Rafik Hariri University
جامعة رفيق الحريري

MECHATRONICS SYTEM DESIGN – MECA443L

PROJECT: HOME AUTOMATION SYSTEM

Adam Manasfi[1], Etaf Al-Zarif[1], Malek Itani[1], Mahmood Al Wattar[1]
[1]Rafik Hariri University, College of Engineering, Department of Mechanical and Mechatronics

ABSTRACT

The objective of this project is design and build a home automation system prototype for paralyzed individuals. The entire system was controlled using MIT App Inventor, a visual programming platform that allowed us to easily create a custom mobile application to control all the actuators.

Keywords: DC Motor, MIT App Inventor, Limit Switch, voice-controlled

1. INTRODUCTION

The goal of this project is to design and build a home automation system that will enhance the lives of people with special needs. Specifically, we aim to create a system that will cater to the needs of paralyzed individuals. The project aims to develop a complete home automation system prototype that will allow users to control various aspects of their house, such as light, gates, doors, water level, and elevator all through a mobile application. This home also includes a security system that can detect any unwanted movements in the area and alert the owner, and a voice-controlled fountain and lighting.

We used will Arduino IDE (or other IDEs) and MIT App Inventor software to build the complete system. We used AutoCAD to design our house prototype that meets the specific constraints, maximum length, width, and depth of 70 cm, 60 cm, and 60 cm, respectively. The design of the house is unique, functional, and logical.

2. MATERIALS AND METHODS

2.1 Methods

The materials used in this experiment are: Arduino NANO, L298N motor drive, 12V Dc motor geared, 12V pump, 5V servo, LED strips, Fly Swatter, Relay, Solar charge controller 30A, hc05 Bluetooth module, sound sensor, Limit Switch, jumper wires.

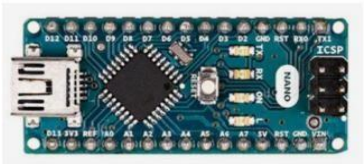


Figure 1: ARDUINO NANO

Arduino Nano: is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. The Arduino NANO board has 22 digital input/output pins and 8 analog input pins including 6 PWM output pins. Its input voltage is 7 to 12V and it operates at 5V. [3]

L298N Motor Drive: This dual bidirectional motor driver, is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily

and independently control two motors of up to 2A each in both directions. It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. [4]

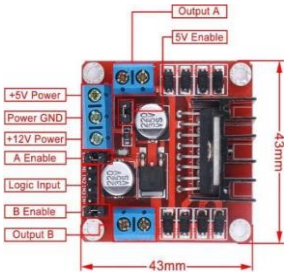


Figure 2: L298N MOTOR

Limit Switch: The limit switch is a mechanical device that requires the physical contact of an object with the switch's actuator to make the contact change state (open/closed). It can be used as part of a large control system, as a safety interlock, or as a counter enumerating object passing a point. It has a "normally open", "normally closed", and a "COM" terminal. When a certain limit is reached, the actuator can deactivate or activate a device to prevent malfunctioning or emergencies. [6]

Figure 3: Limit Switch

12V Pump: type of electric pump that operates on 12 volts DC power, commonly used in automotive, marine, and off-grid applications. Its functionality involves using an electric motor to drive a pump impeller or diaphragm, which creates a flow of fluid through the pump. One example of a 12V pump is a water pump for RVs or off-grid homes, such as the SEAFLO 55-Series 12V Water Pressure Pump.


Figure 4: 12V Pump

5V Servo: type of motor commonly used in robotics, automation, and hobby projects. These servos are designed to rotate to a specific angle or position, and can be controlled using a pulse-width modulation (PWM) signal. The rotation of the servo is controlled by adjusting the duration of the PWM signal, which determines the position of the servo's output shaft. One example of a 5V servo is the TowerPro SG90 Micro Servo, which is a popular and affordable option for many hobby and robotics projects.

Figure 5: 5V Servo

Fly Swatter: A fly swatter is a handheld device used for killing flies and other flying insects. It typically consists of a flat, flexible plastic or metal mesh attached to a handle. The user can swat the fly with the mesh to kill it. In our project, we repurposed a fly swatter to be used as an electrical fence for our security system of the home.


Figure 6: Fly Swatter

MY4N-J Relay: Is an electromagnetic switch that opens and closes a set of contacts when the relay coil is energized. It has 4 contacts with 4 normally opened (NO) and 4 normally closed (NC) terminals. Its contacts can withstand up to 5A at 240V AC or 28V DC. Its coil can be energized by 12V DC. This 14-pin relay is mounted on a relay socket.


Figure 7: MY4N-J Relay

HC-05 Bluetooth Module: The HC-05 is a class 2 Bluetooth module designed for transparent wireless serial communication. It is pre-configured as a slave

Bluetooth device. Once it is paired to a master Bluetooth device such as PC, smart phones and tablet, its operation becomes transparent to the user.

Figure 8: HC-05 Bluetooth Module

MIT App Inventor: MIT App Inventor is an intuitive, visual programming environment that allows everyone even children to build fully functional apps for smartphones and tablets.

Figure 9: MIT App Inventor

Solar Charge Controller: electronic device that regulates the flow of current and voltage from solar panels to batteries, preventing overcharging and ensuring efficient use of solar energy. In our home automation system prototype for people with special needs, we used a 30A solar charge controller to regulate the charging of the system's backup battery using solar panels. This ensured that the battery was charged optimally and efficiently, while also protecting it from overcharging or other potential damage.

Figure 10: Solar Charge Controller

RGB LED Strips: flexible circuit boards with embedded light-emitting diodes (LEDs) that can be cut to different lengths and used for various lighting applications. In our home automation system prototype for people with special needs, we used LED strips to provide ambient lighting in the house prototype and to indicate the status of certain subsystems, such as the security system. The LED strips were controlled by the system's microcontroller and could be turned on or off using the mobile application.

Figure 11: RGB LED Strips

Sound Sensor: electronic component that can detect sound and convert it into an electrical signal. In our home automation system prototype for people with special needs, we used a voice detection sensor to enable voice control of certain subsystems, such as the lights or temperature. By using voice commands, users with limited mobility were able to control the system more easily and independently.

Figure 12: Sound Sensor

2.2 Methods

In this experiment, the aim is to design and build a building management system using Arduino. The house should have an elevator, main gate, garage, lighting, fountain, security system, and a solar panel to help power the house. They should all be controlled via Bluetooth.

2.2.1 Design of the house:

The house was designed on AutoCAD according to the dimension constraints as shown in figure x:

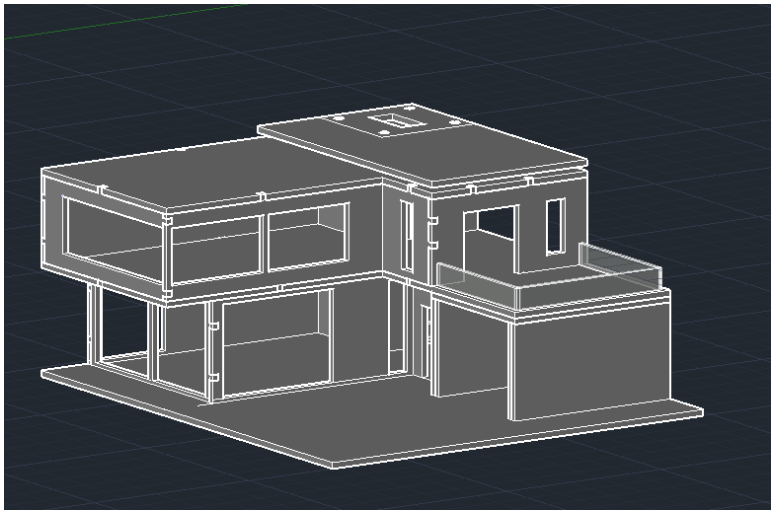
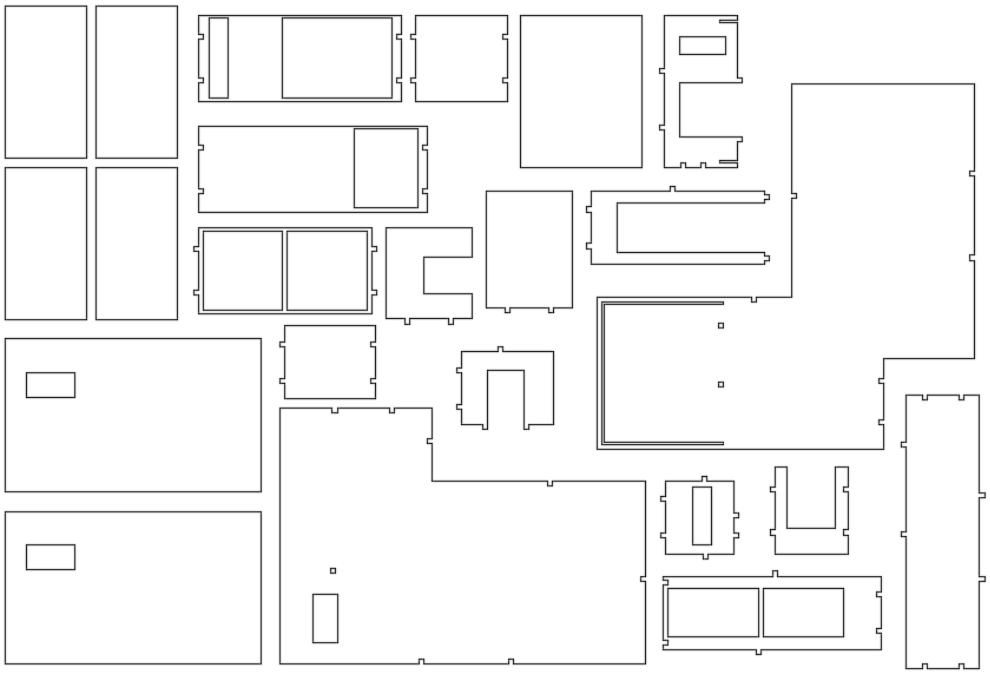


FIGURE 13: **DESIGN OF HOUSE IN AUTOCAD**

The house was designed in a way such that the pieces of the house interlock together like a puzzle. This made it easier for assemble and maintenance. The parts were then cut using a CNC



machine using 8mm MDF. Similarly, windows were cut from 2.5 mm acrylic sheet. The Parts Are Shown In Figure X:

FIGURE 14: HOUSE PARTS LAYOUT FOR CNC CUTTING

2.2.2 Elevator:

The principle of operation of the elevator is as follows. The elevator must go up and down 1 story. For the elevator to know which story is reached, 2 limit switches will be used to determine whether the elevator is on the first floor or the bottom floor. A geared DC motor with a pulley and string will be used to lift the elevator up or down until the elevator box hits one of the two limit switches.

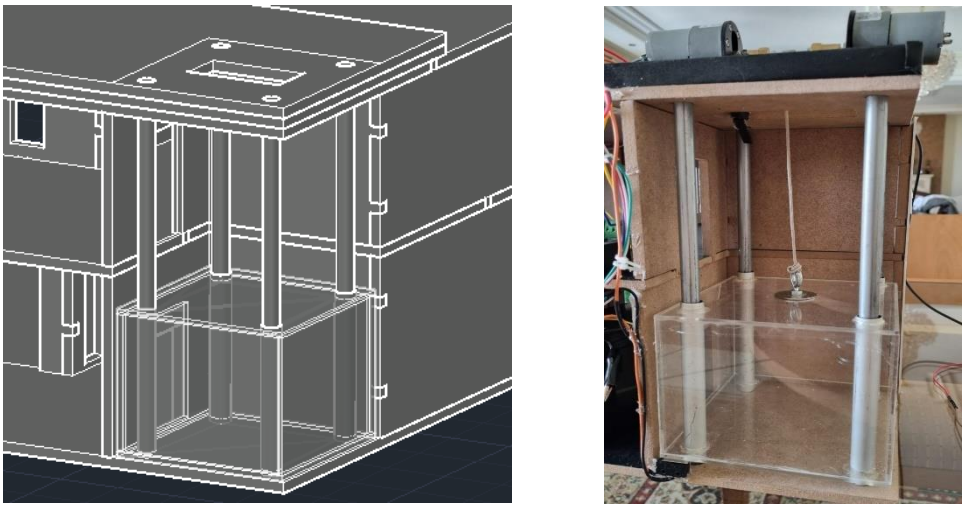


FIGURE 15: ELEVATOR DESIGN

2.2.3 Garage Door:

The garage will use a sliding mechanism. The body of the door should be flexible so that it can bend with no problem. To do this, a few strips of tape will be laid sticky side up, and wooden skewers are cut and placed one after another on the tape such that they form a flexible roll. This flexible roll will then be placed between a rail that will allow the door to slide through it in the direction of opening and closing. To pull the slider, a string is connected to the door which is connected to a rod which is spun by a servo motor using gears. The servo motor was modified by removing its circuit board and potentiometer so that it acts like a 5v geared dc motor. Similar to the elevator, the garage has 2 limit switches that determine when the motor should stop spinning for both positions. For the door to slide up and down smoothly, a heavy metal rod is placed it the bottom of the door so that when the door closes, the weight of the rod will help pull it down. The setup is shown in figure 16:

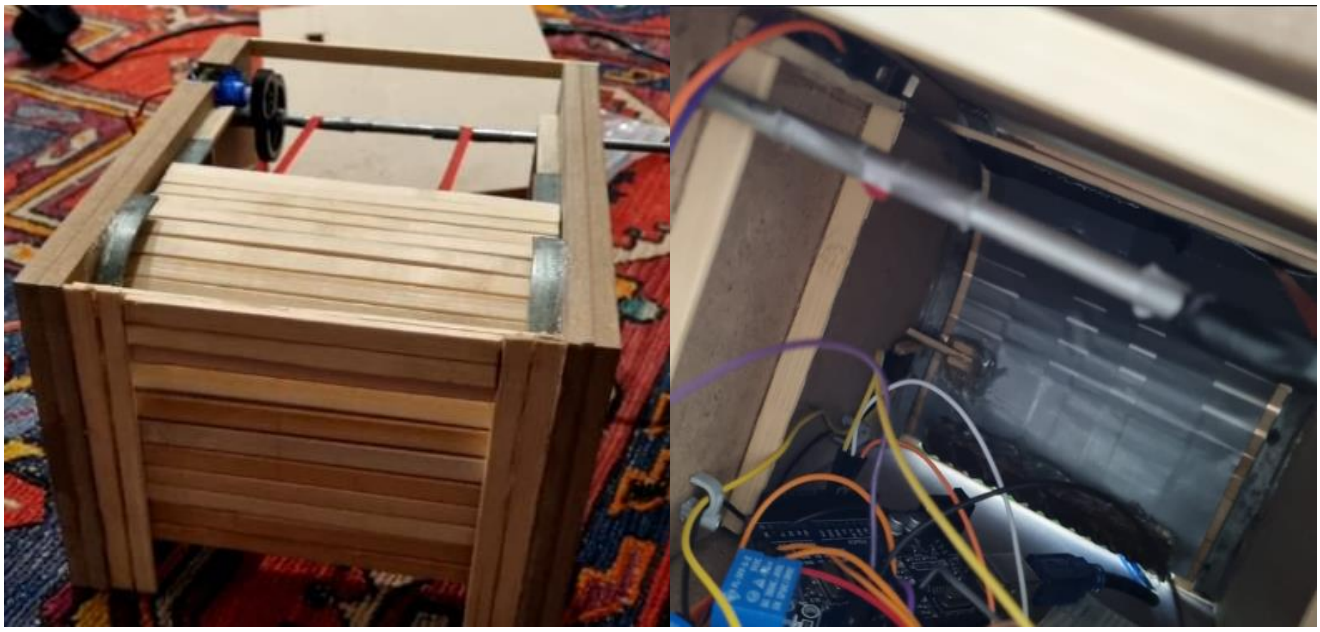


FIGURE 16: GARAGE DOOR MECHANISM

2.2.4 Lighting:

To light the house, RGB LEDs are used. The light colors are either white, red, blue, green or red. A party mode where all the lights flash randomly according to the surrounding voice is included also. They are wired on the ceiling of each floor as seen in figure 17:

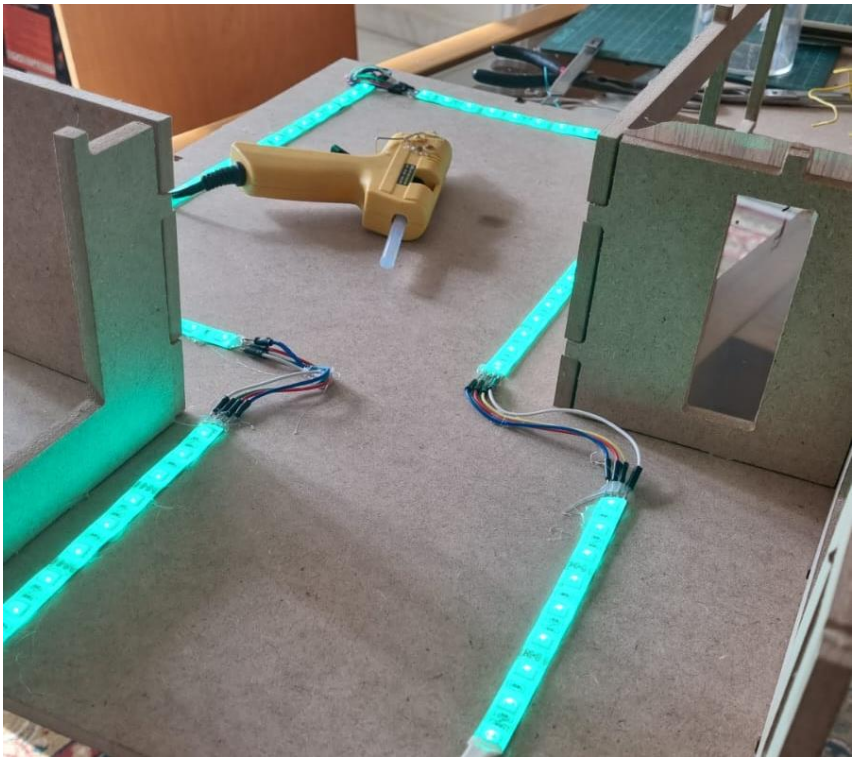


FIGURE 17: HOUSE LIGHTING

2.2.5 Main Door Gate:

It is required to have a main gate in the front of the house that will open and close. To do this a rack and pinion method is best used. Instead of building one from scratch, an already existing system was used which is the DVD drive. Its mechanism is perfect since it already has a rack and pinion along with a motor. A DVD drive was taken apart from an old DVD player. All the sensors and electronics were taken apart except for the motor and gear system. The motor is controlled using a motor drive and the motor is rated at 5 volts. the mechanism is shown in figure 18:

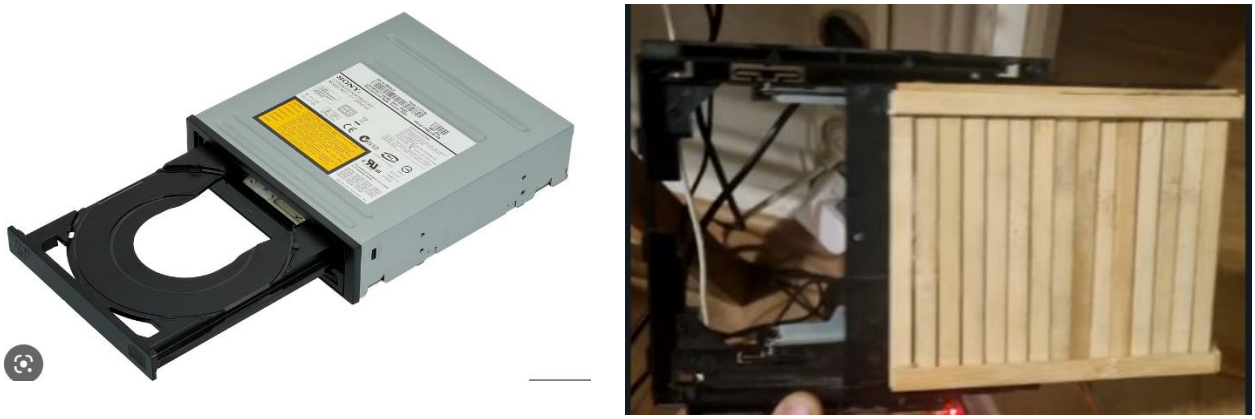


FIGURE 18: MAIN GATE MECHANISM

2.2.6 Fountain:

A fountain will be added using 2 pumps along with a small container to act as a vessel. It is shown in figure 19:

FIGURE 19: WATER FOUNTAIN

2.2.7 Security System:

It is required to install a security system for the house. The house shall have an electric fence to protect it from intruders. The fence was made using a voltage booster circuit from an electric fly swatter. It boots the voltage such that when a conductive object comes in between its 2 terminals, it will short circuit and shock the object in between. For safety, a manual switch is added between the live wires along with a relay in series so that it can be controlled with Arduino. the circuit requires a 3.7-volt lithium-ion battery. The fence is shown in figure 20;

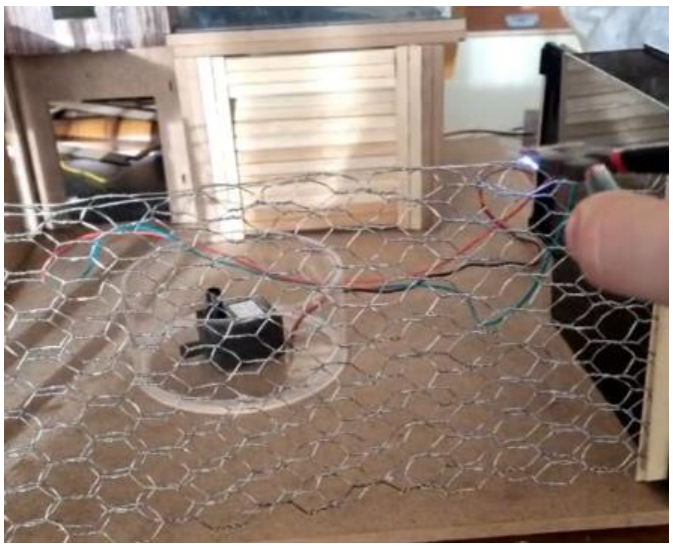


FIGURE 20: ELECTRIC FENCE

2.2.8 Solar:

The house runs on a 12-volt battery as its main power source. It is desired to recharge the battery using solar. To do this, a solar panel is mounted on the roof along with a charge controller connected to the battery. the assembly is shown in figure 21:

FIGURE 21: SOLAR SYSTEM

2.2.9 Water tank:

The water tank must pump up water from a water source at ground level to the roof so that the water can be used in the house. To do this, a simple 12-volt pump is used. The system's installation is shown in figure 22:

FIGURE 22: WATER TANK

2.2.10 Encoder:

Since the house is designed for a disabled person, it is desired to set up an encoder wheelchair that can keep track of how much the person moved. The wheelchair is basically a motor with an encoder which is used to track the distance traveled by the wheelchair. This was done in a pervious lab session.

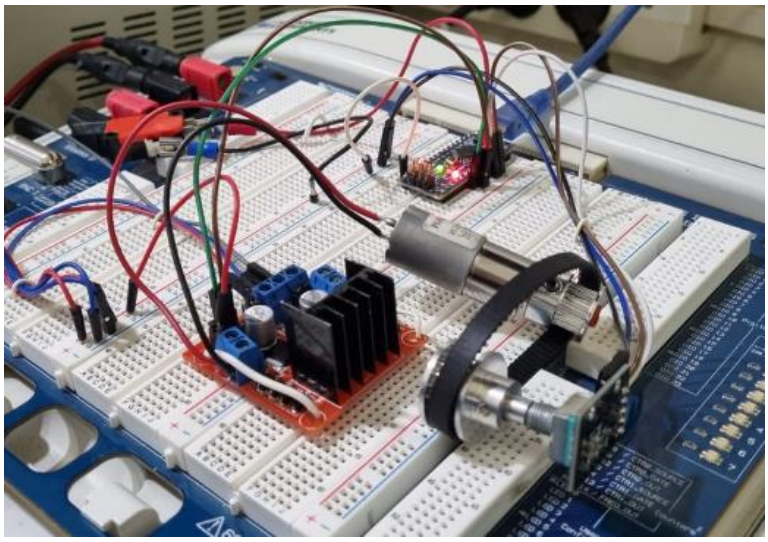


FIGURE 23: SOLAR SYSTEM

2.3 Wiring:

2.3.1 Elevator:

The wiring of the elevator is as follows. Two limit switches are connected to any 2 digital pins, and a geared dc motor is connected to a high amperage motor drive. in the motor drive connect the +VCC to 5 volts and the GND to the ground of the Arduino. enable the motor by connecting the enable pins to 5 volts. then connect the left and right pins of the motor to any 2 PWM pins on the Arduino. The wiring is shown in figure 24:

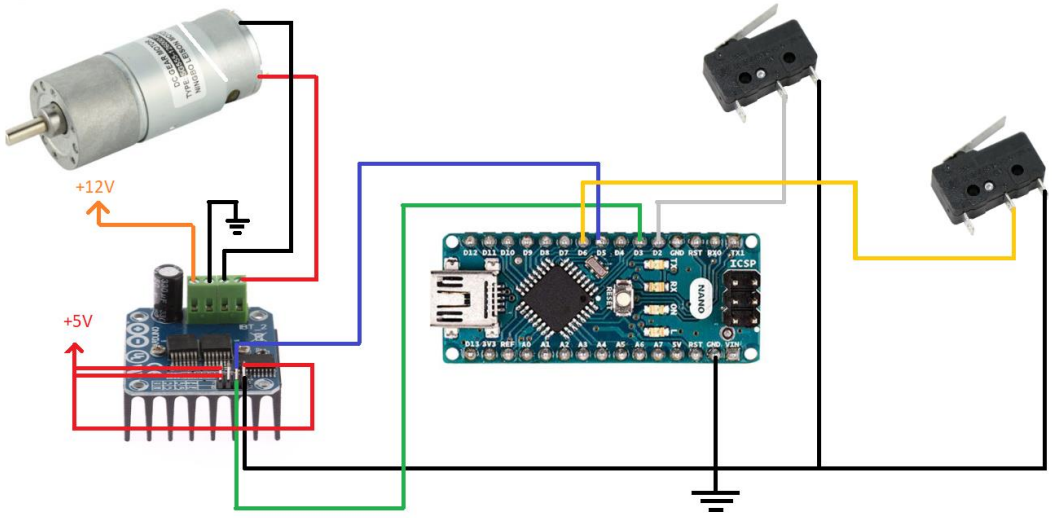


FIGURE 24: WIRING OF THE ELEVATOR

2.3.2 Garage:

The wiring of the garage is similar to that of the elevator. But instead of connecting a geared dc motor, a servo is placed to the motor drive but with 5 volts instead of 12 volts.

2.3.3 Lights:

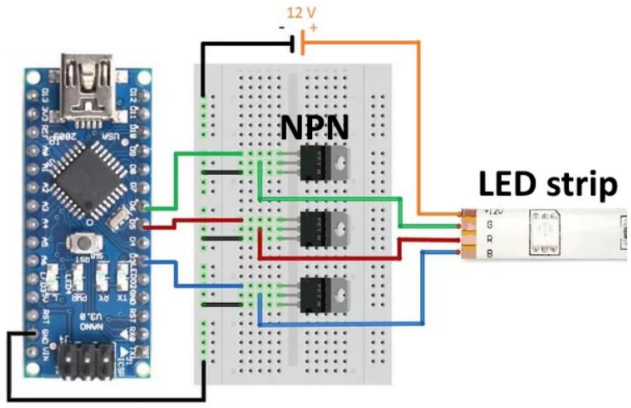


Figure 2 - The wiring of the led strip

FIGURE 25: WIRING OF LIGHTS

2.3.4 Main Door Gate:

The main gate's wiring is also similar to the garage and the elevator except that there are no limit switches and its motor handles 5 volts. Note that once the door reaches its maximum position, it automatically stops even if the motor keeps running without harming the motor. The same is true when the door retracts.

2.3.5 Fountain and tank:

Since both systems require pumps they share the same wiring. Each pump is connected to a relay as shown in figure 26:

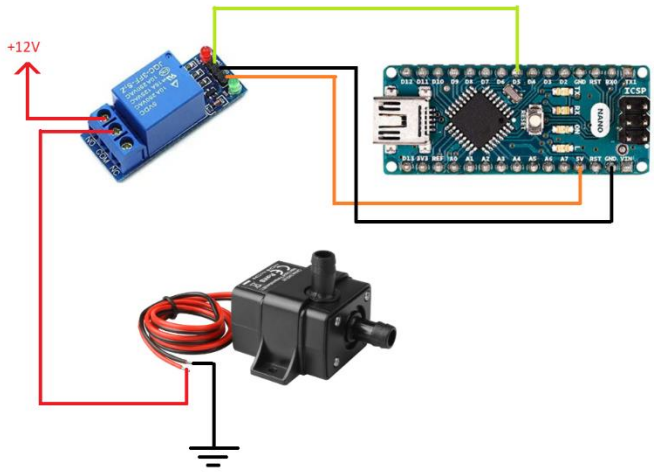


FIGURE 26: WIRING OF PUMPS FOR FOUNTAIN AND TANK

2.3.6 Solar:

The wiring of the solar system is shown in figure 27:

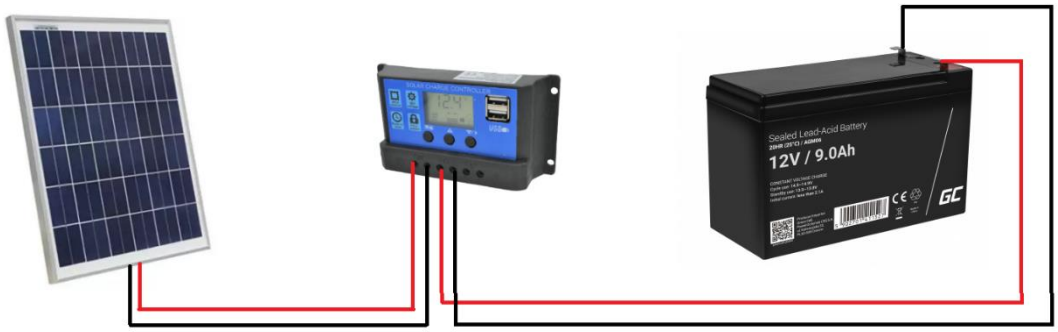


FIGURE 27: WIRING OF SOLAR SYSTEM

2.3.7 Security system:

The wiring of the security system is as follows in figure 28:

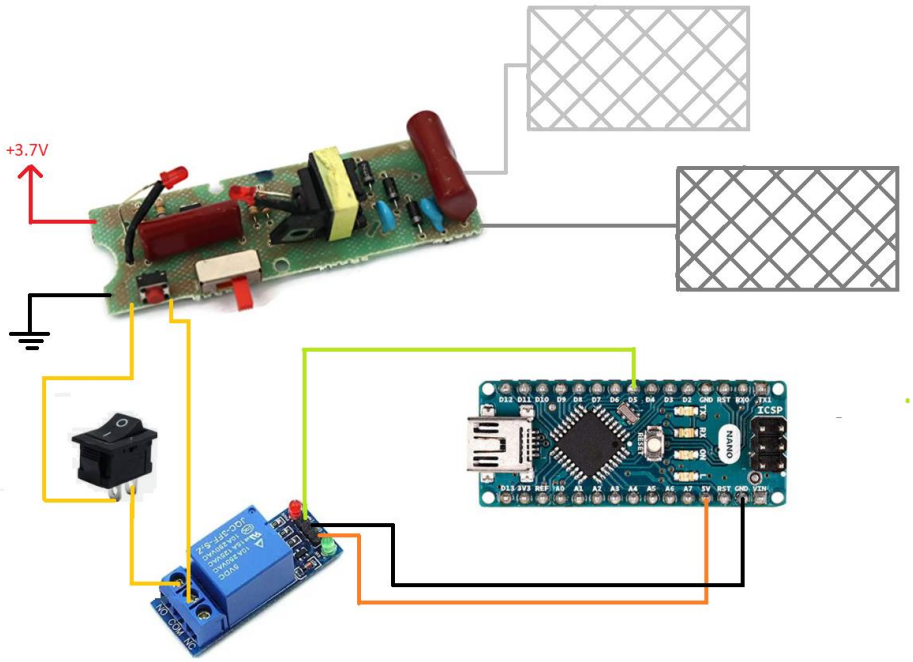


FIGURE 28: WIRING OF SECURITY SYSTEM

2.3.8 HC05 :

To connect the Bluetooth module to the system, connect the VCC and GND pin to the VCC and GND pin of any Arduino. In addition, connect the Tx and Rx of the module to the Rx and Tx of any Arduino respectively. Parallel communication will be used between different Arduinos.

2.4 Bluetooth

2.4.1 Bluetooth

The components that had been used for the home automation project are controlled through Bluetooth device by using MIT companion application. This application will transfer the data that was selected from the application to the Arduino using Bluetooth device HC-05 that will act as a slave and a master at the same time. In this project, the application will be divided into three main screens.

Screen 1 in MIT application is for registering an account by providing a username, password, and confirming the password. After doing that, this account registration will be saved to the second screen by using Tiny DB module that will store the data which are the username and the password that had been provided from the user. If the password and confirm password are not the same, an error will be occurred to the user that will not allow him/her to proceed the registration. Noting that, Tiny DB, Notifier, button, text label, text box, and layout arrangements were used in this screen.

The design screen and the block diagram for screen 1 are shown in the figure below.

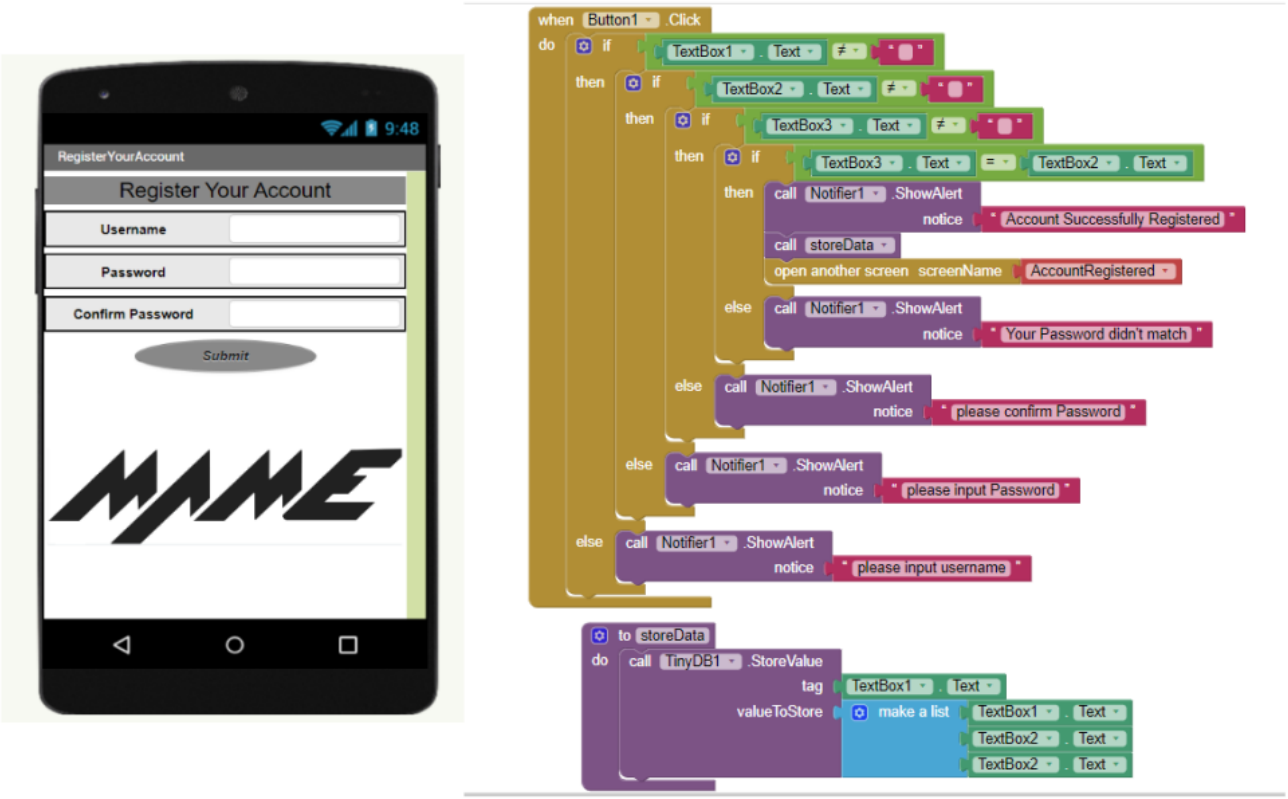


FIGURE 29: DESIGN SCREEN AND BLOCK DIAGARM FOR SCREEN 1 (REGISTER YOUR ACCOUNT)

Screen 2 named as Account Registered will be visible after the user presses the submit button from screen 1. In this screen, the user should write the same username and password selected previously and press login button; otherwise, an error will be displayed on the screen and the user won't be able to proceed the registration. Noting that, Tiny DB, Notifier, button, text label, text box, and layout arrangements were used in this screen.

Figure below shows the design screen and block diagram for Account Registered screen.

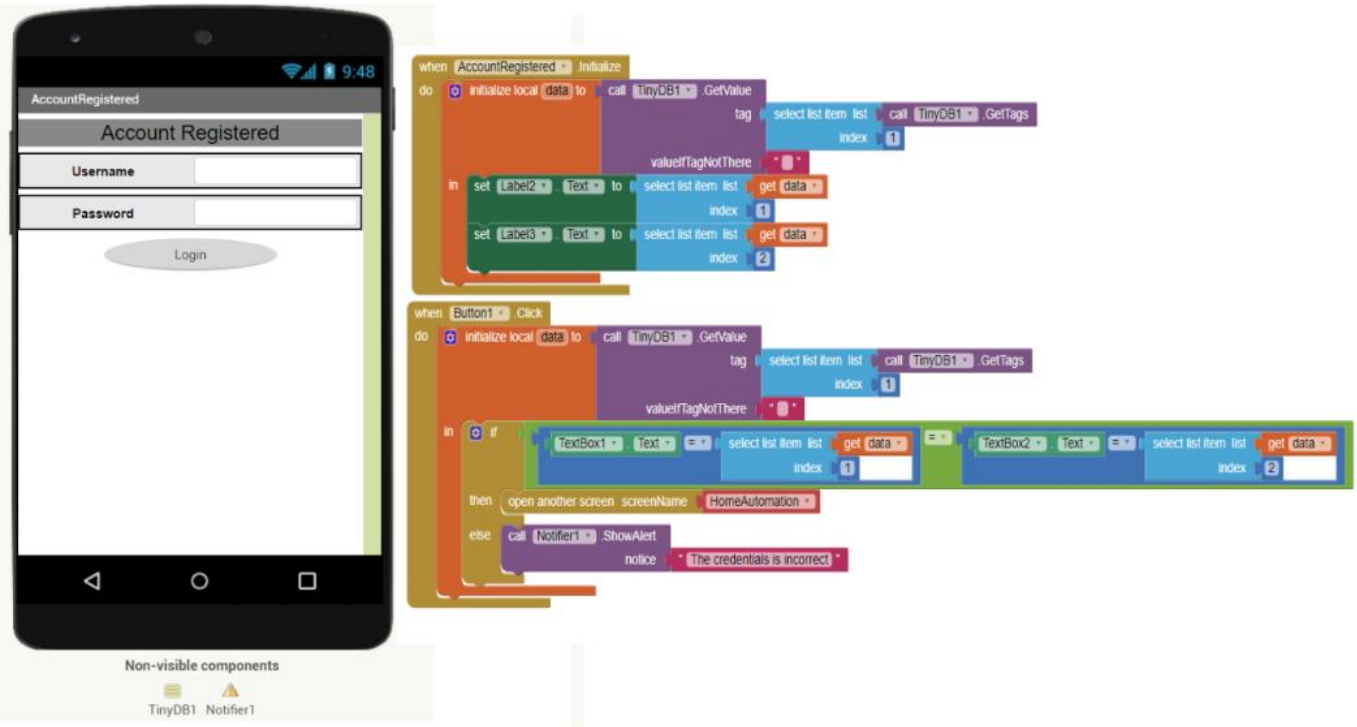


FIGURE 30: DESIGN SCREEN AND BLOCK DIAGARM FOR SCREEN 2

Screen 3 named as home automation screen is divided into multi-screens, the first part of this screen is the Bluetooth list that will show the list of Bluetooth devices nearby by clicking on the Bluetooth picture. After doing that, two buttons are available, if the user pressed the OFF button, then the LED on the Arduino will be off while if the user pressed the ON button, then the LED will be ON which shows that Bluetooth is connected and receiving data from the application. Once doing that, a new layout will be visible named as welcome screen.

Figure below shows the design screen and block diagram for the first layout in screen 3.

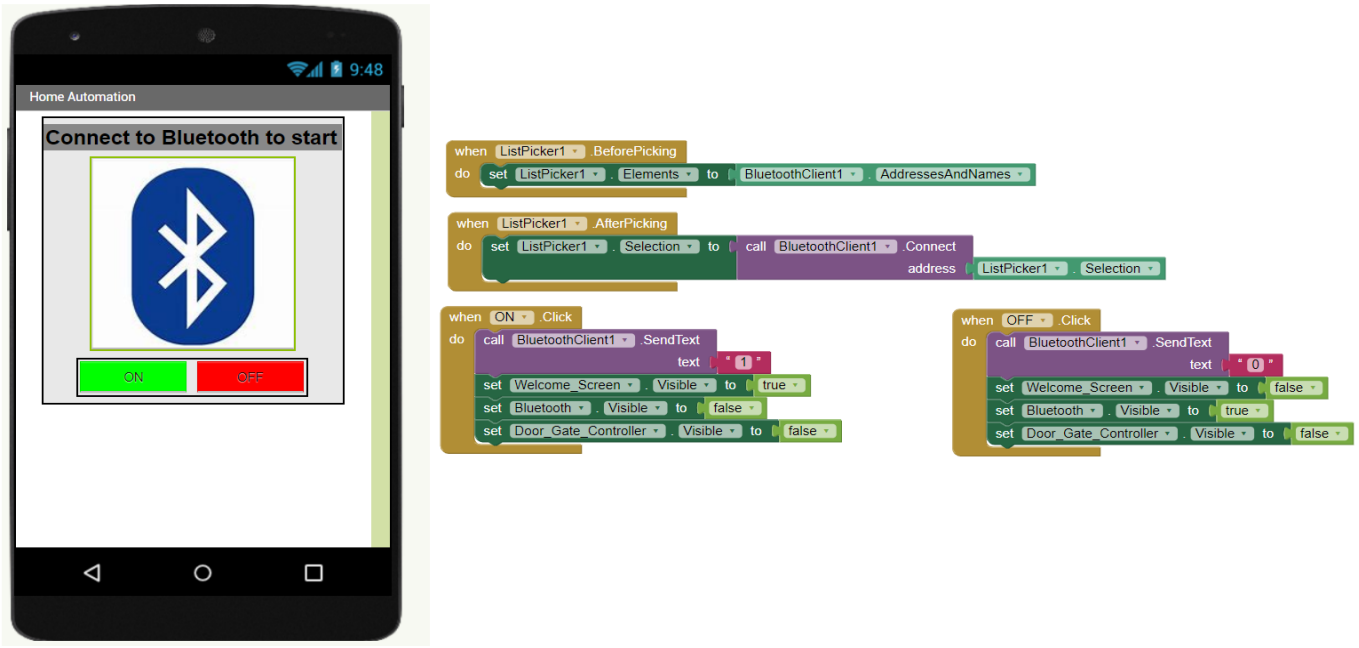


FIGURE 31: DESIGN SCREEN AND BLOCK DIAGARM FOR FIRST LAYOUT IN SCREEN 3

Now in the welcome screen, a set of choices will be visible for the user, go back button is found to go back to the Bluetooth lists, and SOS button that user can call someone if he/she in danger.

Figure below shows the welcome layout and the code block used for the SOS and go back button.

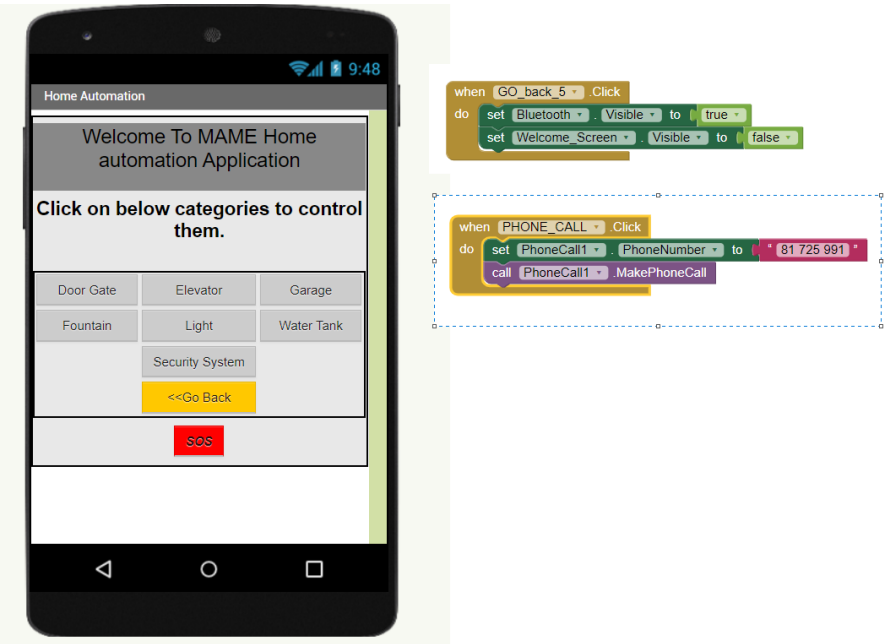


FIGURE 32: DESIGN SCREEN AND BLOCK DIAGARM FOR SECOND LAYOUT IN SCREEN 3

If the user pressed the door gate button, a new layout would be visible, and the old welcome screen would disappear. This new layout will have 3 buttons: Lock, Unlock, and go back button to go back to the welcome layout. Each lock and unlock button will transmit a character that will do a specific operation either to open the door gate or close it.

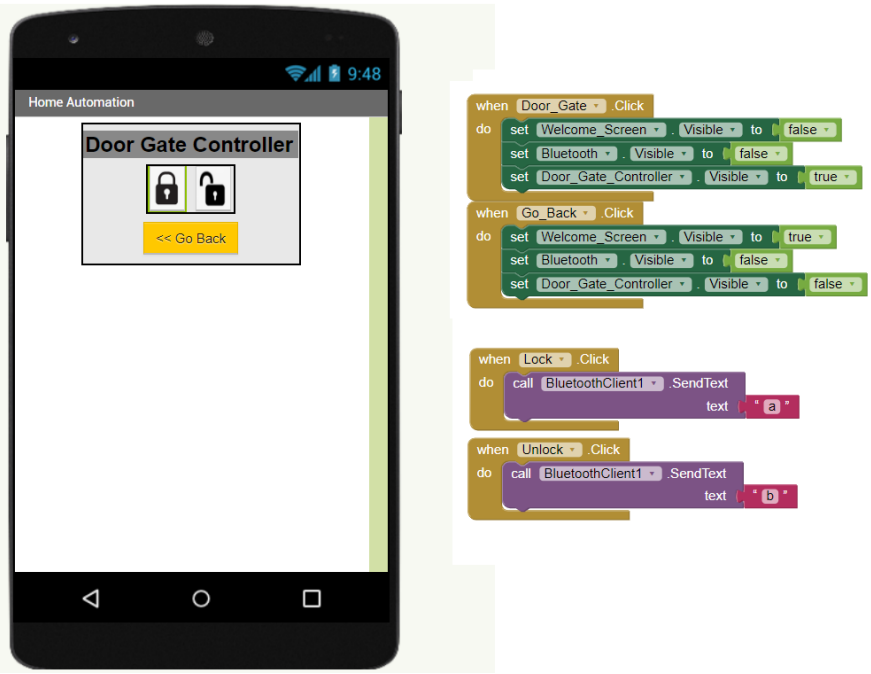


FIGURE 33: DESIGN SCREEN AND BLOCK DIAGARM FOR THIRD LAYOUT IN SCREEN 3

If the user pressed the Elevator button, a new layout would be visible, and the old welcome screen would disappear. This new layout will have 3 buttons: floor1, GF, and go back button to go back to the welcome layout. Each floor1 and GF button will transmit a character that will do a specific operation either the elevator will go up, down, or stay in its position depending to the limit switch value and Bluetooth command as explained in the code section.

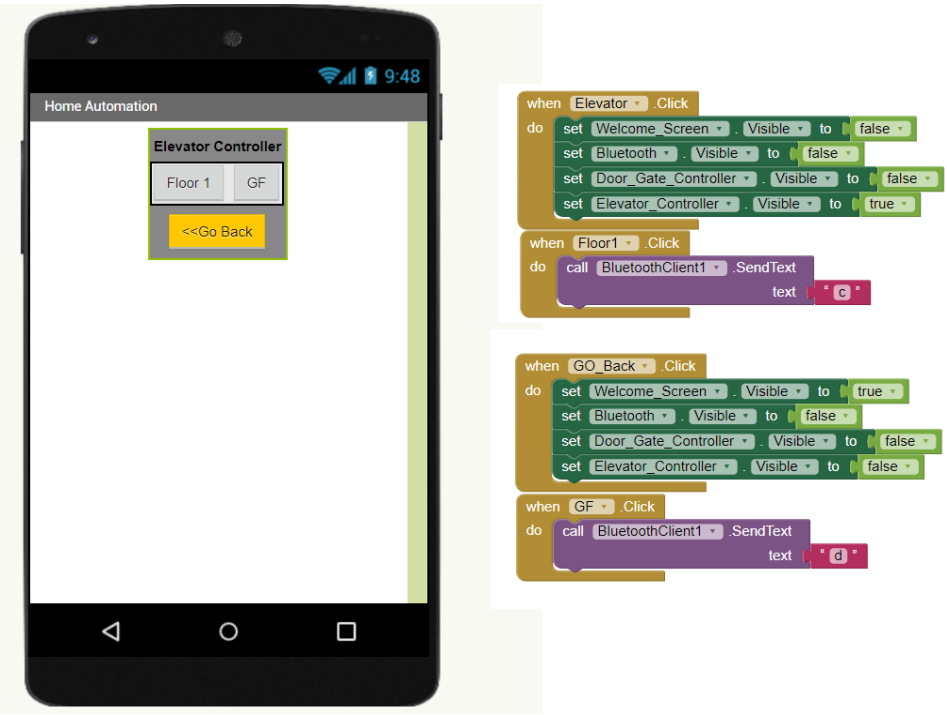


FIGURE 34: DESIGN SCREEN AND BLOCK DIAGARM FOR FOURTH LAYOUT IN SCREEN 3

If the user pressed the garage button, a new layout would be visible, and the old welcome screen would disappear. This new layout will have 3 buttons: open, close, and go back button to go back to the welcome layout. Each open and close button will transmit a character that will do a specific operation either the garage door will go up, down, or stay in its position depending to the limit switch value and Bluetooth command as explained in the code section.

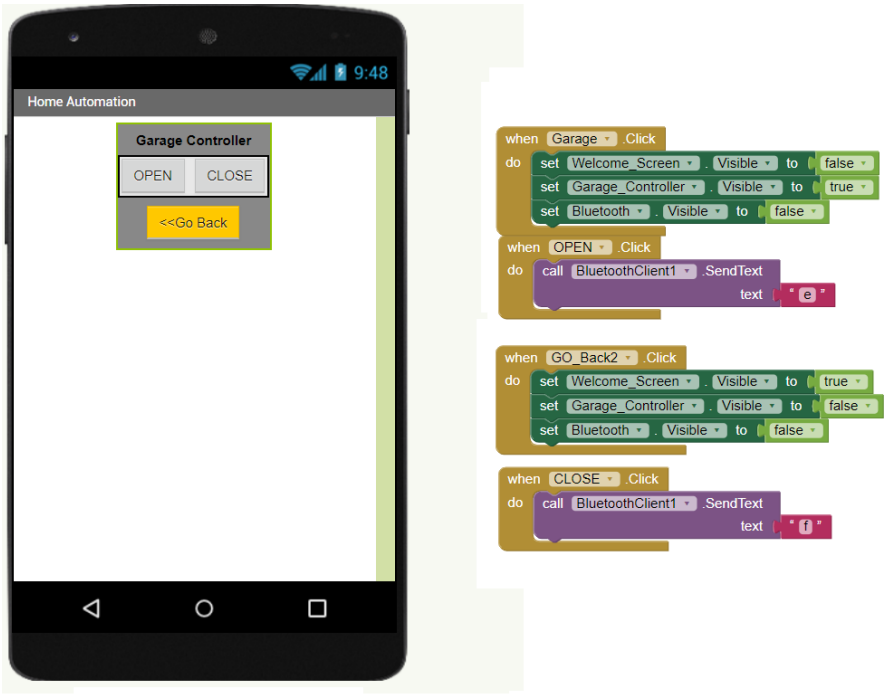


FIGURE 35: DESIGN SCREEN AND BLOCK DIAGARM FOR FIFTH LAYOUT IN SCREEN 3

If the user pressed the fountain button, a new layout would be visible, and the old welcome screen would disappear. This new layout will have 4 buttons: synchronized, not synchronized, OFF, and go back button to go back to the welcome layout. Each button will transmit a character that will do a specific operation whether the 2 pumps will be off, synchronized, or not synchronized.

FIGURE 36: DESIGN SCREEN AND BLOCK DIAGARM FOR SIXTH LAYOUT IN SCREEN 3

If the user pressed the LIGHT button, a new layout would be visible, and the old welcome screen would disappear. This new layout will have 5 buttons: RED, BLUE, GREEN, Party mode, and a go back button to go back to the welcome layout. Each button will transmit a character that will do a specific operation whether the lights will turn RED, BLUE, GREEN, or varying its color depending on the sound surrounding using sound sensor which is called a party mode.

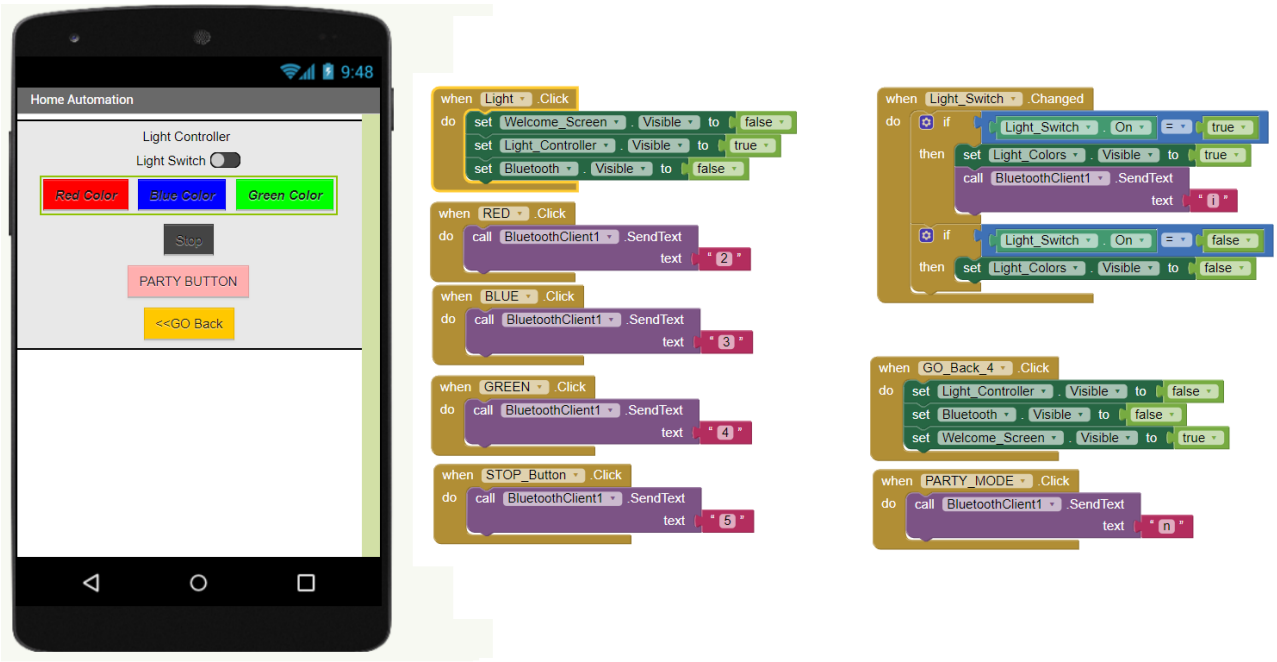


FIGURE 37: DESIGN SCREEN AND BLOCK DIAGARM FOR SEVENTH LAYOUT IN SCREEN 3

If the user pressed the water tank button, a new layout would be visible, and the old welcome screen would disappear. This new layout will have 2 buttons: Add water and a go back button to go back to the welcome layout. The Add water button will make a pump to flow water from one tank found in the ground to other tank found in house roof, the pump will stop pumping water once the water sensor reached the threshold which is 300. If the sensor has already reached its threshold and the Add water button is pressed, nothing will happen, and the pump will not turn on.

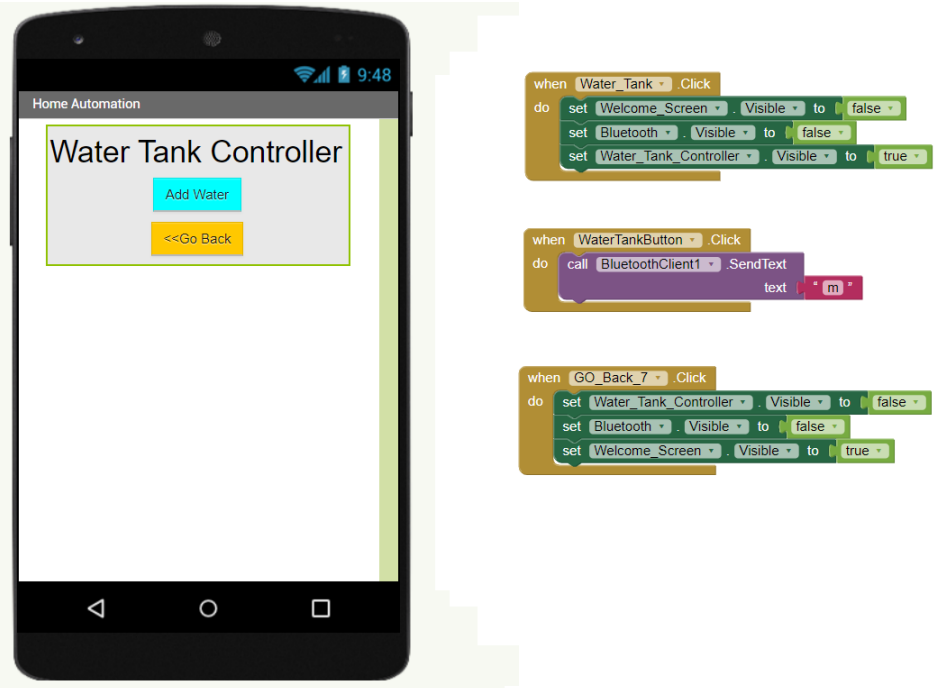


FIGURE 38: DESIGN SCREEN AND BLOCK DIAGARM FOR EIGHTH LAYOUT IN SCREEN 3

If the user pressed the security system button, a new layout would be visible, and the old welcome screen would disappear. This new layout will have 3 buttons: ON, OFF, and the go back button to go back to the welcome layout. Each ON and OFF button will transmit a character that will do a specific operation either the fence will turn ON or OFF.

FIGURE 39: DESIGN SCREEN AND BLOCK DIAGARM FOR NINTH LAYOUT IN SCREEN 3

2.5 Code:

2.5.1 Elevator

Two limit switches are used for the elevator. When pressed, the limit switches output 0V and otherwise 5V. The limit switch below detects that the elevator is on the ground floor and the other detects that the elevator is on the 1st floor.

When the elevator is commanded to move upward, the motors rotate to pull the elevator upward up until the upper limit switch is pressed and outputs 0V.

Similarly, when the elevator is commanded to move to the ground floor, the motors rotate in the opposite direction until the lower limit switch outputs 0V.

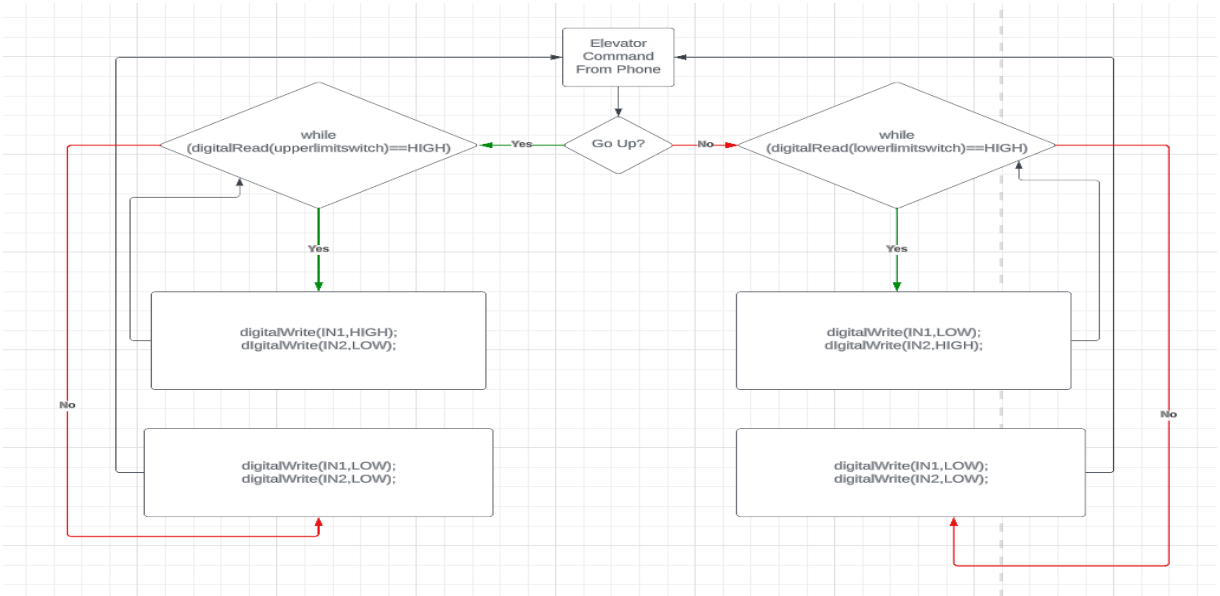


FIGURE 40: Flowchart for Elevator Code

2.5.2 Garage

The garage has the exact same setup and code as the elevator. Upper limit-switch to detect the garage has opened and a lower limit-switch to detect the garage has closed.

2.5.3 RGB LED Strip

The light at the output of the LED strip depends on the current passing through R, G, and B.
For example, if we command an analogWrite(red,255) only at the base of the transistor controlling the red color, the output color would be red.
If we command analogWrite(green,255) only at the base of the transistor controlling the green color would be green.
Similarly, the output color would be blue if we command analogWrite(blue,255) only at the base of the transistor controlling the blue color.
However, commanding analogWrite() at each base of the transistors together with PWM values can output all colors.
During party mode, we read values from the sound sensor to make the LED strip music reactive.

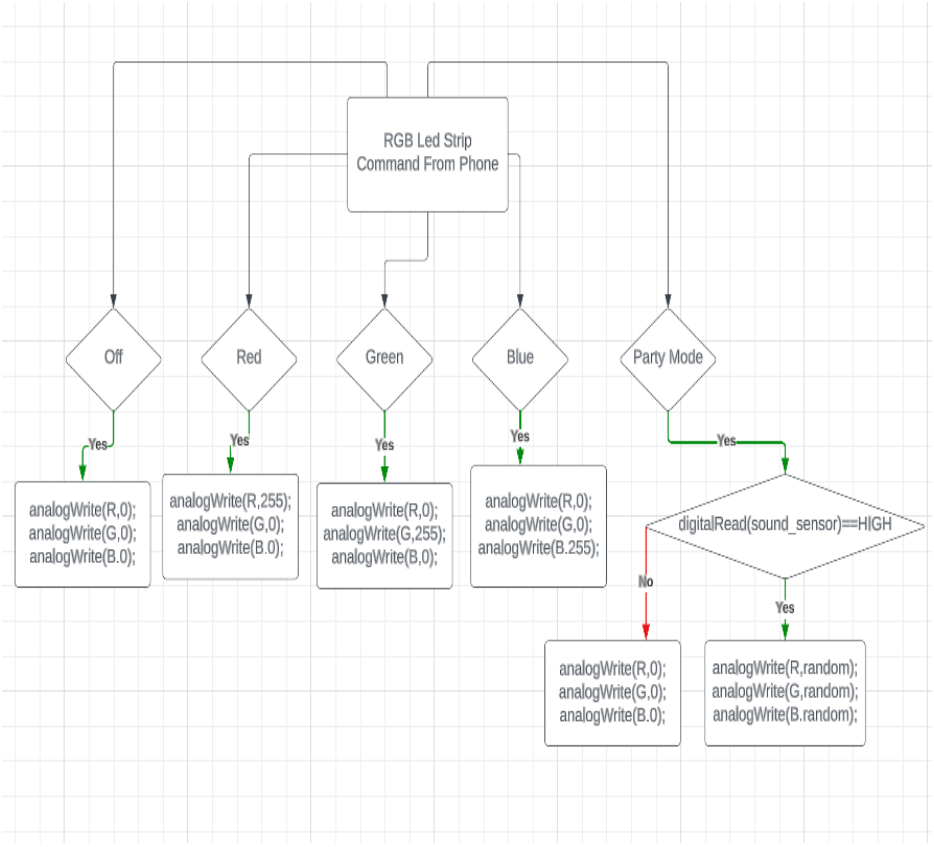


FIGURE 41: Flowchart for RGB LED Strip Code

2.5.3 Fountain

Two relays are used, one for each 12V DC pump. The fountain operates in 3 states: synchronized, unsynchronized, and off.

To make the two pumps synchronized, we simply apply voltage to both relays and cut-off at the same time.

To make the pumps unsynchronized, meaning one starts before the other, we simply use a delay.

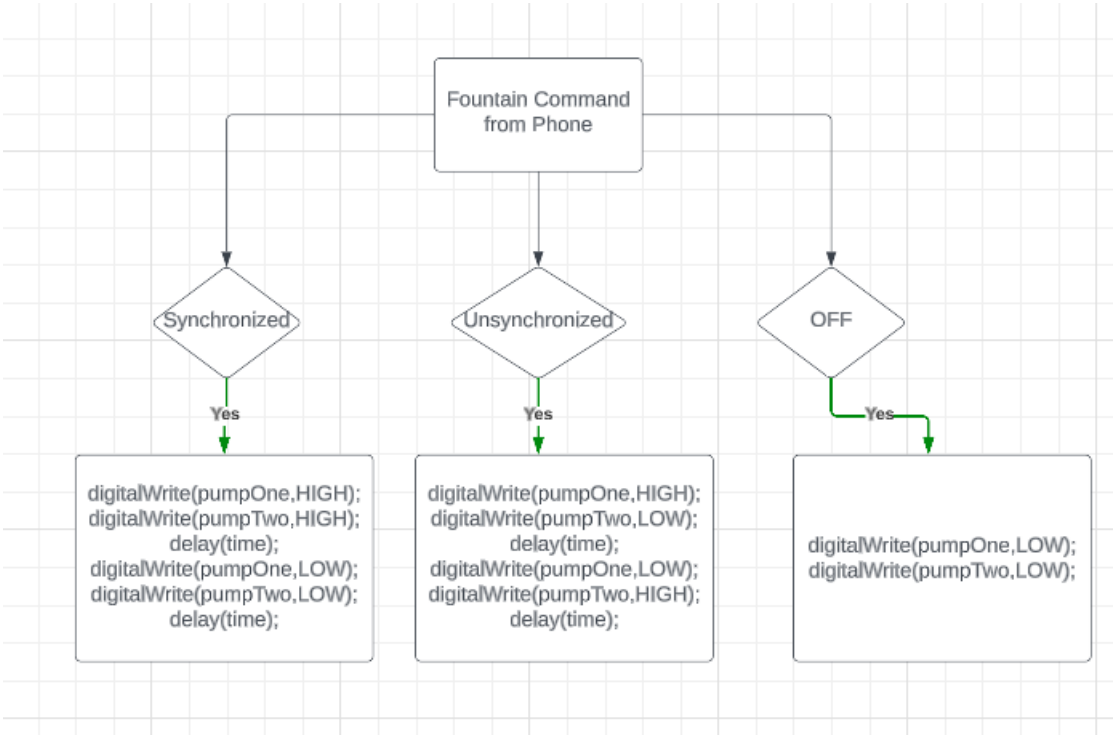


FIGURE 42: Flowchart for Fountain

2.5.4 Electric Fence

Electric fence is a fully electric system. However, as seen in its connections, only a relay, controlled by our microcontroller through the phone, was used for additional safety.

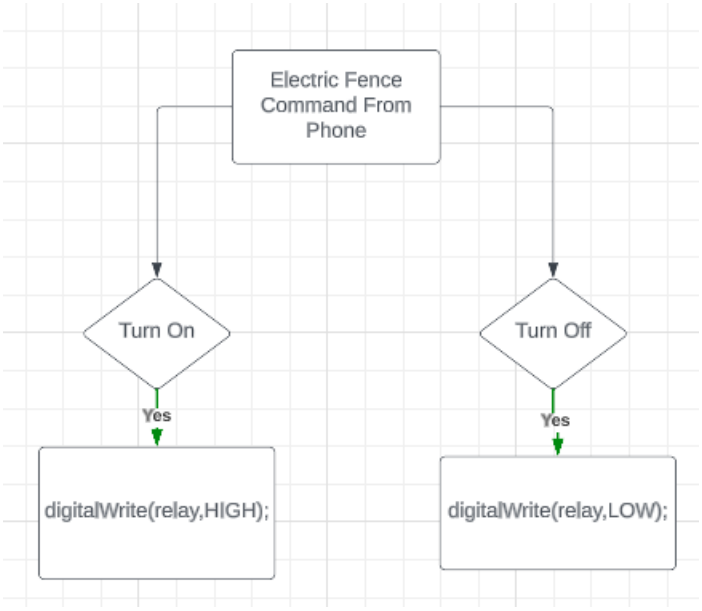


FIGURE 43: Electric Fence Code Flowchart

2.5.5 Gate

Gate is simply commanded to open and close, and we accomplish reversal of polarity across it through the motor driver.

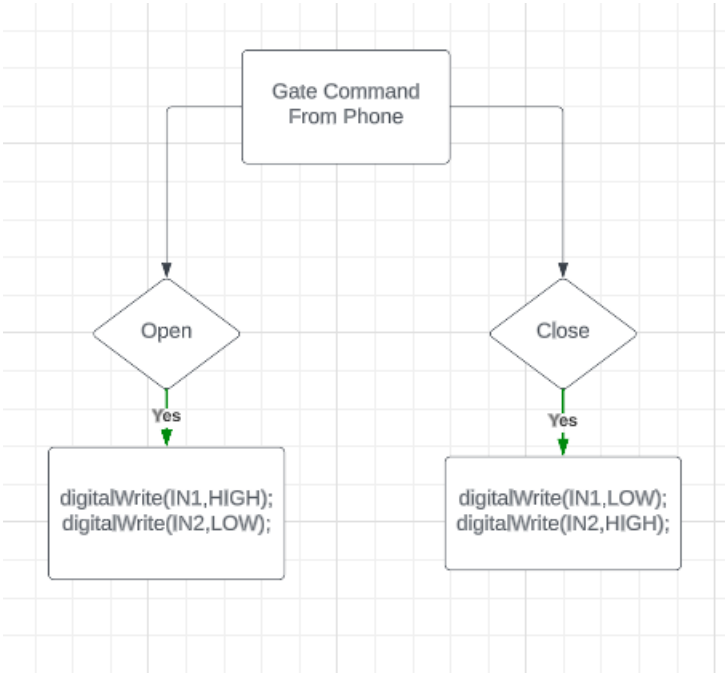


FIGURE 44: Gate Code Flowchart

2.5.6 Encoder

Encoder operation was already explained in experiments 1 and 2 of the course.

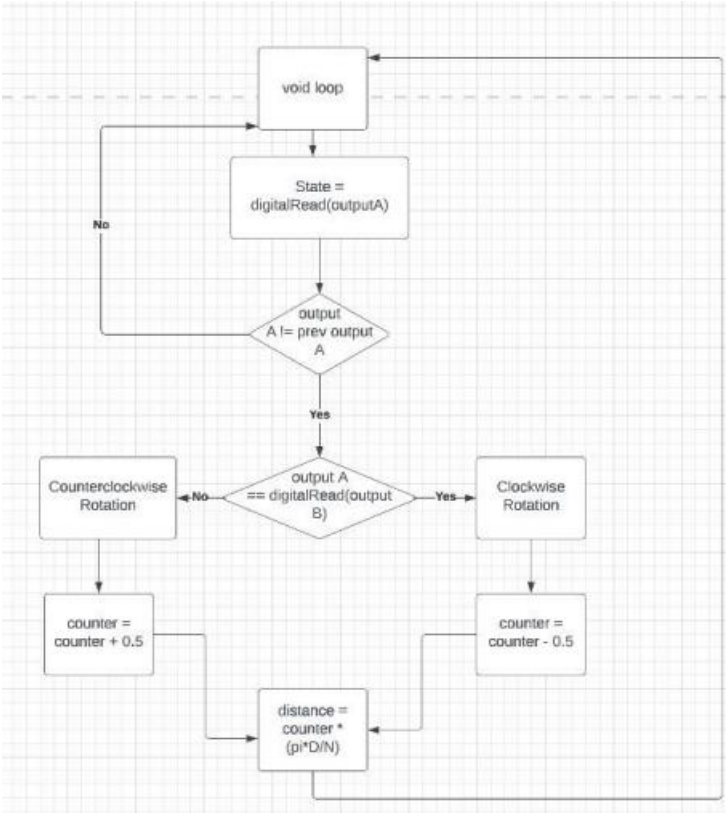


FIGURE 45: Encoder Code Flowchart

2.5.7 Water Tank

Using the fluid level sensor, we read the water level. By activating the relay, water is pumped until the threshold value is reached.

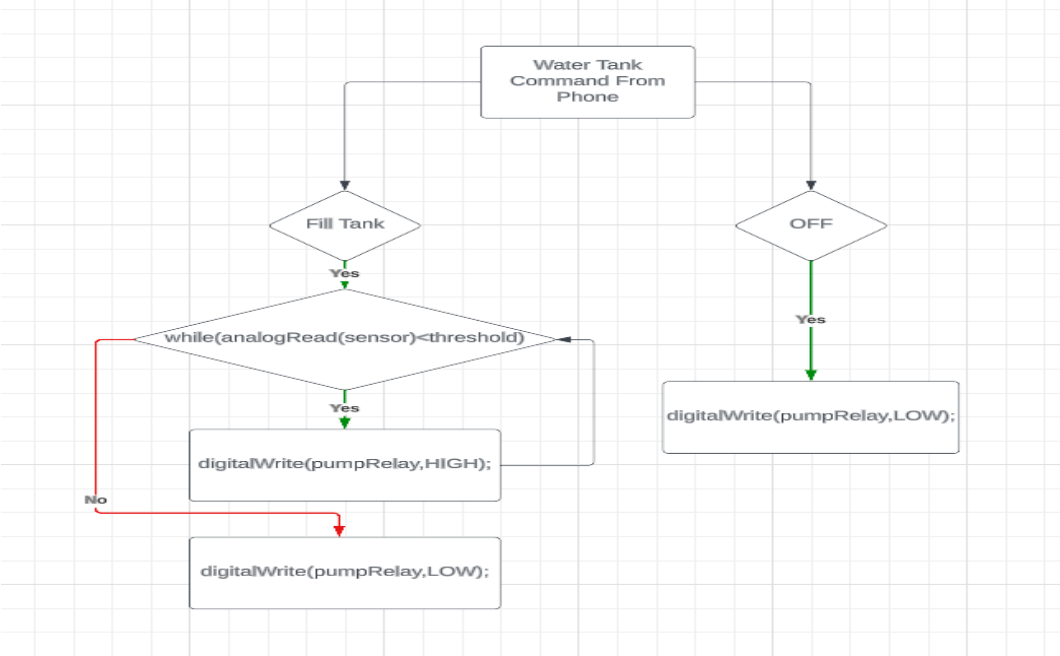


FIGURE 46: Water Tank Flowchart Code

3 RESULTS AND DISCUSSION

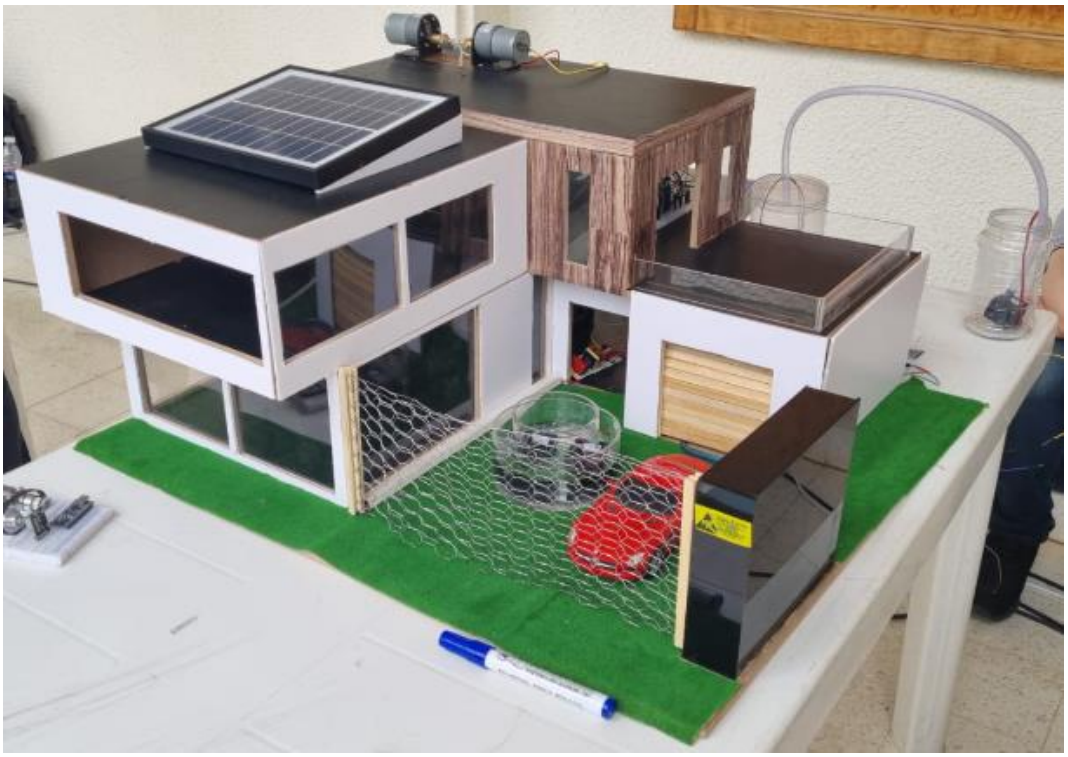


FIGURE 47: FINAL HOUSE MODEL

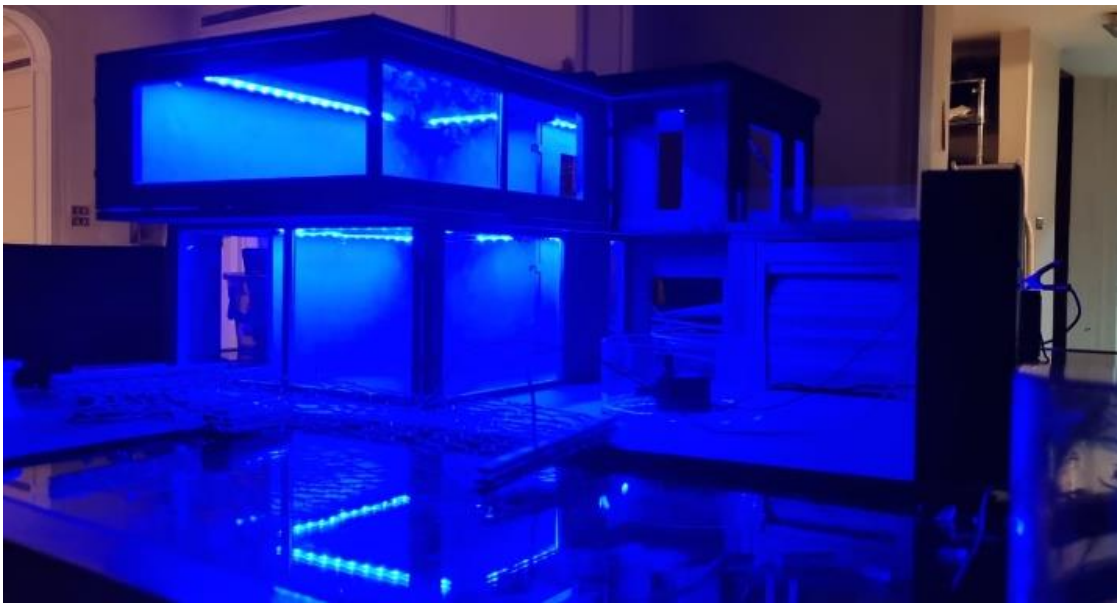


FIGURE 48 : HOUSE LIGHTING

Link to Video of house Working:

https://studentsrhuedu-my.sharepoint.com/:f:/g/personal/itanimm538_students_rhu_edu_lb/ErdRkfEZDRtAlzLcYvZmLeABPSBxmNdA6CUPlIggUaXEiw?e=cNBaNN

4 CONCLUSION

In conclusion, the objective of this project is design and build a home automation system prototype for paralyzed individuals. The entire system was controlled using MIT App Inventor, a visual programming platform that allowed us to easily create a custom mobile application to control all the actuators.

REFERENCES

[1]Arduino,2020.ArduinoNANO.
https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf
[2] HC-05 Buletooth Module
https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/
[3] MIT App Inventor
https://appinventor.mit.edu/

Appendix A:

Code for LED strip and Elevator:

```
#include <Wire.h>
int sound_pin = 2;
bool value_1 = false;
bool value_2 = false;
bool value_3 = false;
char data;
int M_up = 3;
int M_down = 5;
int LS_up = 6;
int LS_down = 8;
int state=1;
unsigned long current, prev_1=0, prev_2=0,prev_3=0;
void setup() {
 // put your setup code here, to run once:
 Wire.begin();
 Serial.begin(9600);
 pinMode(sound_pin,INPUT);
 pinMode(9,OUTPUT);
 pinMode(10,OUTPUT);
 pinMode(11,OUTPUT);
 pinMode(M_up, OUTPUT);
 pinMode(M_down, OUTPUT);
 pinMode(LS_up, INPUT_PULLUP);
 pinMode(LS_down, INPUT_PULLUP);
}
```

```
void loop() {
 int buttonValue1 = digitalRead(LS_up);
 int buttonValue2 = digitalRead(LS_down);
  String command="";
 if(Serial.available()){
  data=Serial.read();
  command=Serial.readString();
  Wire.beginTransmission(9); // transmit to device #9
  Wire.write(x);            // sends x
  Wire.endTransmission();    // stop transmitting
}
 current = millis();
 if (current-prev_1>2000){
 value_1 = !value_1;
 prev_1 = current;
 }
 if (current-prev_2>4000){
 value_2 = !value_2;
 prev_2 = current;
 }
 if (current-prev_3>6000){
 value_3 = !value_3;
 prev_3 = current;
 }
 if (command=="c"){
 while (buttonValue1 == HIGH){
    digitalWrite(M_up, HIGH);
  digitalWrite(M_down, LOW);
  Serial.println("UPP");
  buttonValue1 = digitalRead(LS_up);
  }
  digitalWrite(M_up, LOW);
  Serial.println("STOP");
  digitalWrite(M_down, LOW);

 }
 if (command=="d"){
 while (buttonValue2 == HIGH){
    digitalWrite(M_up, LOW);
  digitalWrite(M_down, HIGH);
    Serial.println("DOWNN");
   buttonValue2 = digitalRead(LS_down);
    }
    Serial.println("STOP");
    digitalWrite(M_up, LOW);
  digitalWrite(M_down, LOW);
```

```
 }
if(command=="n" && state == 0){
 state = 1;
}
else if(command=="n" && state == 1){
 state = 0;
 Serial.print("party");
}
if (state == 0){
 if (digitalRead(sound_pin)==1){
  if (value_1 == 0 && value_2 == 0 && value_3 == 0){
   value_1 = 1;
  }
  digitalWrite(9,value_1);
  digitalWrite(10,value_2);
  digitalWrite(11,value_3);
 }
 else{
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
 }
}
else {
if (data=='4'){
   digitalWrite(9,HIGH); //green
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
  Serial.print("4");

}
if (data=='3'){
   digitalWrite(9,LOW);//blue
 digitalWrite(10,HIGH);
 digitalWrite(11,LOW);
  Serial.print("3");
}
if (data=='2'){
 digitalWrite(9,LOW); //red
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
}

if (data=='5'){
 digitalWrite(9,LOW); //red
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}
```

```
    }
}
```

fountain and fence:

```
    // #include <Wire.h>
    int pumpOnePin = 2;
    int pumpTwoPin = 3;
    int fence=5;
    int time=200;
    int state=3;
    char data=0;
    String command="";

    void setup() {
     //put your setup code here, to run once:
     Serial.begin(9600);
     pinMode(pumpOnePin,OUTPUT);
     pinMode(pumpTwoPin,OUTPUT);
     pinMode(fence,OUTPUT);
    }

    void loop() {
     // put your main code here, to run repeatedly:
     if(Serial.available()){
       data=Serial.read();
       command=Serial.readString();
     }
     if (command=="o"){

       digitalWrite (fence,HIGH);
     }

      if (command=="p"){

       digitalWrite (fence,LOW);
    }

     if (command=="g"){
       state = 0;
     }
     else if (command=="j"){
       state = 1;
     }
     else if (command=="h"){
       state = 2;
     }
     if (state == 0){
```

```
      digitalWrite(pumpOnePin,HIGH);
      digitalWrite(pumpTwoPin,HIGH);
      delay(time);
      digitalWrite(pumpOnePin,LOW);
      digitalWrite(pumpTwoPin,LOW);
      delay(time);
      }

      else if (state == 1){
      digitalWrite(pumpOnePin,LOW);
      digitalWrite(pumpTwoPin,HIGH);
      delay(time);
      digitalWrite(pumpOnePin,HIGH);
      digitalWrite(pumpTwoPin,LOW);
      delay(time);
      digitalWrite(pumpOnePin,LOW);
      }
      else if (state == 2) {
         digitalWrite(pumpOnePin,LOW);
         digitalWrite(pumpTwoPin,LOW);
      }
}
```

Code for garage and gate:

```
    int limitswitchE1=6;
    int limitswitchE2=8;
    int motorG1=9;
    int motorG2=10;
    int test1;
    int test2;
    int dir;
    char data;
    int count=0;

    int IN1 = 3;
    int IN2 = 5;
    void setup() {
     // put your setup code here, to run once:
     pinMode(motorG1,OUTPUT);
     pinMode(motorG2,OUTPUT);
     pinMode(limitswitchE1,INPUT_PULLUP);
     pinMode(limitswitchE2,INPUT_PULLUP);
     pinMode(IN1,OUTPUT);
     pinMode(IN2,OUTPUT);
     Serial.begin(9600);
    }
```

```
void loop() {
 // Serial.println("waiting");
String command="";
//  Serial.println(digitalRead(limitswitchE1));
 // Serial.println(digitalRead(limitswitchE2));


if(Serial.available()){
   data=Serial.read();
   command=Serial.readString();
}

   test1=digitalRead(limitswitchE1);
   test2=digitalRead(limitswitchE2);

 if (data== '1'){
    digitalWrite(13,HIGH);
 }
    if (command=="e"){
     Serial.println("CLOSING");
     while (test2 == HIGH){
    digitalWrite(motorG1,HIGH);
    digitalWrite(motorG2,LOW);
    test2 = digitalRead(limitswitchE2);
    }
    Serial.println("DONE");
    digitalWrite(motorG1,LOW);
    digitalWrite(motorG2,LOW);

 }

    if (command=="f") {
     Serial.println("OPENING");
    while (test1 == HIGH){
    digitalWrite(motorG1,LOW);
    digitalWrite(motorG2,HIGH);
    test1 = digitalRead(limitswitchE1);
    }
    Serial.println("DONE");
    digitalWrite(motorG1,LOW);
    digitalWrite(motorG2,LOW);
    }

    if (command=="b"){ // gate

   digitalWrite(IN1,HIGH);
   digitalWrite(IN2,LOW);
   Serial.println("GATE!");
```

```
        }

        if (command=="a"){
      digitalWrite(IN1,LOW);
      digitalWrite(IN2,HIGH);
       Serial.println("GATE!");
    }

}
```