

ABSTRACT

Security Systems are an important feature of modern residential, office setups and banks. Security systems must be affordable, reliable and effective. Modern complex security systems include several security features like fire, intruders, electronic door lock, heat, smoke, temperature, etc, while some security systems may be a combination of all the security measures.

The RAKGS aims to develop a security system using the application of IOT-**Internet of Things** is the network of device (electronics, software, actuators, and connectivity) allows these things to connect, interact and exchange data. The ideology of project is to establish a security system that has a functionality to alert the owner on his/her mobile number when an intruder is detected.

The model functions as follows, when the right ID is scanned and the right password entered entry is granted with a confirmation message sent to the owner.

When a wrong ID is scanned it alerts the owner with a message sent via GSM module to which he/she can take the action of opening/halting the system. This model allows user to monitor their assets even from a remote place.

1. INTRODUCTION

In this fast paced world, the need for security based systems has increased with time. Smart systems working automatically without human interference have found high demand. Such smart systems can be created with the help of IOT technology.

IOT was invented by Kevin Ashton in early 2000's. IOT is an upcoming technology that makes use of network to control/monitor physical devices which are interconnected.

Thus the basic premise is to have smart sensors collaborate directly without human involvement to deliver a new class of applications. The functioning of the project is on the correlative working of the RFID reader to scan the right ID and get pass when right password is given, if any of the given condition is stated false then access is denied and an alerts is given to the intruder as well as the owner.

The RFID scanner combined with the password authentication makes it achieve higher level of security over intruder. The implement of GSM SIM900A plays a major role in laying control in fingertip.

1.1 OBJECTIVE

The main objective of this project is remote controlling security device and ensures safety over assets and to implement such the model is designed as follows:

- ☺ User can remotely lock or unlock door at anyplace.
- ☺ Detect intruder 24/7.
- ☺ To implement double secure system-RFID scanner and password.
- ☺ SMS alert system that can send sms to enter if any security attack found.
- ☺ The model alerts-reacts and queries the owner for action.
- ☺ Accuracy is validating the user.

1.2 PROBLEM STATEMENT

Today's security system has meet advancement in various places, but still holds few back logs that are addressed:

- ☹ Alert is the main goal of the other security system where action of user is left at stake.
- ☹ Possibility to duplicate keys or theft password.
- ☹ Complete control of the owner is denied.
- ☹ Passive security systems are only available.
- ☹ Security systems are developed only to monitor.
- ☹ Implement of motion sensors can fake intruder alert leading to disturbance.

1.3 EXISTING SYSTEM

The present security systems are designed in a way to capture-alert-monitor model leaving behind the main objective to strengthen login validation.

The modes of Login validation of today's security system is primarily keys else passwords, which can both be broken as each has its own merits and demerits. Alert modules are quite disturbing because even if it detects the owner it alarms hence it creates a chaos.

1.4 PROPOSED SYSTEM

The proposed model has emerged to bread burglary/intruder in ways by overcoming the obstacle faced by trending technologies. The dual authentication mode makes the system impossible to break in and it has accuracy at high calibre, alert messages to alarm user, elimination of object detection sensors omit unwanted chaos.

The GSM module is connected with the servo motor create a remote control for the user to open/close door, given maximum freedom of control to the user. The proposed model is a combination of 89% automation and 11% of human instruction the equal split promotes the uniqueness of the model.

2. SYSTEM SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

- **Platform** : Arduino IDE
- **Operating System** : Windows 7
- **Programming Language** : C++

2.2 HARDWARE REQUIREMENTS

- **Processor** : Intel Core Process
- **Ram** : 512MB
- **Hard Disk** : 1 GB
- **Monitor** : Dell

HARDWARE USED

COMPONENTS	DESCRIPTION
MICROCONTROLLER	Arduino Mega 2560
GSM MODULE	SIM 900A GSM
KEYPAD	4x4 Matrix Membrane Keypad
LCD DISPLAY	16 x 2-character LCD display
BUZZER	Active buzzer 5V
LED (BLUE, GREEN, RED)	5mm Round LED
RESISTOR	220 ohm Resistor
SERVO MOTOR	SG90 Servo - Micro Servo Motor
RFID READER WITH TAGS	REES52 MFRC-522

5V,2A ADAPTER, USB CABLE AND CONNECTING WIRES	Adapter For Gsm Module And Arduino Board Male to Male and Male to Female DuPont wires various lengths.
---	---

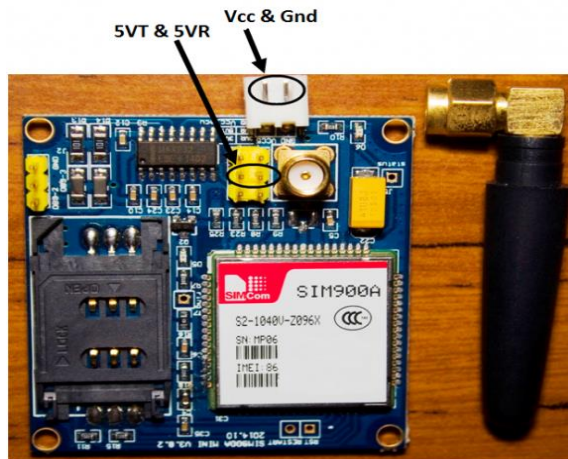
1. ARDUINO MEGA 2560

The Arduino Mega 2560 is a microcontroller board based on a ATmega2560 AVR microcontroller. It has 70 digital input/output pins (of which 15 can be used as PWM outputs and 16 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It can be powered via a USB cable or an AC-to-DC adapter or battery.

Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
USB Host Chip	MAX3421E
Length	101.52 mm
Width	53.3 mm
Weight	36 g

2. GSM SIM 900A

The SIM900A is a complete Dual-band GSM/GPRS solution in a SMT module which can be embedded in the customer applications. Featuring an industry-standard interface, the SIM900A delivers GSM/GPRS 900/1800MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mmx3mm, SIM900A can fit in compact demand of design.



INTERFACE	PIN	DESCRIPTION
Rst	1	Reset the SIM900 module
P	2	Power switch pin of SIM900 module
Tx	3	UART data output
Rx	4	UART data in
-	7	GND
+	8	VCC

3. KEYPAD

A **4X4 KEYPAD** will have **EIGHT TERMINALS**. In them four are **ROWS of MATRIX** and four are **COLUMNS of MATRIX**. These 8 PINS are driven out from 16 buttons present in the MODULE. Those 16 alphanumeric digits on the MODULE surface are the 16 buttons arranged in MATRIX formation.



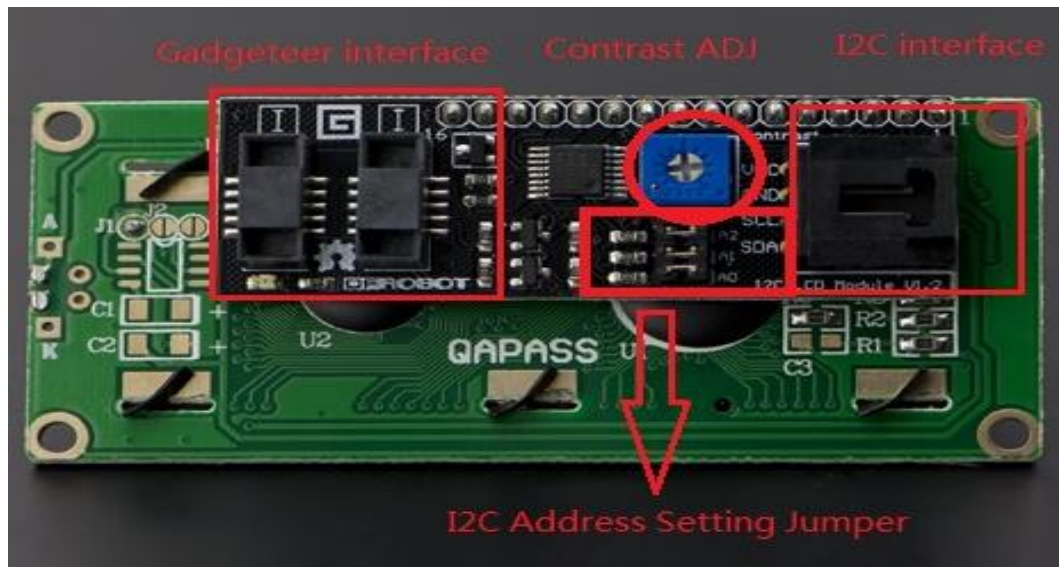
4. LCD

The 16x2 LCD display screen is able to display 16x2 characters on 2 lines, white characters on blue background. It uses an I2C communication interface. It needs only 4 pins for the LCD display: VCC, GND, SDA and SCL.

I2C protocol is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus, where data is transferred bit by bit along a single wire.

I2C only uses two wires to transmit data between devices:

- **SDA (Serial Data)** – The line for the master and slave to send and receive data.
- **SCL (Serial Clock)** – The line that carries the clock signal.



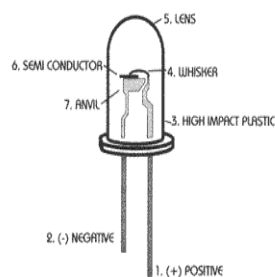
5. BUZZER

A **buzzer** is a small component to add sound features to a system. It is very small and compact 2-pin structure hence can be easily used on breadboard, buzzer can be used by simply powering it using a DC power supply ranging from 4V to 9V.



6. LED (BLUE, GREEN, RED)

A **light-emitting diode (LED)** is a semiconductor light source that emits light when current flows through it.



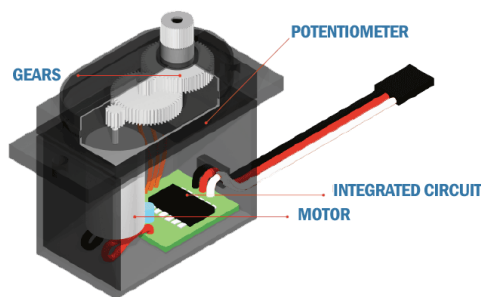
7. RESISTOR

In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.



8. SERVO MOTOR

A **servomotor** is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. The motor is paired with some type of encoder to provide position and speed feedback.



9. USB CABLE, 2 A ADAPTER AND JUMP WIRES

The arduino USB cable and the GSM module 2a adapter are power sources to boot the board on. A jump wire is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype



10. RFID READER WITH TAGS

RFID stands for radio frequency identification and it basically uses the radio waves to read the information on the tag. The RFID tags contain the embedded transmitter and receiver attached to an object. An RFID system consists of two parts: Tag and Reader

RFID TAG

RFID tag contains a chip for storing information about physical object and an antenna to receive and transmit a signal. A RFID tag can usually store 1KB of data but it is enough for storing the name, credit card number, unique identification number, birth date and some more information.



RFID Tags can be passive or active

- A **Passive tag** has no battery and it uses the energy transmitted by the reader.

- An **Active tag** contains a built in battery which makes it able to send a stronger signal and the range increases to 100 feet. Other features are same as the passive tags.

RFID READER

The RFID reader performs two functions: Transmit and receive. The RFID reader contains an antenna, radio frequency module and a control unit.

2.3 TECHNICAL PROFILE

ARDUINO IDE ENVIRONMENT

The **Arduino integrated development environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino board.

The Arduino IDE supports the languages C and C++ using special rules of code structuring.

FEATURES

- **Multi-Platform Application**

Arduino IDE works on the three most popular operating systems: Windows, Mac OS, and Linux. Aside from that, the application is also accessible from the cloud. These options provide programmers with the choice of creating and saving their sketches on the cloud or building their programs locally and upload it directly to the board.

- **Board Management**

Arduino IDE comes with a board management module, where users can select the board they want to work with at the moment. If they wish to change it, they can do so easily from the dropdown menu. Modifying their selection also automatically updates the PORT information with the data they need in relation to the new board.

- **Straightforward Sketching**

With Arduino IDE, users can create programs called sketches that are built with a text editor. The process is a straightforward

one though it has several bells and whistles that make the experience more interactive.

- **Project Documentation**

Arduino IDE offers programmers the option to document their projects. This function allows them to keep track of their advancements and any changes they make every time. Apart from that, documentations allow other people to easily employ the sketches to their own boards.

- **Simple Sketch Sharing**

Aside from saving and archiving sketches and uploading them to the board, Arduino IDE is also capable of sharing sketches (available only on the cloud version). Each sketch is given its own unique URL that users can share with their colleagues and fellow Arduino hobbyists. The recipient then has access to the code; they can save it in the cloud sketchbook or download it for their own use.

- **Vast Library**

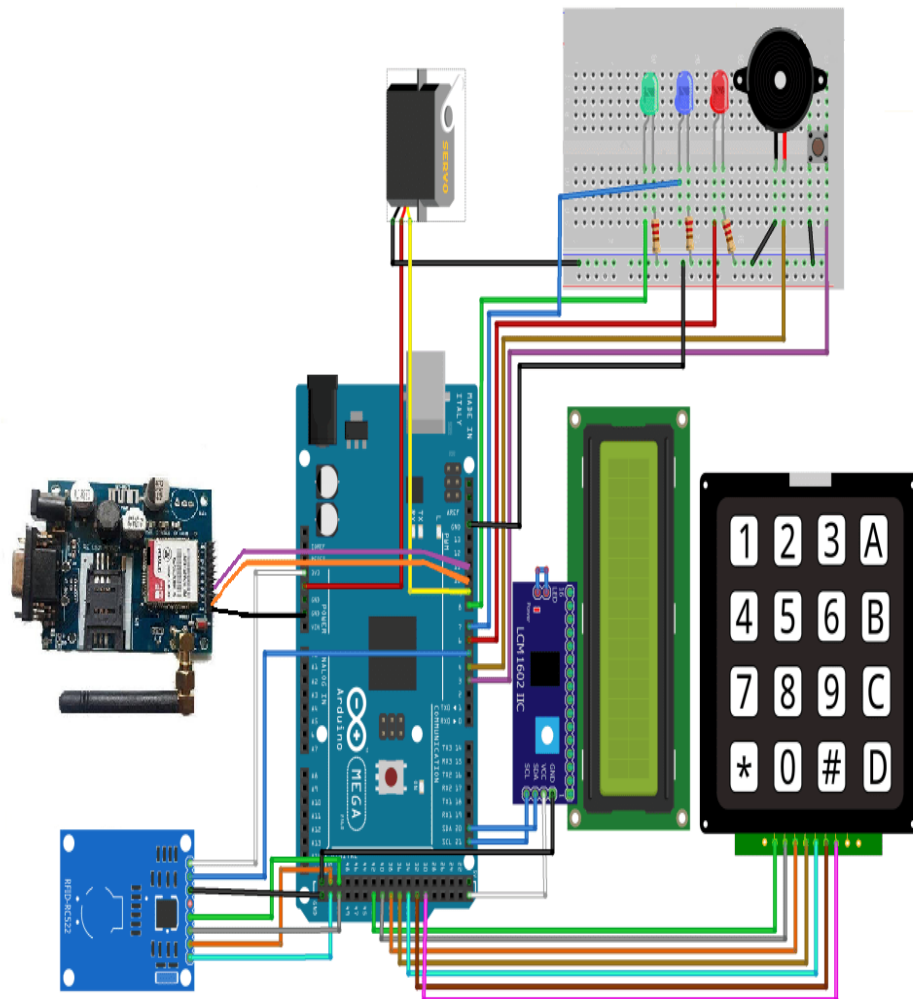
Arduino IDE has more than 700 libraries integrated. These were written and shared by members of the Arduino community that other users can utilize for their own projects without having to install anything. This enables programmers to add a different dimension to their sketches.

3. SYSTEM DESIGN

Systems design could be seen as the application of systems theory to product development.

- **ARCHITECTURAL DESIGN**

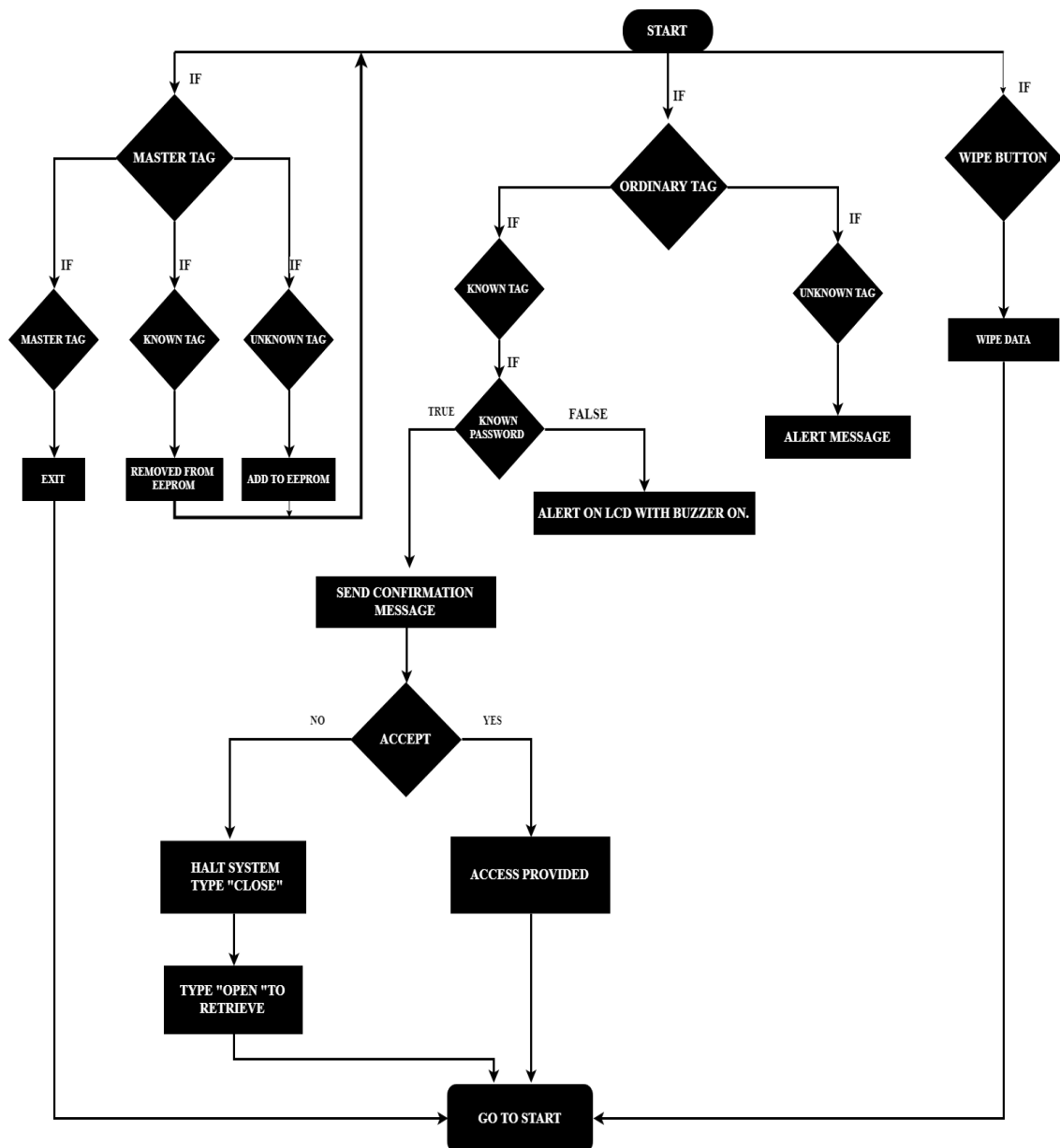
The architectural design of a system emphasizes the design of the system architecture that describes the structure, behaviour and more views of that system and analysis.



- **LOGICAL DESIGN**

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes

graphical) model of the actual system. In the context of systems, designs are included. Logical design includes entity-relationship diagrams (ER diagrams).

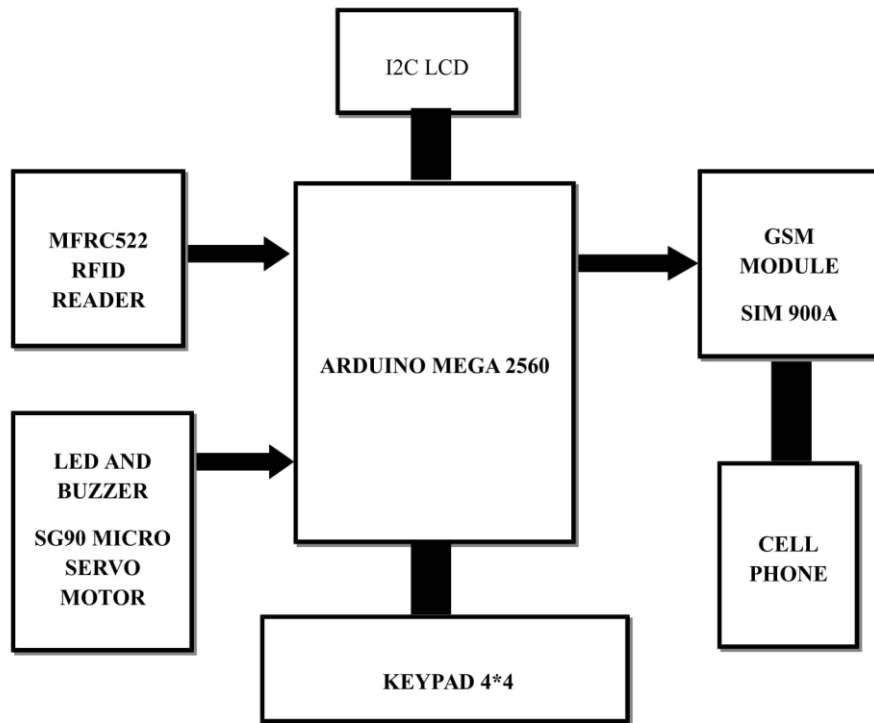


- **PHYSICAL DESIGN**

The physical design relates to the actual input and output processes of the system.

1. **Input requirement,**

2. Output requirements,
3. Storage requirements,
4. Processing requirements,
5. System control and backup or recovery.



4. PROJECT DESCRIPTION

The defined security model has the following modules

1. PROGRAM MODE
2. NORMAL MODE

PROGRAM MODE

It consist of the functionality to define cards and their respective passwords and to remove cards

NORMAL MODE

Enables user to scan cards and enter password. It's the normal functioning of the security system.

WORKING

Define a master tag and password for it. The master tag will act as a programmer and can be used to add or remove other tags. After defining the master tag, other tags can be defined which are use to open the door. To do this, scan the master tag and enter the password for it and it will take the system into **program mode**.

In the program mode, scanning the tags will **add/Remove** these from the system. Scan the tags that need to open the door and enter the password for that tag. The system will store the UID and password of these tags in the EEPROM. Scan the tag again to remove it from the EEPROM. To exit the program mode, scan the master tag.

Now scan the other tags that you have added in the system and enter their passwords to **open the door** and on opening the door lock, it will send us the **confirmation message**. On scanning the wrong tag or on entering the wrong password, the door will remain closed and it will send an **alert message to the owner**.

The owner can also **halt the system** by sending the 'close' message to Arduino. During this time, the system will only look for master tag and messages to unlock.

To **reset the system**, press the reset button of Arduino and then long press the wipe button for 10 seconds. This will remove all the data from the EEPROM including the master tag.

Hardware functioning of the system has been designed using microcontroller **AT MEGA 2560**, **I2C LCD** to display information, **MFRC522 RFID READER** to reads RFID tags for authentication, **SIM900A GSM MODULE** to sends alert messages to the owner, **SG90 MICRO SERVO MOTOR** to rotate door to lock/unlock, **4*4 KEYPAD** to input the valid password and a **BUZZER** for alarm.

The model has many methods which enhances the smooth functioning of the model

granted()-It provides a gesture for valid id and password by rotating the head of the servo and turning on the green led

denied()-It provides a gesture for invalid id and password by turning on the red led and buzzer beep.

getID()-It is used to read the Id of the tag in bytes and it also verifies if the entered card is new and checks if the cards serial detail is readable in these two cases it returns null.

ShowReaderDetails()-checks for the version of the rfid reader.

cycleLeds() and normalModeOn()-shows gestures by turning on/off led and buzzer for respective actions of granted/denied.

readID()-It reads the tags id in terms of bytes and stores it in an array variable.

writeID() and deleteID()-They are used to write and delete tags id from EEPROM.

checkTwo()-It is used to check similarity between two array variables.

findIDSLOT()-It is used to plot the slot number of the card in EERPOM.

findID()-It is used to get the stored position of the card.

isMaster()-It is used to check the specified card is a master card.

monitorWipeButton()-It provides functionality to check if the button is pressed within 10 seconds for wiping the memory of EEPROM completely.

ShowOnLCD()-It prints information on the lcd screen.

storePassword()-It is used to getting password using the getKey() and storing in the EEPROM memory.

Matchpass()-It compares stored password with the entered password.

receive_message()-It provides gsm to get the incoming message open/close.

Send_message()-It is used to send message to the owner.

The combined functioning of the methods with the protocol of each hardware makes the model function cohesively.

5. TESTING AND IMPLEMENTATION

1. SOFTWARE TESTING

Arduino provides tools to assist the implementation of a system. Arduino IDE enables serial communication with Arduino microcontroller through USB. Printing to serial console in the IDE helps to track function events between the hardware and the software layer.

Arduino community open source libraries assist the implementation of the sensors. In this implementation, libraries, which are used, are meant to simplify sensor data fetching and network connectivity.

2 .HARDWARE TESTING

AT MEGA 2560

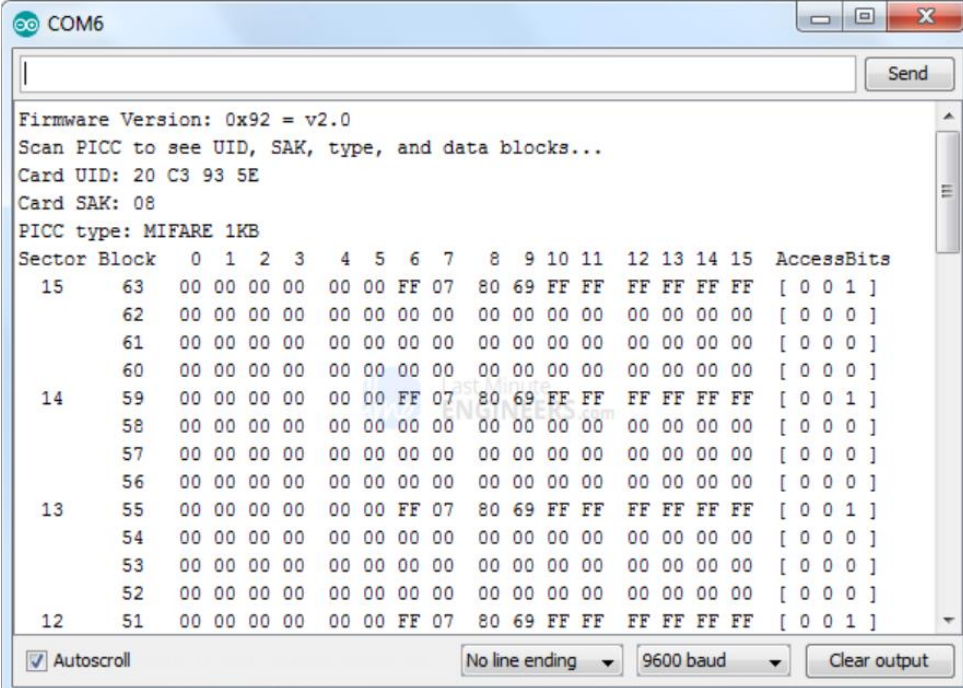
The USB connection with the PC is necessary to program the board and not just to power it up. The Mega2560 automatically draw power from either the USB or an external power supply. The green power LED (labelled PWR) should go on. A few seconds after the upload finishes, the pin 13 (L) LED on the board start to blink (in orange).

SIM900A GSM MODULE

LED STATUS DESCRIPTION		
LED	POWER STATE	DETAILS
PWR LED	ON	RED LED will light immediately
STS LED	ON	GREEN LED will light after 1-2 seconds
NET LED	ON	BLUE LED will starts to blink fast for few seconds(Searching For Network) and becomes slow blinking once the Modem registers with the Network.

MFRC522 RFID READER

When the tag is brought in contact with the reader the proper functioning can be seen via the serial monitor.



```
COM6
Firmware Version: 0x92 = v2.0
Scan PICC to see UID, SAK, type, and data blocks...
Card UID: 20 C3 93 5E
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 AccessBits
15    63  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [ 0 0 1 ]
      62  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
      61  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
      60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
14    59  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [ 0 0 1 ]
      58  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
      57  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
      56  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
13    55  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [ 0 0 1 ]
      54  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
      53  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
      52  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
12    51  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [ 0 0 1 ]

☒ Autoscroll  No line ending  9600 baud  Clear output
```

6. FUTURE ENHANCEMENTS

The existing model in future can be modelled completely into a home security system by adding up fire sensor, smoke sensor and temperature sensor to detect fire/gas leaks and alert to the rescue team and the owner can be sent via the gsm model

In order to capture live footages of the intruder Arucam infused with a wifi-model can be helpful to capture evidences against intruder.

In place of RFID reader finger print sensor can be used for more authentication of the model. Voice recognition model can be infused with the model for verification of the valid user.

Network speed can be improved from 3G to 4G to capture data packets at high speed.

7. CONCLUSION

The model is very robust with two way validation which can be kept for home security as well as bank locker security. Present models are designed only with one validation and concentrate on either monitoring or validating the user, but RAKGS focuses on real time validation and alert.

- ❖ **ROBUST SECURITY SYSTEM**-Protect Valuables.
- ❖ **REMOTE ACCESS**-Owner Has Access Over The Security System From Any Place.
- ❖ **REAL TIME DETECTION** -Omission of Fault Detection.

8. BIBLIOGRAPHY

WEB REFERENCES

1. <https://www.arduino.cc>
2. <https://electronics hobbyists.com>
3. <https://www.instructables.com>
4. <https://howtomechatronics.com>

9. APPENDICES

9.1 SAMPLE CODE

```
#include <EEPROM.h>  
#include <MFRC522.h>
```

```
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <Keypad.h>
#include <Servo.h>
#include <SPI.h>
#include <Wire.h>
SoftwareSerial SIM900(10, 11);
MFRC522 mfrc522(53, 5);
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo myServo;
constexpr uint8_t greenLed = 8;
constexpr uint8_t blueLed = 7;
constexpr uint8_t redLed = 6;
constexpr uint8_t ServoPin = 9;
constexpr uint8_t BuzzerPin = 4;
constexpr uint8_t wipeB = 3;
boolean match = false;
boolean programMode = false;
boolean replaceMaster = false;
uint8_t successRead;
byte storedCard[4];
byte readCard[4];
byte masterCard[4];
char storedPass[4];
char password[4];
char masterPass[4];
boolean RFIDMode = true;
boolean NormalMode = true;
char key_pressed = 0;
uint8_t i = 0;
const byte rows = 4;
const byte columns = 4;
char hexaKeys[rows][columns] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
```

```

};
byte row_pins[rows] = {43, 41, 39, 37};
byte column_pins[columns] = {35, 33, 31};
Keypad newKey = Keypad( makeKeymap(hexaKeys), row_pins,
column_pins, rows, columns);
void setup() {
pinMode(redLed, OUTPUT);
pinMode(greenLed, OUTPUT);
pinMode(blueLed, OUTPUT);
pinMode(BuzzerPin, OUTPUT);
pinMode(wipeB, INPUT_PULLUP);
digitalWrite(redLed, LOW);
digitalWrite(greenLed, LOW);
digitalWrite(blueLed, LOW);
lcd.backlight();
lcd.begin();
SPI.begin();
mfrc522.PCD_Init();
myServo.attach(ServoPin);
myServo.write(10);
SIM900.begin(19200);
ShowReaderDetails();
if (digitalRead(wipeB) == LOW) {
digitalWrite(redLed, HIGH);
lcd.setCursor(0, 0);
lcd.print("Button Pressed");
digitalWrite(BuzzerPin, HIGH);
delay(1000);
digitalWrite(BuzzerPin, LOW);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("This will remove");
lcd.setCursor(0, 1);
lcd.print("all records");
delay(2000);
lcd.clear();
lcd.setCursor(0, 0);

```

```

lcd.print("You have 10 ");
lcd.setCursor(0, 1);
lcd.print("secs to Cancel");
delay(2000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Unpres to cancel");
lcd.setCursor(0, 1);
lcd.print("Counting: ");
bool buttonState = monitorWipeButton(10000);
if (buttonState == true && digitalRead(wipeB) == LOW) {
  lcd.clear();
  lcd.print("Wiping EEPROM...");
  for (uint16_t x = 0; x < EEPROM.length(); x = x + 1)
    if (EEPROM.read(x) == 0) {
      }
    else {
      EEPROM.write(x, 0);
    }
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Wiping Done");
  cycleLeds();
}
else {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Wiping Cancelled");
  digitalWrite(redLed, LOW);
}
}
if (EEPROM.read(1) != 143) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("No Master Card ");
  lcd.setCursor(0, 1);

```



```

lcd.print("Defined");
delay(2000);
lcd.setCursor(0, 0);
lcd.print("Scan A Tag to ");
lcd.setCursor(0, 1);
lcd.print("Define as Master");
do {
  successRead = getID();
  digitalWrite(blueLed, HIGH);
  digitalWrite(BuzzerPin, HIGH);
  delay(200);
  digitalWrite(BuzzerPin, LOW);
  digitalWrite(blueLed, LOW);
  delay(200);
}
while (!successRead);
for ( uint8_t j = 0; j < 4; j++ ) {
  EEPROM.write( 2 + j, readCard[j] );
}
EEPROM.write(1, 143);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Master Defined");
delay(2000);
storePassword(6);
}
for ( uint8_t i = 0; i < 4; i++ ){
  masterCard[i] = EEPROM.read(2 + i);
  masterPass[i] = EEPROM.read(6 + i);
}
ShowOnLCD();
cycleLeds();
SIM900.print("AT+CMGF=1\r");
delay(100);
SIM900.print("AT+CNMI=2,2,0,0,0\r");
delay(100);
}

```

```

void loop () {
  if (NormalMode == false) {
    receive_message();
    getID();
    if ( isMaster(readCard) ) {
      NormalMode = true;
      RFIDMode == true;
      cycleLeds();
    }
  }
  else if (NormalMode == true && RFIDMode == true) {
    do {
      receive_message();
      if (NormalMode == false) {
        break;
      }
      successRead = getID();
      if (programMode) {
        cycleLeds();
      }
      else {
        normalModeOn();
      }
    }
    while (!successRead );
    if (programMode && RFIDMode == true) {
      if ( isMaster(readCard) ) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Exiting Program Mode");
        digitalWrite(BuzzerPin, HIGH);
        delay(1000);
        digitalWrite(BuzzerPin, LOW);
        ShowOnLCD();
        programMode = false;
        return;
      }
    }
  }
}

```

```

else {
  if ( findID(readCard) ) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Already there");
    deleteID(readCard);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Tag to ADD/REM");
    lcd.setCursor(0, 1);
    lcd.print("Master to Exit");
  }
  else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("New Tag,adding...");
    writeID(readCard);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Scan to ADD/REM");
    lcd.setCursor(0, 1);
    lcd.print("Master to Exit");
  }
}
}
else {
  if ( isMaster(readCard)) {
    programMode = true;
    matchpass();
    if (programMode == true) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Program Mode");

      uint8_t count = EEPROM.read(0);
      lcd.setCursor(0, 1);
      lcd.print("I have ");

```

```

lcd.print(count);
lcd.print(" records");
digitalWrite(BuzzerPin, HIGH);
delay(2000);
digitalWrite(BuzzerPin, LOW);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Scan a Tag to ");
lcd.setCursor(0, 1);
lcd.print("ADD/REMOVE");
}
}
else {
if ( findID(readCard) ) {
RFIDMode = false;
ShowOnLCD();
}
else {
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Access Denied");
denied();
send_message("Someone Tried with the wrong tag");
ShowOnLCD();
}
}
}
}
else if (NormalMode == true && RFIDMode == false) {
key_pressed = newKey.getKey();
if (key_pressed)
{
password[i++] = key_pressed;
lcd.print("*");
}
if (i == 4)
{

```

```
delay(200);
if (!(strcmp(password, storedPass, 4)))
{
  lcd.clear();
  lcd.print("Pass Accepted");
  lcd.setCursor(0, 1);
  lcd.print("Door Opened");
  granted();
  send_message("Door Opened \n If it wasn't you, type 'close'");
  RFIDMode = true;
  ShowOnLCD();
  i = 0;
}
else
{
  lcd.clear();
  lcd.print("Wrong Password");
  denied();
  send_message("Someone Tried with the wrong tag");
  RFIDMode = true;
  ShowOnLCD();
  i = 0;
}
}
}
}
```