IoT Based Door Lock System With ESP32, RFID Sensor, and Camera

Javier Ryan Setiadi   
Computer Science and Mathematics  
2702232145Binus University, Jakarta  
[javier.setiadi@binus.ac.id](mailto:javier.setiadi@binus.ac.id)

Johanes Cedrick Wijaya   
Computer Science and Mathematics  
2702213801Binus University, Jakarta  
[johanes.wijaya002@binus.ac.id](mailto:johanes.wijaya002@binus.ac.id)

Marco Linardi   
Computer Science and Mathematics  
2702223046Binus University, Jakarta  
[marco.linardi@binus.ac.id](mailto:marco.linardi@binus.ac.id)

Villyan Sutanto   
Computer Science and Mathematics  
2702291311Binus University, Jakarta  
[villyan.sutanto@binus.ac.id](mailto:villyan.sutanto@binus.ac.id)

*Abstract*—Home security is an essential aspect of creating a safe living environment. In this modern world, technology still struggles to give an affordable and responsive smart security system. Conventional methods such as manual keys and passive cameras often fail to provide real-time security and response, which fails as well to provide sufficient evidence for legal cases. According to FBI data, the burglary rate in 2023 was 250.7 per 100,000 inhabitants—a significant yet still concerning number. This research aims to give the solution of which, giving a cost-effective, user-friendly, smart security system with the help of ESP32 microcontroller, RFID reader, and a camera. The system provides real time evidence collection, provides user interaction through a supporting mobile application that stores all the data which is collected by the ESP32 system. This solution also supports the United Nations Sustainable Development Goal (SDG) 11: Sustainable Cities and Communities, by promoting accessible and secure smart home technology.

Keywords—IoT, door lock, RFID, ESP32, camera, SDG 11

# Introduction

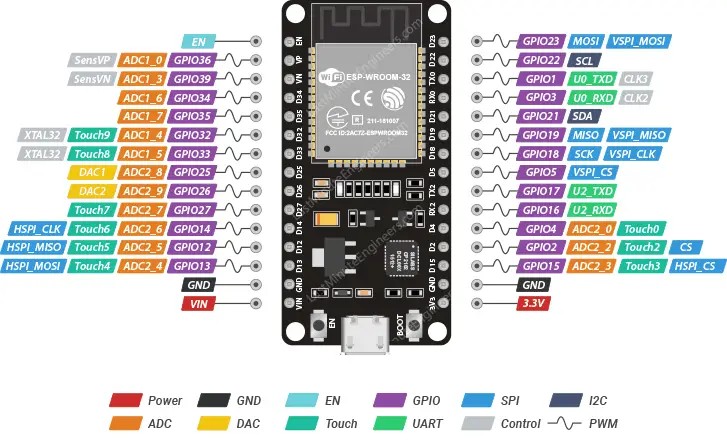
Safety is one of the key things needed for a comfortable home. In the modern age, the needs of a good home security systems which are smart and responsive are scarce or expensive. A lot of past cases involving home burglary that happened without any real identification became a struggle in the conventional home security system such as manual keys or passive cameras. Data shown from the FBI reports implies the burglary rate in 2023 was 250.7 per 100,000 inhabitants, representing a decrease of 22.7 compared to 2022 [1]. This number is still quite high even though lots of new security technology have been issued for the past decade and the police are actively trying to stop this problem. Lack of technology also became a problem in sighting probable reasons to win in court decisions [2].

The aim of this research is to figure out the solution of the problem, lack of evidence towards burglary, and lesser the rate of the burglary itself. In which, develops a cost effective, easy to use, home security system that can collect evidence with cameras. This research is also aimed for reaching the Sustainable Development Goals (SDG) number 11, of which is “Sustainable Cities and Communities”. The usage of ESP32 with the help of RFID system and camera will help to solve problems such as giving a more reliable environment, safe home security, and broad access for consumer’s utility. Conventional RFID might be challenging to use, because of is complicated setups and operations [3]. ESP32 will help with the complications of RFID, with simple wiring and implemental codes in the backend. The system that is used in this paper provides simple use of hardware with ESP32 in hand, an application that supports the hardware and can give notifications, and a database of which the notifications are stored. Hence, this system provides two sided interaction between the user and the hardware with the use of an application.

# Hardware Components

## ESP32

The brains of our internet of things (IoT) operation is based on a microcontroller called ESP32. ESP32 offers a vast features that is compatible to many other sensors and computers. Some of those features includes a dual core processor, Wi-Fi (802.11 b/g/n protocol) and Bluetooth (Bluetooth Low Energy or BLE) connectivity, GPIO pins which can be used for general input/output, and all that consumes low power [4]. The dual core processor is exceptionally powerful in its appliances such as artificial intelligence usage, machine learning, or as a subtle microcontroller that handles a lot of sensor. This may be very useful with the 34 GPIO pins that it has, that also supports a lot of variety interfaces, such as SPI, I2C, UART, and PWM [5]. The operation of sensors, artificial intelligence, and machine learning can be programmed by various frameworks and languages. The most used programming language for ESP32 is C++ [4]. All of these features is specially designed to contribute towards IoT applications with the needs of either Wi-Fi or Bluetooth connectivity. Other appliable contributions include robotics and automations, audio and multimedia, and environmental monitoring [5].



*Picture 2.1 ESP32 Pinouts* [6]

## Modules and Sensors

1. OLED LCD (I2C 1.3 Inch)

OLED LCD is one of few components of the Arduino modules, where it can output multiple information of which it’s given. Data/information that is produced by the Arduino module such as texts, integers, or pixelated pictures, can be displayed by using OLED LCD. *Organic Light-Emitting Diode* (OLED) is perfect for this research because it doesn’t consume too much voltage for it to run correctly. This OLED LCD has 128 x 64 pixel of which, 128 columns and 64 rows. The normal font that is used in this LCD is 5x7 font. But if another type of font is being used, either it being larger or smaller, the number of characters that can be generated will varied based on the remaining pixels that are available [7]. The specification of OLED LCD 1.3 Inch is showed on the 2.1 table.

A small blue and white electronic device

AI-generated content may be incorrect.

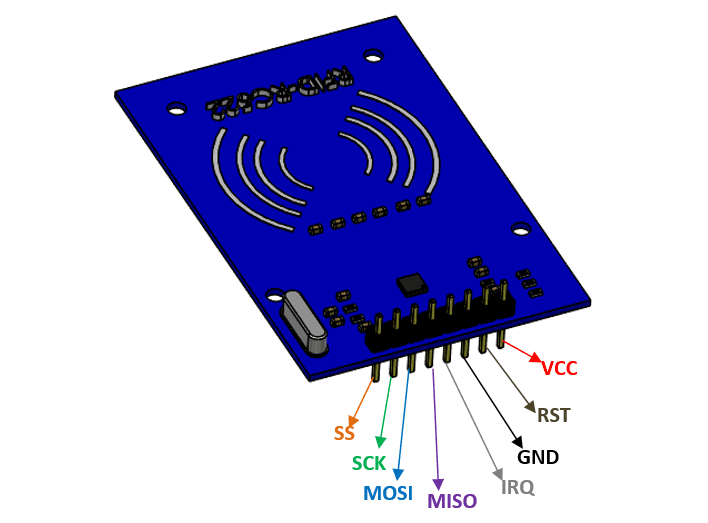
*Picture 2.2 OLED LCD I2C 1.3 Inch* [8]

|  |  |  |
| --- | --- | --- |
| No | Specification | Description |
| 1 | Size | 1.3 OLED |
| 2 | Interface | I2C |
| 3 | LCD Dimention | 36.2 x 31.3 x 1.2 mm (max) |
| 4 | Resolution | 128 x 64 Pixel |
| 5 | Active Area | 21.74 x 11.2 mm |
| 6 | Driver IC | SH1106 or SSD1306 |
| 7 | Display Color | Blue or White |
| 8 | Display Type | OLED or Negative or Transmissive |
| 9 | Operating Temp | -40°C to +85°C |
| 10 | Storage Temp | -45°C to +90°C |
| 11 | VCC | 2.2 – 5.5 Volts |

*Table 2.1 OLED LCD I2C 1.3 Inch Specification*

1. RFID Sensor (RC522)

RFID Sensor RC522 is a module that utilise radio frequencies as its main data. RFID stands for Radio Frequency Identification which identification method uses radio transmission frequencies that’s provided with RFID label or transponder to save and receive long ranged data [9]. The main logic behind the use of RFID is with the use of an item, preferably a tag or a card, which is filled with a transponder code that can be read by the RFID sensor. Examples of the code are often being an identification code, an ID code, or the user’s name. RFID sensor can check whether the code that was scanned is correct based on the database and can act base on this information. The use of RFID sensor can be found at numerous tasks such as libraries and bookstores, packet tracking, building access control, etc [10]. The specification of the RFID Sensor RC522 will be shown by the 2.2 table.



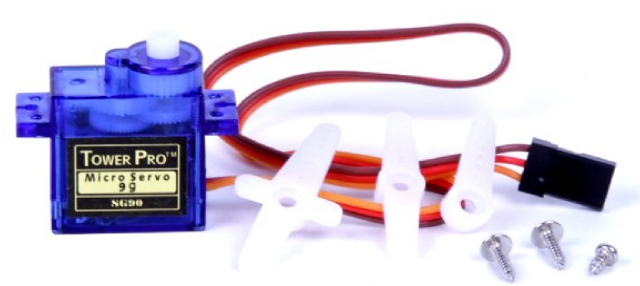
*Picture 2.3 RFID RF522 Sensor With Pinouts* [11]

|  |  |  |
| --- | --- | --- |
| No | Specification | Description |
| 1 | Frequency | 13.56 MHz |
| 2 | Voltage | 2.5V to 3.3V |
| 3 | Communication | SPI, I2C, UART |
| 4 | Data Rate | 10 Mbps |
| 5 | Read Range | 5 cm |
| 6 | Current | 13-26 mA |

*Table 2.2 RFID RF522 Specification*

1. Servo (SG90)

The SG90 servo a low cost, lightweight, efficient servo for any types of projects. Servo uses PWM or Pulse Width Modulation for it to be controlled correctly [12]. This signal is given to the servo by a microcontroller, either digitally or analogically [13]. The use of this servo can be modified based on the use of it, either it operates all day such as in a drone, or segmented as in a car wiper. This servo specifically doesn’t cover that much in range, it only covers 180 degrees (90 in each direction), so the servo’s arm doesn’t give a full rotation. The modulation of this servo also needs a quite substantial amount of voltage and current. Otherwise, it may not work correctly (jitters or not moving at all).



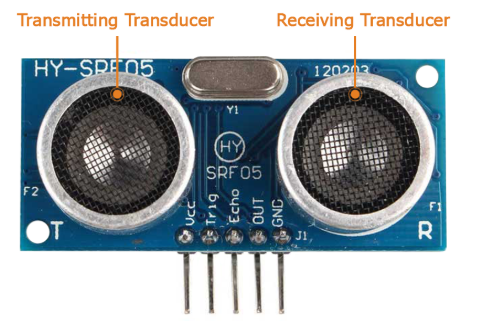
*Picture 2.4 Servo SG90 With 3 Arms* [14]

|  |  |  |
| --- | --- | --- |
| No | Specification | Description |
| 1 | Modulation | Analog |
| 2 | Torque | 4.8V: 25.00 oz-in (1.80 kg-cm) |
| 3 | Speed | 4.8V: 0.12 sec/60° |
| 4 | Weight | 0.32 oz (9.0 g) |
| 5 | Dimensions | Length: 0.91 in (23.0 mm) Width: 0.48 in (12.2 mm) Height: 1.14 in (29.0 mm) |
| 6 | Motor Type | 3-pole |
| 7 | Gear Type | Plastic |
| 8 | Rotation/Support | Bushing |
| 9 | Pulse Width | 500-2400 µs |
| 10 | Connector Type | JR |
| 11 | Current (idle) | 10 mA |
| 12 | Current (under weight) | 100 – 250 mA |
| 13 | Current (maximum) | 650 – 1000 mA |

*Table 2.3 Servo SG90 Specification Based On Servo Database* [15]

1. Ultrasonic Sensor (HY-SRF05)

Ultrasonic sensor is a sensor that uses ultrasonic waves to determine the distance between the sensor itself, and the following target. The distance that is generated is quite accurate despite multiple variables can be accounted such as colour, different lightings, and obstacles. The ultrasonic sensor which is provided as an Arduino module, comes in 2 series, HC-SR04 and HY-SRF05. HY-SRF05 is an upgraded version of the HC-SR04 where it gives larger ultrasonic range (2 cm – 400 cm), sensor range of 4 meters (HC-SR04 only provides 3 meters), range accuracy can reach 3mm, and has more pins which is more useful for even more accurate data generation [16]. The module works by using I/O trigger for at least 10 ultrasonic high level signal, then it sends 8 40kHz signals to detect whether there is a pulse signal in response, if there is a signal, with the high level signal returning, the time also can be measured from the high output IO duration of sending the ultrasonic signal. This returning data is then processed by using the formula:



*Picture 2.5 Ultrasonic Sensor HY-SRF05* [17]

|  |  |  |
| --- | --- | --- |
| No | Specification | Description |
| 1 | Trigger Pin Format | 10 uS Digital Pulse |
| 2 | Sound Frequency | 10 kHz |
| 3 | Echo Pin Output | 0 VCC |
| 4 | Echo Pin Output | Test Distance (formula above) |
| 5 | Measurement Range | 2 cm to 4.5 m |
| 6 | Measurement Resolution | 0.3 cm |
| 7 | Measurement Angle | Up to 15 degree |
| 8 | Measurement Rate | 40 Hz |
| 9 | Supply Voltage | 4.5 V to 5.5 V |
| 10 | Supply Current | 10 mA to 40 mA |
| 11 | Connector | 5 pin male connectors |
| 12 | Static Current | < 2mA |

*Table 2.4 Ultrasonic Sensor HY-SRF05 Specifications* [16]

1. Active Buzzer

Active buzzer is a module that can output a buzzing noise which is perfect for alerting user in case of errors, correct attempts, and else. Buzzer are categorized into two, which are active buzzer and passive buzzer. The difference between these two buzzers are, when active buzzer is connected to a power source, it will immediately produce sound, but the passive buzzer will stay silent. The main idea behind a buzzer is, it produces sound just like humans talking. Vibrations that is caused by current being delivered to a coil inside of the buzzer, makes the coil vibrate, hence creating sound [18].

A close-up of a black device

AI-generated content may be incorrect.

*Picture 2.6 Passive Buzzer and Active Buzzer* [18]

## Arduino IDE

Arduino IDE is the main brains of this ESP32 operating system. ESP32 uses multiple programming languages, but with Arduino IDE, we use C++ as the main programming language. Arduino IDE provides useful libraries that can be used to support multiple codes, essentially supporting a couple different ESP32’s with different developer and manufacturers as well. The libraries that are supported by Arduino IDE provide additional functionality that enables easy integration of various sensors, actuators, and components [4]. Such libraries that are used in this research are listed on the table below.

|  |  |  |
| --- | --- | --- |
| No | Library | Description |
| 1 | Wifi.h | Provides Wi-Fi connectivity functions for ESP32. |
| 2 | WiFiClientSecure.h | Enables HTTPS (secure HTTP) connections using SSL/TLS. |
| 3 | ESPAsyncWebServer.h | Asynchronous web server for ESP32. |
| 4 | HTTPClient.h | Enables sending HTTP requests (GET, POST, etc.) over Wi-Fi. |
| 5 | esp\_camera.h | Enables image capture with ESP32-CAM module. |
| 6 | ArduinoJson.h | Efficient JSON parsing and generation for constrained devices. |
| 7 | base64.h | Provides functions to encode/decode Base64 strings. |
| 8 | Wire.h | I2C communication support (used with I2C sensors/displays). |
| 9 | U8g2lib.h | OLED display library supports many controllers (e.g., SH1106, SSD1306). |
| 10 | SPI.h | SPI communication supports high-speed peripheral communication. |
| 11 | MFRC522.h | Library for MFRC522 RFID module over SPI. |
| 12 | Arduino.h | Core Arduino functions and macros (included automatically in most sketches). |
| 13 | ESP32Servo.h | Enables servo motor control via PWM on ESP32. |
| 14 | Ticker.h | Lightweight timer to run tasks periodically without the function delay() |
| 15 | time.h | Provides time and date functions via NTP or manual setting. |

*Table 2.5 Libraries Used in The Project*

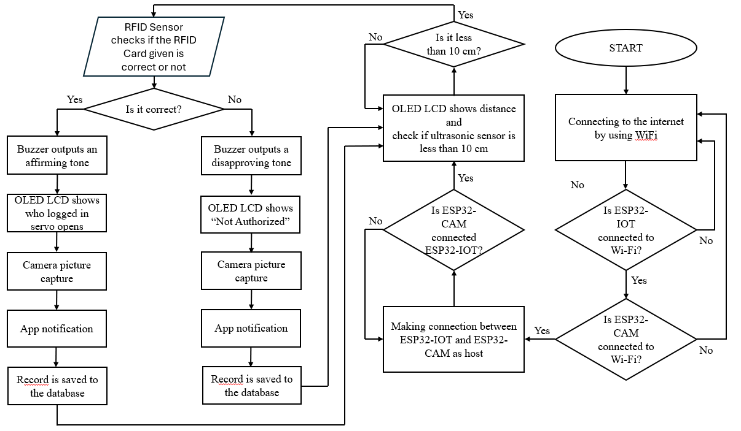
## Relation Between Components

The use of 2 ESP32s, ESP32 for IoT components, and ESP32-CAM is needed in this research. If either one is used, it isn’t powerful enough to sustain all the sensors that are needed for components to work correctly. The ESP32-CAM will be the host between the ESP32s so they can communicate with each other. The process can work because the ESP32-CAM will generate an IP address which can be connected to, and the ESP32 for IoT components will connect to the IP address directly.

The sensors that are used in this research work simultaneously. Firstly, when the ESP32s are connected to the internet, the ultrasonic sensor will start to scan its surroundings, searching for something that can move close enough and then activate the RFID scanner. If there’s an object that is less than 10cm to the ultrasonic sensor, buzzer will give a starting tone, and RFID sensor will be on standby if there’s a RFID card or tag that will be scanned. If not, then the ultrasonic sensor will stay on reading state repeatedly. Meanwhile all this happened, OLED LCD will always output the distance between the ultrasonic sensor to an object.

The RFID sensor will wait for an RFID card to be scanned. If there’s no card or tag that is read in 25 seconds, buzzer will give a long tone which identifies that there’s no card being read, and OLED LCD will output “Timeout! Try Again”. If there’s a card being read, then the system will check whether the card’s credentials are correct or not. In this research we use the cardholder’s name that is already stored within each card, to be the validation variable. If the cardholder’s name is verified, then the system will continue to open the door lock. Otherwise, the system will have the door lock still on close and continue to the next step.

The systematics to open the door lock is, if the RFID card is verified, buzzer will output an affirming tone with OLED LCD outputting who opened the door, for example “Hello, Marco Linardi”. Then the ESP32-CAM will capture a picture of the door opener, which is sent to the database, which the application took and displayed it in the application and gave a notification as well about who logged in, based on the RFID cardholder’s name. Then, after all those processes, the servo will turn to the open state, indicating the door lock is already opened. In contrast, when the RFID card isn’t verified, the buzzer will give a disapproving tone, and OLED LCD will outputs “Not Authorized”. The difference between before is, the servo will not open because it the user isn’t authorized. But the camera capture will still work because we want to know who tried to open the door lock. This information is sent to the database, and the application will take that information, display it on the app, and give a warning notification that someone is trying to open the door.



*Picture 2.7 Flowchart of The Components*

# Android App and Database

Another crucial part of this research is from behind the scenes. The android app and database needs to work in cohesion so the project can work properly. The database will store the information gained from the microcontrollers, and the application will display those information with a more interesting display, a gallery of pictures, whom have tried to enter the house, and a detailed information about whom have tried to enter the house as well.

## Android Studio

This research uses android studio as its main program to make the android app. Android studio is perfect for this research because the support that Google gave to this program makes it perfect for integrating it with other google program, which is Firebase, as our database. Android studio includes built in features such as APK packaging, emulator testing, layout previews, and version control integration which is very useful for excluding bugs, testing the app before exporting it, and this makes it perfect for android app development. Other built-in shoes are layout editors, which makes it easier for making UI designs, with drag and drops, APK analyzer, which helps to reduce bloat in the app itself, profiler for analyzing memory usage, CPU performance, and network activity, and lastly logcat viewer, for live log output and debugging the app itself [19]. Android studio heavily relies on XML (eXtensible Markup Language) for designing the UI for the app itself. UI elements like buttons, text views, images, and layouts are defined in XML layout files. Strings, colors, dimensions, styles, and themes are declared in XML resource files. The last is the app’s essential information, components, permissions, and hardware requirements are also set with XML.

## Firebase

Firebase is a database that is provided by Google. This database operates based on a cloud-host that allows data to be stored across multiple clients at the same time. Firebase is perfect for this project, because this project also uses Android Studio from Google as well. This will make sure compatibility between the two programs is compatible with each other. The database that will be used in this project will store a couple of important pieces of information. Such as date and time of which the data is created, image, validation, and the name of the user. The date and time of data will be created if there’s an attempt of door unlock in the IoT system. Image, validation, and name will follow as important credentials that will be generated when an RFID card is scanned. Another important aspect of the database is password. The password is saved by using SHA-256 (Secure Hash Algorithm 256-bit). SHA-256 is widely used in the security system to make any input (for example a password) and turns it into a fixed string of 64 characters long. After this string has been hashed, it cannot be reversed. So, the security of the string that has been hashed is definite. Base64 is also used in this project to save pictures that are captured by the ESP32-CAM. Base64 will transfer images to be saved as plain text, making it compatible with the style of Firebase’s database. It also helps with the size of which the image will be stored. As a string, it will take less memory space for image to be saved, rather than being an image. While there’s no disadvantage when decoding is to a mobile application. The image that is stored within the firebase will be decoded on the application, making it appear as an image again.

# Result and Discussion

The project is successfully finalized. The ESP32 and ESP32-CAM connects greatly with other modules, sensors, and the application as well as the database. The prototype of the model has been made with the help of cardboards, glue, and electrical tapes. We call this project as “Mr. Secure”.

A logo for a home security system

AI-generated content may be incorrect.

*Picture 3.1. Mr. Secure Logo*

A couple of struggles that is faced with this project is mainly because of the servo. The SG90 servo sometimes doesn’t move when it is supposed to. This is mainly caused by the lack of power that goes into the servo. The counter to this issue is to use external battery pack for powering the servo in itself. We use 4 serried batteries that is paralleled another 4 serried batteries. The type of battery that is used for this is “ABC Super Power 1.5 V”. With this, we have a total of 6 Volts, paralleled with 6 Volts. We only use the battery pack for servo, so we can estimate the voltage needed for the servo. The normal voltage so a servo can be optimal under weight is 5 volts. From table 2.3, we can take the maximum current for example as 1A. If the normal voltage of servo is used, we can use the formula:

Where *V* is voltage, *I* is the current, *R* is the resistance, we can get the normal resistance of servo under maximum load which is 5 Ω (Ohm). With this, we can know the appropriate current which is 6V/5Ω which is 1.2A, and it is sufficient for the servo. But why do we need another 6V total of batteries being paralleled to the servo? The maximum current that a serried channel of batteries only can output a total of 0.5A to 1A. This is still a bit under of what is needed for the servo. In addition, we use another set of serried batteries hence making the current output doubled. Now the servo’s current needs are already sustained, and the voltage is more stable as well. Hence, making the likelihood of servo dropping less than before.

The systematics of the project works just as explain before on part II.D. where the relation between components is explained. The lock which is used in this Mr. Secure prototype is just a simple slide latch door lock, with the servo moving it.

A cardboard box with wires and wires

AI-generated content may be incorrect.

*Picture 3.2. Mr. Secure Off State*

A blue electronic device with a screen and a red light

AI-generated content may be incorrect.

*Picture 3.3. Mr. Secure First On State, Trying To Connect To Wi-Fi*

*A blue circuit board with a red light

AI-generated content may be incorrect.*

*Picture 3.4. Mr. Secure Has Connected To The Wi-Fi*

When the system is connected to power, both of the ESPs will immediately try to connect to Wi-Fi, to secure a stable connection. After securing a stable connection, both the ESP32s will try to connect to each other, by having ESP32-CAM as the host. The ESPs can achieve this by having ESP32-IoT connects to ESP32-CAM’s IP.

A blue electronic device with a red light

AI-generated content may be incorrect.

*Picture 3.5. Mr. Secure Normal State When Turned On, Ultrasonic Sensor Will Always Scan For Nearest Object*

*A blue electronic device with a red light

AI-generated content may be incorrect.*

*Picture 3.6. Mr. Secure Detects An Object Being Less Than 10cm*

*A blue circuit board with a red light

AI-generated content may be incorrect.*

*Picture 3.7. Mr. Secure Waits For A RFID Card Or Tag To Be Detected*

After being connected to each other, the system will come to it’s default state, that is the ultrasonic sensor will continuously search its surroundings and LCD will display the distance of its ultrasonic sensor to the nearest thing possible. If there’s an object near the ultrasonic sensor being less than 10cm, it will be detected and buzzer will give a long beep. The LCD will also print the message “Object detected! Scan your card”. After the beep, LCD will display the message “Scan your card”.

*A hand holding a card with a letter on it

AI-generated content may be incorrect.*

*Picture 3.8. Mr. Secure Scanned A RFID Card And Displays The Cardholder’s Name*

A close up of a device

AI-generated content may be incorrect.

*Picture 3.9. Mr. Secure Has Validated The Cardholder's Name, The Door Will Open*

The user will now scan an RFID card or RFID tag to open the door. If the cardholder’s name is verified, then the door lock will open, the buzzer will give an approving tone, and LCD will output “Hello, (card holder’s name) Access Granted”. The ESP32-CAM will capture an image a then those data will be sent to firebase, so it will be stored safely. The format of which data being saved is by ID, time created (createdAt), image, validation (isValid), and name. The ID will be auto generated if there is a new record that will be added to the database. Time created will be saved by the format of “Year-Month-DayTHour:Minutes:Seconds”. Validation is saved by Boolean value, as in true or false, and lastly, name will be saved from the cardholder’s name. If the card isn’t verified, buzzer will output a disapproving tone, the door will not open, and LCD will output “Access Denied Unauthorized!”.

A screenshot of a computer

AI-generated content may be incorrect.

*Picture 3.10. Application Database Record*



*Picture 3.11. Mr. Secure Detects That The Cardholder’s Name Isn’t Verified So The Door Isn’t Open*

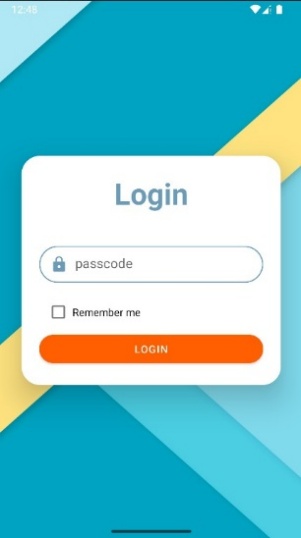
A blue circuit board with a red light

AI-generated content may be incorrect.

*Picture 3.12. Mr. Secure Will Output Timeout Notification Where There Is No Card Scan On The Time Interval*

The database record will If there’s no scan on the RFID scanner, buzzer will output a long beep and LCD will output the text “Timeout! Try Again…”. All of those records, either there is a successful RFID scan, unsuccessful RFID scan, or RFID timeout, will be recorded on the database.

The application will have a login page, with username and password. This process surely will have validation steps, so the user who logged on to the application is verified. There is also a remember me option, so the user wont need to login every time they wanted to use the application.



*Picture 3.13. Mr. Secure Login Page*

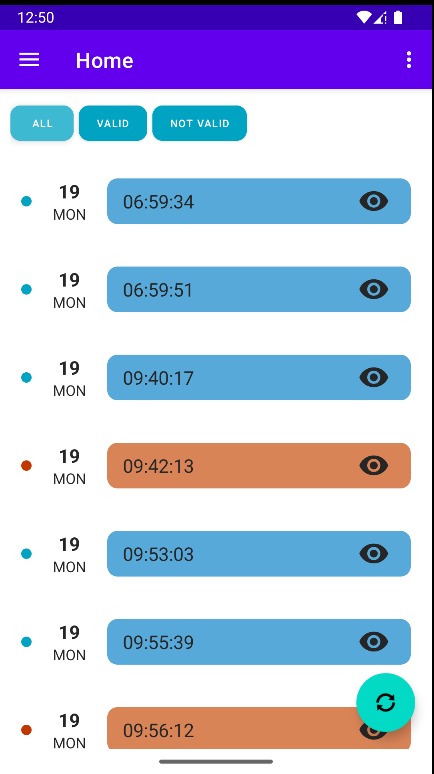


*Picture 3.14. Mr. Secure Welcome Notification*



*Picture 3.15. Mr. Secure Wrong Passcode Notification*

The application’s default page will display a bunch of records that is tries to unlock the door. The records will be shown in a list, where blue coloured records are the ones with the validated user, and the orang coloured records are the ones with unknown user, which the door didn’t open onto. The app also has a filter which can show specific records, either the valid ones, or the non-valid ones. This ensure easy access for user, whether they wanted to know who tried to unlock their doors.

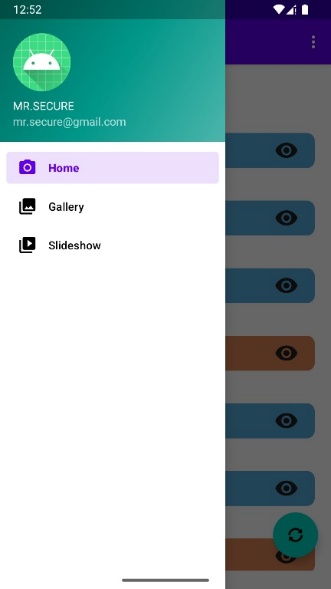


*Picture 3.16. Mr. Secure Landing Page*

The app also have a refresh button to ensure real-time protection is being handed to the user. This method also become very handy if burglars tried to prey the door by brute forcing the RFID system. The help of taking an image whenever there is an attempt to open the door will be very beneficial for crime cases, to give a sufficient prove. The app also have other feature like slideshow and gallery, to see all of the pictures that have been collected over the time.



Picture 3.17. Mr. Secure Landing Page With Filters Applied



Picture 3.19. Mr. Secure’s Other Features



Picture 3.18. Mr. Secure’s Gallery

##### Conclusion and *Suggestions*

The project is very cost sufficient and easy to make for home owners who really liked to make their own home security services. The simplicity of the design and easy use of modules and microcontrollers. There’s a couple suggestions for further research as in, more efficient power supply can be achieved rather than the use of power banks. More than that, the use of 8 batteries for servo only is very over the top.

##### References

1. Federal Bureau of Investigation, “Crime data explorer,” U.S. Department of Justice, 2025. [Online]. Available: <https://cde.ucr.cjis.gov/LATEST/webapp/>. [Accessed: May 29, 2025].
2. F. Navarrete, Á. L. Garrido, C. Bobed, and R. Trillo-Lado, “Ontology-driven automated reasoning about property crimes,” *Business & Information Systems Engineering*, 2024. [Online]. Available: <https://doi.org/10.1007/s12599-024-00886-3>. [Accessed: May 29, 2025].
3. M. S. Z. M. Zabidi, K. H. Yusof, A. S. Ahmad, N. H. Shahadan, K. A. Yusof, and K. Amri Yusof, “IoT RFID door security system,” *J. Phys.: Conf. Ser.*, vol. 2312, p. 012092, 2022, doi: 10.1088/1742-6596/2312/1/012092. [Accessed: May 29, 2025].
4. D. Hercog, T. Lerher, M. Truntič, and O. Težak, “Design and implementation of ESP32-based IoT devices,” *Sensors*, vol. 23, no. 15, p. 6739, 2023, doi: 10.3390/s23156739. [Accessed: May 29, 2025].
5. M. Samiullah, M. Z. Irfan, and A. Rafique, “Microcontrollers: A comprehensive overview and comparative analysis of diverse types,” ResearchGate, Sep. 22, 2023. [Online]. Available: <https://www.researchgate.net/publication/374016434_Microcontrollers_A_Comprehensive_Overview_and_Comparative_Analysis_of_Diverse_Types>. [Accessed: May 30, 2025].
6. Last Minute Engineers, “ESP32 pinout reference: Which GPIO pins should you use?” [Online]. Available: <https://lastminuteengineers.com/esp32-pinout-reference/>. [Accessed: May 30, 2025].
7. F. D. Tanugraha, *Tampilan data teks pada OLED menggunakan transmisi LoRa*, Internship Report, Faculty of Technology and Informatics, Universitas Dinamika, 2021. [Online]. Available: <https://eprints.utdi.ac.id/10178/3/3_203310032_BAB_II.pdf>. [Accessed: May 30, 2025].
8. Shenzhen LCD, “1.3 inch 128x64 display module I2C OLED,” [Online]. Available: <https://id.shenzhenlcd.com/oled/1-3-inch-128x64-display-module-i2c-oled.html>. [Accessed: May 30, 2025].
9. T. B. Chandrawati, R. F. Christanti, and R. Sanjaya, “Deteksi buku perpustakaan fakultas dengan aplikasi RFID berbasis web,” *Jurnal Proxies*, vol. 1, no. 1, pp. 32–36, 2012. [Online]. Available: <https://repository.unika.ac.id/18/1/Journal%2011-Deteksi%20Buku%20Perpustakaan%20Fakultas%20dengan%20Aplikasi%20RFID%20Berbasis%20Web.pdf>. [Accessed: May 30, 2025].
10. S. A. Dewanto, M. Munir, B. Wulandari, and K. Alfian, “MFRC522 RFID technology implementation for conventional merchant with cashless payment system,” Universitas Negeri Yogyakarta, Indonesia. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1742-6596/1737/1/012012/pdf>. [Accessed: May 30, 2025].
11. Components101, “RC522 RFID module,” [Online]. Available: <https://components101.com/wireless/rc522-rfid-module>. [Accessed: May 30, 2025].
12. G. di Pasquo, “SG90 servo characterization,” ResearchGate, Aug. 2021. [Online]. Available: <https://www.researchgate.net/publication/353754375_SG90_Servo_Characterization>. [Accessed: May 30, 2025].
13. R. A. Hayubi, S. Aulia, D. A. Gunawan, S. Hidayatullah, and D. Aribowo, “Implementasi sistem penggerak servo SG 90 berbasis Arduino Uno dengan kontrol sudut dinamis,” *Mars: Jurnal Teknik Mesin, Industri, Elektro dan Ilmu Komputer*, vol. 2, no. 6, pp. 130–140, Dec. 2024. [Online]. Available: <https://doi.org/10.61132/mars.v2i6.535>. [Accessed: May 30, 2025].
14. Tower Pro, “SG90 9g micro servo datasheet,” [Online]. Available: <http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf>. [Accessed: May 30, 2025].
15. Servo Database, “TowerPro SG90 servo,” [Online]. Available: <https://servodatabase.com/servo/towerpro/sg90>. [Accessed: May 30, 2025].
16. Micros, “HY-SRF05 Precision Ultrasonic Sensor,” *Micros Datasheet*, [Online]. Available: <https://www.micros.com.pl/mediaserver/M_HY-SRF05_0003.pdf>. [Accessed: May 31, 2025].
17. J. Sebai, “How to use the HY-SRF05 ultrasonic distance sensor,” [Online]. Available: <https://jawhersebai.com/tutorials/how-to-use-the-hy-srf05-ultrasonic-distance-sensor/>. [Accessed: May 31, 2025].
18. Lithium-ion, “Active and Passive Buzzer – Discussed,” *Instructables*. [Online]. Available: <https://www.instructables.com/Active-and-Passive-Buzzer-Discussed/>. [Accessed: May 31, 2025].
19. Google, "APK Analyzer," *Android Developers*, 2024. [Online]. Available: <https://developer.android.com/studio/debug/apk-analyzer?utm_source=chatgpt.com>. [Accessed: Jun. 1, 2025].