Internet of Things Report

Abstract – This report presents the design and development of a smart access system for a home garage door using the Arduino Uno as the central microcontroller. This system integrates an ultrasonic sensor, a RFID reader, a 16x2 LCD display, and a micro-servo motor. The ultrasonic sensor detects an object within a given range and when no object is detected, the system will remain inactive to reduce unnecessary power usage especially on the LCD display. Once an object is detected, it will prompt a message for user to scan for ID on the LCD display. If a valid tag is read, it displays a “Welcome Home” message, a green LED light will turn on, and the servo motor will rotate to open the door. After three invalid ID scans, the LCD screen will print “Illegal User”, a red light will be triggered and activate the buzzer alarm. This creates an additional security function for the system. Although the intended application is for a garage door, building a life size model is not feasible within the timeframe. Therefore, a smaller model was created to demonstrate the functions and idea.

1 Introduction

In recent years, the adoption of smart automation systems has grown significantly as homes and industries seek convenience, security, and efficiency. Some smart systems application includes lights, door locks, and switches. Traditional garage door systems typically use handheld remotes or manual operation in opening the garage door. However, this may pose as a security risk if the controller is lost, duplicated or accessed by unauthorised individuals. As home automation continues to advance, there is an increasing demand for more reliable, user-friendly, and secure entry systems. As smart door locks are becoming more common in homes, creating a smart garage door system is also becoming a feasible and practical idea.

This IoT project proposes a proof-of-concept smart garage door system to demonstrate the idea and function of a smart garage door system. The system is built around the Arduino Uno microcontroller which is versatile and compatible with a wide range of sensors and electronic modules. An ultrasonic sensor is incorporated to detect the presence of a vehicle or object, to ensure that the system activates when necessary. The RFID module is used for the primary authentication method, requiring users to present an authorised tag before granted access. A 16x2 LCD display provides real-time instructions and feedback to the user, while the micro-servo motor simulates the opening and closing of garage door of the model. Additional modules such as LED lights and a buzzer help to signal successful entry, invalid attempts, or security alerts.

Through the integration of these components, this project shows how a low-cost microcontroller platform can be used to create a secure and interactive access-control solution. Although the final system is a scaled-down model due to time and material constraints, it’s able to showcase the core functions, workflow, and potential applications of a fully automated garage door access system.

2 System Design

2.1 Project planning

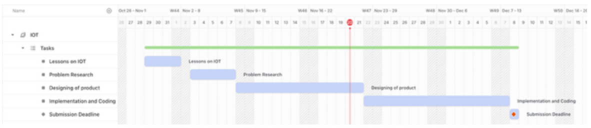


Fig 1. Gantt Chart for project planning

The timeline for this project started from 29th October 2025 and ends on the 8th of December 2025. During these 6 weeks period, there are several tasks set to ensure the completion of the project. As there was a concurrent project ongoing, time is constraint and must be properly planned for both projects to be completed.

The first task was to learn about “Internet of Things”. There was a mixture of online and physical classes that happened on the first week, 29th October 2025 to 1st November 2025. During this period, there was also hands on sessions on building the circuit board and program it through the Arduino IDE app. This was important as it must be used in this project.

The next task would be on the research of a problem. Firstly, it was to think of a problem followed by researching on current solutions on the issues. The initial idea of a solution was also done during this period. The duration set for this task was 5 days from 3rd - 7th November 2025.

The designing of IoT product was allocated 14 days, from 8th - 21st November 2025. During this period, the circuit design and activity diagram was tasked.

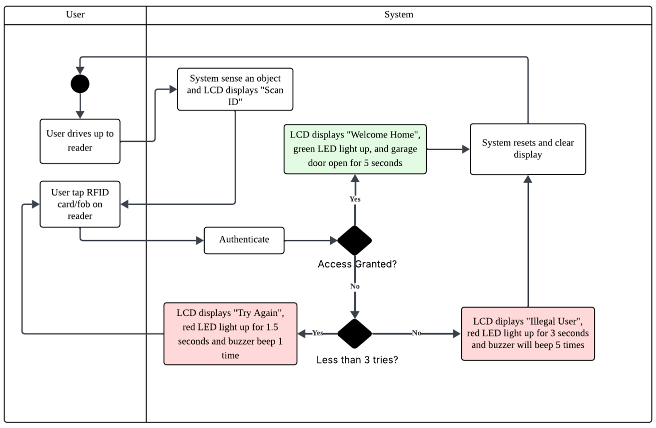
Lastly were the implementation and coding for the project. This period was the most intensive where it took 16 days, starting from 22nd November - 7th December 2025. The prototype creation, Arduino codes, testing and video recording were done during this period.

2.2 Activity Diagrams

There are 2 activities for this smart garage door system. First is when a user taps a valid ID on the RFID reader and the other is when a user taps an invalid ID.

Valid ID tapped on the RFID reader
1. Users drive up to the front porch, stopping in front of the reader
2. Ultrasonic sensor detects an object and display "Scan ID" on the LCD display
3. Users tapped the ID key on the RFID reader
4. Upon authentication, LCD will display "Welcome Home!"
5. Green LED will light up for 5 seconds
6. The garage door will open for 5 secs for the car to drive in

Invalid ID tapped on the RFID reader
1. Users drive up to the front porch, stopping in front of the reader
2. Ultrasonic sensor detects an object and display "Scan ID" on the LCD display
3. Users tapped the ID key on the RFID reader
4. Upon authentication, LCD will display "Try Again"
5. A red LED will light up for 1.5 seconds
6. Buzzer will also start for 1.5 seconds
7. After 3 tries of invalid ID scan, the LCD displays "Illegal User"
8. A red LED will light up and the buzzer will sound for 3 seconds


User
System
User drives up to reader
System sense an object and LCD displays "Scan ID"
User tap RFID card/fob on reader
Authenticate
Access Granted?
Yes
LCD displays "Welcome Home", green LED light up, and garage door open for 5 seconds
System resets and clear display
No
Less than 3 tries?
Yes
LCD displays "Try Again", red LED light up for 1.5 seconds and buzzer beep 1 time
No
LCD diplays "Illegal User", red LED light up for 3 seconds and buzzer will beep 5 times

2.3 Circuit Design

In this IoT system, listed below are the required components or modules needed.

1. HC-SR04 Ultrasonic Sensor
2. LCD1602 Module
3. RC522 RFID Module
4. Servo Motor SG90
5. Uno R3 Controller Board
6. 830 Tie-Points Breadboard
7. 400 Tie-Points Breadboard
8. Power Supply Module
9. Active Buzzer
10. Green LED
11. Red LED
12. Potentiometer 10k
13. Jumper Wires
14. Resistors (330Ω)

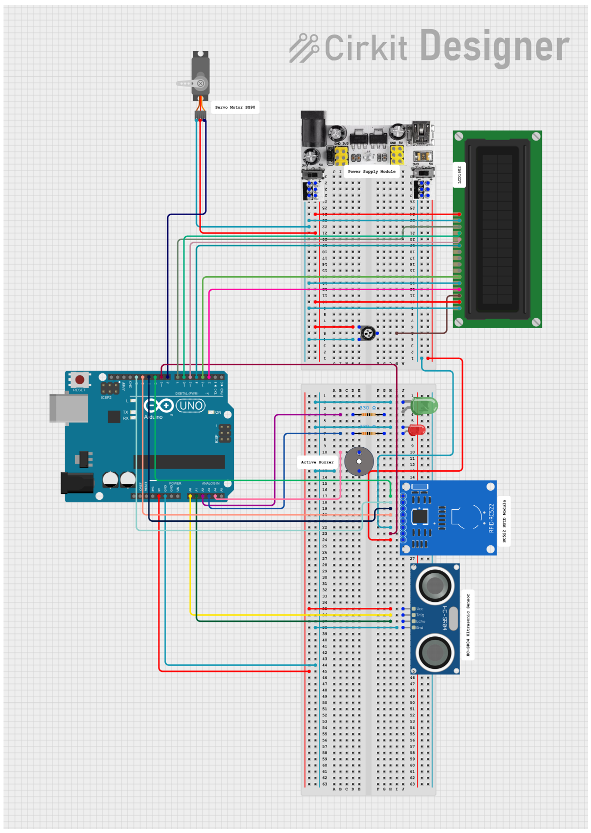


Fig 3. Circuit Design for smart garage door system

Initially the RC522 RFID module could not read the RFID card or tag when it was drawing power from the Uno board itself. I went on to the internet and research on the possible causes. From the research, the RFID module requires a steady 3.3v power [1]. Therefore, I went to test the powerline for the RC522 module by using the code written below and the Arduino IDE. From the test, the reading shows numbers around 242 - 250. This analog reading got to be recalculated with the following formula [2].

$$Voltage = \frac{analogRead\ value}{1023} \times 5.0$$

```
void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}
void loop() {
  Serial.println(analogRead(A0));
  digitalWrite(13, HIGH);
  delay(500);
  digitalWrite(13, LOW);
  delay(500);
}
```
Fig 4. Code to get analogRead Value

From the calculation, the approximate voltage from the Uno board is only around 1.2v which is not sufficient to power the RFID module. Thus, adding an additional power supply module can help support in the power delivery. It can also be used to help power the LCD screen and servo motor at 5v so that the Uno can supply steady power to the other components such as the Ultrasonic Sensor, LEDs and buzzer.

2.4 Costing

After selecting the module and components needed for this system, a costing table can be created to see the total cost for the system. The pricelist for each item can be retrieved from the Kuriosity website [3]

Components/Modules	**Price**	**Qty**	**Cost**
HC-SR04 Ultrasonic Sensor	$4.00	1	$4.00
LCD1602 Module	$8.80	1	$8.80
RC522 RFID	$6.50	1	$6.50
Servo Motor SG90	$7.00	1	$7.00
Uno R3 Controller Board	$45.00	1	$45.00
830 Tie-points breadboard	$4.00	1	$4.00
400 Tie-points breadboard	$3.50	1	$3.50
Power Supply Module	$4.00	1	$4.00
Active Buzzer	$1.00	1	$1.00
LED Kit (100 pcs)	$6.40	1	$6.40
Potentiometer 10k	$4.00	1	$4.00
Jumper wires Male to Male 10cm (40 pcs)	$3.00	1	$3.00
Jumper wires Male to Female 20cm (40 pcs)	$3.50	1	$3.50
Resistor Kit	$13.40	1	$13.40
		Total Cost	$114.10

3. Implementation

3.1 Prototype Creation

In this prototype system, it consists of 4 parts excluding the Arduino Uno controller. The 4 different parts consists of the different components needed to run the system.

The first part is for the **Garage Door,** and it is used to open and close the garage door. The component/module is:

1. Servo Motor SG90

The second and third part is for the **Display**. These parts are to operate the LCD display and provide power supply to the LCD module and servo motor as well. The components/modules are:

1. LCD1602 Module
2. 400 Tie-Points Breadboard
3. Power Supply Module
4. Potentiometer 10k
5. Jumper Wires

The last part is for the **Sensors and RFID reader**. This is the main part of the system as it consists of modules and sensors that will operate the whole system. The components/modules are:

1. HC-SR04 Ultrasonic Sensor
2. RC522 RFID Module
3. 830 Tie-Points Breadboard
4. Active Buzzer
5. Green LED
6. Red LED
7. Jumper Wires
8. Resistors 330Ω

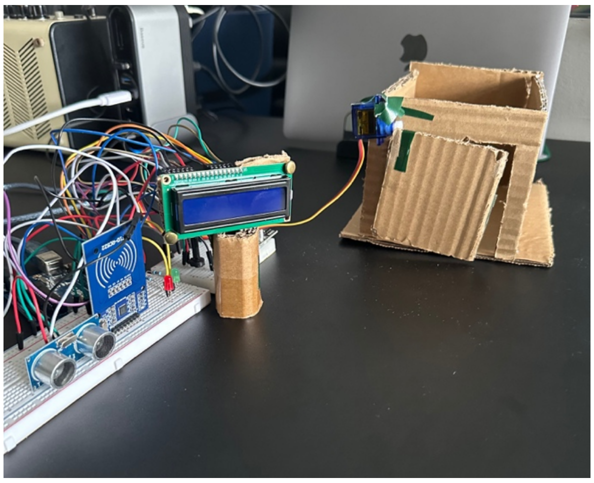


Fig 5. Prototype

3.2 Code Snippets

3.2.1 Libraries

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>
#include <LiquidCrystal.h>
```

Fig 6. Libraries set up for Arduino IDE

The code started with adding the necessary libraries to be used for all the components. The RC522 RFID requires to use the protocol from serial peripheral interface (SPI) library and the MFRC522 library. To operate the servo motor and LCD display, I used the "Servo" and "LiquidCrystal" library respectively.

3.2.2 Pin and Object Definitions

```
// ---------- PIN DEFINITIONS ----------
#define LCD_RS   2
#define LCD_E    3
#define LCD_D4   4
#define LCD_D5   5
#define LCD_D6   6
#define LCD_D7   7

#define SERVO_PIN    8

#define RFID_RST_PIN 9
#define RFID_SS_PIN  10

#define ULTRASONIC_TRIG A0
#define ULTRASONIC_ECHO A1

#define GREEN_LED A2
#define RED_LED   A3
#define BUZZER    A4
```

Fig 7. Defining pin usage

```
// ---------- OBJECTS ----------
LiquidCrystal lcd(LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7);
MFRC522 mfrc522(RFID_SS_PIN, RFID_RST_PIN);
Servo doorServo;
```

Fig 8. Objects definition

Next, I proceed to define the pin usage for the different modules connected to the Uno board. These pins will be used to transmit data or power to the different modules on the breadboard or direct connection. As the RFID reader and the LCD screen require several data pins on the Arduino Uno board, it only leaves the analog pins for the other components. After defining the pins, I went on to define each pin to the different objects used such as the lcd, RC522, and servo motor.

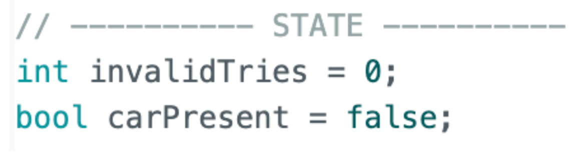
3.2.3 Configurations

```
// ---------- CONFIG ----------
const int OPEN_ANGLE = 90;   // servo angle when door opens
const int CLOSED_ANGLE = 180;  // servo angle when door is closed
const int DISTANCE_THRESHOLD_CM = 8;  // ultrasonic trigger distance
const int MAX_INVALID_TRIES = 3;

// CARD UID: A3 5C 8C 14
byte authorizedUID[] = { 0xA3, 0x5C, 0x8C, 0x14 };
const byte authorizedUIDSize = 4;
```

Fig 9. Configurations settings

I went on to configure the requirements for the system. Some configuration settings are the servo motor angle for the open and closing of garage door which was set at 90° and 180° respectively. I have also set the distance threshold for the ultrasonic sensor to 8 cm and the maximum number of invalid tries to 3. For this prototype, as I only have 1 key card and 1 key fob for the RFID reader, I set the UID for the key fob as the authorised user in the configurations as well.

3.2.4 Variables Setting



```
// ---------- STATE ----------
int invalidTries = 0;
bool carPresent = false;
```

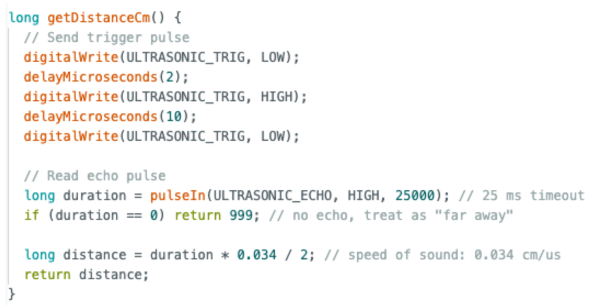
Fig 10. State Variables

I then set two variables, "invalidTries" and "carPresent" which will be used in the later codes. The invalid tries variable is defined as 0, so that during the loop it can be increased by 1 and can be reset easily back to 0. For the presence of car variable, I have set it as a Boolean which only returns true or false. If there is car present, it will return true and other code will be followed.

3.2.5 Helper Functions

This part of the code is where I have defined some helper functions that can be reference in the loop codes to run the system. It is called a helper function as the code perform a specific part of a larger task to help the main function. It is good to use helper functions so that it can help make the codes to look more clean and organised [4].

getDistanceCm()

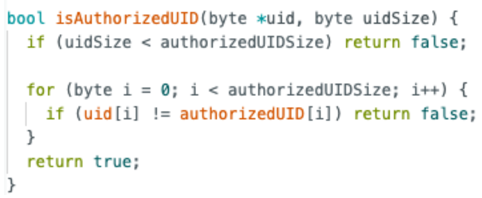


```
long getDistanceCm() {
  // Send trigger pulse
  digitalWrite(ULTRASONIC_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(ULTRASONIC_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(ULTRASONIC_TRIG, LOW);

  // Read echo pulse
  long duration = pulseIn(ULTRASONIC_ECHO, HIGH, 25000); // 25 ms timeout
  if (duration == 0) return 999; // no echo, treat as "far away"

  long distance = duration * 0.034 / 2; // speed of sound: 0.034 cm/us
  return distance;
}
```

Fig 11. getDistanceCm() helper function code

The first helper function is the getDistanceCm(). This helps to calculate the distance read from the readings of the ultrasonic sensor. This would be used to calculate distance from object and can be used in conjunction with the distance threshold to detect the presence of an object.

isAuthorizedUID()

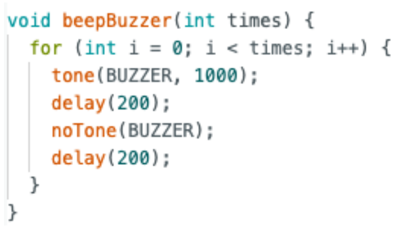


```
bool isAuthorizedUID(byte *uid, byte uidSize) {
  if (uidSize < authorizedUIDSize) return false;

  for (byte i = 0; i < authorizedUIDSize; i++) {
    if (uid[i] != authorizedUID[i]) return false;
  }
  return true;
}
```

Fig 12. isAuthorizedUID() helper function code

This helper function is task in the checking of the scanned UID and comparing it to the authorised UID. It is a Boolean function so it will return true or false. There are 2 checks in this function. The first is to check on the UID byte size, if the scanned UID size is lesser than the authorised size it will return false. The next is to compare each individual byte with the authorised. If there's a mismatch in byte, it will return as false. Lastly, if everything matches the authorised UID, then it will return as true.

beepBuzzer()

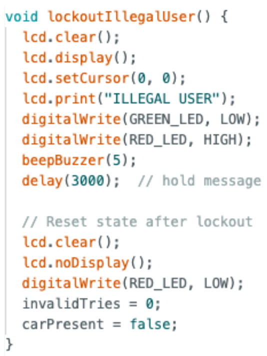


```
void beepBuzzer(int times) {
  for (int i = 0; i < times; i++) {
    tone(BUZZER, 1000);
    delay(200);
    noTone(BUZZER);
    delay(200);
  }
}
```

Fig 13. beepBuzzer() helper function code

The next function is a function to activate the buzzer. This will be helpful so that this function can be easily reference in other parts of the code. This function requires to input a number, which will tell the code the number of times the buzzer should be.

lockoutIllegalUser()

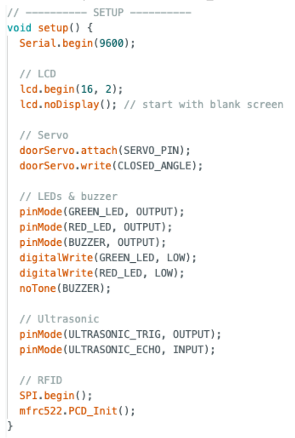


```
void lockoutIllegalUser() {
  lcd.clear();
  lcd.display();
  lcd.setCursor(0, 0);
  lcd.print("ILLEGAL USER");
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(RED_LED, HIGH);
  beepBuzzer(5);
  delay(3000);  // hold message

  // Reset state after lockout
  lcd.clear();
  lcd.noDisplay();
  digitalWrite(RED_LED, LOW);
  invalidTries = 0;
  carPresent = false;
}
```

Fig 14. lockoutIllegalUser() helper function code

The last helper function is the lock an illegal user function. This function will be used once the maximum number of tries is reached. This is the code that will display the "Illegal User" on the LCD, activate the red light and beep the buzzer 5 times. Then after 3 seconds it will reset the state by clearing the display, deactivate the LED and reset the 2 state variables back to 0 and false.

3.2.6 System Setup

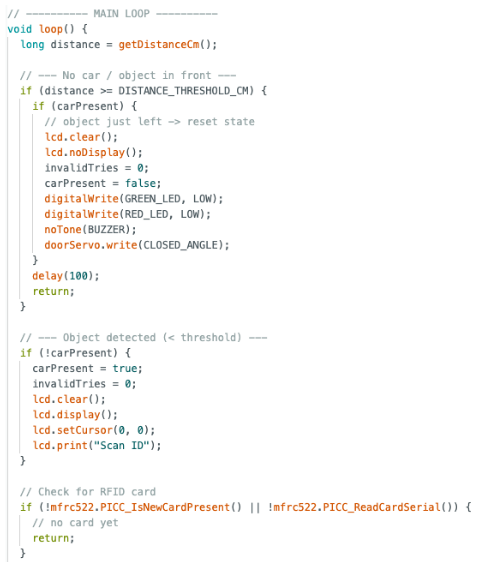


```
// ---------- SETUP ----------
void setup() {
  Serial.begin(9600);

  // LCD
  lcd.begin(16, 2);
  lcd.noDisplay(); // start with blank screen

  // Servo
  doorServo.attach(SERVO_PIN);
  doorServo.write(CLOSED_ANGLE);

  // LEDs & buzzer
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(RED_LED, LOW);
  noTone(BUZZER);

  // Ultrasonic
  pinMode(ULTRASONIC_TRIG, OUTPUT);
  pinMode(ULTRASONIC_ECHO, INPUT);

  // RFID
  SPI.begin();
  mfrc522.PCD_Init();
}
```

Fig 15. Code to initialise the components

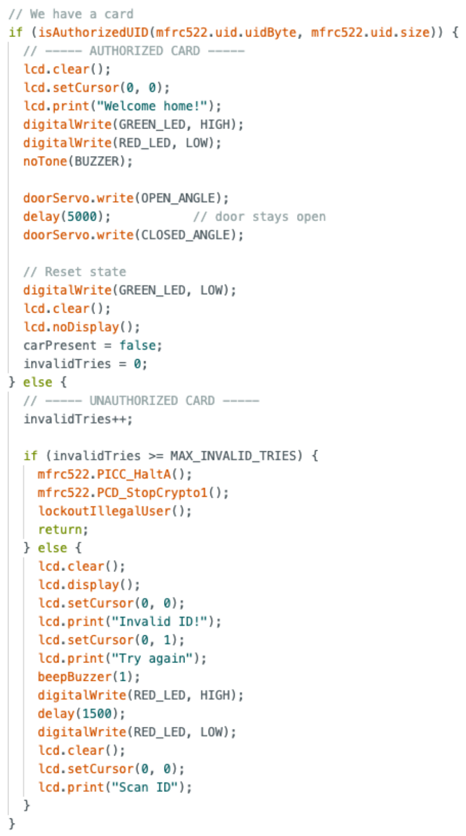
In the setup() function, the serial communication was first initialised to set up the environment. Next the other components such as the LED, buzzer, ultrasonic sensor are initialised. The LCD display was also initialised and set display as blank. After initialising the servo motor, the angle of it was set to the closed-door angle. For the RFID module, it was initialised and the SPI hardware was started so that it can establish communication with the Arduino.

3.2.7 System Main()



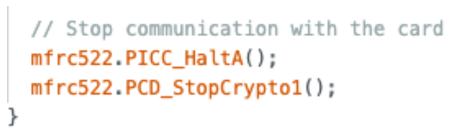
```
// ---------- MAIN LOOP ----------
void loop() {
  long distance = getDistanceCm();

  // --- No car / object in front ---
  if (distance >= DISTANCE_THRESHOLD_CM) {
    if (carPresent) {
      // object just left -> reset state
      lcd.clear();
      lcd.noDisplay();
      invalidTries = 0;
      carPresent = false;
      digitalWrite(GREEN_LED, LOW);
      digitalWrite(RED_LED, LOW);
      noTone(BUZZER);
      doorServo.write(CLOSED_ANGLE);
    }
    delay(100);
    return;
  }

  // --- Object detected (< threshold) ---
  if (!carPresent) {
    carPresent = true;
    invalidTries = 0;
    lcd.clear();
    lcd.display();
    lcd.setCursor(0, 0);
    lcd.print("Scan ID");
  }

  // Check for RFID card
  if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) {
    // no card yet
    return;
  }
```

Fig 16. Main loop code first part



```
// We have a card
if (isAuthorizedUID(mfrc522.uid.uidByte, mfrc522.uid.size)) {
  // ----- AUTHORIZED CARD -----
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Welcome home!");
  digitalWrite(GREEN_LED, HIGH);
  digitalWrite(RED_LED, LOW);
  noTone(BUZZER);

  doorServo.write(OPEN_ANGLE);
  delay(5000);          // door stays open
  doorServo.write(CLOSED_ANGLE);

  // Reset state
  digitalWrite(GREEN_LED, LOW);
  lcd.clear();
  lcd.noDisplay();
  carPresent = false;
  invalidTries = 0;
} else {
  // ----- UNAUTHORIZED CARD -----
  invalidTries++;

  if (invalidTries >= MAX_INVALID_TRIES) {
    mfrc522.PICC_HaltA();
    mfrc522.PCD_StopCrypto1();
    lockoutIllegalUser();
    return;
  } else {
    lcd.clear();
    lcd.display();
    lcd.setCursor(0, 0);
    lcd.print("Invalid ID!");
    lcd.setCursor(0, 1);
    lcd.print("Try again");
    beepBuzzer(1);
    digitalWrite(RED_LED, HIGH);
    delay(1500);
    digitalWrite(RED_LED, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Scan ID");
  }
}
```

Fig 17. Main loop code second part

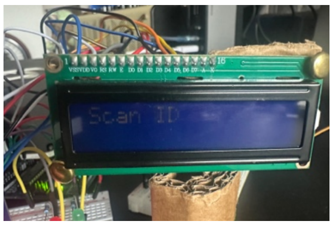


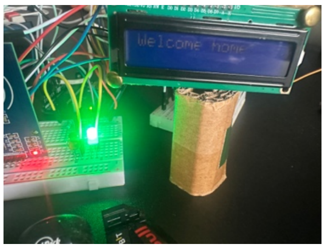
```
  // Stop communication with the card
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
}
```

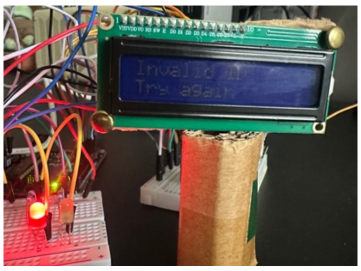
Fig 18. Main loop code third part

In the main loop, the system will undergo multiple steps as shown in the codes. I first set a variable for distance with the helper function to get the distance with the ultrasonic sensor. Next part would be when there's no car or object in front of the sensor, the system will be in reset state where the LCD display is cleared, all the LED lights and buzzer will be off, the servo motor at a closed angle, and the variables reset to 0. The function is then exited. The code is then followed by when an object is detected by the distance recorded lower than the threshold value. It will print "Scan ID" on the LCD. The code will then use the RFID reader to check if RFID card is present and readable before the code continues to run. The code will then authenticate the RFID card, and if it is the approved user, proceed with the authorized card code. If an unauthorised card was scanned, the unauthorised code will run. The number of tries will increase by 1 until the invalid tries is equal or more than the maximum tries, it will call the lockoutIllegalUser() helper function. During the lockout, there are two codes, "_HaltA()" and "_StopCrypto1()" to stop communication with the card. It will be run at the end of the code to end the main loop.

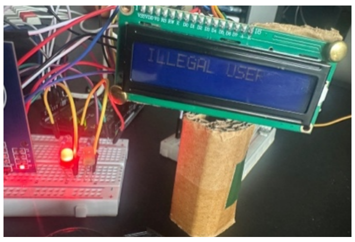
4. Test Cases

Test Case ID	1
Function	Sensor senses an object
Test Description	- Ultrasonic sense an object - LCD displays "Scan ID"
Expected Results	Door remains close and LCD display phrase
Actual Results	Door remains close and LCD display phrase
Test Outcome	Passed
Screenshots	

Test Case ID	2
Function	Using an authorised RFID card/fob
Test Description	- Authorised RFID card/fob - LCD displays "Welcome home" - Green LED light up for 5 seconds - Motor open door for 5 seconds
Expected Results	Garage door opens for 5 seconds
Actual Results	Garage door opens for 5 seconds
Test Outcome	Passed
Screenshots	

Test Case ID	3
Function	Using an unauthorised RFID card/fob (less than 3 tries)
Test Description	- Unauthorised RFID card/fob - LCD displays "Try Again" - Buzzer activated for 1 time - Red LED light up for 1.5 seconds - Motor remains at closed angle
Expected Results	Garage door remains closed
Actual Results	Garage door remains closed
Test Outcome	Passed
Screenshots	

Test Case ID	4
Function	Using an unauthorised RFID card/fob (3 tries and more)
Test Description	- Unauthorised RFID card/fob - LCD displays "Illegal User" - Red LED light up for 3 seconds - Buzzer activated for 5 times - Motor remains at closed angle
Expected Results	Garage door remains closed
Actual Results	Garage door remains closed
Test Outcome	Passed

Screenshots	

5. Discussion

5.1 Summary

This prototype was able to display the feasibility of using a low-cost technology to create a more secure and efficient smart garage door system. Although the servo motor can't be directly used for the opening and closing of garage door, wiring can be done to establish a connection with existing door motors. By having this prototype of a smart garage door system, implementing a similar setup in homes becomes more achievable. With further enhancements, such as integrating internet connection and mobile application control, the system can be expanded into a fully functional smart-home solution. Overall, this project shows how affordable components and microcontroller-based automation can enhance convenience and security in everyday living environments.

5.2 Advantages

5.2.1 Scalable

This project has the potential to scale beyond the prototype stage. Although, the current system uses the Arduino components, the architecture allows expansion to enhance functionality and real-world deployment. The RFID reader module can be upgraded to allow more types of verification and increase security. These possibilities shows that the prototype is not limited as a small-scale demonstration, with the right upgrades, it can become a robust and scalable smart system for residential and commercial use.

5.2.2 Cost Effective

The components used in this prototype are inexpensive which can be considered as cost effective. Any additional cost for upgrade to the system can be based on the individual module and inexpensive options. In addition, there are libraries and codes readily available which allows the system to be easily implemented and does not require a professional help. Thus, making the system a cost-effective one.

5.3 Limitations and Disadvantages

5.3.1 Servo Motor Limitations

The first limitation is that the prototype uses a basic servo motor to operate the garage door. However, for a life size garage door, the motor cannot support it as it does not have the mechanical strength and limited torque. Therefore, it is a disadvantage in terms of using the same motor for a life size project. There's a need to purchase additional motors that can support the garage door weight, and additional wiring and configurations must be made.

5.3.2 Fixed UID authentication

In the prototype, it only allows a fixed UID for authentication and any addition of authorised UID needs to be edited in the code. This limits the scalable approach as it will be quite tedious to make edits in the code for every UID to be authorised. It is also not user friendly, where it restricts people that have no experience in coding, to make the necessary changes. A robust system would require a more scalable approach, such as allowing multiple IDs to be stored locally or integrating a cloud-based database management.

5.3.3 Wiring Situation

Lastly, the prototype operates on wired components and lacks wireless connectivity components. This reduces the practicality of a modern smart home environment as it could not be remotely controlled. In addition, the prototype shows many jumper wires being used. To scale in a life size garage door, longer and bigger wires got to be used so that the system in the home can communicate with the components build outside of the homes such as the sensors and reader.

5.4 Future Enhancements

From the discussion of the limitations of the prototype, there are future enhancements that can help tackle them.

5.4.1 Wi-Fi Connectivity

As we are building towards a smart home, having a wireless connectivity over Wi-Fi is important. With the additional of a Wi-Fi module, such as the ESP32, it can pave the way for many more integrations. Some integrations can include cloud storage of UID and opening the garage door through a phone app.

5.4.2 Sensor

Currently the current system, after granted access by an authorised UID, the garage door will open for 5 seconds. However, sometimes this might not be feasible as some cars might not clear the garage door in 5 seconds. Therefore, to solve this issue, there can be an additional sensor that will sense once the car clears the garage door before proceeding to close it.

5.4.3 Access Management

In the current set up, the additional and removal of authorised UID are through codes in the Arduino IDE. This makes the whole process quite tedious. To tackle this issue, there can be a phone application that is able to manage all the access. With the enhancement of wireless connectivity, it makes this enhancement possible.

6. References

[1] J. Joseph, “Arduino RFID Tutorial: Complete RC522 Guide with Code & Projects,” *CircuitDigest*, Sep. 01, 2025. https://circuitdigest.com/microcontroller-projects/interfacing-rfid-reader-module-with-arduino

[2] Arduino, “Read analog Voltage | Arduino documentation.” https://docs.arduino.cc/built-in-examples/basics/ReadAnalogVoltage/

[3] Kuriosity, “DIY Electronics in Singapore: Arduino, Raspberry Pi, microbit, M5Stack,” *Kuriosity*. https://kuriosity.sg/

[4] GeeksforGeeks, “What are the Helper functions ?,” *GeeksforGeeks*, Jul. 23, 2025. https://www.geeksforgeeks.org/javascript/what-are-the-helper-functions/

Appendix A

I have recorded a video to demonstrate how the system operates. It consists of the prototype demonstration and the code explanation. The video and the code file have been uploaded to a google drive folder that can be accessed with the following link.

https://drive.google.com/drive/folders/1qk_DPyW3ITeSchNqCIdhCXCIaI1JkEt0?usp=drive_link