Electronic Door Lock Using RFID & GSM

With the rising interest in security, frameworks with high reliability and fast response times are essential for industries and enterprises. Radio-Frequency Identification (RFID) is one of the most significant areas of future development and is gaining substantial attention from both the scientific community and industry. In this work, an RFID-based door access control system using Arduino is developed. The system utilizes RFID ID tags and an RFID reader to compare the scanned tag data with the stored data in the microcontroller. Based on the validation, the door is operated using a servo motor or solenoid lock mechanism.

The main aim of the project is to design an RFID-based door locking/unlocking system using Arduino, where only registered individuals can gain access using their RFID cards. The central control unit is the Arduino UNO microcontroller, interfaced with RFID reader and tags, relay module, solenoid lock, buzzer, and a **GSM module**.

When a user scans their RFID tag on the reader, the microcontroller checks whether the card is registered. If it is valid, the system unlocks the door via the relay and solenoid lock, allowing access. If the tag is unrecognized, the system activates a buzzer as a security alert and **sends an SMS notification via the GSM module** to a predefined mobile number, alerting the concerned authority of the unauthorized access attempt. The relay functions as a switch to lock or unlock the door. The entire process is managed by an Arduino program written in embedded C.

Main objective of this project:

- To lock/unlock the door using RFID technology.
- This system gives a buzzer if an unauthorized person tries to access the door.
- Relay based switching mechanism.
- Using Arduino to achieve this task.

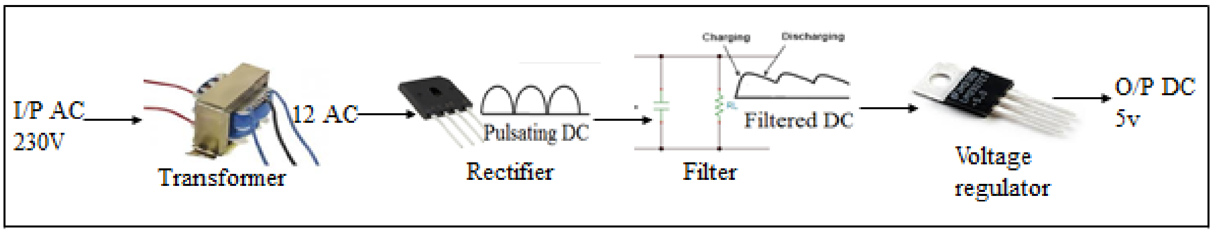
The main building blocks of the project are:

- Arduino microcontroller.
- Adapter power supply.
- RFID reader.
- Relay.
- Solenoid lock.
- GSM.
- Buzzer.

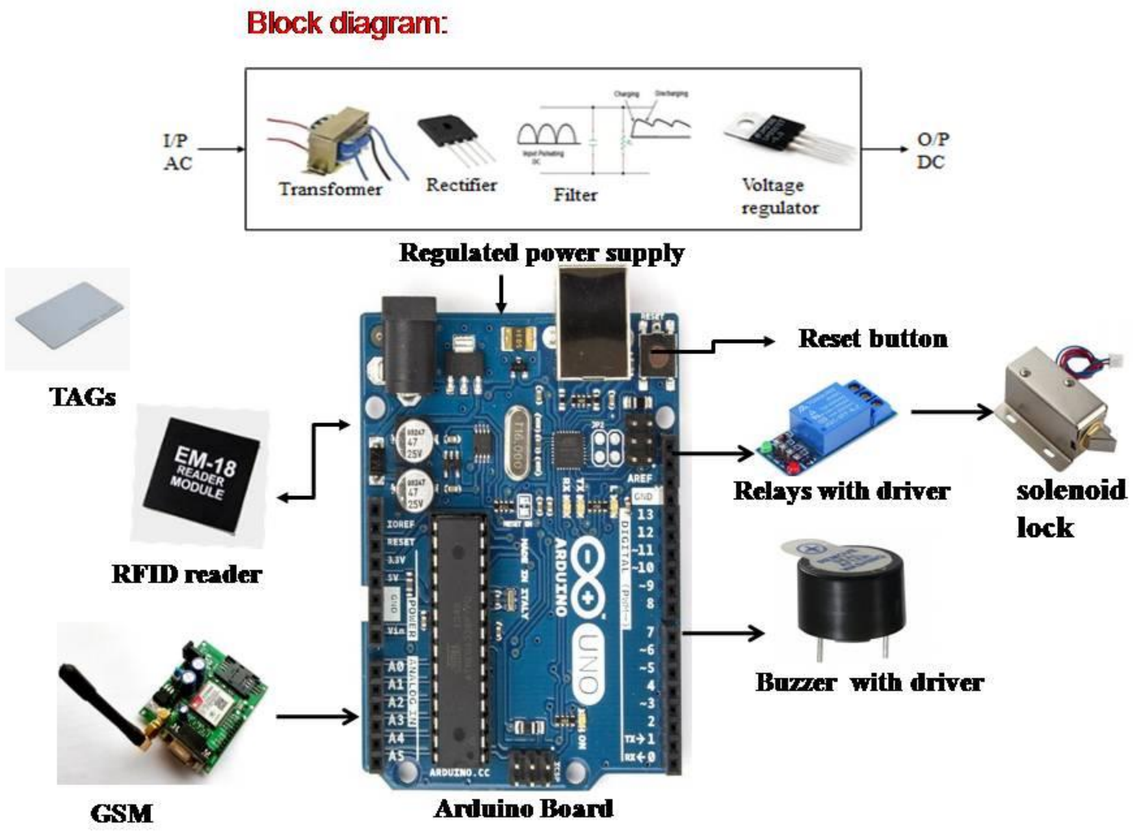
Software's used:

1. Arduino IDE for compiling and dumping code into controller
2. Express SCH for Circuit design.

Regulated power supply:



Block diagram:



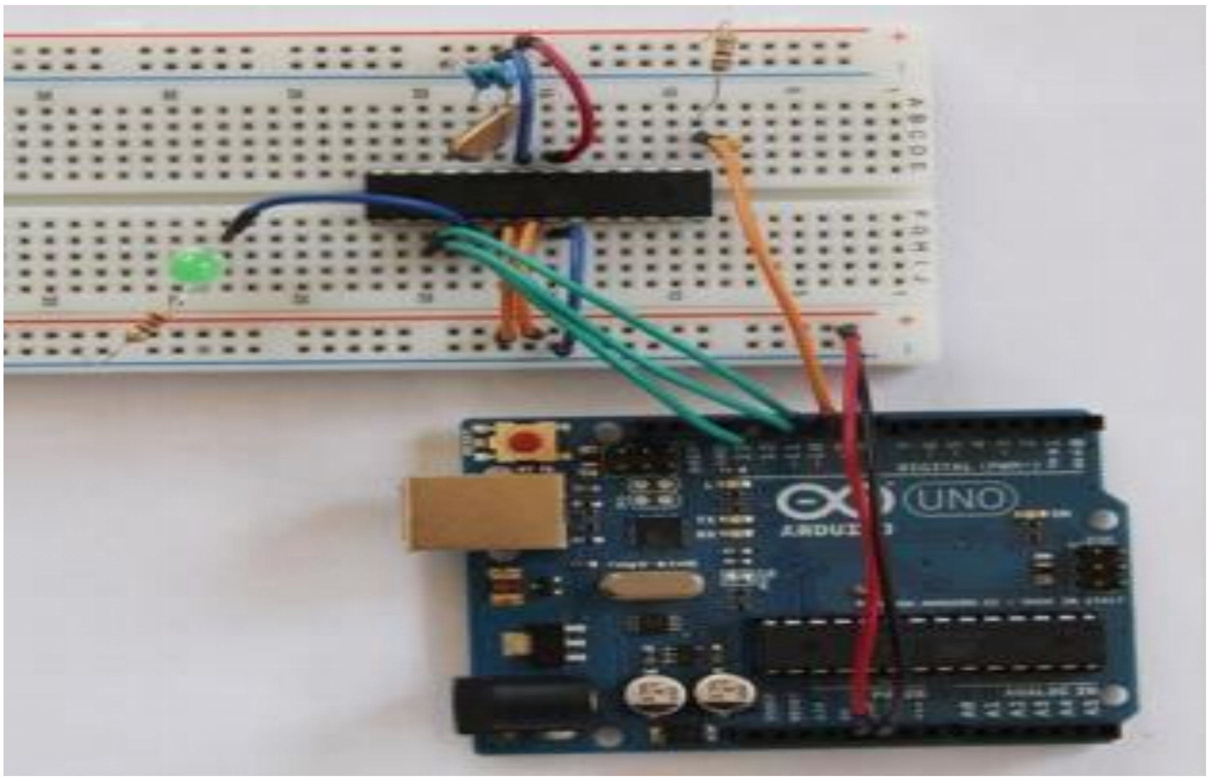
PROCEDURAL STEPS FOR COMPILATION, SIMULATION & DUMPING:

COMPILATION & SIMULATION STEPS:

STEP 1: PARTS

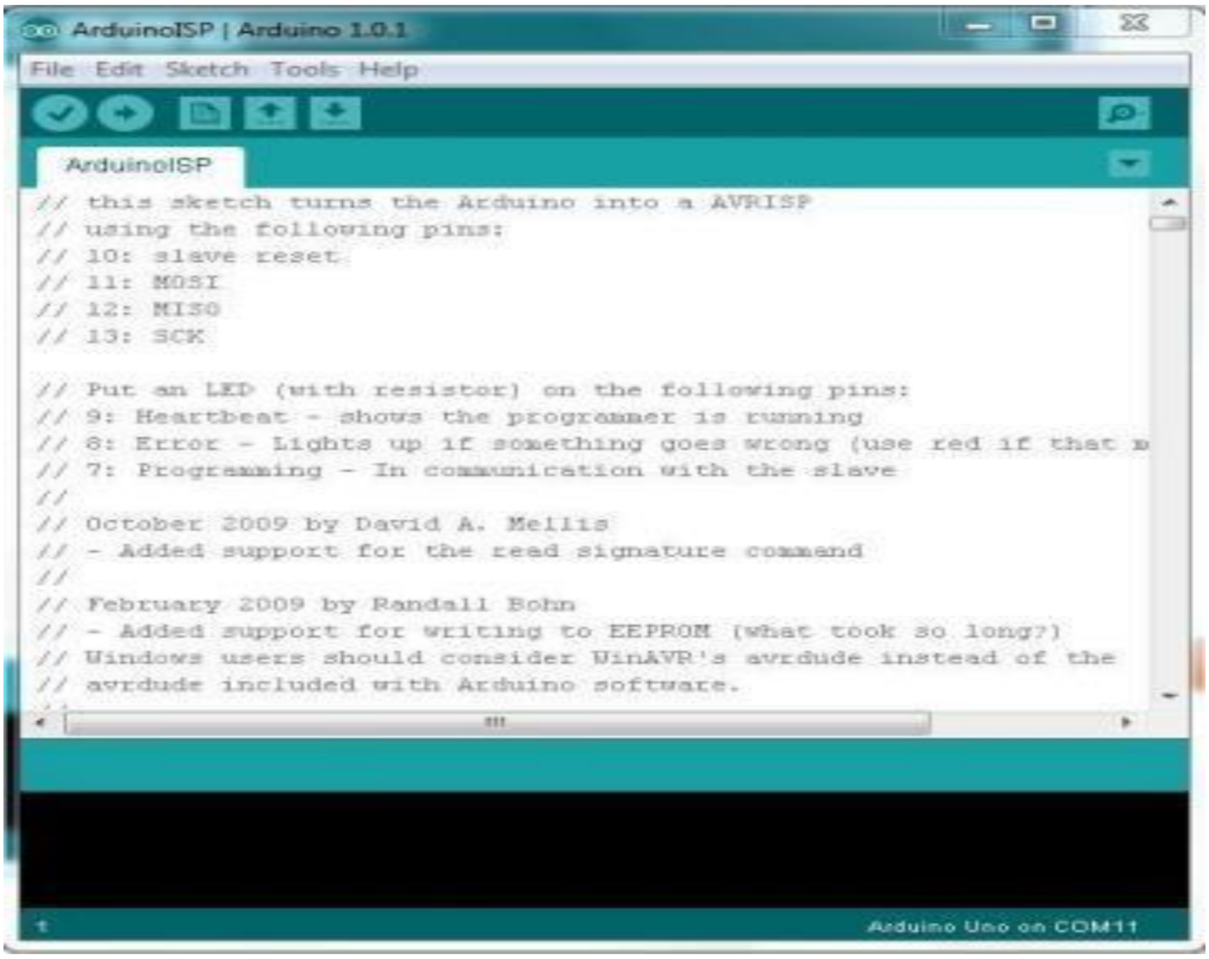
1 x Arduino on a Breadboard
1 x Arduino UNO
Connecting Wires
Arduino IDE installed on your PC

STEP 2: THE APPROACH



We use the Arduino UNO to boot-load the ATmega328 that is sitting on the Arduino-on-a- Breadboard. This is fairly straightforward having an ATmega328P-PU, but needs an extra step for an ATmega328-PU.

STEP 3: PROGRAM YOUR ARDUINO UNO AS AN ISP



```
// this sketch turns the Arduino into a AVRISP
// using the following pins:
// 10: slave reset
// 11: MOSI
// 12: MISO
// 13: SCK

// Put an LED (with resistor) on the following pins:
// 9: Heartbeat - shows the programmer is running
// 8: Error - Lights up if something goes wrong (use red if that m
// 7: Programming - In communication with the slave
//
// October 2009 by David A. Mellis
// - Added support for the read signature command
//
// February 2009 by Randall Bohn
// - Added support for writing to EEPROM (what took so long?)
// Windows users should consider WinAVR's avrdude instead of the
// avrdude included with Arduino software.
```

We need to program the Arduino UNO to act as an ISP (In-System Programmer), so that it can burn the bootloader onto the Breadboard chip.

1. Open the Arduino IDE
2. Open the Arduino ISP sketch (under File, Examples)
3. If you're using version 1.0 of the IDE:

Search for void heartbeat and change the line that reads:

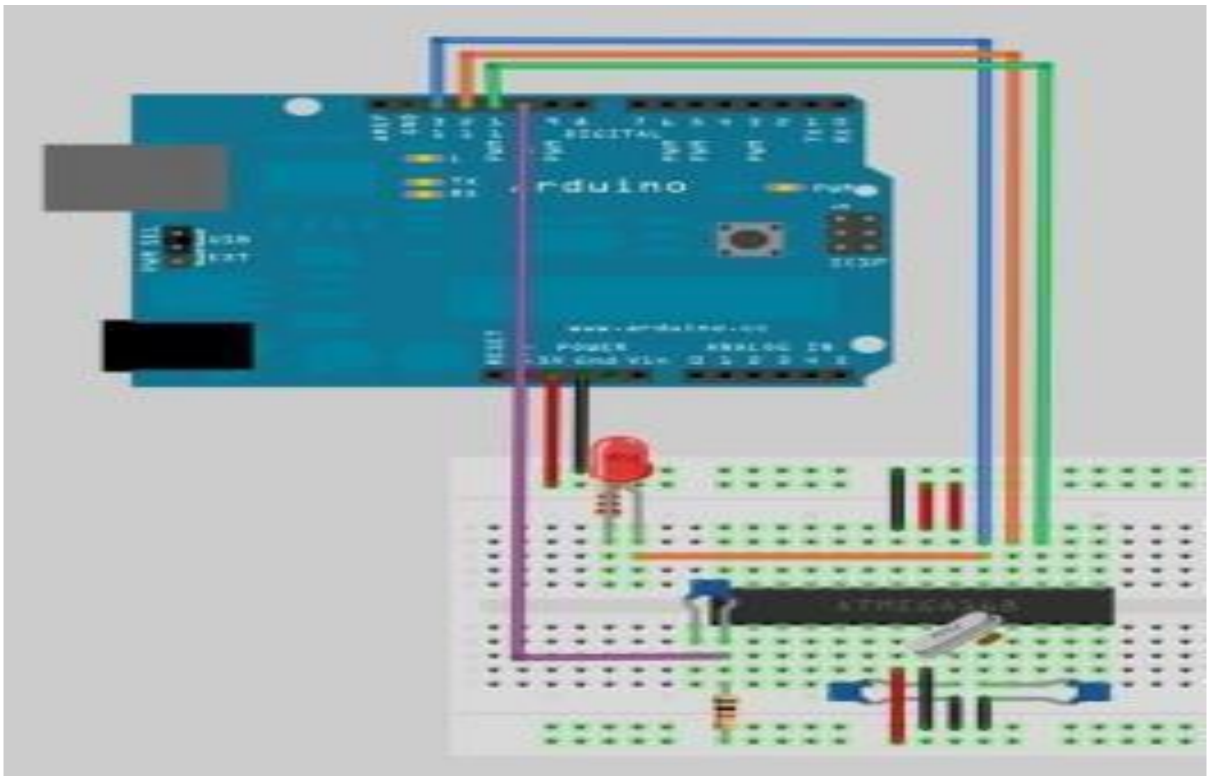
delay (40);

to

delay (20);

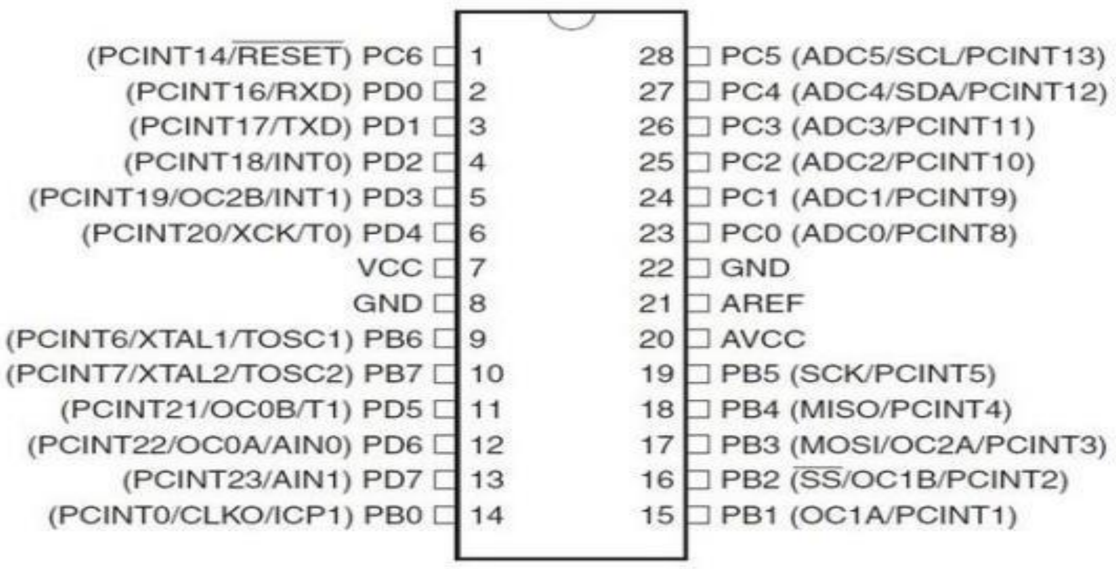
Connect your UNO to the PC, making sure it's not connected to the Arduino on a Breadboard. Ensure your UNO is selected under the Boards menu option, and upload the sketch.

STEP 4: CONNECT YOUR ATmega328



Now connect your ATmega to your UNO as follows:

- UNO 5v ---> ATmega pin 7 (VCC)
- UNO GND ---> ATmega pin 8 (GND)
- UNO pin 10 ---> ATmega pin 1 (RESET)
- UNO pin 11 ---> ATmega pin 17 (MOSI)
- UNO pin 12 ---> ATmega pin 18 (MISO)
- UNO pin 13 ---> ATmega pin 19 (SCK)



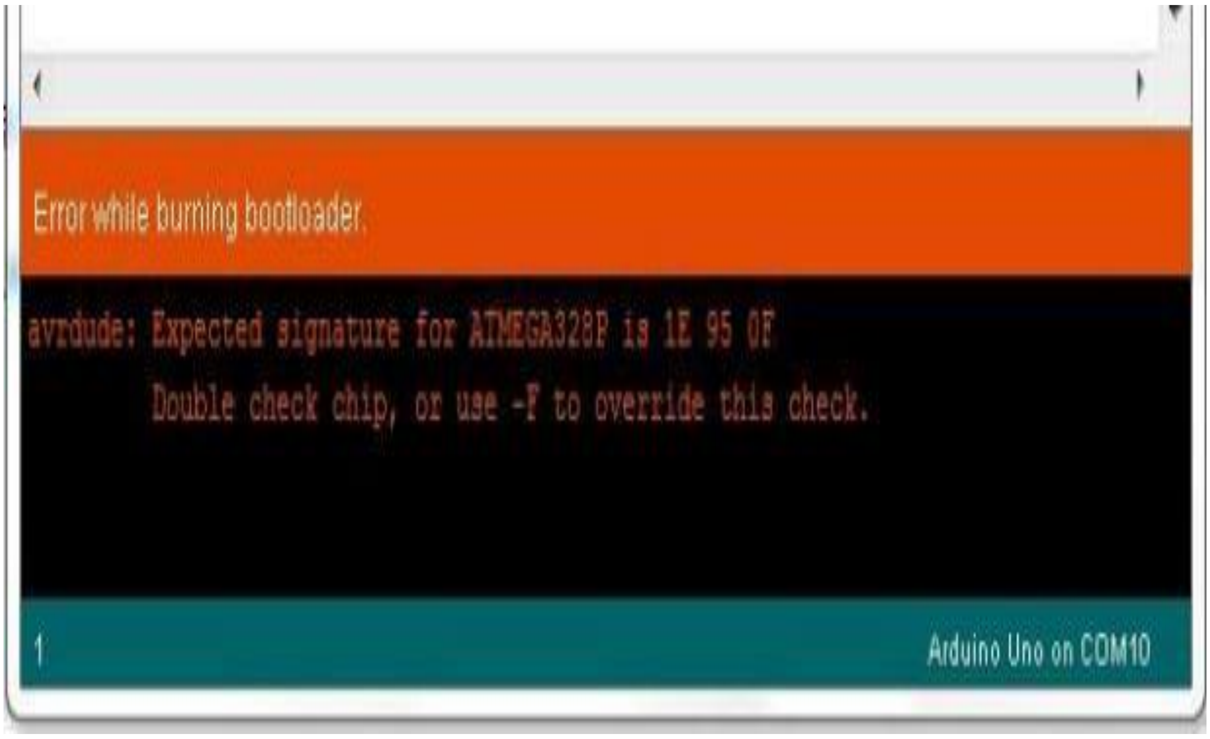
STEP 5: WHICH ATmega328 ARE YOU USING?

I learnt the hard way that there is more than one type of ATmega328. The two variants that are of interest to us are the ATmega328-PU and the ATmega328P-PU. The **-PU** suffix means that the chips are in a PDIP package, the format we need for our breadboard.

The **328P** is a pico power processor, designed for low power consumption, and is used on the Arduino boards. Given low power consumption this is first choice. The **328** does not have pico Power technology, and is not used on the Arduino boards – and is not explicitly supported by the Arduino IDE. What this means is that we can easily boot load the ATmega328P, but not the ATmega328. Unfortunately the websites that sell these chips don't always differentiate between them and forums are filled with people struggling to use the ATmega328-PU. Luckily there is a workaround - take a look at my Crash Bang website.

STEP 6: ATmega328-PU WORK AROUND

Each microprocessor has a *signature* – a unique code that identifies its model. When you boot load a chip (or even upload a sketch) the Arduino IDE checks that the chip selected matches the type it's connected to. Even though the ATmega328-PU in essence functions in the same way as the ATmega328P-PU, it has a different signature, and one that isn't recognised by the Arduino IDE.



Error while burning bootloader.

```
avrdude: Expected signature for ATMEGA328P is 1E 95 0F
         Double check chip, or use -F to override this check.
```

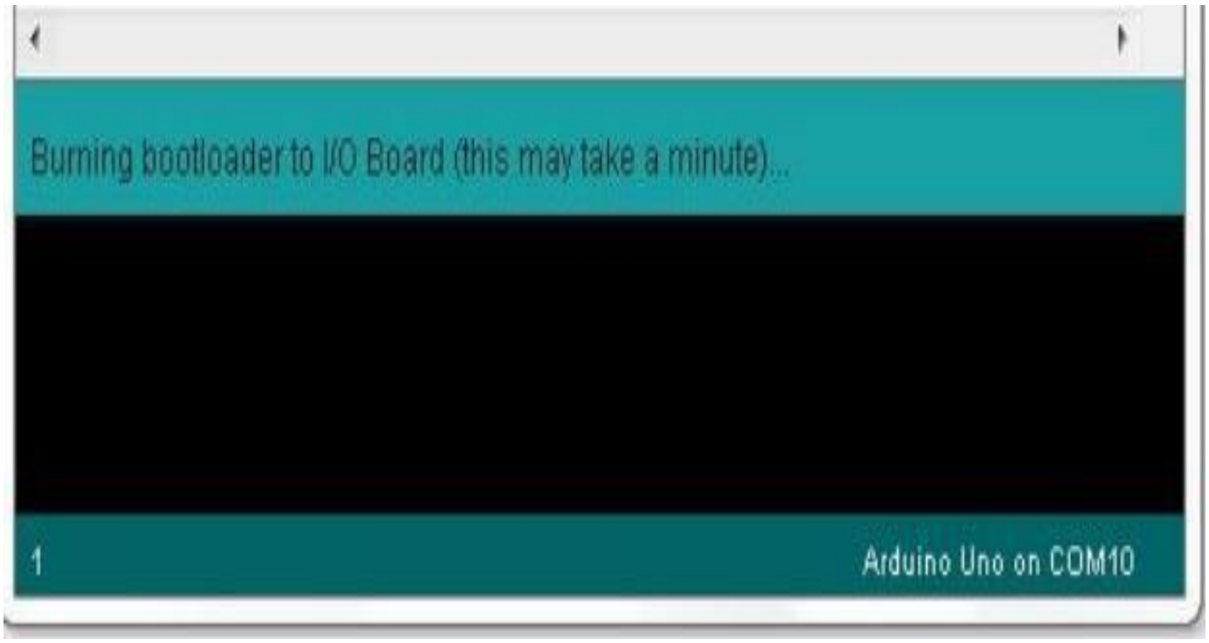
In your Arduino folder, find the subfolder:**..\hardware\tools\avr\etc**

1. Make a backup copy of the file: **avrdude.conf**
2. Open the file **avrdude.conf** in a text editor
3. Search for: "**0x1e 0x95 0x0F**" (this is the ATmega328P signature)
4. Replace it with: "**0x1e 0x95 0x14**" (this is the ATmega328 signature)
5. Save the file
6. Restart the Arduino IDE
7. Continue with the rest of the steps in the instruct-able, and once boot loading is complete restore the backup copy you made.

STEP 7: BOOTLOAD THE ATmega328

In the Arduino IDE, from the **Tools** menu:

- under the **Board** option choose **Arduino UNO**
- under the **Serial Port** option ensure the correct port is selected
- under the **Programmer** option choose **Arduino as ISP**

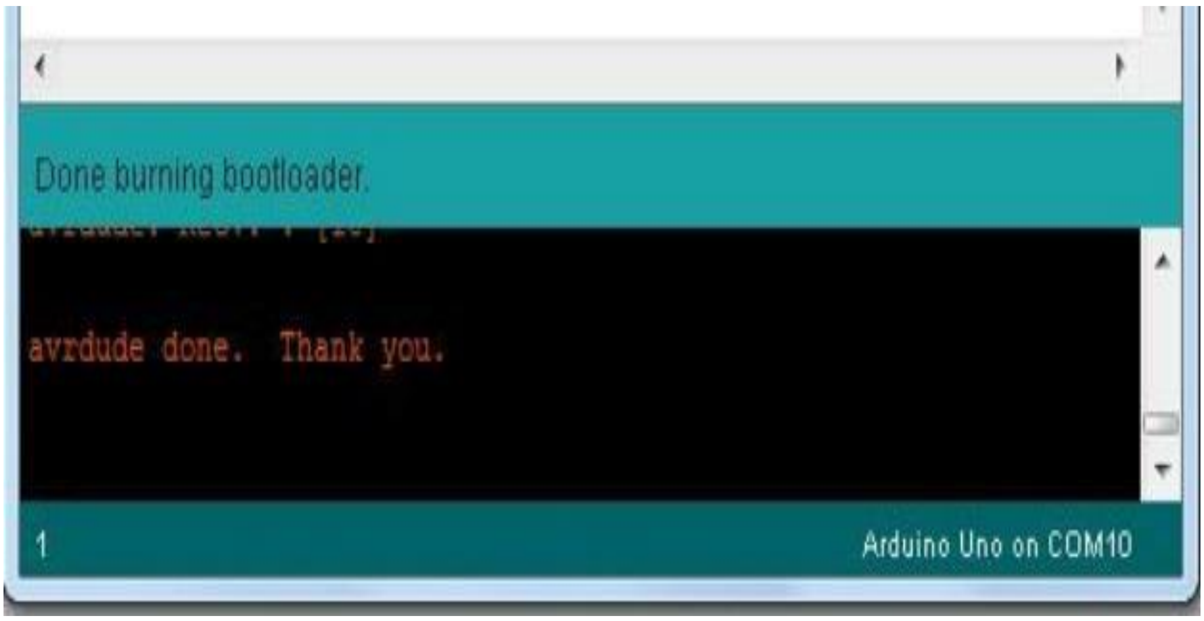


Burning bootloader to I/O Board (this may take a minute)...

1

Arduino Uno on COM10

To burn the Bootloader, choose **Burn Bootloader** from the **Tools** menu You should see a message **"Burning bootloader to I/O Board (this may take a minute)"** Once the bootloader has been burned, a message of confirming the success gets displayed. **"Congratulations: You're now ready to load sketches onto your Arduino on a breadboard!"**



Done burning bootloader.

```
avrdude done.  Thank you.
```

1

Arduino Uno on COM10

6.2 SAMPLE CODE

```
#include <stdio.h>
#include <string.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial (8, 9); //  RX, TX
char ch;
int led = 13;
int relay = 10;
int count = 0;
char *c1="160052EE72D8";
void setup()
{
  mySerial.begin(9600);
  Serial.begin(9600);
  pinMode(led,OUTPUT);
  pinMode(relay,OUTPUT);
  digitalWrite(led,1);
  delay(700);
  digitalWrite(led,0);
  delay(700);
  digitalWrite(led,1);
  delay(700);
  digitalWrite(led,0);
}
void loop()
{
  String ddata = "";
  int flag = 0;
  while(1)
```

```
{
   count = 0;
   delay(160);
   if (Serial.available() > 0)
   {
    i = 0;
    while(1)
    {
      char inChar = Serial.read();
       if(strcmp(inChar, c1) == 0)
       {
          digitalWrite(13, LOW);   // turn the LED on (HIGH is the voltage level)
          digitalWrite(relay, HIGH);   // turn the LED on (HIGH is the voltage level)
           mySerial.println("AT+CMGF=1");
           delay(200);
           mySerial.println("AT+CMGS=\"+919100326021\"");
           delay(300);
           mySerial.println("*** Authorised ***");
           delay(200);
          delay(3000);
          digitalWrite(relay, LOW);   // turn the LED on (HIGH is the voltage level)
          break;
       }
       else
       {
          digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
           mySerial.println("AT+CMGF=1");
           delay(200);
           mySerial.println("AT+CMGS=\"+919100326021\"");
           delay(300);
```

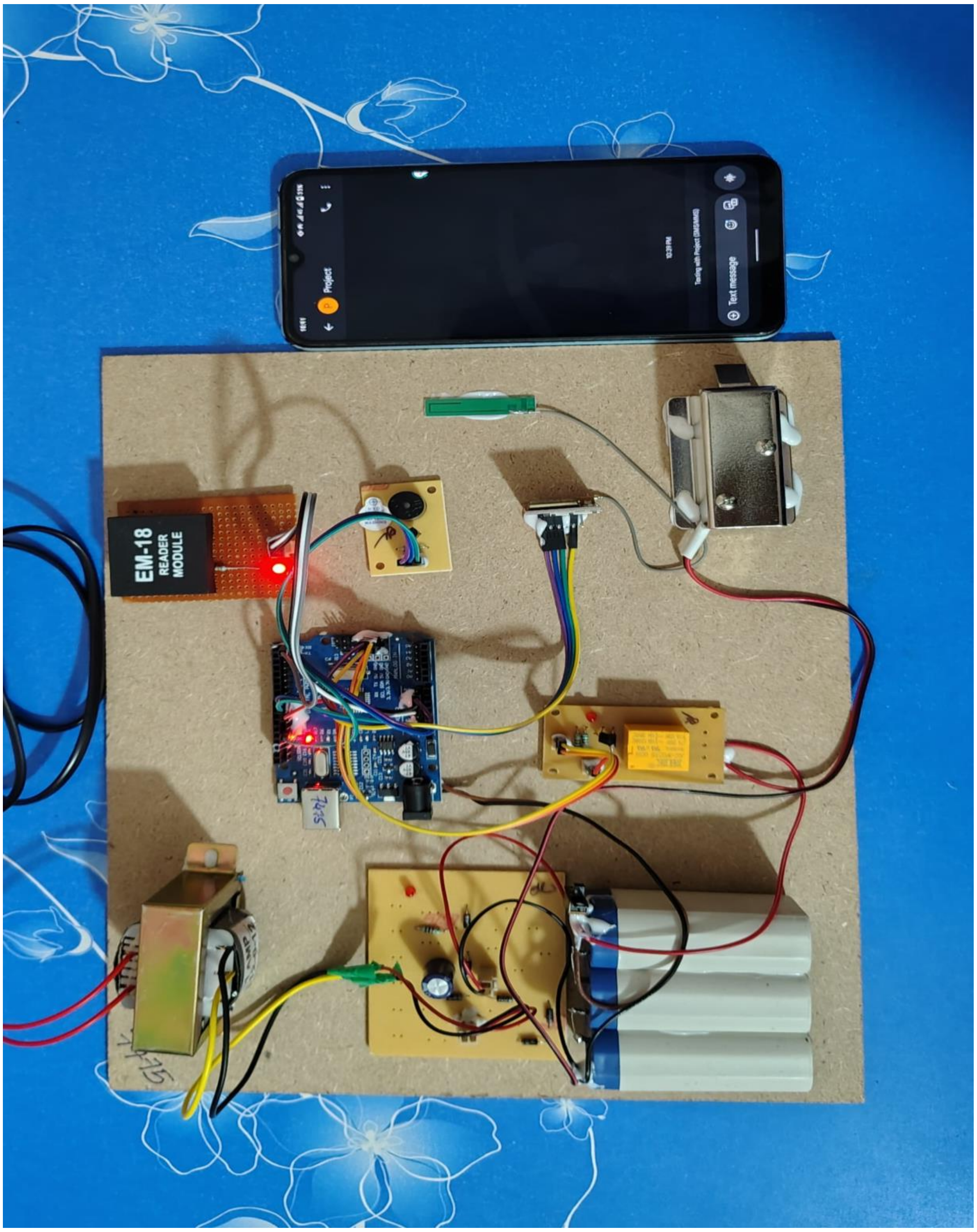
```
                 mySerial.println("***ALERT*** Un-Authorised ");
                 delay(200);
                digitalWrite(13, LOW);   // turn the LED on (HIGH is the voltage level)
                break;

              }
             }
           }
          }
     }
    }
```

TESTING & TEST CASES

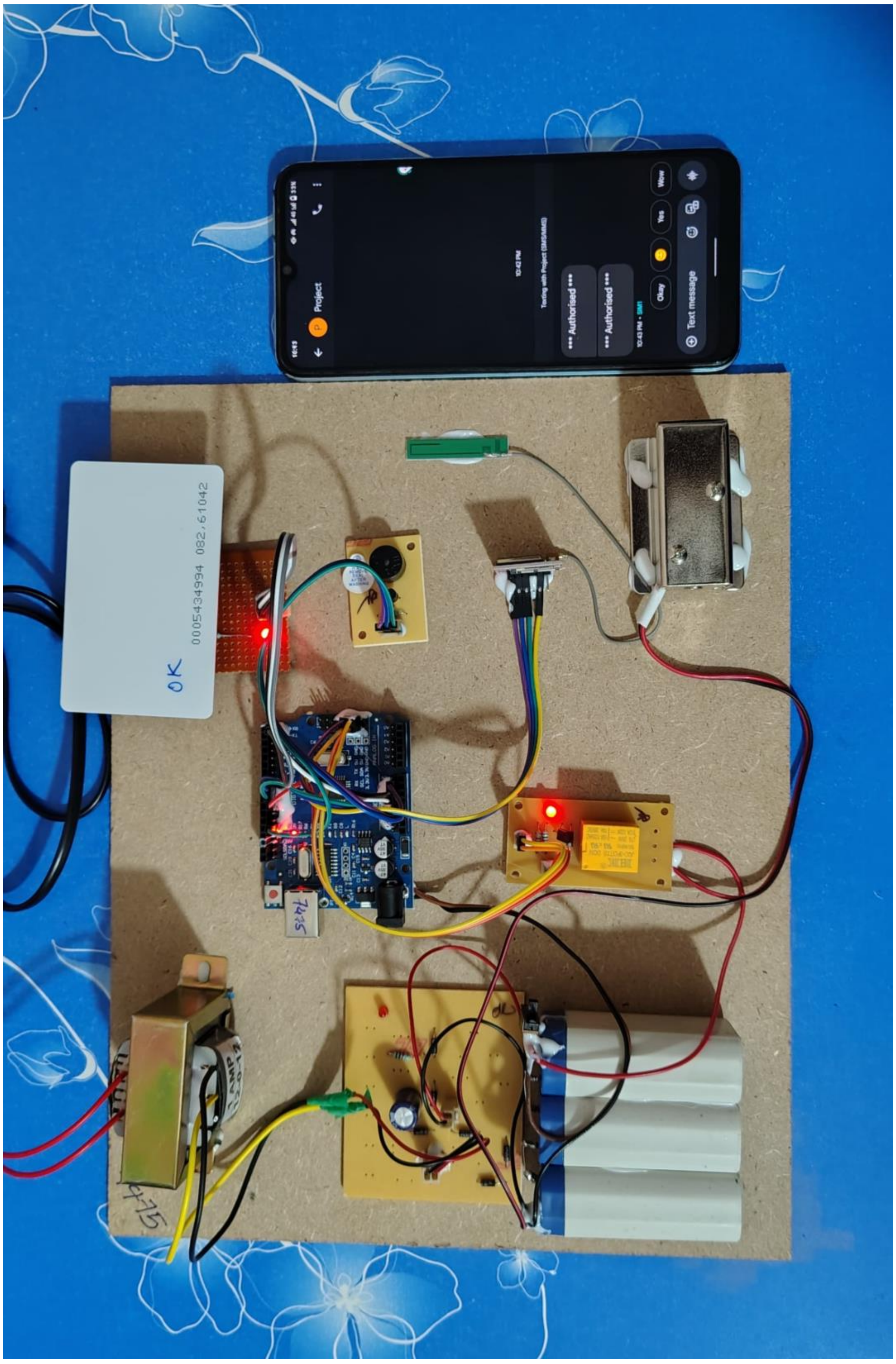
S.NO	TEST CASES	TESTCASE DESCRIPTION	EXPECTED OUTPUT	OBTAINED OUTPUT	TEST RESULT
1.	Relay Switch On	Using an Authorized card tap on Reader	Give Access	Access Granted	Pass
2	Relay Switch Off	Using an Authorized card tap on Reader	No Access	Access Denied	Pass
3	Buzzer	If unauthorize card is Tap on Reader	Activate Buzzer	Buzzer Activates	Pass
4	Buzzer	If authorize card is Tap on Reader	No Buzzer	Buzzer not Activated	Pass
5	RFID-Reader	Tap on reader	Read Card	Card Readed	Pass

OUTPUT SCREENS

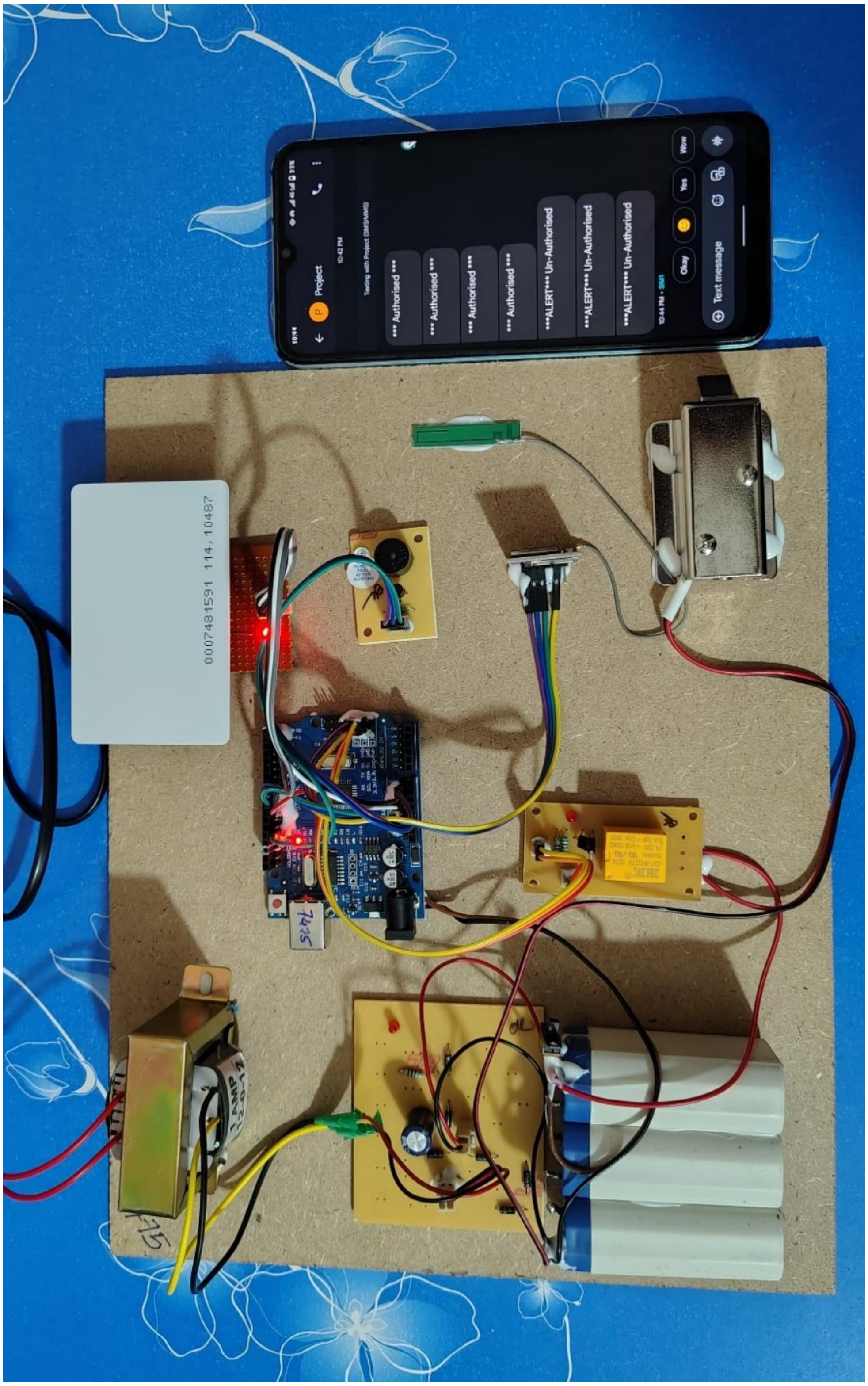
1.Power supply ON, Led Blinks and System initializes.



2.Using an Authorized card tap on Reader



3.Using an Unauthorized card tap on Reader



Project

10:42 PM

Texting with Project (SMS/MMS)

*** Authorised ***

*** Authorised ***

*** Authorised ***

*** Authorised ***

ALERT Un-Authorised

ALERT Un-Authorised

ALERT Un-Authorised

10:44 PM · SIM1

CONCLUSION

In conclusion, the **Electronic Door Lock System using RFID and Arduino** provides an efficient, secure, and cost-effective solution for access control. By integrating **RFID technology** with the **Arduino microcontroller**, the system ensures that only authorized users can access a restricted area through RFID tag validation. The use of additional components like a **relay-controlled solenoid lock**, **buzzer for feedback**, and an optional **GSM module** for remote alerts enhances the system's reliability and functionality. The design is scalable, energy-efficient, and easy to modify, making it suitable for **homes, offices, laboratories**, and other secure environments. Overall, this project demonstrates how embedded systems and automation can significantly improve traditional security systems, laying a strong foundation for further innovations such as **IoT-based door locks** and **multi-factor authentication**. Integrating features of all the hardware components used have been developed in it. The presence of every module has been reasoned out and placed carefully, thus contributing to the best working of the unit. Secondly, the project has been successfully implemented using highly advanced ICs with the help of growing technology. Thus, the project has been successfully designed and tested.