thanks to which the reliability and stability of this system should be guaranteed.

The Raspberry Pi runs:

- MQTT Server - Service for communication with microcontrollers
- Apache 2 - Web server
- MariaDB - SQL database for measured values.
- Python Scripts - Covers all logic in communication with microcontrollers, as well as storing the data in a database.

4.1 Communication

All communication takes place wirelessly via Wi-Fi. The protocol by which individual devices communicate is MQTT. MQTT (formerly: Message Queuing Telemetry Transport, today MQ Telemetry Transport) is a simple and undemanding protocol for transmitting messages between clients via a central point - a broker. Thanks to this simplicity, it is easy to implement it even in embedded devices and it is spread relatively quickly. For the MQTT protocol, the transmission is performed using TCP and uses the publisher-subscriber design pattern. So there is one central point (the MQTT broker) that takes care of exchanging messages. In our case, the broker will be Raspberry Pi. **Figure 5** shows a block diagram of MQTT Communication. Messages are sorted into so-called topics, and the device either publishes in the given topic (publish), i.e. it sends data to a broker, which stores and distributes them to other devices, or is subscribed to a topic or more topics (subscribe). The broker then sends all messages with the given topic to the device. Of course, one device can be a publisher in some topics and at the same time a subscriber in others.

4.2 Input panel

Its purpose will be to lock a house by pressing a key, as well as to unlock it by entering a 4-digit numeric code. At the same time, it serves as an element to alert the intruder via an alarm. In the source code, we have programmed macros that allow you to define how many times it is possible to enter a pin when the house is locked, the time delay of attempts, the default display of data on the screen, reading keystrokes and communicating using the MQTT protocol. Furthermore, in the source code, which we will not mention here, we perform Wi-Fi configurations and check whether a connection to Wi-Fi has taken place, followed by repeated attempts, LED signaling and the reconfiguration of the device. Another functionality programmed in the input panel is the function for setting the MQTT client, which defines the address of the broker, the communication port and instructions with the settings for receiving messages from the broker. The main loop of the program first calls a function that verifies if the ESP is still connected to the Wi-Fi network, and if not, it tries to connect it several times. If this still does not work, the device will restart. Subsequently, a function is called that verifies the connection

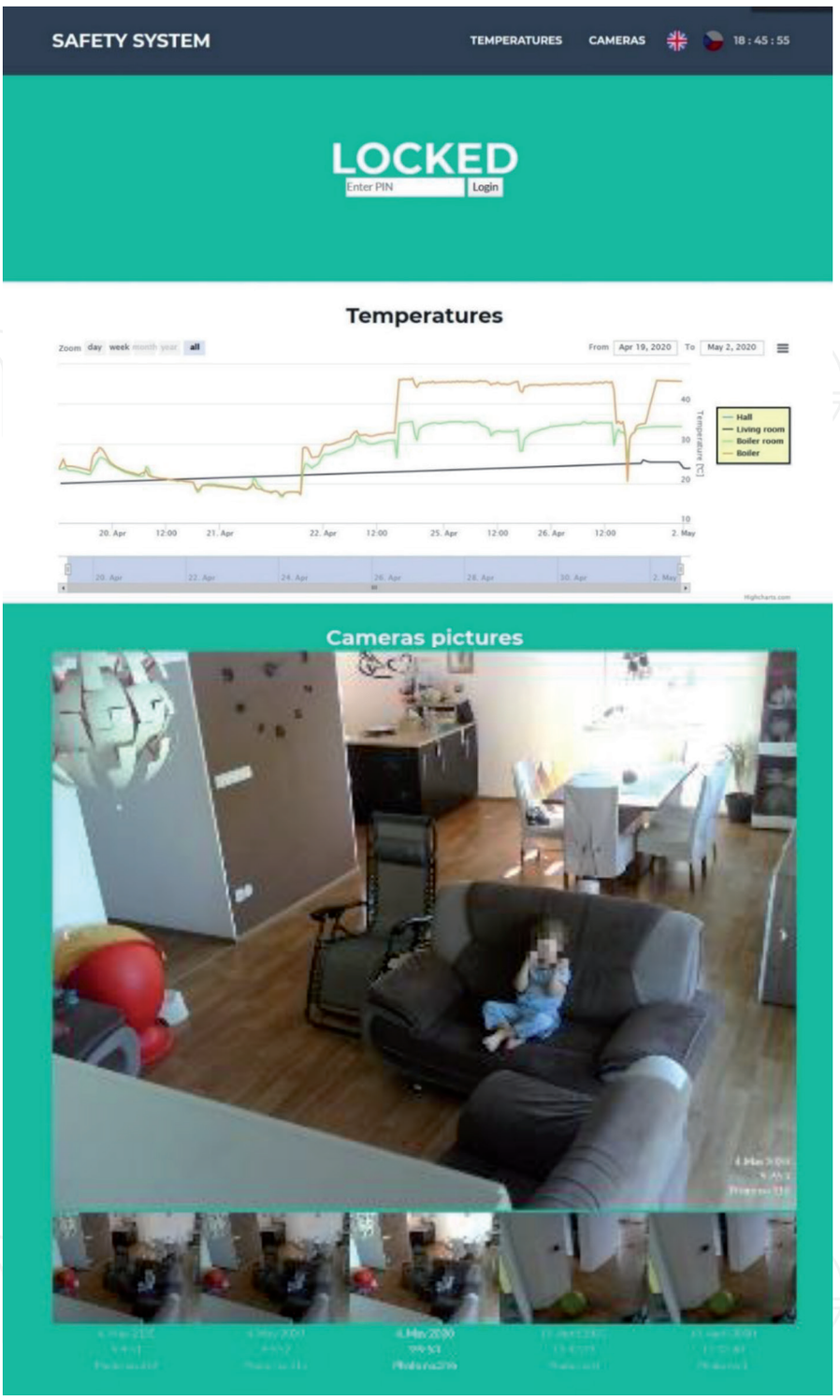

Figure 4.
Measuring and monitoring application.

Figure 5.
MQTT communication.

to the MQTT broker and, if connected, reads incoming messages from the broker. Incoming messages include the status of whether the house is locked, in which case this fact is shown on the display, and if the locked state detects an intruder from one of the camera modules, then an alarm is triggered in the form of a passive speaker, which emits a constant tone of a specific frequency. Other functions that needed to be solved included verifying key presses at a certain time interval, resolving the display backlight when entering data, verifying an incorrectly entered Pin code and then blocking further attempts for a certain time interval, as well as sending a message when trying to crack a password. The module then further solves the verification logic of the house locking status and the communication with the control unit. The PIN code is stored in a database located on the control unit. The control unit transmits the house status and the current PIN code via secure communication.

4.3 Camera module

Its primary goal is to record movement in a specific area if the module has been activated in advance. Motion is recorded using a PIR sensor, and then photos are captured at one-second intervals while the PIR sensor detects movement. The secondary goal is to monitor the temperature and send this data to the control unit. The source code in **Figure 6** contains macros that allow you to configure how long the delay between each captured image is, as well as the delay between recording the temperature. Other macros are used to define the pins to which the PIR module and the temperature sensor are connected, as well as the accuracy with which the value on the sensor is measured. Furthermore, RGB LED cathodes are defined here. In terms of software, this module solves a total of 6 libraries, which ensure the sending of the captured image of the server, the loading of the camera, connection to the Wi-Fi network, communication using the MQTT protocol and connection and work with the temperature sensor.

The software of this module implements functions for verifying the connection to the Wi-Fi network and checking the connection to the MQTT broker. This code fragment is shown in **Figure 7**. Other functions are checking the measuring

```
//<><><><><><><><><><><><><><><><><><><><><><><><><><><><>//
//===============LIBRARIES=====================//
//<><><><><><><><><><><><><><><><><><><><><><><><><><><><>//
#include "esp_http_client.h" // http client
#include "esp_camera.h"  // configuration and camera handling
#include <WiFi.h> // Wi-Fi network connection header
#include <PubSubClient.h> // MQTT broker connection
#include <OneWire.h>   // temperature sensor connection
#include <DallasTemperature.h> // temperature reading

#define capture_interval 1000 // microsec. between photos capture (1 sec)
#define temp_interval 300000 // microsec. between temperature measuring (5min)

#define PIR 15 // PIR pin
#define ONE_WIRE_BUS 14 // Temperature sensor pin
#define TEMPERATURE_PRECISION 12 // Measurement precision
#define LedR 4 // R RGB LED
#define LedG 2 // G RGB LED
#define LedB 13 // B RGB LED
/*
 * yellow                - lost Wi-Fi communication
 * violet                - lost Mqtt broker communication
 * red                   - camera module error
 * green blinking        - I'm starting
 */
```

Figure 6.
Including the libraries and macro definitions.

```
void loop()
{
  WifiCheck();

  mqttRun();

  temp_current_millis = millis();
  if (temp_current_millis - temp_last_capture_millis > temp_interval) // If the time
      // for the next temperature measurement has elapsed
  {
    temp_last_capture_millis = millis();
    Temperature();
  }

  if (locked) // If house is locked
  {
    bool PIRSENSOR = digitalRead(PIR);
    if (PIRSENSOR) // If movement is captured
    {
      if (!last_PIR)
      {
        Serial.println("Public Intruder");
        MQTTclient.publish(PublishIntruder, MQTTid, MQTTpubQos); // Publish message
      }
      current_millis = millis();
      if (current_millis - last_capture_millis > capture_interval)  // 1 sec elapsed
      {
        last_capture_millis = millis();
        take_send_photo(); // Také a photo and send snapshot
      }
    }
    else
    {
    last_PIR = PIRSENSOR;
    }
  }
}
```

Figure 7.
Fragment of the source code of the ESP32 camera module.

interval, processing the measured values and sending it to the broker. Another check finds out if the broker received the message about a locked house. If this happens, then the movement in the area is checked by reading the value from the PIR sensor. If the movement has been detected, a house intrusion report is sent to the broker. At the same time, it is checked whether the time required to take another screenshot has elapsed since the last loop. If so, a function is called to take a picture using the connected camera. This image is then sent and then saved in the directory.

4.4 Control unit

The purposes of this unit are as follows: To mediate communication using the MQTT protocol, to store incoming data in a database and to enable access to them, and also to provide an HMI interface. We use the MariaDB database to store data, which is a relational database that is community developed by the successor branch of MySQL.4 users with different rights to use the database. This is mainly to ensure that we secure access to information as much as possible. The primary user is the Administrator with rights related to the *esp_db* database, in which all tables are stored, telephone numbers, measured values, captured photographs and the individual states of closing and opening the house. Users are also WRITE, who can write into the individual TABLEs and users READ who can only read data.

The telephone numbers and names of the users to whom these telephone numbers belong are stored in the DB Phones table. The other two columns of this table indicate the rights of individual telephone numbers, one column indicates the right to unlock or lock the house via SMS, while the other column indicates the manipulation of the gateway. The last is LOGIN, which stores a 4-digit PIN to unlock the house.

As already mentioned, Raspberry runs a web server, which is mediated by the Apache 2 service. This service runs several PHP scripts supplemented by an HTML structure. JavaScript then takes care of the dynamics of the pages together with a jQuery. Everything is unified in design and formatted using a CSS style.

The application screen on the webserver is divided into three parts. The first is the option to log in by entering the PIN code. After a successful login, the house can be unlocked or locked remotely. We see this directly in the application on the mobile device in **Figure 8**.

In the next section, there is a *HighCharts* graph, which clearly shows the course of temperatures over time measured on individual devices. In the chart, it is possible to select a specific device from which we want to see the course of temperatures. It is also possible to choose which time range we want to display the temperature. There are also additional options in which graphs can be exported either to an image or, for example, to an Excel worksheet or similar files. In the last section called the Cameras picture, we will find all captured photos from all connected camera devices that are connected to the set. These images are sorted according to the date that is displayed for each photo and are assigned a serial number for greater clarity. The site can be switched according to the language preference to the local environment, if necessary. This site has also been customized for mobile devices such as smartphones. We assume that these sites will mainly use these devices.

4.5 Python scripts

These scripts mediate communication for the MQTT protocol, as well as for storing data in a database and communication with the GSM module. There are three python files on Raspberry, one of which acts as an MQTT client. It listens to everything that is sent to the broker and performs a specific action based on the incoming message. The second python script contains functions that allow access

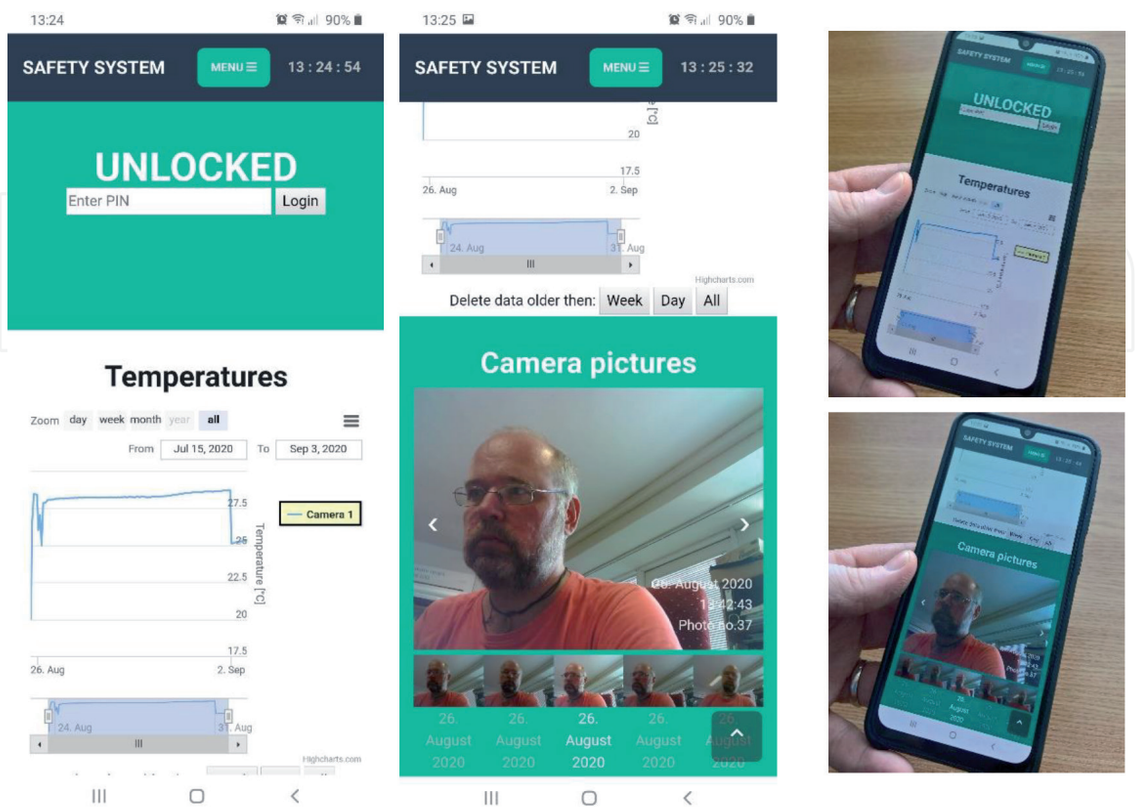

Figure 8.
Web server – Application on a mobile device.

to the database. The third communicates via a serial line with the GSM module and mediates communication with the end-user.

The Python script contains functions that are used to retrieve values from the database, such as the current state of the house (locked/unlocked). Another function is used to write values into tables with temperatures that come as a message from the camera modules via the MQTT protocol.

The first one, *on_message*, is called when the MQTT broker sends a message. An example of this code is shown in **Figure 9**. In this function, the message is then filtered using a topic, where we are interested in what the name of the message is. In the case of a topic called Refresh, we know that a new device has just joined the communication and is requesting an update of the data. Therefore, a message is sent to this command, which calls a function that reads the current pin code and house status from the database. It sends these messages after an encrypted communication.

The second function then takes care of changing the state of the house (unlocking/locking). If it occurs, the message will be retransmitted after 5 seconds to make it 100% certain that all connected devices will receive the message. At the same frequency, messages are also sent about the state of the intruder (i.e. if one of the camera devices detected movement), if the house is in a locked state.

The script that mediates communication with the GSM module first sets up this module. It then goes into an endless loop in which it checks to see if an intruder has entered the house. If so, then an SMS message is sent to all telephone numbers that are registered in the database. If a telephone number is dialed, the number is first verified whether it is written in the database and has the appropriate rights to open the gateway. Then there is contact on the relay module, which is connected to the Raspberry Pi, it closes and thus the gate opens. We added this function at the request of the owner of the house on which the system is installed. By sending an SMS "lock" or "unlock" it is possible to manipulate the security system of the house. An example of this code is shown in **Figure 10**. Of course, when sending an SMS, we first check whether the number from which the SMS was sent is registered in the database together with the appropriate permissions. Subsequently, an SMS message is sent to this telephone number. This message contains the current time, the status of the house (unlocked/locked), the last measured temperatures from all sensors and, last but not least, the remaining credit on the SIM card (so that the user knows

```
    def on_message(client, userdata, message):
        global last_LockState, last_IntruderState
        if ("Refresh" in message.topic):
            print("Requirement for Refresh")
            client.publish('Read/ESP/Lock', payload=last_LockState, qos=1, retain=False)
            time.sleep(1)
            client.publish('Read/ESP/Pin', payload=ReadFromDB("Read/ESP/Pin"), qos=1, retain=False)
            time.sleep(1)
            client.publish('Read/ESP/Intruder', payload=last_IntruderState, qos=1, retain=False)
        elif ("Write" in message.topic):
            print("Topic: " + message.topic + "  Data: " + message.payload.decode('utf-8'))
            SaveToDB(message.topic, message.payload.decode('utf-8'))

def publish_state():
    global last_LockState, last_IntruderState
    while client.connected_flag:
        time.sleep(2)
        if ReadFromDB('Read/ESP/Lock') is not last_LockState:
            last_LockState = ReadFromDB('Read/ESP/Lock')
            client.publish('Read/ESP/Lock', payload=last_LockState, qos=1, retain=False)
        if last_LockState and ReadFromDB("Read/ESP/Intruder") is not last_IntruderState:
            last_IntruderState = ReadFromDB("Read/ESP/Intruder")
            client.publish('Read/ESP/Intruder', payload=ReadFromDB("Read/ESP/Intruder"), qos=1, retain=False)
```

Figure 9.
Fragment of python script for MQTT communication.

```
while True:
    reply = bytes.decode(ser.read(ser.inWaiting()))
    if reply !="":
        print (reply)
        if "+CLIP:" in reply:  # if calling
            phoneNumber = reply[reply.index("+CLIP: ") + 8:reply.index("+CLIP: ") + 20]
            for x in range(len(PhoneTable)):
                if PhoneTable[x][0] in phoneNumber and 1 is PhoneTable[x][2]:
                    print("i am opening the gate")
                    GPIO.output(17, GPIO.LOW)
                    time.sleep(2)
                    GPIO.output(17, GPIO.HIGH)
                    # SendSMS(phoneNumber)
            ser.write(str.encode('ATH\r'))  # stop call

        if '+CMT: ' in reply: # if SMS
            phoneNumber = reply[reply.index("+CMT: ") + 8:reply.index("+CMT: ") + 20]
            print (phoneNumber)
            for x in range(len(PhoneTable)):
                if PhoneTable[x][0] in phoneNumber and 1 is PhoneTable[x][1]:
                    if "lock" in reply.lower():
                        print("i am locking the house")
                        ChangeState("true")
                        time.sleep(1)
                        SendSMS(phoneNumber)
                    elif "unlock" in reply.lower() or "unlocking" in reply.lower():
                        print("i am unlocking the house")
                        ChangeState("false")
                        time.sleep(1)
                        SendSMS(phoneNumber)
                    else:
                        SendSMS(phoneNumber)
        ser.flushInput()  # Clear buf
        ser.flushOutput()  # Clear buf
    time.sleep(1)
```

Figure 10.
Fragment of python script with GSM module communication.

when it is necessary to recharge the mobile credit). At the request of the owner, a function has been added that controls the boiler temperatures, which are sent from one of the modules. The owner is informed by an SMS message when a limit of 80° C is exceeded.

5. System security measures

The proposed system must, of course, comply with safety principles. As these are a number of interconnected technologies, there are several recommendations for security measures, from completely common to complex solutions. The system uses a number of hardware modules, communication elements, a web server, a database, configurable access to the control unit, an input pin to unlock the system and all this must be secured. The basic safety recommendation is to separate the wireless network for the IoT modules of our system from the wireless network of the house in which we use PCs, laptops and other common devices. This can be done either by the total separation and operation of two separate networks or by different variants of VLAN and micro segmentation of the network, depending on the available hardware of the specific solution. In our case, it is a completely separate network.

Another important element is the principle of preventive protection of the Raspberry Pi control unit, securing SSH remote access, compliance with security principles and best practices for MariaDB database such as Avoiding running *mysqld*

as root, Limit *ssh* access, Limit *sudo* access on the MariaDB and the use of security plugins [37]. The limitation of the number of attempts to enter the input Pin on the instrument panel is solved by the possibility of configuring the selected time delay after the unsuccessful entry of 3 consecutive attempts, including sending a notification to a mobile phone to pre-selected phone numbers.

Last but not least, securing the communication of the MQTT protocol is also important, as this protocol was not originally designed for security, but for its availability and undemanding implementation. We can apply the security of this communication in implementations of later Mosquitto brokers. This is done using authentication and authorization mechanisms that allow you to add plugins. This is done using authentication and authorization mechanisms that allow you to add plugins. This is a client authentication using a client IDs, Access Control List or x509 Clients certificates. To protect the contents of your MQTT messages you can use TLS or SSL Security and Payload Encryption.

6. Installation and testing of security system

To be able to implement the device into operation and place it in the house, these devices must have mechanical protection and at the same time, allow the most convenient possible installation. For this reason, we decided to produce cases for camera modules and input panels with a keyboard and touch screen on a 3D printer.

It is necessary to insert ESP32, an LCD, a keyboard, a speaker and DC-DC voltage converter in the input panel, which will reduce the distribution voltage of 24 V DC to the required 5 V DC. The housing is designed for wall mounting around the front door, with a height suitable for the user. The camera module contains the following elements: ESP32-CAM, temperature sensor, a PIR sensor, a RGB LED, IR LED, a NPN transistor, a constant current source, a DC-DC converter and an antenna for Wi-Fi. We designed the housing so that the camera can be mounted in the corner of the room so that it blends in with the surrounding walls as much as possible and does not disturb the integrity of the room in any way. The camera is placed in a rotating sphere to be set in which area of the room we want to record. After the implementation of hardware modules, the 3D printing of assembly boxes and the mounting and software implementation of the entire system, the testing and installation phase took place, first in the laboratory and after verification of the functionality of all modules assembly into a residential house. **Figure 11** shows

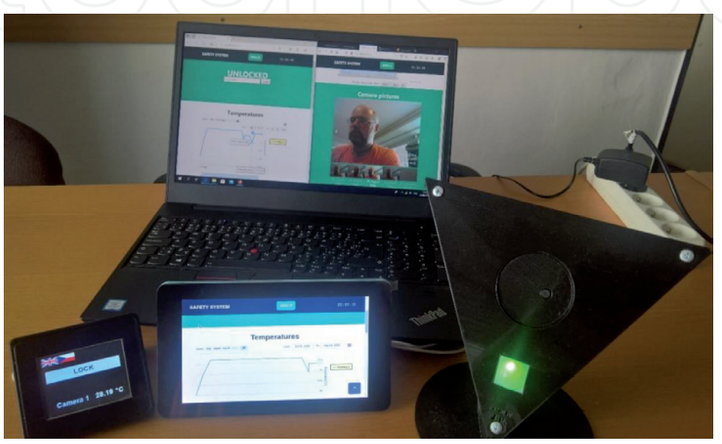

Figure 11.
Testing of hardware modules and software functionality.

Figure 12.
Installation of the ESP32 cam module in the living room.

the testing of the input panel, the touch screen, one camera module and the display of monitoring temperatures and captured photos on the webserver in individual modes simulating a closed and unlocked house. This test also related to testing the alarm and sending messages by the GSM module to pre-selected telephone numbers.

We installed three camera modules and two input panels in the house, one with a touch screen and the other with a membrane keyboard for more comfortable handling in a dusty room, with one module for collecting temperatures in the boiler room and directly from the boiler and one Raspberry Pi Zero W control unit.

All components of the device are powered from a 12 V source with direct voltage so that even on long routes, the voltage of individual devices is at least 5 V. **Figure 12** shows one of the three camera modules that has just been activated and signals its initialization with the color green. You can see the input panels in figures in the Hardware Modules chapter. On the left, we can see a panel with a membrane keyboard, which is located in the boiler room, and the other is located at the main entrance. Raspberry Pi Zero W is located in the distribution board, together with a GSM modem and a relay module. The individual devices were tested of course, after connecting all the modules, some parts of the code were fine-tuned to ensure the greatest possible comfort of use and reliability.

7. Price of the created monitoring system

The price of the created monitoring system is around 250 euros, when, of course, we add up only all the purchased hardware modules. The price of the 3D printing of the designed boxes is negligible. We did not include the price of our own development work and software implementations in the price of the system. The advantage of the whole system is that when it is extended by other measuring IoT modules in the current configuration, the final price increases only by a few Euros, when increasing the number of existing measuring units, the final price increases in the price of a particular module, whether it is another ESP32 module equipped with a camera and sensors or input or display units, the price of which is given in the table. The price of the resulting device could be reduced, but our goal was not to choose the cheapest components. In the production of more pieces of the monitoring system and with the quantity discount of selected components, we would get even lower prices. Prices in the **Table 1** are given according to the European local market, it is possible to get lower prices elsewhere.

Device	Item	Price in Euro
Control unit	Rpi Zero	12
	SD card	10
	GSM Modem	8
	Relay module	2
	DC – DC converter	4
	RTC real time module	2
1 piece	Total	38
Camera module	ESP32 – CAM	14
	Temperature sensor	1,9
	PIR sensor	2,3
	DC – DC converter	1,5
	RGB LED	0,6
	Resistances 1kΩ, 4k7Ω	0,4
3 pieces	Total (rounded)	62
Input panel with Touch display	ESP32	8
	Touch display	25
	DC – DC converter	2
1 piece	Total	35
Input panel	ESP32	8
	Keyboard	1,5
	LCD Char Display	4,5
	Alarm	3
	DC – DC converter	2
1 piece	Total	19
Measurement device	ESP32	8
	Temperature sensor	2
	Outdoor temperature sensor	4
	DC – DC converter	2
1 piece	Total	16
Power supply	Power supply	30
Electrical wiring	House wiring, cables, LED indicators	30
Total price		230 Euro

Table 1.
Price of components.

8. Conclusion

We designed a camera system with an ESP32 chip, which was installed on a house; the system is connected to its Wi-Fi network. The set was extended using input panels at the main entrance and the entrance to the boiler room. It was necessary to create a parent device that would serve as a data store and control for

the entire report. This was done using a Raspberry Pi, whose function is to pass communication to all other elements in the assembly. At the same time, it serves as a repository of photographs captured from camera modules and a web server is running on it, which allows access to data on this unit.

In an idle state, the camera modules are inactive (they do not take camera pictures). However, if the house is locked, which can be done by using two input panels or using a web server or by sending an SMS command, then after a certain time delay, the camera modules will be activated after leaving the house. These check whether motion has been detected on the PIR sensors. If this situation occurs, then this information is sent to the control unit and sent to all camera modules. Immediately afterwards, images are taken from all camera modules with a second interval. An alarm located in one input panel is also activated, and an SMS message is sent from the control unit via a GSM modem to predefined telephone numbers stating that an intrusion into the house has occurred. Images taken from the camera modules are sent via HTTP post to the control unit, where they are stored.

ESP32 modules are also equipped with temperature sensors for monitoring the temperature in the rooms. These temperatures are processed and sent to the control unit, where they are stored in the database. The whole set is further expanded by one device that reads the temperatures in the boiler room and the temperature directly from the boiler. This makes it possible to monitor the dependence between the temperatures of the boiler and in the individual rooms. Thanks to this, a house could be heated more efficiently.

The web server, which also runs on this unit, and ensures easy and clear access to all data that is continuously stored here. The current status of the house and the possibility of locking with it after entering the PIN code is displayed on the webserver. The application displays a graph that clearly plots the temperatures measured on individual devices as a function of time. There is the possibility of filtering a certain device from which the data were measured, as well as the possibility of displaying a certain time interval. The last section of the website is a gallery of photos that were taken on camera modules and are sorted from the most recent, with each photo showing the exact time when it was captured.

A GSM modem is connected to the control unit, which enables the remote manipulation of the house via an SMS message, but only for selected telephone numbers. By sending an "unlock" or "lock" command, this action is performed. Back then, the user is informed about the execution via an SMS message, which contains the current state of the house (unlocked/locked). The last measured temperatures from all connected sensors and the credit balance on the SIM card are also sent. If any other SMS message is sent, again only from selected numbers, the user is sent an SMS with the same content as when sending the order. When the telephone number rings, the relay is connected, which is connected to the gate for entering the property of the house. At present, we are also dealing with taking camera pictures in poor visibility conditions or complete darkness. This problem could be solved by removing the IR filter located on the camera lens and illuminating the room by adding, for example, IR LEDs, the light of which would not attract attention because it is not visible to the human eye.

Acknowledgements

This work was supported by the European Regional Development Fund in the Research Centre of Advanced Mechatronic Systems project,

CZ.02.1.01/0.0/0.0/16_019 /0000867 within the Operational Programme Research, Development and Education and the project SP2020/571 Research and Development of Advanced Methods in the Area of Machines and Process Control by the Ministry of Education, Youth and Sports.

Author details

Marek Babiuch* and Jiri Postulka
Department of Control System and Instrumentation, Technical University of Ostrava, Ostrava, Czech Republic

*Address all correspondence to: marek.babiuch@vsb.cz

References

[1] Foltýnek P, Babiuch M and Šuránek P. Measurement and data processing from Internet of Things modules by dual-core application using ESP32 board. *Meas. Control*. 2019; 7-8. DOI: 10.1177/0020294019857748.

[2] Dokic K, Martinovic M and Radisic B. Neural Networks with ESP32 - Are Two Heads Faster than One? *Conference on Data Science and Machine Learning Applications, CDMA 2020*. DOI: 10.1109/CDMA47397.2020.00030.

[3] Biswas SB. and Tariq Iqbal M. Solar Water Pumping System Control Using a Low Cost ESP32 Microcontroller. *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE).* DOI: 10.1109/CCECE.2018.8447749.

[4] Kodali RK and Valdas A. MQTT Based Monitoring System for Urban Farmers Using ESP32 and Raspberry Pi. *International Conference on Green Computing and Internet of Things, ICGCIoT 2018*. DOI: 10.1109/ICGCIoT.2018.8752995.

[5] Ram CRS, Ravimaran S, Krishnan RS, Ismail, M, et al. Internet of Green Things with autonomous wireless wheel robots against green houses and farms. *Int. J. Distrib. Sens. Netw*. 2020; 6. DOI: 10.1177/1550147720923477.

[6] Sarjerao BS and Prakasarao A. A Low Cost Smart Pollution Measurement System Using REST API and ESP32. *International Conference for Convergence in Technology, I2CT 2018*. DOI: 10.1109/I2CT.2018.8529500.

[7] Abdullah AH, Sudin S, Ajit MIM, et al. Development of ESP32-based Wi-Fi Electronic Nose System for Monitoring LPG Leakage at Gas Cylinder Refurbish Plant. *nternational Conference on Computational Approach in Smart Systems Design and Applications, ICASSDA 2018.* DOI: 10.1109/ICASSDA.2018.8477594.

[8] Atmajaya D, Kurniati N., Astuti W, et al. Digital Scales System on Non-Organic Waste Types Based on Load Cell and ESP32. *East Indonesia Conference on Computer and Information Technology: Internet of Things for Industry, EIConCIT 2018*. DOI: 10.1109/EIConCIT.2018.8878667.

[9] Allafi I and Iqbal T. Design and implementation of a low cost web server using ESP32 for real-time photovoltaic system monitoring. *2017 IEEE Electrical Power and Energy Conference, EPEC 2017.* DOI: 10.1109/EPEC.2017.8286184.

[10] Kljakić S, Rajs V, Bodić M and Cvetković N. Position Regulation System with Camera and Microcontroller ESP32. E*UROCON 2019 - 18th International Conference on Smart Technologies*. DOI: 10.1109/EUROCON.2019.8861899.

[11] Rai P and Rehman M. ESP32 Based Smart Surveillance System. *International Conference on Computing, Mathematics and Engineering Technologies, iCoMET 2019*. DOI: 10.1109/ICOMET.2019.8673463.

[12] Gunasagaran R, Kamarudin LM and Zakaria A. Embedded Device Free Passive (EDfP) System: Sensitivity of ESP32. *IEEE Student Conference on Research and Development (SCOReD)*. DOI: 10.1109/SCORED.2018.8710808.

[13] Misal SR, Prajwal SR, Niveditha HM, et al. Indoor Positioning System (IPS) Using ESP32, MQTT and Bluetooth. *Fourth International Conference on Computing Methodologies and Communication 2020*, DOI: 10.1109/ICCMC48092.2020.ICCMC-00015.

[14] Murmu PP, Paul H, Roopa JJ and Timothy AJ. A Novel modernistic

techniques in women security system using ESP32 and Arduino Uno. *International Conference on Signal Processing and Communication, ICSPC 2019*. DOI: 10.1109/ICSPC46172.2019.8976745.

[15] Lojka T, Miškuf M and Zolotová I. Industrial IoT Gateway with Machine Learning for Smart Manufacturing. *IFIP Advances in Information and Communication Technology*, Volume 488, 2016, DOI: 10.1007/978-3-319-51133-7_89.

[16] Aghenta LO and Iqbal MT. Low-Cost, Open Source IoT-Based SCADA System Design Using Thinger.IO and ESP32 Thing. *Electronics 2019;8.*DOI: 10.3390/electronics8080822.

[17] Mishra A, Reichherzer T, Kalaimannan E, et al. Trade-offs involved in the choice of cloud service configurations when building secure, scalable, and efficient Internet-of-Things networks. *Int. J. Distrib. Sens. Netw*. 2020; 2. DOI: 10.1177/1550147720908199.

[18] Aydos M, Vural Y and Tekerek A. Assessing risks and threats with layered approach to Internet of Things security. *Meas. Control*. 2019; 5-6. DOI: 10.1177/0020294019837991.

[19] Catelani M, Ciani L, Bartolini, A, et al. Characterization of a low-cost and low-power environmental monitoring system. *International Instrumentation and Measurement Technology Conference 2020*. DOI: 10.1109/I2MTC43012.2020.9129274.

[20] Škraba A, Kolozvari A, Kofjac D., et al. Prototype of Group Heart Rate Monitoring with ESP32. *8th Mediterranean Conference on Embedded Computing, MECO 2019*. DOI: 10.1109/MECO.2019.8760150.

[21] Kushnir V, Koman B and Yuzevych V. IoT Image Recognition System Implementation for Blind Peoples Using esp32, Mobile Phone and Convolutional Neural Network. *International Scientific and Practical Conference on Electronics and Information Technologies, ELIT 2019*. DOI: 10.1109/ELIT.2019.8892289.

[22] Ghosh D, Agrawal A, Prakash N and Goyal P. Smart Saline Level Monitoring System Using ESP32 And MQTT-S. *IEEE 20th International Conference on e-Health Networking, Applications and Services, Healthcom 2018*. DOI: 10.1109/HealthCom.2018.8531172.

[23] Nemec D, Janota A, Gregor M, et al. Control of the mobile robot by hand movement measured by inertial sensors, *J. Electr. Eng*. 2017; 99,4. DOI: 10.1007/s00202-017-0614-3

[24] Cupkova D, Kajati E, Mocnej J, et al. I. Intelligent human-centric lighting for mental wellbeing improvement. *Int. J. Distrib. Sens. Netw*. 2019; 9. DOI: 10.1177/1550147719875878.

[25] Nikolov N and Nakov O. Research of Secure Communication of Esp32 IoT Embedded System to.NET Core Cloud Structure using MQTTS SSL/TLS. *International Scientific Conference Electronics, ET 2019*. DOI: 10.1109/ET.2019.8878636.

[26] Iqbal A and Iqbal T. Low-cost and Secure Communication System for Remote Micro-grids using AES Cryptography on ESP32 with LoRa Module. *IEEE Electrical Power and Energy Conference, EPEC 2018*. DOI: 10.1109/EPEC.2018.8598380.

[27] Fan C and Ding Q. A novel wireless visual sensor network protocol based on LoRa modulation. *Int. J. Distrib. Sens. Netw.* 2018; 14, Issue 3. DOI: 10.1177/1550147718765980.

[28] Yao F, Yang SH and Xia B. A Zigbee based home automation: System design and implementation. *Meas. Control*.

2008; 41, Issue 10, pp. 310-314. DOI: 10.1177/002029400804101003.

[29] Asensio Á, Marco Á, Blasco R, et al. Protocol and architecture to bring things into internet of things *Int. J. Distrib. Sens. Netw*. 2014. DOI: 10.1155/2014/158252.

[30] Puliafito A, Celesti A, Villari M, et al. Towards the integration between IoT and cloud computing: An approach for the secure self-configuration of embedded devices. *Int. J. Distrib. Sens. Netw*. 2015. DOI: 10.1155/2015/286860.

[31] Babiuch M, Foltynek P and Smutny P. Using the ESP32 Microcontroller for Data Processing. *International Carpathian Control Conference, ICCC 2019*. DOI: 10.1109/ CarpathianCC.2019. 8765944.

[32] Maier A, Sharp A and Vagapov Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. *nternet Technologies and Applications, ITA 2017*. DOI: 10.1109/ ITECHA.2017.8101926.

[33] Carducci CGC, Monti A, Schraven MH, et al. Enabling ESP32-based IoT Applications in Building Automation Systems. *IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2019*. DOI: 10.1109/METROI4.2019.8792852.

[34] Takacs G, Vachálek J and Rohal'-Ilkiv B. Online Structural Health Monitoring and Parameter Estimation for Vibrating Active Cantilever Beams Using Low-Priced Microcontrollers. *Shock Vib.* 2015. DOI: 10.1155/2015/506430.

[35] *Nextion: HMI displays* [online]. 2020 [cit. 2020-08-24]. Available from: https://nextion.tech/basic -series-introduction/

[36] Mocnej J, Pekar A, Seah WKG, et al. Quality-enabled decentralized IoT architecture with efficient resources utilization. *Rob. Comput. Integr. Manuf.*, Volume 67, 2020. DOI: 10.1016/j. rcim.2020.102001.

[37] Percona Live. *MariaDB Security Features and Best Practices* [online]. May 2019 [cit. 2020-09-04]. https://www. percona.com/live/19/sessions/mariadb-security-features-and-best-practices