MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

IoT based Smart home security system with remote access control and management

Bachelor's Thesis

VÍT NAKLÁDAL

Brno, Fall 2022

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

IoT based Smart home security system with remote access control and management

Bachelor's Thesis

VÍT NAKLÁDAL

Advisor: Bacem Mbarek, PhD

Department of Applied Informatics

Brno, Fall 2022

MUNI
FI

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Vít Nakládal

Advisor: Bacem Mbarek, PhD

Acknowledgements

First of all, I would like to thank my supervisor Bacem Mbarek, Ph.D., for his feedback, but also for his patience and trust in me. After that, I would like to thank my family and my friends for supporting me during my studies.

Abstract

This work aims to create a home security system that competes with current solutions, primarily in price. It uses a cheap module with a built-in camera, often used for IoT projects. In case of unexpected events, the monitoring device takes a photo and informs the mobile application user, who can view the photo in this application. As an intermediary between the monitoring device and the mobile application, the Google Cloud Platform is used.

Keywords

IoT, ESP32, PlatformIO, Home security, Google Cloud, Flutter, Mobile App

Contents

List of Tables

List of Figures

1 Introduction

For adequate home security, it is essential to constantly monitor the home for anything suspicious. For monitoring, the administrator can employ a person who will be physically present at the household and will monitor the given object. Such a solution does not need any preparation but is relatively expensive. A cheaper alternative may be to use a device that takes care of this automatically. Such a device can work on the principle of investigating various events, such as whether someone entered the home, and then informing the administrator that something suspicious is happening. In addition, this warning message can contain information about why the given device evaluated the situation as suspicious (for example, that it detected movement using a motion sensor) and, in addition, other information that would help administrators clearly determine whether it is a false alarm or a real danger. This additional data is helpful because evaluating a hazard from a photo or video is a relatively challenging task for a device but not for a person.

The following sections describe the current options used for home security and their advantages and disadvantages. It also describes what problems of current solutions this work tries to face and how it tries to achieve them. The last section, the Thesis outline, briefly summarizes all the chapters in this thesis.

1.1 Background

Nowadays, the Internet of Things (IoT) [1] is increasingly popular [2] and is evolving across many fields. Using a network of IoT devices inside a building can be used for indoor localization of vehicles [3], or sensors in trash bins can inform the city's cleaning company and help them plan a more efficient route for waste collection [4]. Along with the development of wireless Internet access in households, new possibilities for home security are opening up.

IoT in Home Security

Security is an important thing to deter a potential robber. The so-called passive security, such as door locks, may no longer be enough nowadays, so extending this security with so-called active security is a good idea. Have sensors in and around the house that detect robbers and inform the owner well in advance to react. It may not only be a robber, but even in the event of an unexpected fire or gas leak, it is good to know about it as soon as possible to minimize damage.

One of the most common security is using a camera with motion detection. Such a solution can be straightforward to set up. The camera allows it to connect to the wireless Internet via Wi-Fi and run on batteries simultaneously, so it does not need to run any cables. An example of such a product is the Google Nest Cam [5]. It also has a motion and sound sensor to activate recording and send the video to the owner's mobile phone to save energy. Even the situation evaluation does not have to be sole to the owner. Thanks to image analysis, this camera can even classify what it is, to a limited extent. For example, a similar Google Nest Doorbell device [6] can distinguish between the arrival of a postman with a package, acquaintances, or even alert suspicious activity with a recommendation to call an emergency service. Other features include saving the record to cloud storage and automatically turning security on / off by locating the owner's mobile [7]. Additionally, multiple cameras can be used together and possibly combined with a Google Nest Protect smoke detector [8], which can also detect movement and light and serve as a practical night light. It should be clear from the description that this security method is straightforward and that each device can do quite a lot.

As an alternative, building a home security system can be made up of simpler and smaller devices. For example, the Zigbee wireless communication protocol [11] allows you to create energy-efficient and small devices that securely share data. Examples of shared data from the device can be information about temperature, light, movement, and other measurable data, as well as control signals for controlling lights, electrical outlets, and other things. Another advantage is that they can run on battery power for several years [12]. The owner can place these devices around the house and then use the application to manually set what happens at different values obtained by each

(a) Nest Cam

(b) Nest Doorbell

(c) Nest Protect

Figure 1.1: Google Nest products; *source: Google Store* [9][10][8]

device. To fulfill the security function, it is usually necessary to connect it to some other device, such as a camera monitoring the home or an electronic lock that can connect to the network [13]. Thanks to it, the owner can easily change access to the household for specific people or even generate a temporary key for potential guests.

1.2 Motivation

The problem with currently available solutions might be for someone at a higher price. Security with Google Nest or similar products costs a relatively large amount, and it is necessary to pay a subscription to take advantage of all their features. As a result, they are not suitable for securing cheaper and poorer households.

The security system in question in this work will be easy to set up and cheaper than the mentioned solutions, with the offered options as close as possible to the already available devices.

1.3 Objective of the thesis

This thesis offers a device with a motion-tracking sensor and a camera to capture the current state of the household. This device connects wirelessly to the Internet using Wi-Fi, and once connected, it starts monitoring the household. If there is any movement in the home while

the device is active, the device takes a picture and immediately saves it to the remote server, the so-called cloud. The cloud records this insertion of a new photo and sends a message to a mobile phone with the necessary application installed. This application responds to the message by sending a notification to the mobile phone, alerting the administrator that something is happening in the monitored household. At the same time, it downloads the created photo, thus allowing the administrator to see how the particular household currently looks.

1.4 Limitations

The most significant limitation this system has is that the assembled system currently does not have the option of managing access to the household. he section] in possible future changes mentions this. The device used as a en brings additional limits in the form of the delay between detecting the danger and informing the administrator and the photos' quality. Another limitation of this system is that the household only tries to protect itself from burglars. This system does not solve a fire or a possible gas leak.

More limitations are mentioned throughout the rest of the work.

1.5 Thesis outline

The detailed description of how this thesis achieved this solution is described as follows:

- **Chapter 1:** Introduces the topic and describes state-of-the-art.
- **Chapter 2:** Describes some terms and tools used in this bachelor thesis.
- **Chapter 3:** Provides an implementation overview and describes the system's functioning in parts or as a whole.
- **Chapter 4:** Shows the system's functioning using measured data and an example of taken photos.
- **Chapter 5:** Summarizes the whole thesis and shows where it can continue.

2 Theoretical background

This chapter briefly describes some concepts this thesis is built on, together with their advantages and disadvantages.

2.1 ESP32-CAM AI-Thinker

ESP32-CAM AI-Thinker [14] is a very cheap development board based on ESP32 SoC [15], which is a series of cheap, low-power system on chip microcontrollers with integrated Wi-Fi and Bluetooth [16]. Some features include, for example, 34x programmable general-purpose input/output (GPIO) pins, analog-to-digital signal converter (ADC), digital-to-analog signal converter, SD card host controller, 5 μA deep sleep current with the possibility of waking up by GPIO interrupt, timer or by ADC measurements. This last mentioned feature opens a possibility to run it on the battery, and its potential usage is discussed in section 5.2.3. Furthermore, security features include flash encryption and hardware-accelerated calculation of AES, SHA-2, and RSA [17]. Currently, this work is not utilizing the chip to its maximum. Thanks to its possibilities, however, various options for expanding this system open up.

Features and advantages

The development board has a built-in 2MPx camera with a flashlight and micro SD card reader connected to the ESP32 SoC. Specifically, it is the OV2640 camera, which is also an SoC. Some advantages include, for example, taking up to 15 photos per second in 1600x1200 resolution, noise reduction, and automatic exposure or white balance. The most significant advantage is the compression of photos into JPEG format, which can reduce the resulting photo size by almost 25 times [18]. Thanks to this, the target chip only needs a smaller amount of free memory and, at the same time, does not need to use computing power to compress the photo.

The development board also contains a voltage regulator chip at 3.3V usable for peripherals, 4MB PSRAM to help the camera run

smoothly [19], a microSD card reader, and up to 10 additional GPIO pins (if the microSD card reader is not used).

Disadvantages

The disadvantage of this development board is the limitation of usable pins if we want to use the microSD card reader. In this case, a maximum of 4 GPIO pins (GPIO 0, 1, 3, 16) can be used, with restrictions such as, for example, that when the device is turned on, there must not be a positive voltage on the GPIO 0 pin. Due to the limited number of usable pins, it is also impossible to use ADC when using Wi-Fi. More information about usable pins of ESP32 and this development board are available[1]. The lack of pins, even supporting ADC, can be partially solved by connecting the necessary devices using I2C communication[2]. However, this will result in additional costs. A further and unsolvable limitation might be the lack of enough flash memory for the compiled code. For example, the section about Limitations of the Monitoring node shows that the application in this thesis used about 93% of flash memory.

2.2 Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is a suite of Google's cloud computing services [20] that are distributed among many regions [21]. Each service is paid monthly on how much was used by the application. However, Google offers limits to a sublist of services, which, if not exceeded, results in no fee for that service [22]. This free possibility is great for experimenting or even small projects.

Among the first services that GCP offered was App Engine, which served as a tool for software developers to deploy and run applications without worrying about infrastructure [23]. Other services include, for example, Cloud Storage, Firestore, and Cloud Functions. These three services are also services that GCP shares together with the

1. See `https://randomnerdtutorials.com/esp32-pinout-reference-gpios/` and `https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/`
2. See `https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/`

Firebase development platform [24], which provides detailed documentation and cross-platform SDKs to help build and ship apps on many platforms, including Android [25]. Each service is configurable, for example, via the graphical web interface Google Cloud console [26], or by using the command line with gcloud CLI tools [27].

The security system this thesis is about uses these services, therefore, will be described in more detail below.

2.2.1 Cloud Storage

It is a service that enables data storage on Google Cloud. The data is immutable and can be in any format [28]. More specifically, data is stored in Buckets with a defined geographic area where the data will be physically stored [29]. There can be an unlimited number of these Buckets; however, the rate of creating and deleting Buckets is limited [30]. The possibilities for securing access to data and the data itself are via using Identity and Access Management (IAM), about which there is more details[3]. Another security for the data is their encryption (which is by default), and, for example, the need for authentification for users who want to manipulate the data [31].

2.2.2 Firestore

Firestore is a flexible, easy-to-use NoSQL database for developers [32]. The data in the NoSQL database is stored in documents that are stored in collections. A document can contain different data, including another nested collection, creating a structured hierarchy [33]. An example of the data structure for the NoSQL database is visible in Figure 2.1.

One of the advantages of the Firestore database is that it will take care of the infrastructure needed in the background, i.e., it will scale up or down depending on the load. To retrieve data, the Firestore offers an expressive querying mechanism, for example, queries combining filtering and sorting. The Firebase SDK for Firestore offers end devices the possibility of real-time updates, and offline mode support [34].

3. See `https://cloud.google.com/storage/docs/access-control/iam`

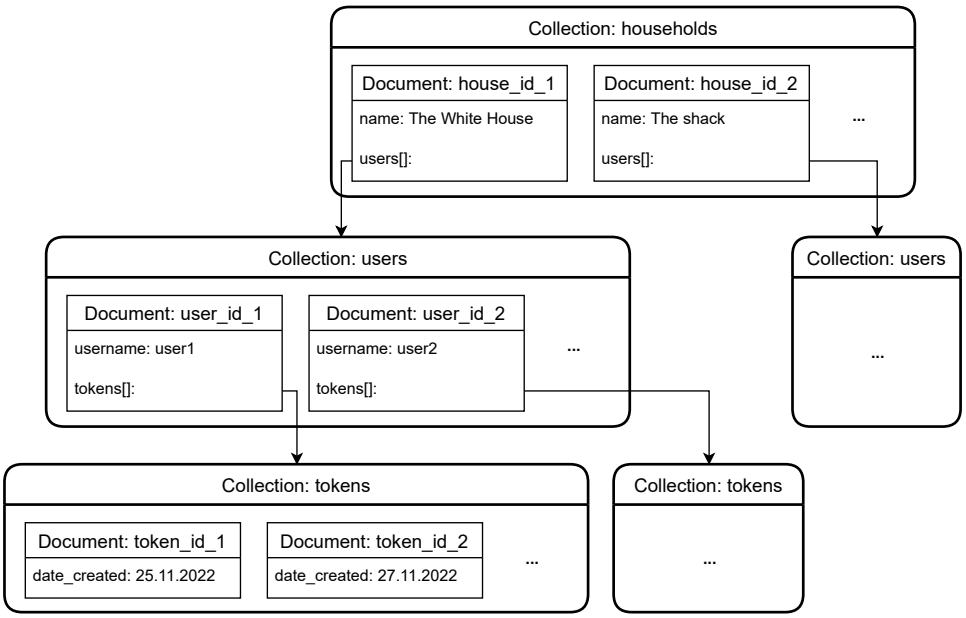


Figure 2.1: Example of the data structure of NoSQL database

2.2.3 Cloud Functions

Cloud Functions allow the execution of functions without a fixed server in the background. Thanks to this, there is no need to keep the server running, and deal with its possible outages, scaling it up or down based on the usage. Cloud Functions is taking care of it, creating a very robust code execution solution and greatly simplifying the work of software developers.

The function execution is triggered only in case of the occurrence of the event. The event is bound to the function execution. Because it is a serverless solution, only the execution of the function is charged, not the time for which it is deployed [35].

2.3 Flutter

Flutter is an open-source framework supported by Google for multi-platform app development [36]. It was officially launched in 2017 [37], and its popularity is gradually increasing, while the popularity of

alternatives is stagnating or even decreasing [38]. Due to its growing popularity and the fact that it is open source, it is also maintained and expanded by a growing community of people.

It uses Dart as the platform-independent programming language to create applications, and this programming language is used both for creating UI and application logic [36]. Dart supports both ahead-of-time (AOT) and just-in-time (JIT) compilation. Thanks to the compilation into the native code of the target device, it enables the quick start and operation of the application. On the contrary, just-in-time compilation gives a great advantage to developers by making changes in the code visible immediately. So fixing bugs or deploying new functions can be tried immediately [39].

Disadvantages

Although the popularity of using Flutter is increasing, it is still a relatively new framework and does not have a large extension base. Other disadvantages include the larger application size of the resulting application and the limited number of platforms it supports so far [39].

2.4 PlatformIO

PlatformIO is a professional tool for embedded systems engineers facilitating the development of applications for embedded systems [40]. It aims to simplify the complicated process of setting up development software and then to deploy the application to the target [41] device. It is mainly used as an IDE extension to the popular Visual Studio Code [42].

Its main features include, for example, the possibility of debugging without complex configuration and the need to know information about the debugging server used by the developer [43]. Another feature is support for Unit Testing, both on the host machine and multiple embedded devices, with the subsequent aggregation of results into the resulting test report [44]. Static Code Analysis is here to identify potential bugs, vulnerabilities, and security threats. This feature checks the source code and notifies the developer if there is a problem before

the developer compiles and deploys the application on the device [45]. Finally, thanks to PlatformIO, it is straightforward to deploy the same code on different development platforms by setting multiple environments in the Project Configuration File [42].

3 System design

This chapter describes the construction and functionality of the entire system. A brief description of the functioning of the entire system is described in Figure 3.1 and consists of the following three parts:

1. Monitoring node
2. Google Cloud Platform
3. Mobile phone

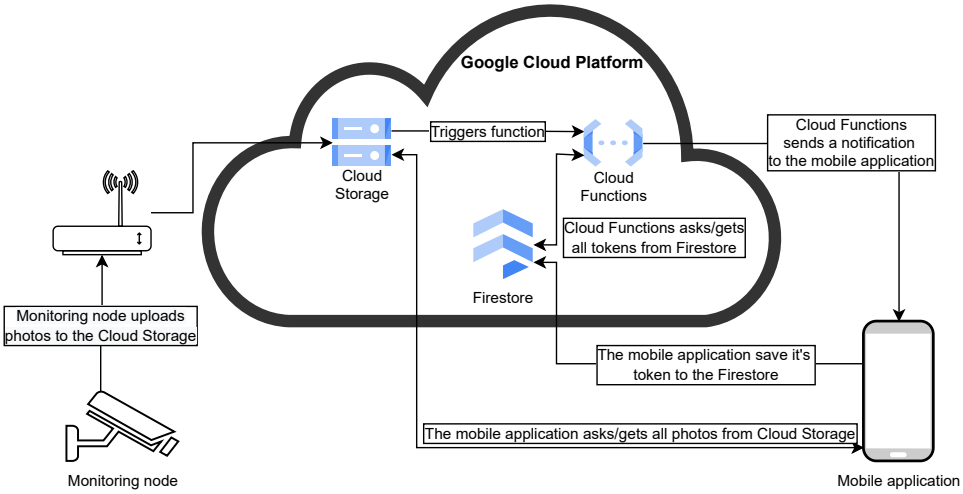


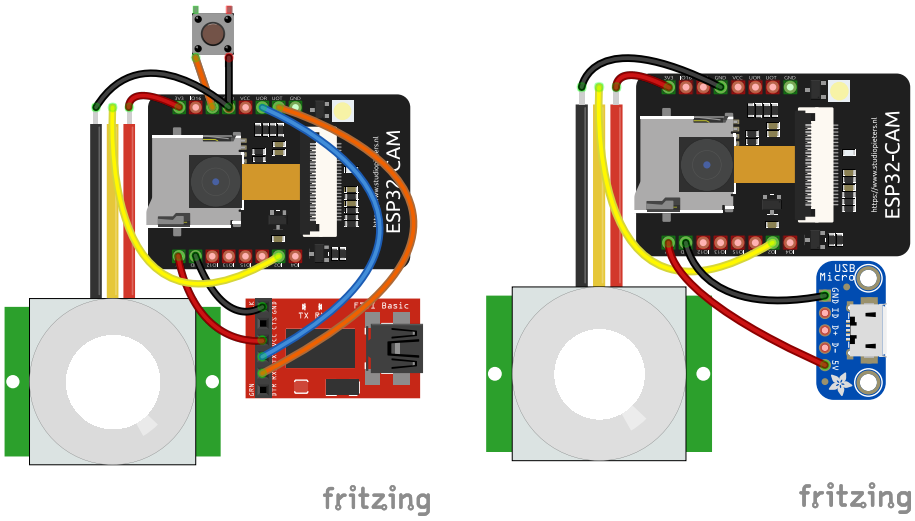
Figure 3.1: System workflow diagram

3.1 Monitoring node

The monitoring node is a device that is physically present in the household and monitors it. Its task is to take a photo when detecting movement, which it uploads to Google Cloud Storage using a Wi-Fi router, through which it connects to the Internet. Setting up of it is split into the Hardware and Software level.

3.1.1 Set up of monitoring node at Hardware level

The monitoring node consists of an ESP32-CAM AI-Thinker development board (Section 2.1 details it) and the AM312 motion sensor [46]. Figure 3.2, Table 3.1, and Table 3.2 shows the wiring diagram of the entire monitoring node, either during development (with additional USB to Serial converter and support button for uploading the code) - variant (a) or during regular operation - variant (b).



(a) Develop variant **(b)** Production variant

Figure 3.2: Schematic of monitoring node; tool used: Fritzing [47]

Table 3.1: Wiring description for develop variant

ESP32 dev. board pin	Target pin
IO0	one side of button
GND	second side of button
GND	GND on PIR sensor
3V3	Power pin on PIR sensor
IO2	Signal pin on PIR sensor
GND	GND on USB to Serial converter
5V	VCC on USB to Serial converter
UOR	TX on USB to Serial converter
UOT	RX on USB to Serial converter

Table 3.2: Wiring description for production variant

ESP32 dev. board pin	Target pin
GND	GND on PIR sensor
3V3	Power pin on PIR sensor
IO2	Signal pin on PIR sensor
GND	GND on USB
5V	5V on USB

3.1.2 Set up of monitoring node at Software level

For writing source code, analyzing the written code, uploading the code to the device, and displaying the output from the device, the Visual Studio Code with the PlatformIO extension is used. Visual Studio Code is a multi-platform, free, open-source source code editor supported by the community and Microsoft, allowing the installation of many different extensions [48]. PlatformIO greatly simplified adding the necessary libraries and compiling and deploying the application on the ESP32 development board. Section 2.4 provides more details about PlatformIO. The application is written using C programming language [1].

1. See `https://en.wikipedia.org/wiki/C_(programming_language)`

The application itself is built on this example [49]. The credentials had to be generated using the instructions from the same example to access and upload the pictures from the development board to the Cloud Storage [50]. Figure 3.3 provides a lightweight description of the operation of the example application. The version that this thesis is using has added motion tracking using a PIR sensor and a changed part of obtaining and uploading a photo. Any code execution is interrupted once movement occurs. While this interruption happens, a bit is set in the queue, signaling that movement has occurred. This interruption reduces the chance of not taking a photo if the sensor detects movement.

Acquiring and uploading a photo runs in a separate *take_picture_task* task created after a successful connection to Cloud Storage. This task first verifies whether a bit is set in the queue, signaling that a movement has occurred. If so, the queue will be emptied, and the photo will be taken and uploaded to Cloud Storage. In case of an upload error, the current task will be destroyed, and the device will try to re-establish the connection to Cloud Storage. Figure 3.4 describes the changes in the original *task_gcp* and the new *take_picture_task*.

Limitations of the Monitoring node

One of the most significant drawbacks is that the monitoring node uses hard-defined credentials in the source code to connect to Wi-Fi and Google Cloud Storage. Thus making it more secure and user-friendly is one of the most necessary upgrade for practical use. Another issue might be because of needing more flash memory where the program is stored - without any optimization, the PlatformIO showed the program used about 93%.

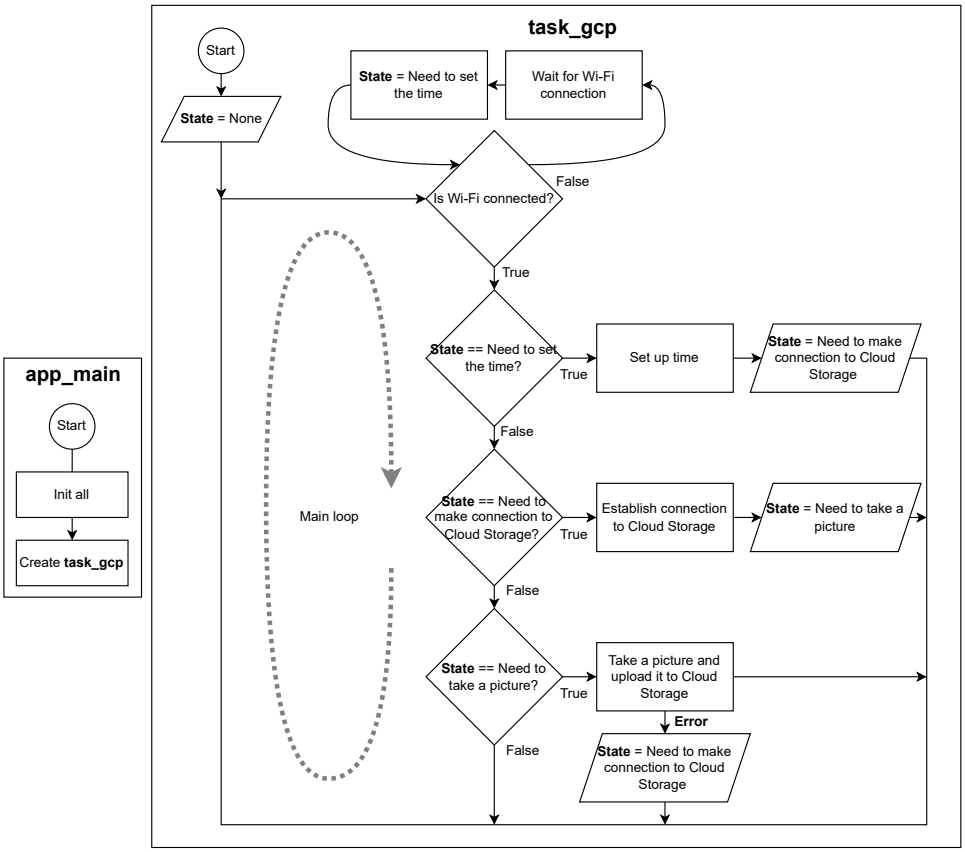


Figure 3.3: Flowchart of used example [49]

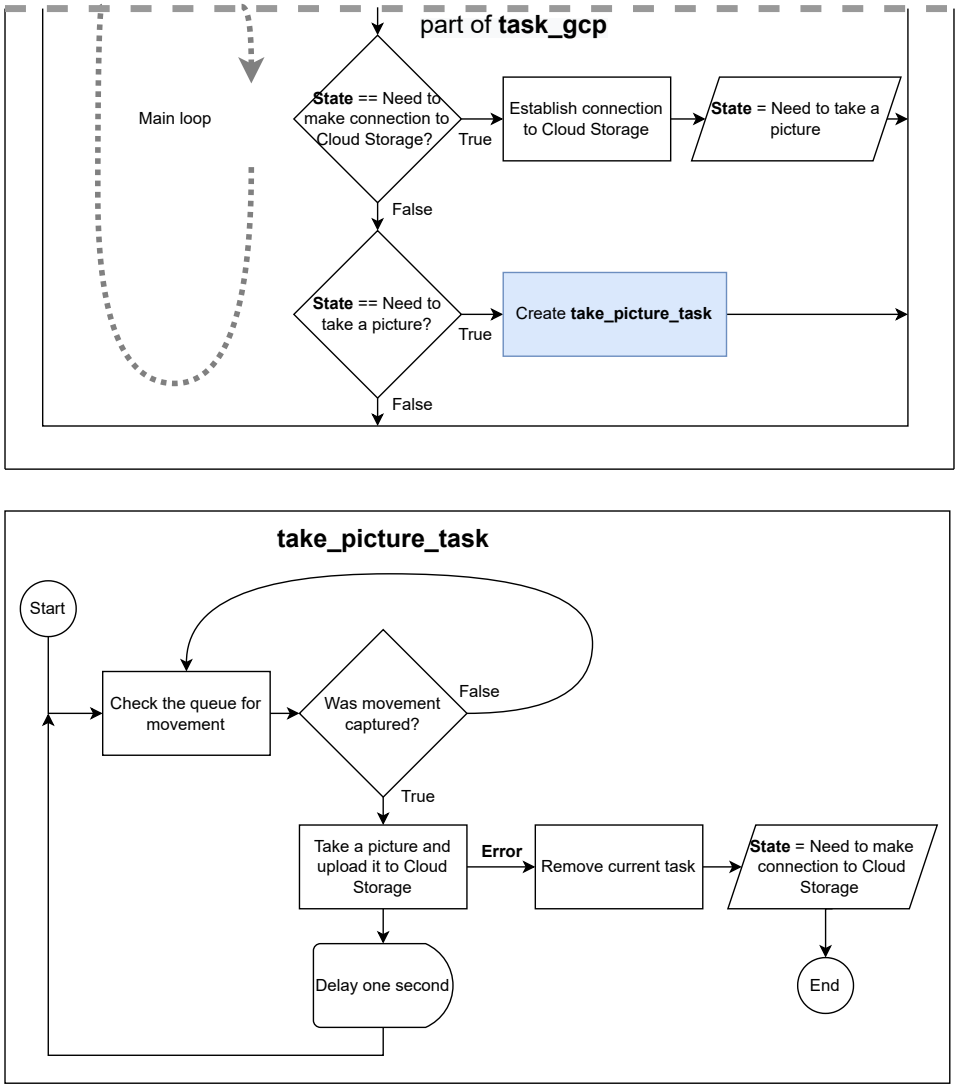


Figure 3.4: Flowchart of the current application

3.2 Google Cloud Platform (GCP)

Section 2.2 describes what GCP and some of its services are. In the security system, this thesis is about, the GCP acts as an intermediate between the monitoring node and the mobile phone. It stores photos

from the monitoring node, which it then makes available to the mobile phone and, at the same time, informs the mobile phone in the event of a newly encountered photo. GCP services used in this security system are:

- Cloud Storage is a data storage service that stores photos taken by the monitoring node. The mobile phone then obtains the photo from it.
- Cloud Functions is a service that executes code that starts executing based on some event. If the monitoring node uploads a new photo to Cloud Storage, Cloud Functions will run a code that will send a message to all user's mobile phones of this security system.
- Firestore is a NoSQL database storing all mobile users' tokens from the Firebase Cloud Messaging (FCM) service[2] to directly send messages to them.

3.2.1 Set up of Cloud Storage

Using Firebase wizard to create a GCP project will automatically configure Cloud Storage with a bucket to store data. For testing purposes, it was appropriate to create the bucket in the us-east1, us-west1, and us-central1 regions, which are the only ones that offer the possibility of use, within limits, for free [51].

The monitoring node uses the bucket's and project's names to access it.

3.2.2 Set up of Cloud Functions

When creating a new Cloud Function, what is its trigger is immediately specified. The purpose of this function is to inform mobile users that there will be movement in the monitored object. As soon as movement occurs, the monitoring node takes a photo and uploads it to Cloud Storage. This action - uploading a new photo to Cloud Storage - is used as a trigger to start the function. The function itself is written in Python 3.7 and works as follows:

2. See https://firebase.flutter.dev/docs/messaging/overview/

1. Download all user tokens from the Firestore database to send a targeted message.
2. It will use the FCM service to send a message to the mobile phones with the given tokens.

3.2.3 Set up of Firestore

Firestore works in two modes: datastore and native mode [52]. Since the mobile application is accessing it, it was helpful to create it in native mode or convert it to this mode [53].

Limitations of GCP

Security was not considered when configuring GCP. So, for example, photos stored in Cloud Storage are accessible to everyone, and changing data in the Firestore database is also possible for everyone.

3.3 Mobile phone

The administrator uses the mobile phone to interact with the security system. He can view the photos of the household taken by the monitoring node via a mobile phone using a remote cloud. Also, he receives a notification if the monitoring device detects movement. The administrator must use the mobile application to interact directly with the security system.

3.3.1 Set up of mobile application

As a development tool for writing and analyzing the source code, the Android Studio IDE is used. It is free, built on JetBrains' IntelliJ IDEA[3], and officially supported by Google [54]. Another necessary thing for developing Flutter applications is installing the Flutter SDK. The procedure is available [4]. See Section 2.3 for more details on Flutter.

The application itself is straightforward - the main page contains a list of images and provides help in the upper right corner. When the

3. See https://www.jetbrains.com/idea/
4. See https://docs.flutter.dev/get-started/install

user turns on the application, all images from Cloud Storage will be displayed automatically. The user can manually refresh the images by pulling the entire list down. To delete individual images, the user can swipe them away.

In addition, the application allows the user to receive a notification that a new photo has been uploaded to Cloud Storage and, therefore, receive information about the newly detected movement. The display of the notification differs if the application runs in the background or the foreground. If it runs in the background, a classic notification will be displayed (similar to a notification alerting to a new message). If it is in the foreground, this notification is no longer displayed; instead, the application generates a smaller label informing about the same. To receive notifications reliably, these notifications are targeted at so-called tokens. During the launch phase, the application obtains this token and immediately saves it to the Firestore database. Cloud Functions takes these tokens from this database and sends notifications to mobile phones using these tokens when needed.

The functioning of the mobile application within the entire security system describes following two simplified sequence diagrams:

- Figure 3.5 Describes the scenario when the user turns on the application and browses photos.

- Figure 3.6 Describes the scenario when the monitoring device detects movement and thus gradually informs the user.

Limitations of mobile application

The major limitation of this mobile application is that it has to run on a device with Android 4.4 and newer. Another limitation is a long time (several seconds) needed to get all photos and the need to refresh the list to see the new ones manually. While the testing phase, the bug prevented displaying a list of photos if there were too many of them (about 50+).

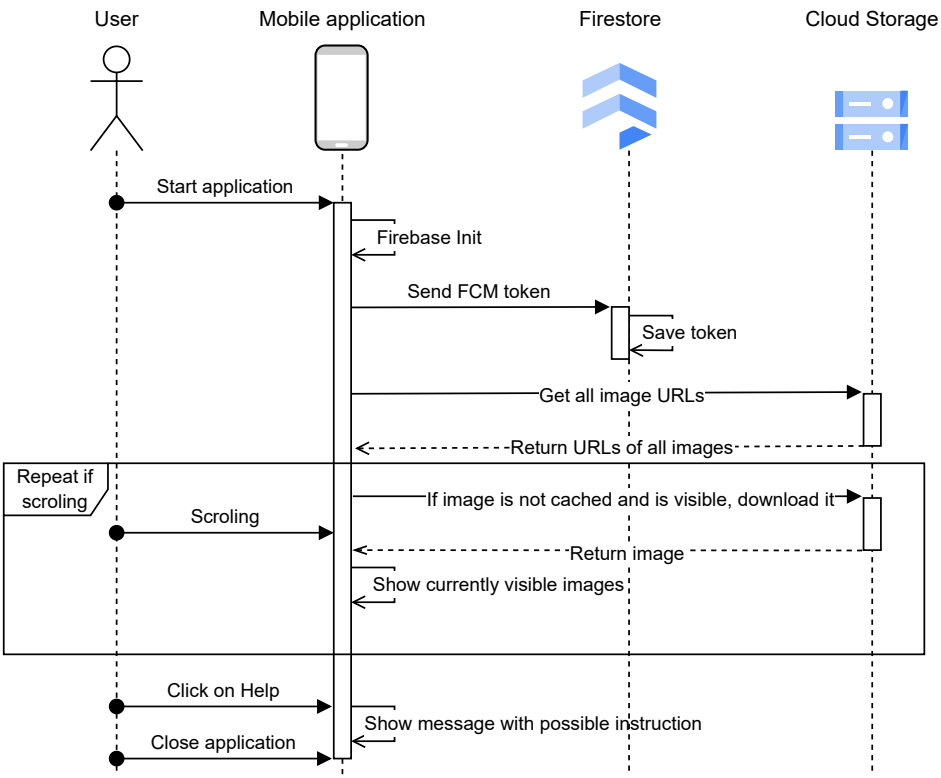


Figure 3.5: Sequence diagram of application startup and browsing the photos

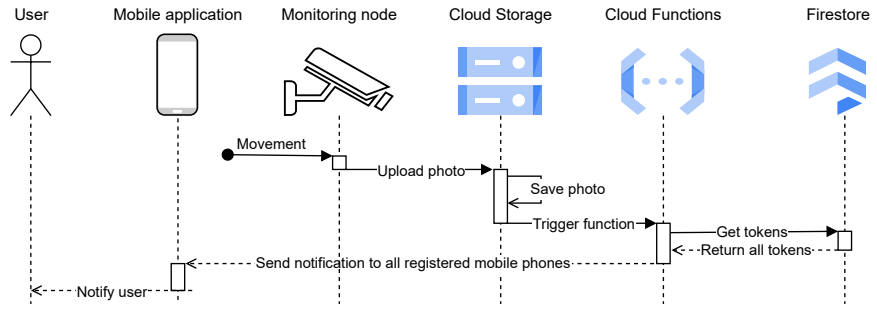


Figure 3.6: Sequence diagram of detecting movement

4 Measurements and analysis

This chapter describes the measured values of the assembled security system. Part of the results is also a description of how the measurement took place.

Test environment setup

Each operation is measured ten times in total. Five times when the monitoring node is in direct line of sight with the Wi-Fi router at a distance of fewer than 1m and five times when there is no direct line of sight, and the distance between the monitoring node and the Wi-Fi router is approximately 10m. The power supply is provided through the USB to Serial converter connected via USB to the computer.

4.1 Duration of operations

This section shows the duration of some operations performed by the monitoring node. Measured operations include:

- Load Time: describes how long the monitoring node is powering to its active state[1].
- Reload Time: describes how long it takes for the monitoring node to reconnect to Wi-Fi, which was previously turned off, to be again in an active state.
- Notify time: describes how long it takes to notify the user of the application from the time the monitoring node detects movement. Precisely the time between movement detection on the monitoring node and receiving notification on mobile phone with screen off.

1. The active state is the monitoring node's state where the monitoring node is detecting movement and is ready to take and upload a photo. It starts with the creation of the take_picture_task.

Table 4.1: The time required to power on the monitoring node to the active state (in seconds)

In sight with Wi-Fi router	28,8	33,8	28,1	27,9	28,7
Not in sight with Wi-Fi router	27,9	29,5	34,9	27,3	28,5

Table 4.2: The time required to turn on the monitoring node to the active state in case of Wi-Fi connection loss (in seconds)

In sight with Wi-Fi router	13,1	14,2	14,2	14,8	13,6
Not in sight with Wi-Fi router	14,4	16,9	16,7	14,3	19,6

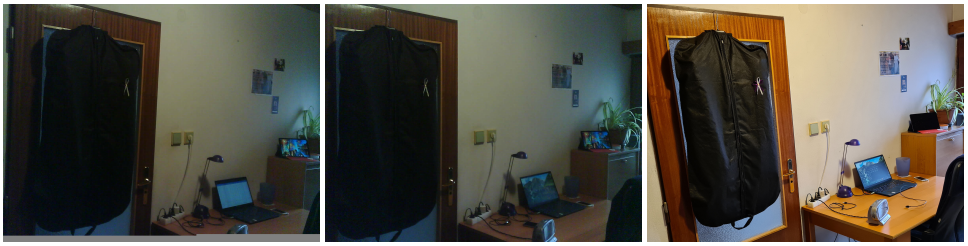
Table 4.3: The time required to notify the user when motion is detected (in seconds)

In sight with Wi-Fi router	4,3	3,5	3,2	3,7	2,9
Not in sight with Wi-Fi router	23,5	4,3	4,3	3,9	4,1

4.2 Picture quality

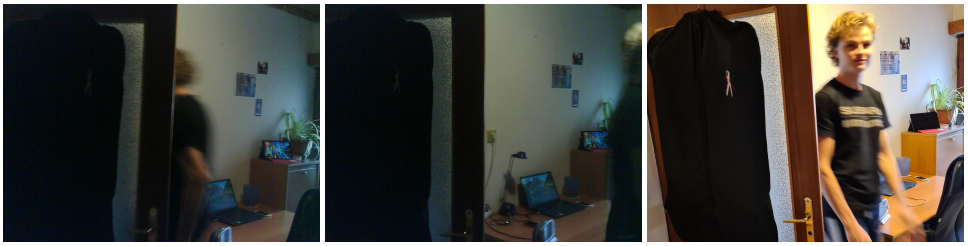
Because this security system is trying to cost as little as possible, it is necessary to know if the built-in camera is applicable. This section shows the quality of pictures taken by the monitoring node, compared to pictures taken by mobile phone. As mobile phone is used Samsung Galaxy S20 [55]. The example pictures from the monitoring node are the actual data sent to the Cloud Storage after successful motion detection.

4.2.1 Good light condition



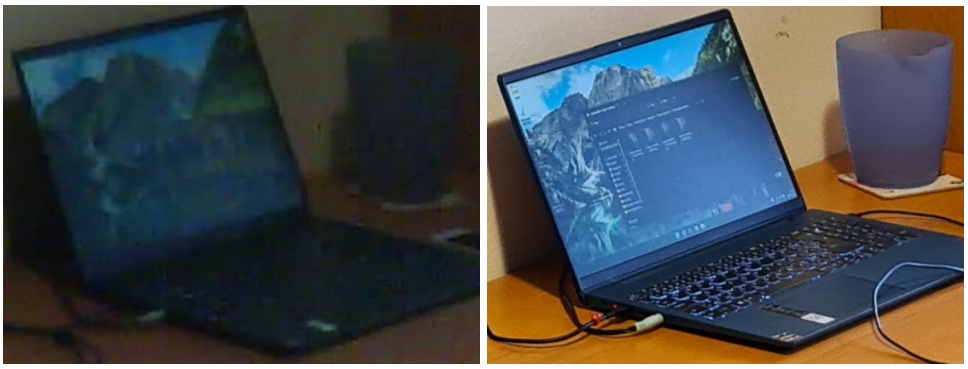
(a) Worse connection **(b)** Good connection **(c)** Mobile phone

Figure 4.1: Example photos of the static scene in good light condition



(a) Worse connection **(b)** Good connection **(c)** Mobile phone

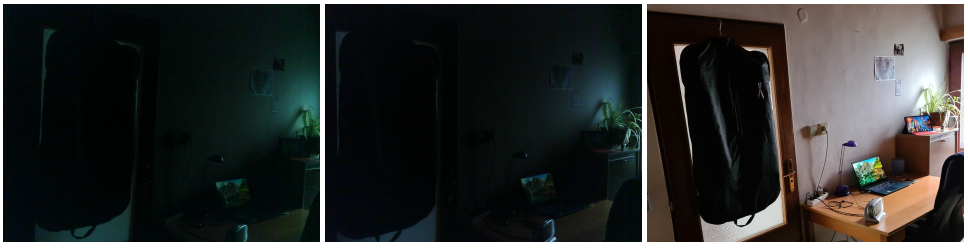
Figure 4.2: Example photos of the scene with movement in good light condition



(a) Monitoring node **(b)** Mobile phone

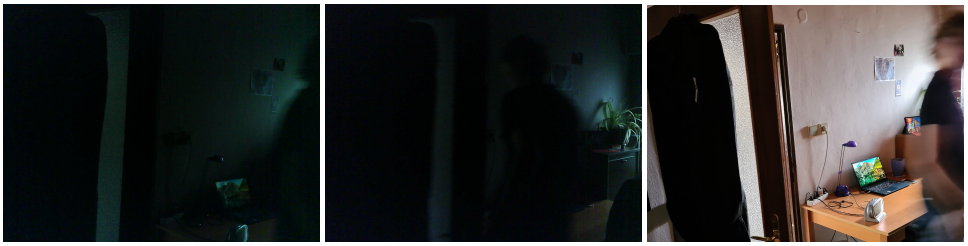
Figure 4.3: Comparison of details in good light condition

4.2.2 Lousy light condition



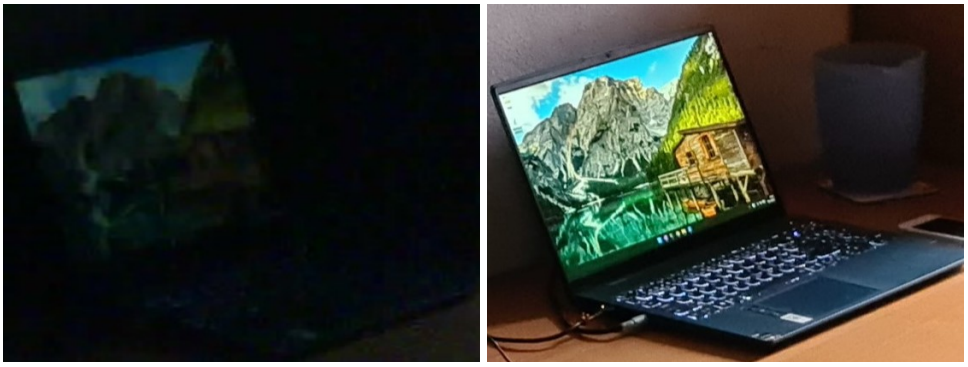
(a) Worse connection **(b)** Good connection **(c)** Mobile phone

Figure 4.4: Example photos of the static scene in lousy light condition



(a) Worse connection **(b)** Good connection **(c)** Mobile phone

Figure 4.5: Example photos of the scene with movement in lousy light condition



(a) Worse connection **(b)** Good connection

Figure 4.6: Comparison of details in lousy light condition

It is visible that there is a significant gap in the quality between the phone camera and the monitoring node camera. As the light conditions deteriorate, the camera result becomes unusable; therefore, any security depends only on the correct movement detection by the PIR sensor. This problem could be eliminated by automatically turning on the light simultaneously when motion is detected. This feature is, however, not implemented.

5 Conclusion

This thesis began by describing the issue of home security and the advantages and disadvantages of currently existing solutions. A possible disadvantage of the current solutions was the higher price, which the solution of this thesis tried to face. So the thesis described creating a home security system based on three main parts. The first part is the monitoring node, i.e., the physical device in the household, which monitors the household as needed. It is based on the low-cost ESP32-CAM AI-Thinker development board with a built-in camera and an added motion sensor. The second part is an intermediary between the first and third part, being on the Internet, and keeps the photos from the monitoring node safe and, at the same time, informs the third part, i.e., the mobile application, and allows it to view these photos. Google Cloud Platform mediates this part. Finally, the third part is the already mentioned, very simple mobile application built on the Flutter framework, which has recently been growing in popularity.

Together with other tools and terms used, these parts are first described theoretically. The following is a description of their incorporation in the assembled security system with a specific implementation and a list of disadvantages that this specific implementation contains. The penultimate chapter shows actual measured data of the assembled security system, such as the duration of some operations or the quality of photos taken by the monitoring node. Finally, this chapter summarizes everything, returns to the state-of-the-art solutions with comparison, and describes the assembled system's possible future expansion.

5.1 Comparing to the state-of-the-art solutions

The current implementation has minimal functionality and security and is, therefore, unable to compete with state-of-the-art solutions. Nevertheless, what Table 4.3 showed is that it offers a relatively fast reaction time in case of motion detection (less than four seconds in case of a good Wi-Fi connection). Overall, it offers the basis for further expansion to build a tailored, customized security system.

5.2 Further work and possible expandability

5.2.1 Improving security and user-friendliness

This is one of the most needed upgrades before all the other ones. Currently, the taken photos are publicly accessible, and the same applies to the Firestore database. This issue can be solved by restricting access to the Cloud Storage and Firestore database only to users who first authorize themselves in the mobile application.

For better user-friendliness, enabling a straightforward configuration of the monitoring node's access data to the Wi-Fi network and Cloud Storage is necessary. It could be done by adding an initialization state in the monitoring node, during which it connects to the mobile phone. In addition, the mobile application will have a configuration screen added.

5.2.2 Adding access control using an RFID reader

As the title of the bachelor's thesis suggests, the current version was initially supposed to be able to control access to the household. However, that did not happen, so this section describes this possible extension.

One possible implementation might be to purchase an RFID reader [56] and use a library that allows reading serial numbers of the RFID tags [57]. In the new screen of the mobile application, the user can set a list of allowed serial numbers, and the application will advertise this list to Firestore.

The monitoring node would roughly change as follows: Regular downloading of the list of allowed RFID tags would be added to the main task_gcp. In addition, a new task would be added, which the monitoring node executes only if the user taps a tag to the reader. This task verifies whether it is an allowed tag and activates the pin to which the door lock will be connected. This extension could either work on the already used ESP32-CAM AI-Thinker development board, or the access control device could be independently based on a much cheaper ESP8266 SoC [58] with, for example, this [59] library used.

5.2.3 Powered by batteries

Among the main advantages of the ESP32 SoC is the ability to go to sleep, thereby reducing power consumption and extending the end device's battery life. The so-called deep sleep, in which the ESP32 only operates the internal Ultra Low Power Coprocessor, consumes merely 0.15 mA [60]. In this state, it can still react to external events, such as motion detection using a PIR sensor, by waking up the SoC and performing the relevant operation. In combination with a GSM module, it can be a desirable solution for places without a Wi-Fi signal or an electrical network within reach.

5.2.4 Adding another sensors

If some pins remain free, it is possible to occupy them with additional sensors, thus expanding home security possibilities. An example of another sensor might be the MQ2 Gas sensor for detecting combustible gas, more specifically LPG, Propane, and Hydrogen [61].

A The content of IS MU archive

The IS MU archive contains the implementation part of this thesis. It is split into the following structure of three folders:

- *ESP32* - contains two folders, first representing the whole project called *repository* and second *source_codes* showing the source codes of the application.
- *Cloud Functions* - contains the source code used for the cloud function and the file containing needed dependencies.
- *Mobile application* - contains *repository* folder with all the files used in the Flutter project and second folder *source_codes* limited to only files with source codes.

Bibliography

1. *Internet of things* [online]. Wikipedia, the free encyclopedia, 2022-11-16 [visited on 2022-11-18]. Available from: https://en.wikipedia.org/w/index.php?title=Internet_of_things&oldid=1122269037.
2. DAHLQVIST, Fredrik; PATEL, Mark; RAJKO, Alexander; SHULMAN, Jonathan. *Growing opportunities in the Internet of Things* [online]. McKinsey & Company, 2019-07-22 [visited on 2021-11-11]. Available from: https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things.
3. OSCAR, Silver. *An Indoor Localization System Based on BLE Mesh Network* [online]. 2016 [visited on 2021-11-12]. Available from: https://liu.diva-portal.org/smash/get/diva2:935745/FULLTEXT01.pdf. MA thesis. Linköping University.
4. *Smart Bins in Dublin* [online]. Smart Dublin, 2016-01-25 [visited on 2021-11-11]. Available from: https://smartdublin.ie/smart-bins/.
5. *Meet the family of Nest Cams.* [Online]. Google Store [visited on 2022-04-05]. Available from: https://store.google.com/us/category/nest_cams.
6. *Home security starts at the front door.* [Online]. Google Store [visited on 2022-04-05]. Available from: https://store.google.com/us/category/nest_doorbells.
7. *How Nest cameras automatically switch to Home or Away* [online]. Google Nest Help [visited on 2022-04-05]. Available from: https://support.google.com/googlenest/answer/9246474.
8. *Nest Protect smoke and CO alarm* [online]. Google Store [visited on 2022-04-05]. Available from: https://store.google.com/us/product/nest_protect_2nd_gen.
9. *Nest Cam (Battery) - Google Store* [online]. Google Store [visited on 2022-12-12]. Available from: https://store.google.com/gb/product/nest_cam_battery?hl=en-GB.

10. *Nest Doorbell (wired) - Google Store* [online]. Google Store [visited on 2022-12-12]. Available from: https://store.google.com/gb/product/nest_doorbell_wired?hl=en-GB.

11. *Zigbee* [online]. Wikipedia, the free encyclopedia, 2022-04-04 [visited on 2022-04-05]. Available from: https://en.wikipedia.org/w/index.php?title=Zigbee&oldid=1080982688.

12. *Zigbee* [online]. Wikipedia, the free encyclopedia, 2022-11-21 [visited on 2022-11-22]. Available from: https://en.wikipedia.org/w/index.php?title=Zigbee&oldid=1123040166#Use_cases.

13. *Aqara Smart Door Lock N100* [online]. Aqara [visited on 2022-04-05]. Available from: https://www.aqara.com/en/smart_door_lock_n100.html.

14. *ESP32-CAM* [online]. Ai-Thinker Co. [visited on 2022-11-14]. Available from: http://www.ai-thinker.com/pro_view-24.html.

15. *System on a chip* [online]. Wikipedia, the free encyclopedia, 2022-11-13 [visited on 2022-11-14]. Available from: https://en.wikipedia.org/w/index.php?title=System_on_a_chip&oldid=1121653808.

16. *ESP32* [online]. Wikipedia, the free encyclopedia, 2022-11-07 [visited on 2022-11-22]. Available from: https://en.wikipedia.org/w/index.php?title=ESP32&oldid=1120566063.

17. *ESP32* [online]. Wikipedia, the free encyclopedia, 2022-11-07 [visited on 2022-11-22]. Available from: https://en.wikipedia.org/w/index.php?title=ESP32&oldid=1120566063#Features.

18. *OV2640 – Specs, Datasheets, Cameras, Features, Alternatives* [online]. Arducam [visited on 2022-11-22]. Available from: https://www.arducam.com/ov2640/.

19. *ESP32-CAM AI-Thinker Board – All about GPIO Pins* [online]. Microcontrollerslab.com [visited on 2022-11-22]. Available from: https://microcontrollerslab.com/esp32-cam-ai-thinker-pinout-gpio-pins-features-how-to-program/.

20. *Google Cloud Platform* [online]. Wikipedia, the free encyclopedia, 2022-11-23 [visited on 2022-11-24]. Available from: https://en.wikipedia.org/w/index.php?title=Google_Cloud_Platform&oldid=1123377072.

21. *Google Cloud Service Health* [online]. Google [visited on 2022-11-24]. Available from: https://status.cloud.google.com/index.html.

22. *Firebase Pricing* [online]. Google [visited on 2022-11-24]. Available from: https://firebase.google.com/pricing.

23. MCDONALD, Paul. *Introducing Google App Engine + our new blog* [online]. Google App Engine Blog, 2008-04-07 [visited on 2022-11-28]. Available from: http://googleappengine.blogspot.com/2008/04/introducing-google-app-engine-our-new.html.

24. *Firebase & Google Cloud* [online]. Google [visited on 2022-11-24]. Available from: https://firebase.google.com/firebase-and-gcp.

25. *Firebase* [online]. Google [visited on 2022-11-24]. Available from: https://firebase.google.com/.

26. *Google Cloud console* [online]. Google [visited on 2022-11-28]. Available from: https://cloud.google.com/cloud-console.

27. *gcloud CLI overview* [online]. Google [visited on 2022-11-28]. Available from: https://cloud.google.com/sdk/gcloud.

28. *What is Cloud Storage?* [Online]. Google [visited on 2022-11-24]. Available from: https://cloud.google.com/storage/docs/introduction.

29. *About Cloud Storage buckets | Buckets* [online]. Google [visited on 2022-11-28]. Available from: https://cloud.google.com/storage/docs/buckets#buckets.

30. *Quotas & limits | Buckets* [online]. Google [visited on 2022-11-28]. Available from: https://cloud.google.com/storage/quotas#buckets.

31. *What is Cloud Storage? | Securing your data* [online]. Google [visited on 2022-11-24]. Available from: https://cloud.google.com/storage/docs/introduction#securing_your_data.

32. *What is Cloud Firestore?* [Online]. Back4App [visited on 2022-11-25]. Available from: https://blog.back4app.com/what-is-cloud-firestore/.

33. *Firestore | Firebase | How does it work?* [Online]. Google [visited on 2022-11-25]. Available from: https://firebase.google.com/docs/firestore#how_does_it_work.

34. *Firestore | Firebase* [online]. Google [visited on 2022-11-24]. Available from: https://firebase.google.com/docs/firestore#key_capabilities.

35. *Cloud Functions | Google Cloud* [online]. Google [visited on 2022-11-24]. Available from: https://cloud.google.com/functions.

36. KOCHMAŃSKI, Michał. *Flutter or Xamarin For Mobile Development in 2022? 8 Reasons For Choosing Flutter* [online]. Monterail sp. z o.o, 2022-04-07 [visited on 2022-11-25]. Available from: https://www.monterail.com/blog/flutter-vs-xamarin.

37. *Flutter (software)* [online]. Wikipedia, the free encyclopedia, 2022-11-22 [visited on 2022-11-25]. Available from: https://en.wikipedia.org/w/index.php?title=Flutter_(software)&oldid=1123197779.

38. *Stack Overflow Trends* [online]. Stack Exchange Inc [visited on 2022-11-25]. Available from: https://insights.stackoverflow.com/trends?tags=flutter%5C%2Creact-native%5C%2Cxamarin%5C%2Ckotlin%5C%2Cswift.

39. KARASAVVAS, Theodoros 'Theo'. *Why Flutter is the most popular cross-platform mobile SDK* [online]. Stack Overflow, 2022-02-21 [visited on 2022-11-25]. Available from: https://stackoverflow.blog/2022/02/21/why-flutter-is-the-most-popular-cross-platform-mobile-sdk/.

40. *What is PlatformIO? — PlatformIO latest documentation* [online]. PlatformIO [visited on 2022-11-25]. Available from: https://docs.platformio.org/en/latest/what-is-platformio.html.

41. *What is PlatformIO? — PlatformIO latest documentation | Problematic* [online]. PlatformIO [visited on 2022-11-25]. Available from: https://docs.platformio.org/en/latest/what-is-platformio.html#problematic.

42. *Professional collaborative platform for embedded development* [online]. PlatformIO [visited on 2022-11-25]. Available from: https://platformio.org/.

43. *Debugging — PlatformIO latest documentation* [online]. PlatformIO [visited on 2022-11-25]. Available from: https://docs.platformio.org/en/latest/plus/debugging.html.

44. *Unit Testing — PlatformIO latest documentation* [online]. PlatformIO [visited on 2022-11-25]. Available from: https://docs.platformio.org/en/latest/advanced/unit-testing/index.html.

45. *Static Code Analysis — PlatformIO latest documentation* [online]. PlatformIO [visited on 2022-11-25]. Available from: https://docs.platformio.org/en/latest/advanced/static-code-analysis/index.html.

46. *AM312 datasheet* [online]. Alldatasheet [visited on 2022-12-09]. Available from: https://html.alldatasheet.com/html-pdf/1179499/ETC2/AM312/109/1/AM312.html.

47. *Welcome to Fritzing* [online]. Fritzing [visited on 2022-12-09]. Available from: https://fritzing.org/.

48. *Documentation for Visual Studio Code* [online]. Microsoft [visited on 2022-11-28]. Available from: https://code.visualstudio.com/docs.

49. BRANDÃO, Raphael. *Example google_storage* [online]. GitHub, 2022-07-20 [visited on 2022-01-12]. Available from: https://github.com/raphaelbs/esp32-cam-ai-thinker/tree/master/examples/google_storage.

50. BRANDÃO, Raphael. *Example google_storage | Steps* [online]. GitHub, 2022-07-20 [visited on 2022-01-12]. Available from: https://github.com/raphaelbs/esp32-cam-ai-thinker/tree/master/examples/google_storage.

51. *Free cloud features and trial offer | Google Cloud Free Program | Storage* [online]. Google, 2022-11-30 [visited on 2022-12-02]. Available from: https://cloud.google.com/free/docs/free-cloud-features#storage.

52. *Choosing between Native mode and Datastore mode | Cloud Datastore Documentation | Google Cloud* [online]. Google, 2022-11-30 [visited on 2022-12-02]. Available from: https://cloud.google.com/datastore/docs/firestore-or-datastore.

53. *Changing between Native mode and Datastore mode | Cloud Datastore Documentation | Google Cloud* [online]. Google, 2022-11-30 [visited on 2022-12-02]. Available from: https://cloud.google.com/datastore/docs/firestore-or-datastore#changing%7B%5C_%7Dbetween%7B%5C_%7Dnative%7B%5C_%7Dmode%7B%5C_%7Dand%7B%5C_%7Ddatastore%7B%5C_%7Dmode.

54. *Android Studio* [online]. Wikipedia, the free encyclopedia, 2022-11-19 [visited on 2022-12-02]. Available from: https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=1122711818.

55. *Samsung Galaxy S20 - Full phone specifications* [online]. GSMArena [visited on 2022-12-09]. Available from: https://www.gsmarena.com/samsung_galaxy_s20-10081.php.

56. *MFRC522 Standard performance MIFARE and NTAG frontend* [online]. NXP Semiconductors N.V., 2022-04-27 [visited on 2022-12-12]. Available from: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

57. BOBIJA, Alija. *MFRC522 library* [online]. Github, 2022-08-26 [visited on 2022-12-12]. Available from: https://github.com/abobija/esp-idf-rc522.

58. *ESP8266 Wi-Fi MCU I Espressif Systems* [online]. Espressif Systems [visited on 2022-12-12]. Available from: https://www.espressif.com/en/products/socs/esp8266.

59. BALBOA, Miguel André. *MFRC522* [online]. Github, 2022-06-06 [visited on 2022-12-12]. Available from: https://github.com/miguelbalboa/rfid.

60. *ESP8266 Wi-Fi MCU I Espressif Systems* [online]. Espressif Systems [visited on 2022-12-12]. Available from: https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/.

61. *MQ-2 Semiconductor Sensor for Combustible Gas* [online]. Pololu [visited on 2022-12-12]. Available from: https://www.pololu.com/file/0J309/MQ2.pdf.