ITE233 - Introduction to Internet of Things
FINAL GROUP PROJECT

IOT Home Automation System

Professor: Ajarn Atikom Srivallop

Section 1

Ms. Riya Shrestha 1906250006
Mr. Vlas Nagibin 1907020003
Mr. Dalmacio Ntutumu Medang 1907180009
Ms. Fardina Kabir 1910080005

The following ideas represent IOT Home Automation System for ITE233 project:

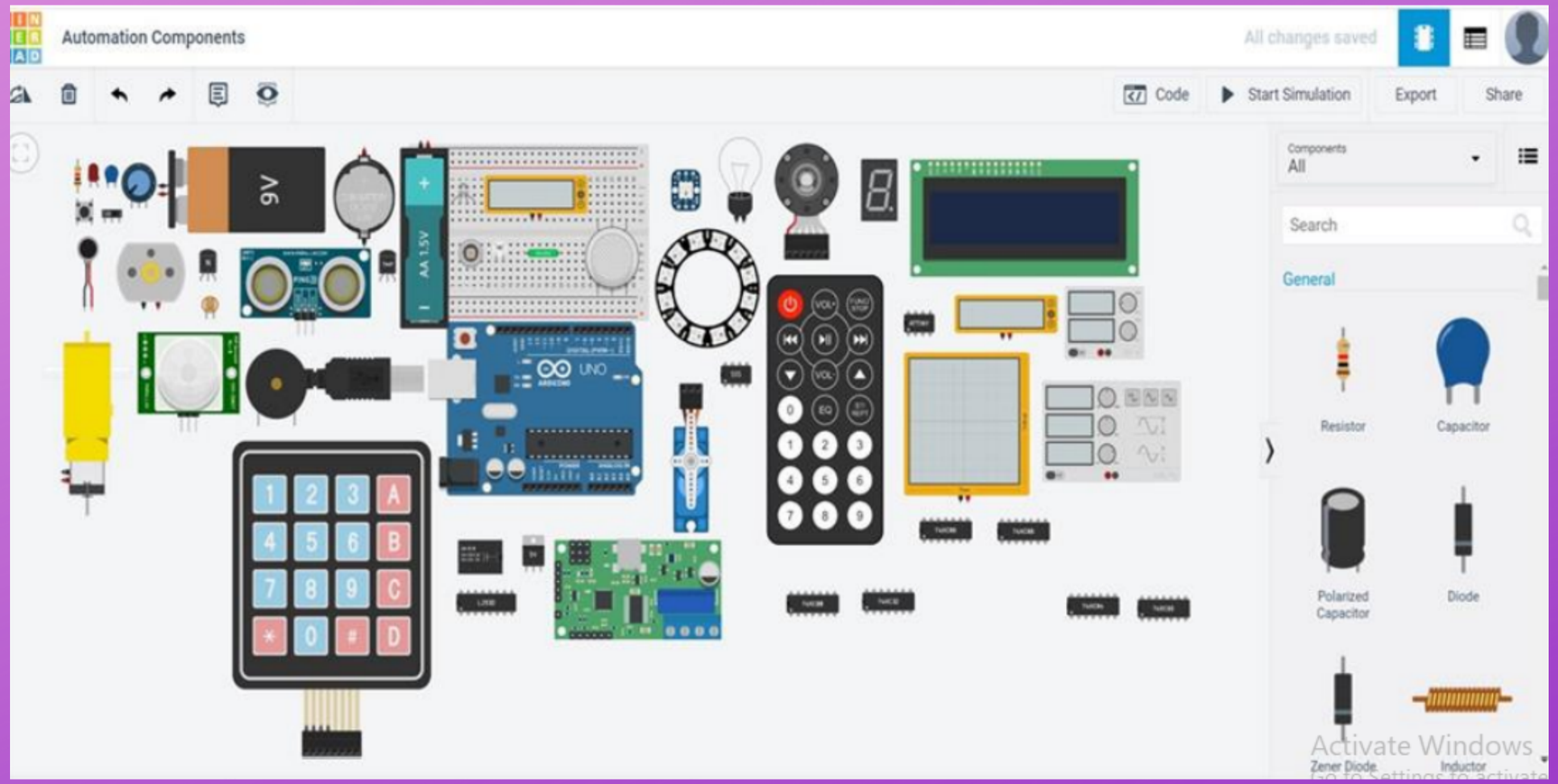
- Home security system with keypad and IR remote control
- Automated Car parking system
- Sensor-Based Smart Home Automation

For every ideas, we presented samples of data generated by blynk app to represent statistics about the ideas from credible sources.

In the end, we will look at Virtual platform difficulties and future possibilities.

The Simulation platform used is TinkerCAD

TinkerCAD has the following componenets:



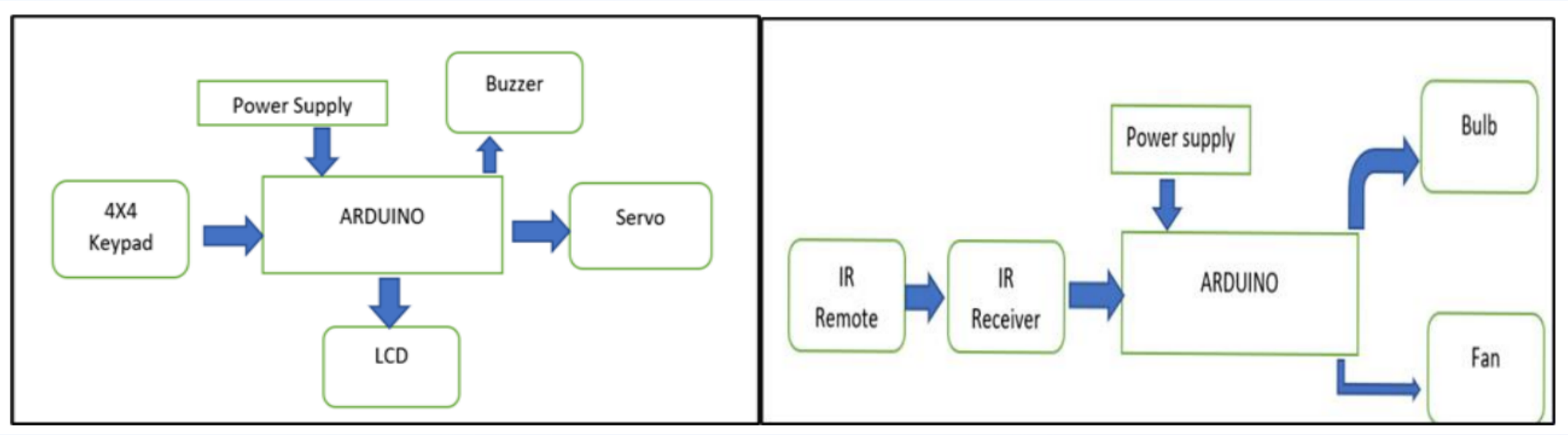
1. Automatic Home security system with keypad and IR remote control

List of components used in this application:

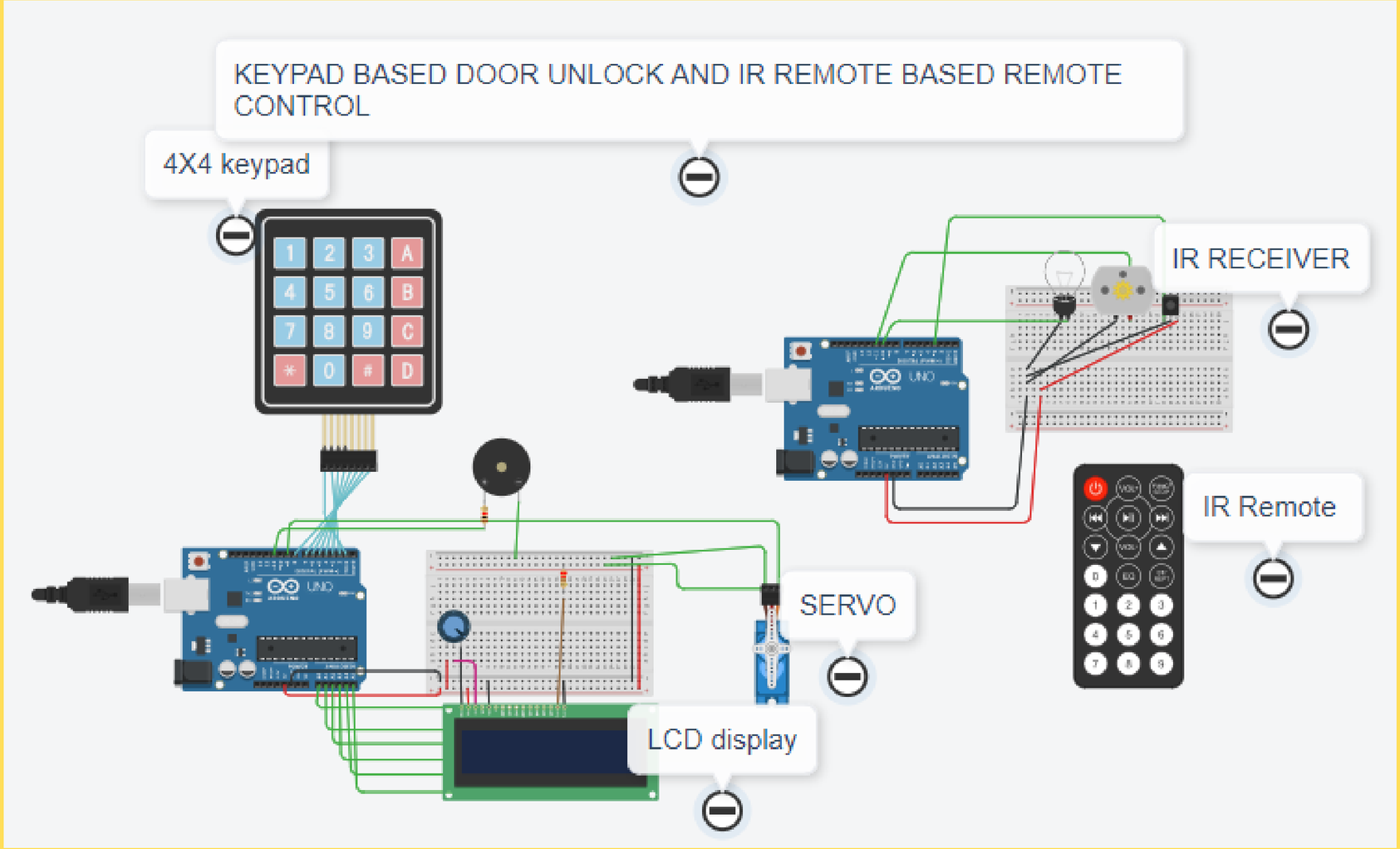
An Arduino microcontroller, and a smartphone—will be used to power the Blynk system, and a deadbolt was also used for practical simulation with the Blynk app.

Name	Quantity	Component
KEYPAD1	1	Keypad 4x4
SERVO1	1	Micro Servo
PIEZO1	1	Piezo
U1 U5	2	Arduino Uno R3
U2	1	LCD 16 x 2
Rpot1	1	250 kΩ, Potentiometer
R1	1	220 Ω Resistor
R2	1	1 kΩ Resistor
U3	1	IR sensor
L1	1	Light bulb
M2	1	DC Motor

Block diagram for Home security system with keypad and IR remote control



The circuit along with the components



Tinkercad simulation video with explanation of the function:

Text codng for keypad based door unlock/lock

```
#include <Keypad.h>
#include <LiquidCrystal.h>
#include <Servo.h>
Servo myservo;
LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);
#define Password_Lenght 7
int pos = 0;
char Data[Password_Lenght];
char Master[Password_Lenght] = "123456";
byte data_count = 0, master_count = 0;
bool Pass_is_good;
char customKey;

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
```

```
};
bool door = true;
int buzzer=11;
byte rowPins[ROWS] = {1, 2, 3, 4}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {5, 6, 7, 8}; //connect to the column pinouts of the keypad

Keypad customKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS); //initialize an ins

void setup()
{
  myservo.attach(9);
  pinMode(buzzer,OUTPUT);
  ServoClose();
  lcd.begin(16, 2);
  lcd.print(" KEYPAD SECURITY");
  lcd.setCursor(0, 1);
  lcd.print("**HELLO**");
  delay(1000);
  lcd.clear();

}
```

```
void loop()
{
  if (door == 0)
  {
    customKey = customKeypad.getKey();

    if (customKey == '#')

    {
      lcd.clear();
      ServoClose();
      lcd.print("  Door is close");
      delay(2000);
      door = 1;
    }
  }

  else Open();
}

void clearData()
```

```
void clearData()
{
  while (data_count != 0)
  {
    Data[data_count--] = 0;
  }
  return;
}

void ServoOpen()
{
  for (pos = 180; pos >= 0; pos -= 5) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}

void ServoClose()
{
  for (pos = 0; pos <= 180; pos += 5) { // goes from 180 degrees to 0 degrees
```

```
void ServoClose()
{
  for (pos = 0; pos <= 180; pos += 5) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);                 // tell servo to go to position in variable 'pos'
    delay(15);                          // waits 15ms for the servo to reach the position
  }
}

void Open()
{
  lcd.setCursor(0, 0);
  lcd.print(" Enter Password");

  customKey = customKeypad.getKey();
  if (customKey) // makes sure a key is actually pressed, equal to (customKey != NO_KEY)
  {
    Data[data_count] = customKey; // store char into data array
    lcd.setCursor(data_count, 1); // move cursor to show each new char
    lcd.print(Data[data_count]); // print char at said cursor
    data_count++; // increment data array by 1 to store new char, also keep track of the
  }
```

```
  lcd.print(Data[data_count]); // print char at said cursor
  data_count++; // increment data array by 1 to store new char, also keep track of the
}

if (data_count == Password_Lenght - 1) // if the array index is equal to the number of
{
  if (!strcmp(Data, Master)) // equal to (strcmp(Data, Master) == 0)
  {
    lcd.clear();
    ServoOpen();
    lcd.setCursor(0,0);
    lcd.print("  Door is Open");
    lcd.setCursor(0,1);
    lcd.print("welcome home");
    door = 0;
  }
  else
  {
    lcd.clear();
    lcd.print("  Wrong Password");
    digitalWrite(buzzer,HIGH);
```

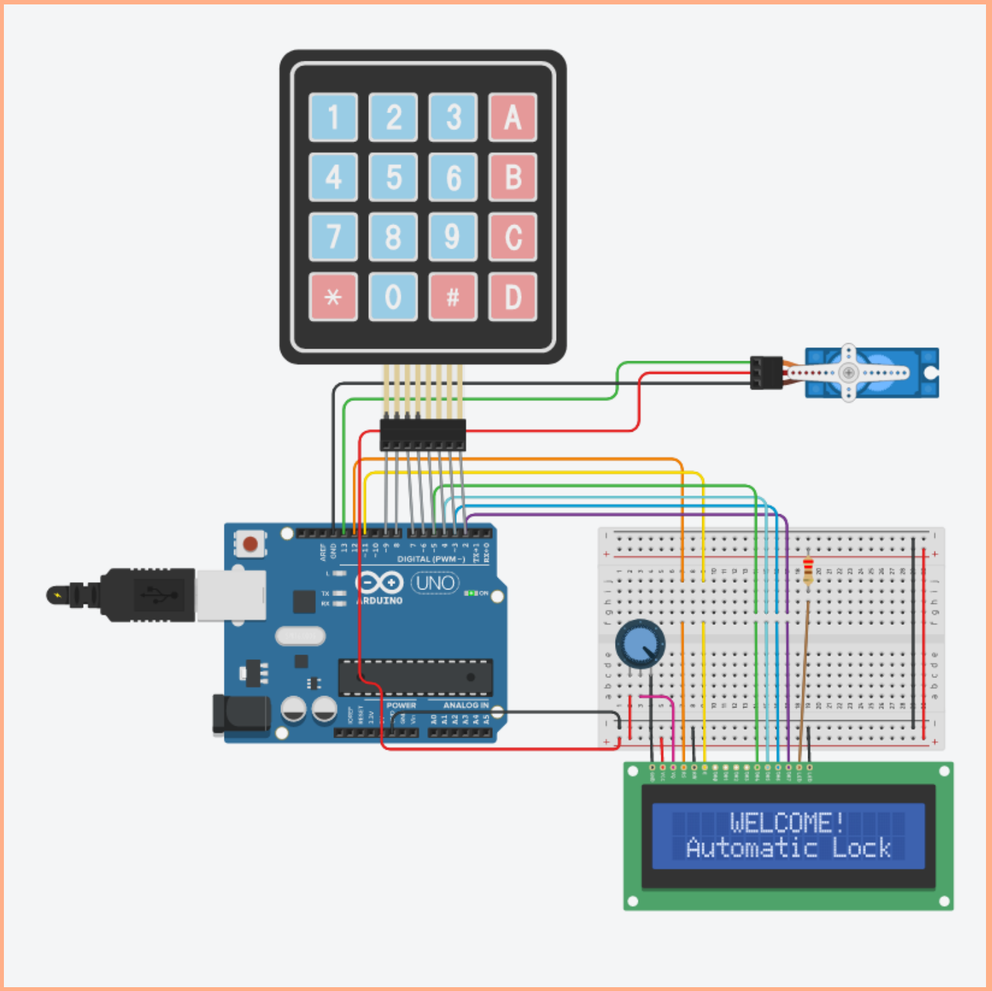
```
      ServoOpen();
      lcd.setCursor(0,0);
      lcd.print("  Door is Open");
      lcd.setCursor(0,1);
      lcd.print("welcome home");
      door = 0;
    }
    else
    {
      lcd.clear();
      lcd.print("  Wrong Password");
      digitalWrite(buzzer,HIGH);
      delay(1000);
      digitalWrite(buzzer,LOW);
      door = 1;
    }
    clearData();
  }
}
```

Coding (Block)

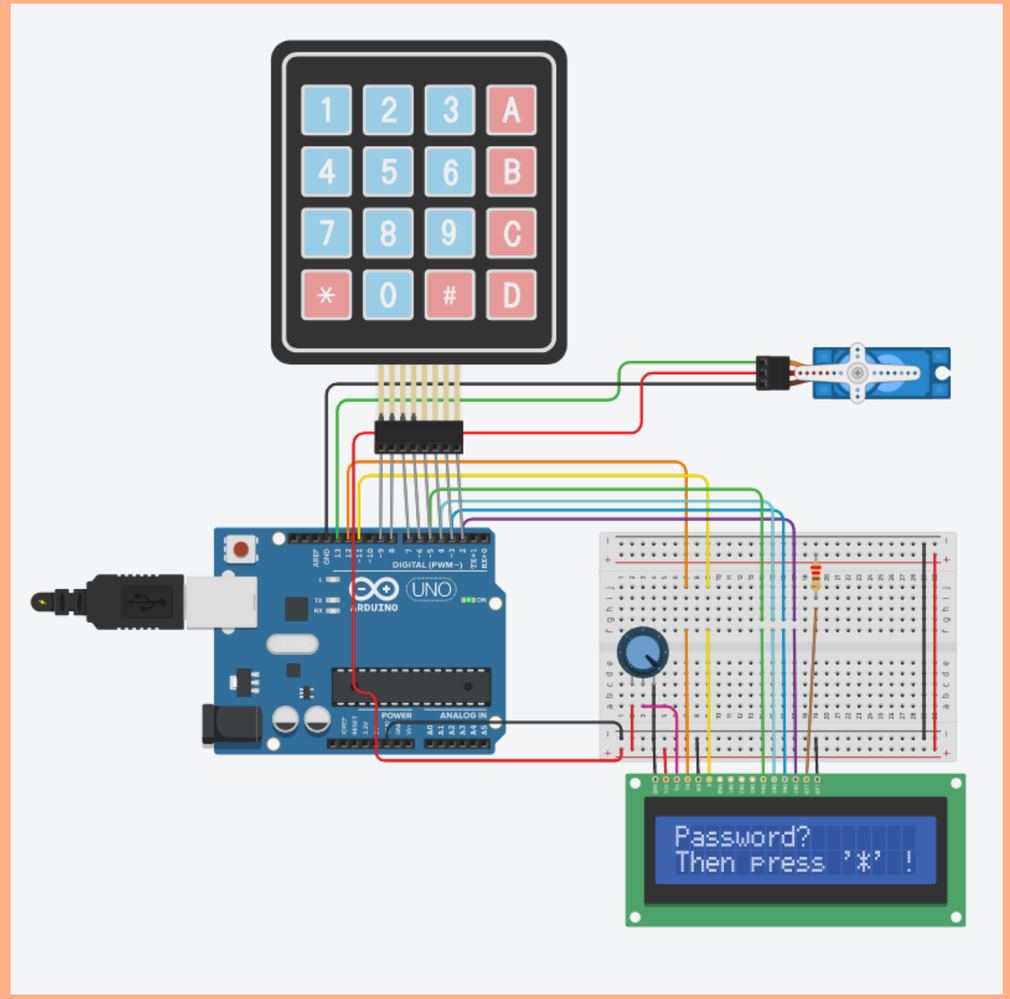
```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Outcome of the simulation

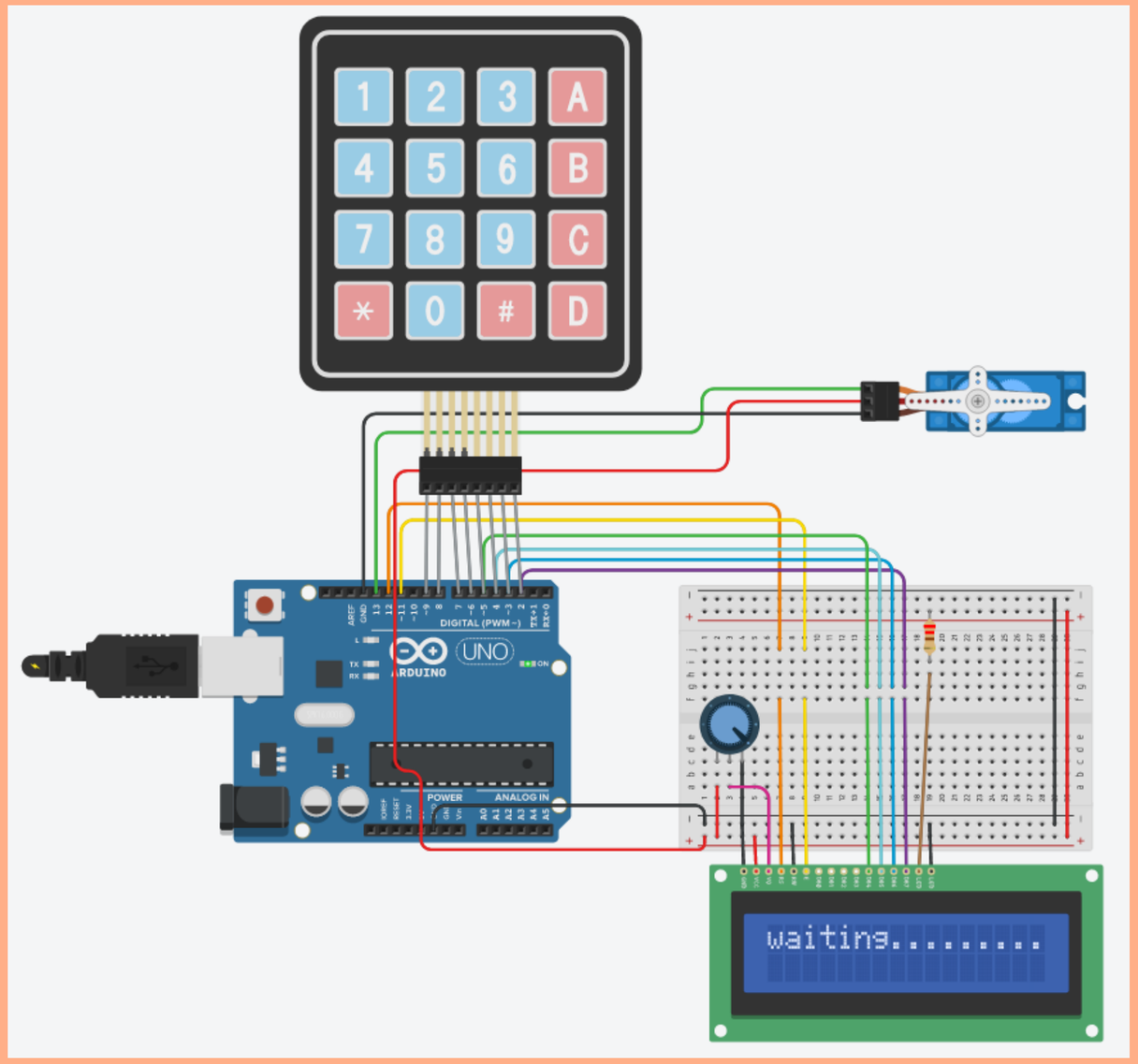
When Arduino Uno is connected, the LCD displays a welcome message



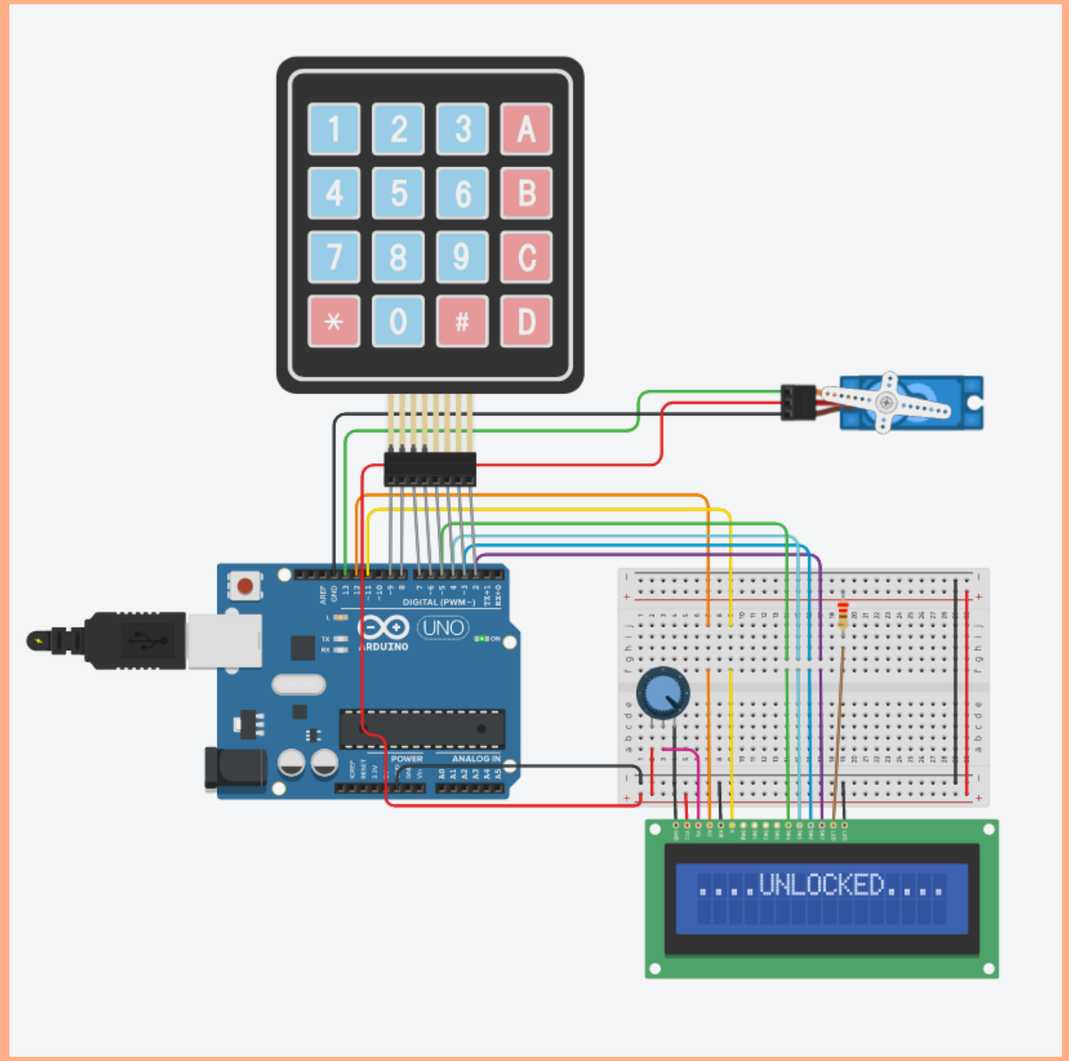
Then, it asks for the password for the door to unlock



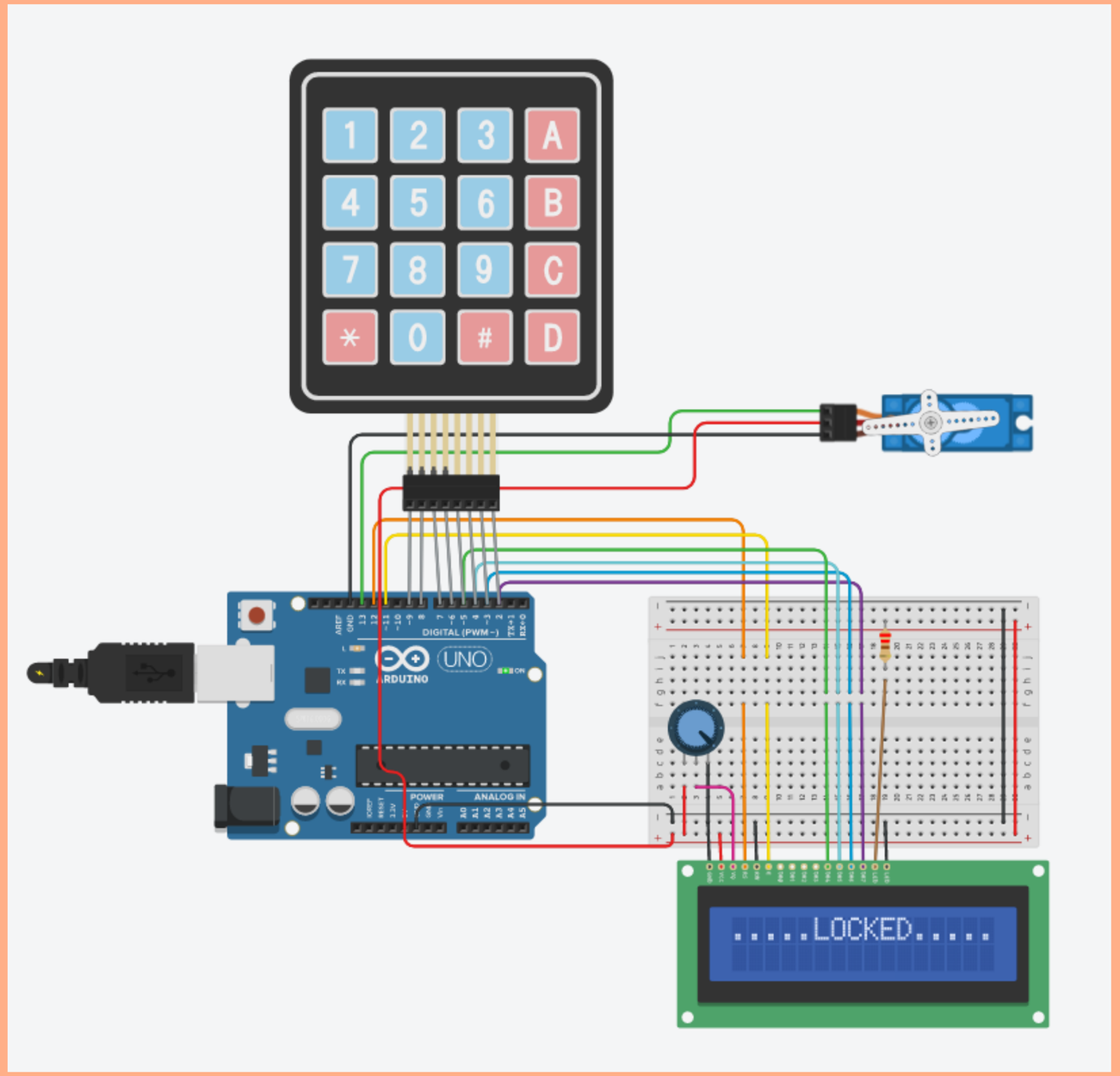
It then waits for the user to type in the password



When the right password is entered, the door unlocks and waits precisely 2 seconds before locking itself. The state of the door (unlocked/locked) is shown.



Waits for 2 seconds.



- Three key pieces of hardware—an Arduino microcontroller, a Servo motor, and a smartphone—will be used to power the Blynk system.

- The servo motor and sensor are managed by the IoT Controlled Arduino. The Arduino will thereafter be in standby mode and await the user's monitoring action. The information obtained through a phone will always appear on phone applications, whether they are locked or unlocked.

- Making sure the door is locked or unlocked is the first step in the method. The user will receive a notification on their phone if the door is unlocked. The user may then use their phone to lock the door.

- On the other hand, even when they are distant from the door, the user simply locks or unlocks the door using their phone. The project's schematic, shown in Figure 12, has a servo motor that detects and keeps track of the door lock.

- The door will notify the user when it is locked or unlocked. The user may then use the Blynk app on their phone to lock or open the door.

Source used mostly for the idea of keypad-based door unlock/lock : https://tatiuc.edu.my/ijset/index.php/ijset/article/download/93/59

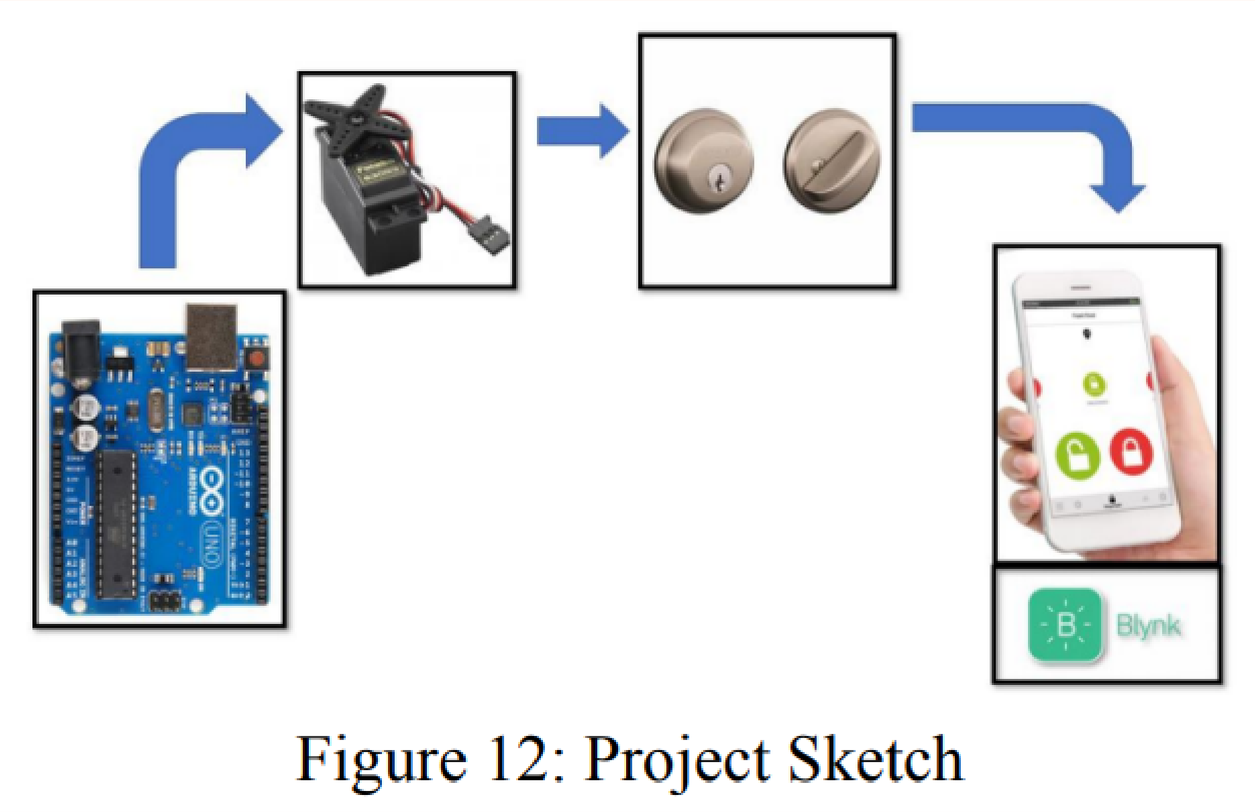
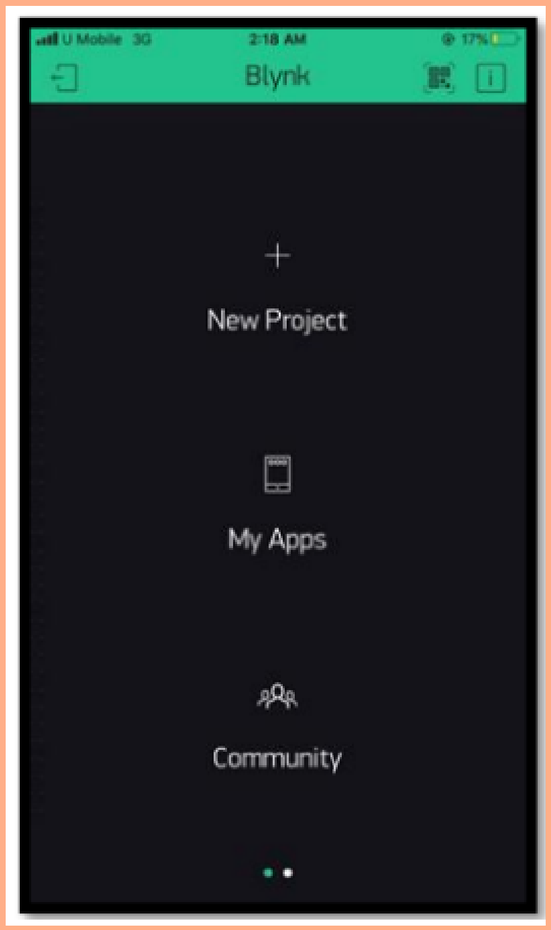
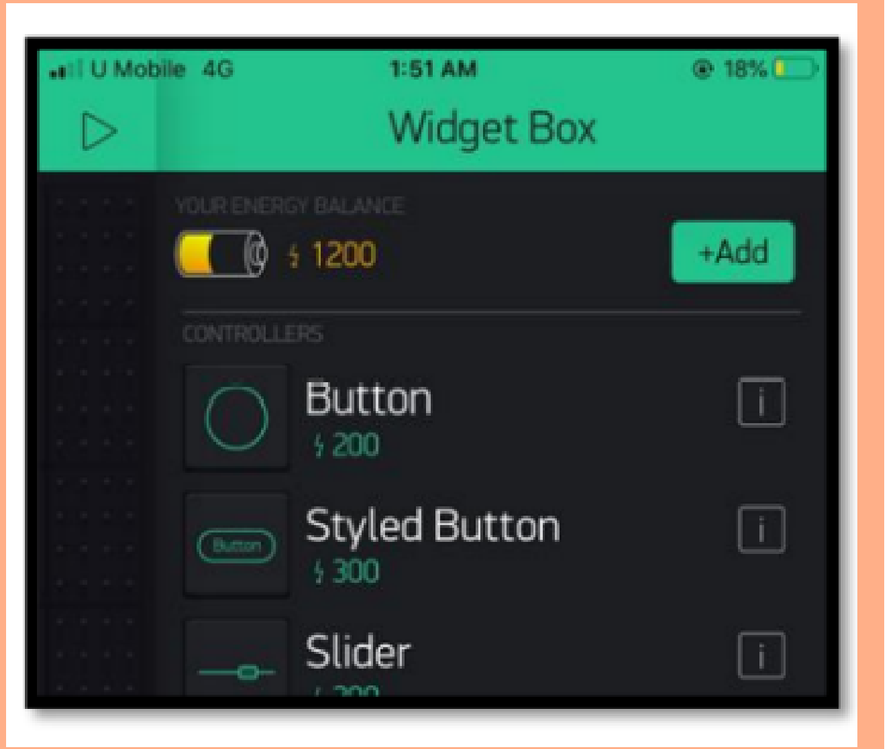
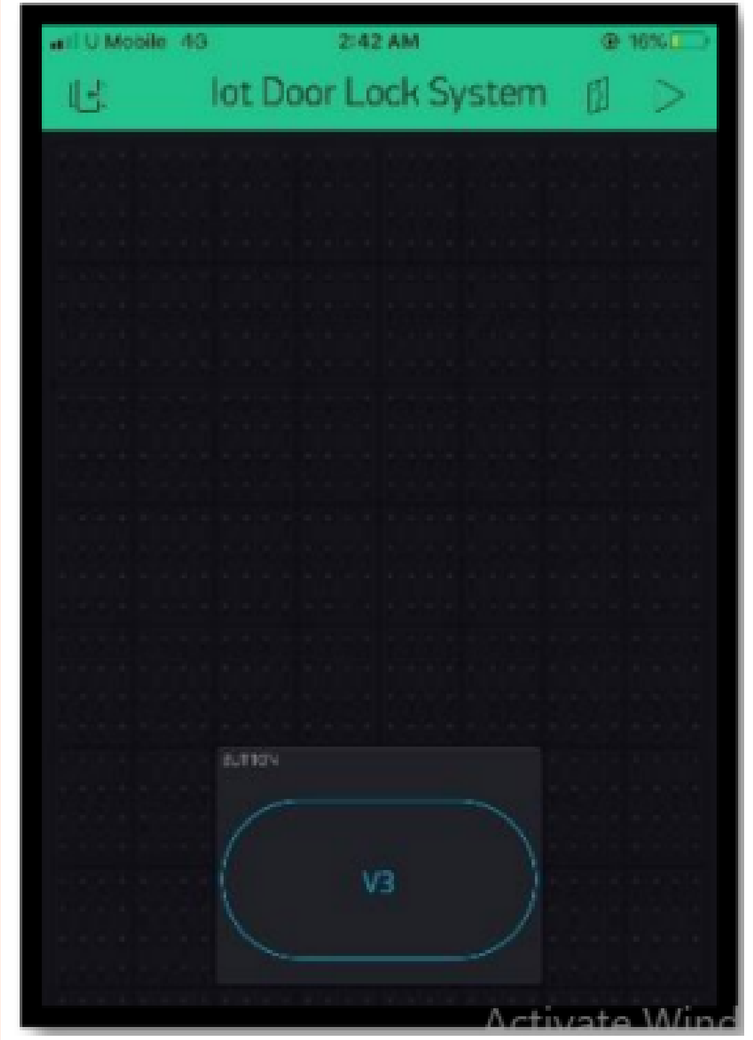
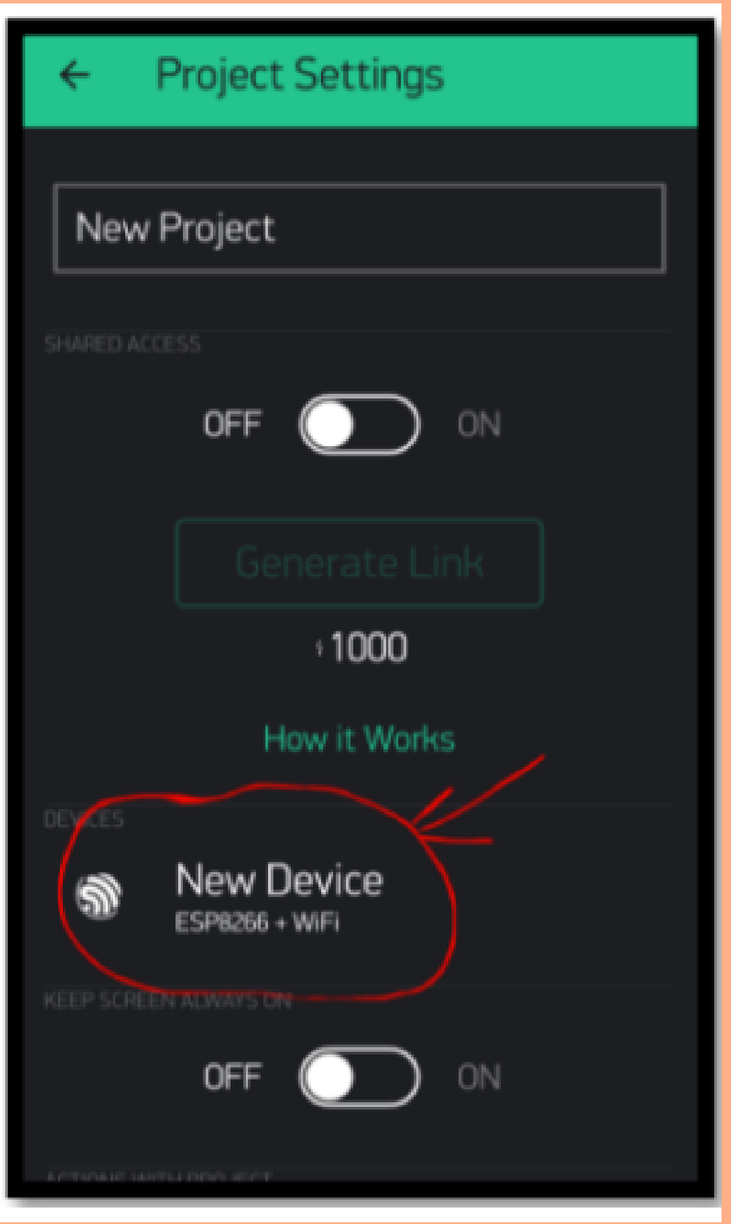


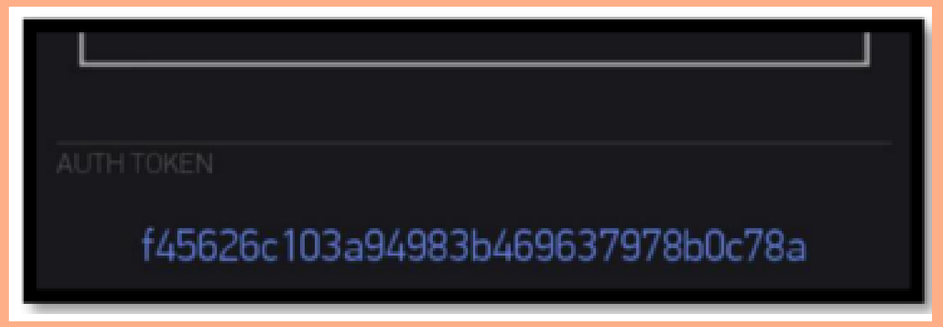
Figure 12: Project Sketch

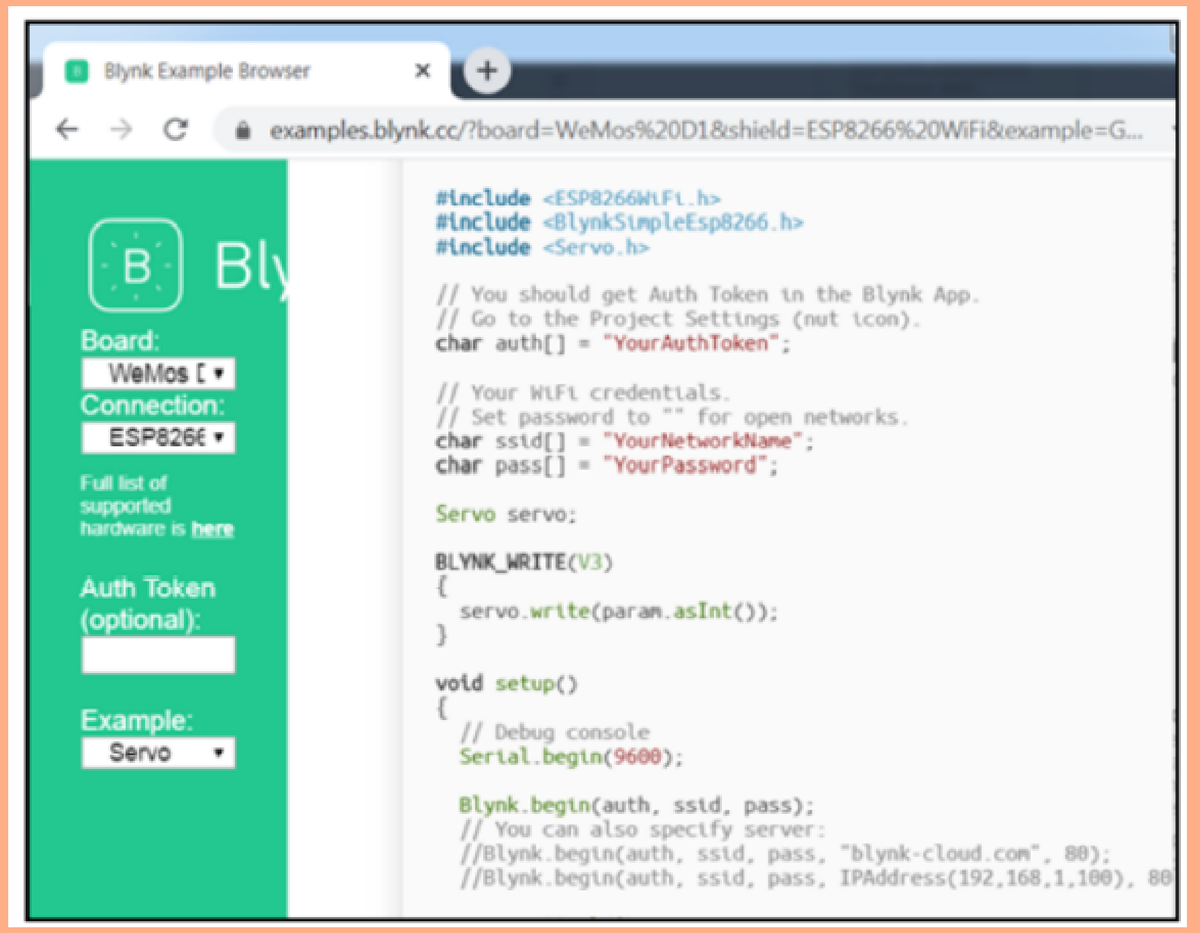



U Mobile 4G
1:51 AM
18%
Widget Box
YOUR ENERGY BALANCE
1200
+Add
CONTROLLERS
Button
200
Styled Button
300
Slider


U Mobile 4G
2:42 AM
16%
Iot Door Lock System
V3


Project Settings
New Project
SHARED ACCESS
OFF
ON
Generate Link
1000
How it Works
New Device
ESP8266 + WiFi
KEEP SCREEN ALWAYS ON
OFF
ON


AUTH TOKEN
f45626c103a94983b469637978b0c78a


Blynk Example Browser
examples.blynk.cc/?board=WeMos%20D1&shield=ESP8266%20WiFi&example=G...
Board:
WeMos D
Connection:
ESP8266
Full list of supported hardware is here
Auth Token (optional):
Example:
Servo
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Servo.h>
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";
Servo servo;
BLYNK_WRITE(V3)
{
servo.write(param.asInt());
}
void setup()
{
// Debug console
Serial.begin(9600);
Blynk.begin(auth, ssid, pass);
// You can also specify server:
//Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
//Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 80

1. **Control With the Blynk App, relay & NodeMCU ESP8266 to display data for IR remote control for a light bulb.**

(Source: https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-switch-using-nodemcu-and-the-blynk-app)

1. Download the Blynk app from the Google Play Store (for Android users) or the Apple App Store (for iOS users) (iPhone users).

2. Following installation, sign up with an e-mail address that we have access to.

3. We will receive authentication for our projects via this e-mail.

4. After we've signed up, click the "+" symbol to start a new project.

5. Name the project "IoT light Switch";

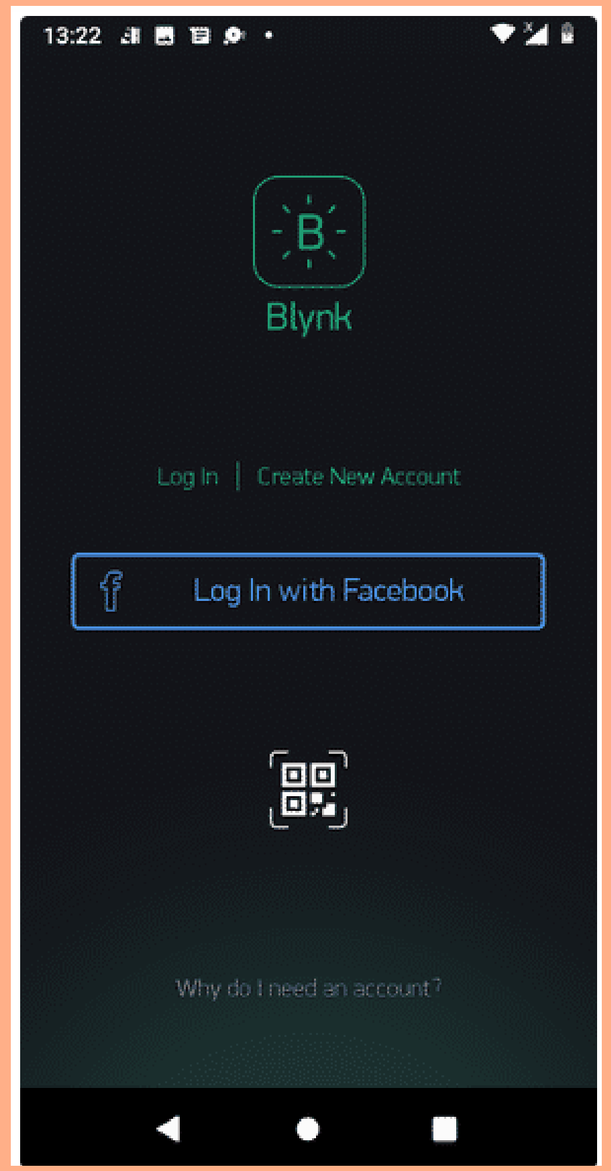
6. select NodeMCU as the device and WiFi as the connection type;

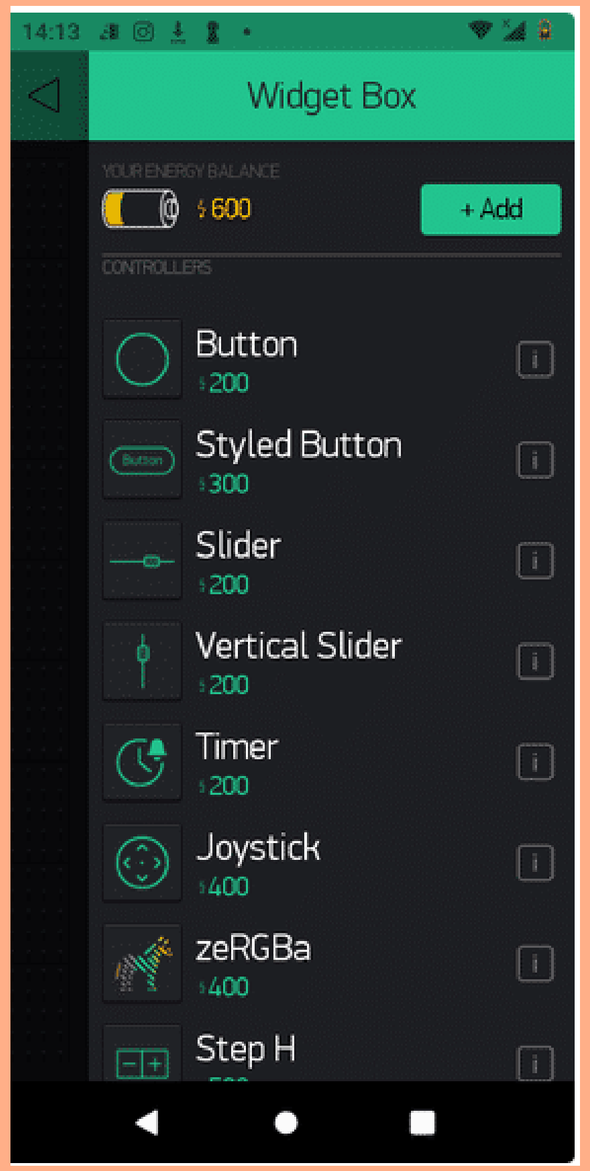
7. hit the create button at the bottom.

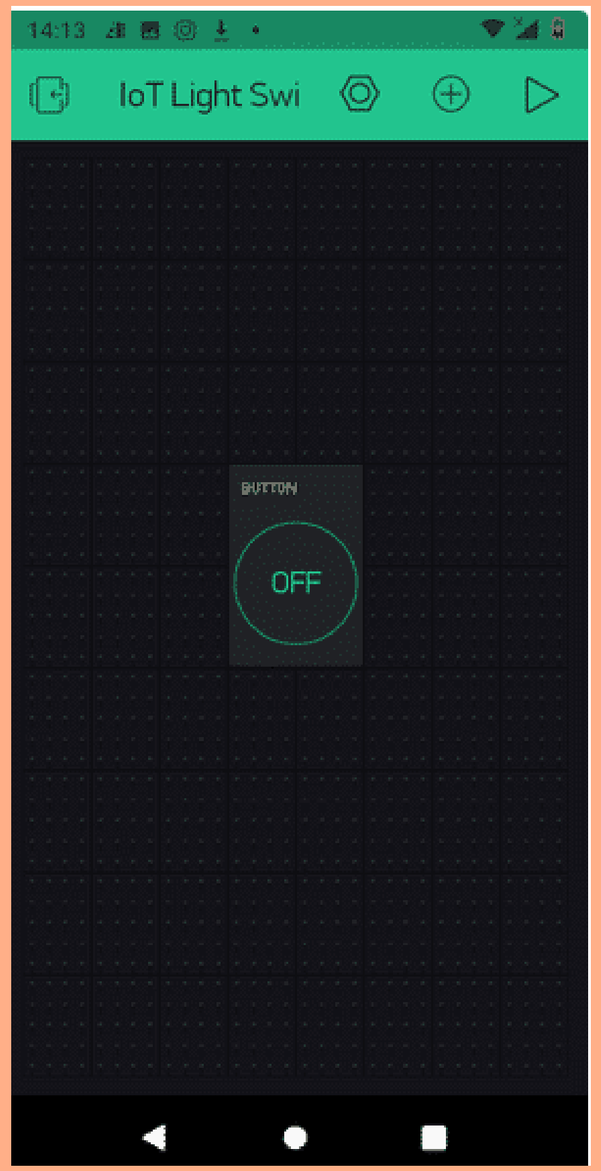
8. Add a button by tapping the "+" symbol.

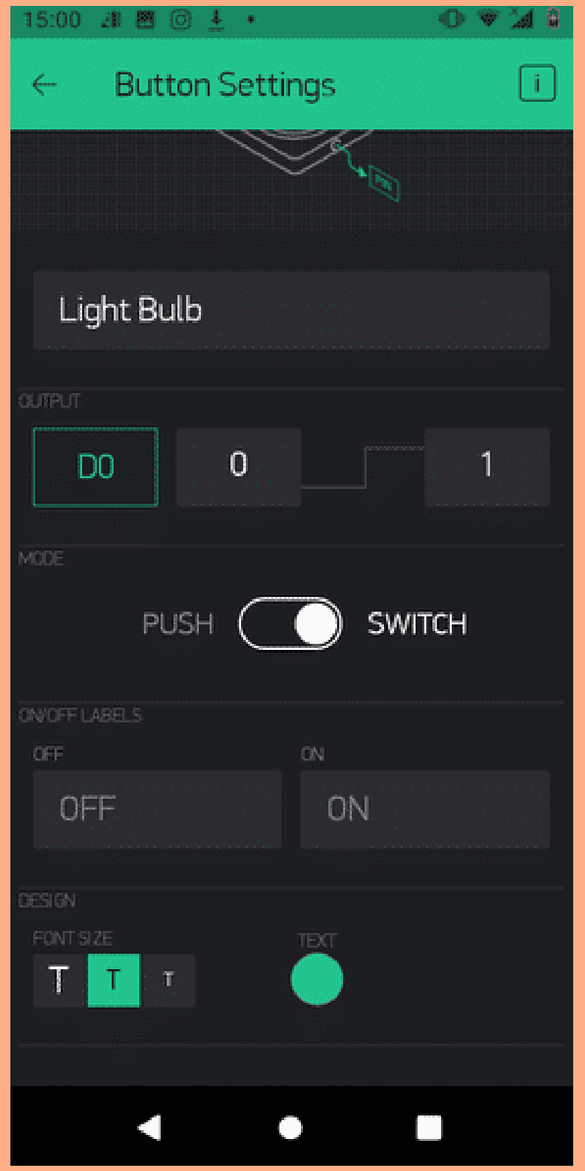
9. Click the button to access configuration and setup options.

10. Give the button the name "Light Bulb," attach the output pin to digital pin 0 (DO), and set the logic to 0-1.


13:22
Blynk
Log In | Create New Account
Log In with Facebook
Why do I need an account?


14:13
Widget Box
YOUR ENERGY BALANCE
600
+ Add
CONTROLLERS
Button
200
Styled Button
300
Slider
200
Vertical Slider
200
Timer
200
Joystick
400
zeRGBa
400
Step H


14:13
IoT Light Swi
BUTTON
OFF


15:00
Button Settings
Light Bulb
OUTPUT
D0
0
1
MODE
PUSH
SWITCH
ON/OFF LABELS
OFF
OFF
ON
ON
DESIGN
FONT SIZE
TEXT

- **The keyless entry system substantially streamlines entry processes to the point where you need to look for the finest home security system, which is the most visible enhancement smart technology brings to door locks.**
- **One of the most often lost objects is a set of house keys. The user does not need to worry about managing yet another key or how to transfer the key to another person, making it more convenient. You won't have to worry about misplacing or losing your keys because it provides quick authentication and unlocks.**
- **The goals of this project have been accomplished. The goals of this project have been accomplished. By utilizing the Blynk application framework, this project seeks to construct an IoT monitoring system. This project consists of an Internet of Things (IoT) system that was constructed using an Arduino Wemos D1 as the microcontroller and a Blynk monitoring program as the microprocessor.**
- **By completing this project, virtual keys provide users with a useful security alternative that allows them to quickly lock and open doors on their Android or Apple mobile using free software. Locks and door unlocking may both be controlled by a single Blynk application. Most significantly, by integrating IoT apps into the system, it should be user-friendly and even accessible to non-technical users.**

2. **The Code** (Blynk app)

- **Connect our nodeMCU to the computer's USB port using a USB power supply cord.**
- **Go to Tools > Board in the Arduino IDE and choose the proper NodeMCU board, such as NodeMCU 1.0. (ESP -12E Module).**
- **Go to Tools > Port > and select the proper port to select the pertinent port.**
- **To access the code, click File > Examples > Blynk > Boards Wifi > NodeMCU (select to include code). The code will resemble this:**

```
/*************************************************************
  Download latest Blynk library here:
    https://github.com/blynkkk/blynk-library/releases/latest

  Blynk is a platform with iOS and Android apps to control
  Arduino, Raspberry Pi and the likes over the Internet.
  You can easily build graphic interfaces for all your
  projects by simply dragging and dropping widgets.

    Downloads, docs, tutorials: http://www.blynk.cc
    Sketch generator:           http://examples.blynk.cc
    Blynk community:            http://community.blynk.cc
    Follow us:                  http://www.fb.com/blynkapp
                                http://twitter.com/blynk_app

  Blynk library is licensed under MIT license
  This example code is in public domain.
```

```
19 *************************************************************
20  This example runs directly on NodeMCU.
21
22  Note: This requires ESP8266 support package:
23    https://github.com/esp8266/Arduino
24
25  Please be sure to select the right NodeMCU module
26  in the Tools -> Board menu!
27
28  For advanced settings please follow ESP examples :
29   - ESP8266_Standalone_Manual_IP.ino
30   - ESP8266_Standalone_SmartConfig.ino
31   - ESP8266_Standalone_SSL.ino
32
33  Change WiFi ssid, pass, and Blynk auth token to run :)
34  Feel free to apply it to any other example. It's simple!
35 *************************************************************/
```

```
35  *************************************************************/
36
37 /* Comment this out to disable prints and save space */
38 #define BLYNK_PRINT Serial
39
40
41 #include <ESP8266WiFi.h>
42 #include <BlynkSimpleEsp8266.h>
43
44 // You should get Auth Token in the Blynk App.
45 // Go to the Project Settings (nut icon).
46 char auth[] = "YourAuthToken";
47
48 // Your WiFi credentials.
49 // Set password to "" for open networks.
50 char ssid[] = "YourNetworkName";
51 char pass[] = "YourPassword";
```

```
53 void setup()
54 {
55   // Debug console
56   Serial.begin(9600);
57
58   Blynk.begin(auth, ssid, pass);
59   // You can also specify server:
60   //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
61   //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
62 }
63
64 void loop()
65 {
66   Blynk.run();
67 }
68
```

- **Now copy our authentication code and paste it into char auth in your email inbox.**
- **Then enter our network credentials, where pass is the password for the network we are using and ssid is the network name you are using.**

```
char auth[] = "YourAuthToken";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";
```

- **We may upload the code after everything is set up.**
- **Tap the "play" icon after launching the Blynk app on our smartphone. Now, we'll be able to turn on and off our light bulb.**

Over Wi-Fi, BLYNK controls a servo motor (source: https://www.instructables.com/Servo-Motor-Controlled-With-BLYNK-Over-WiFi/)

Step 1: Things We Need

Hardware:

- Servo SG90
- NodeMCU
- MicroUSB cable
- Jumper Wires

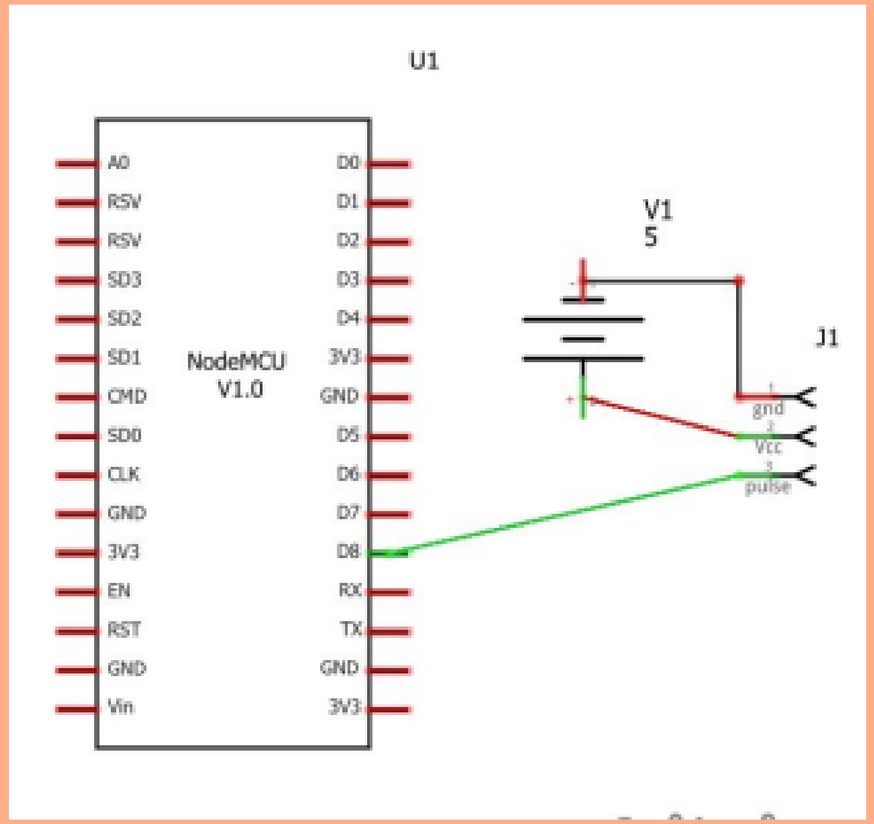
Software:

- Arduino IDE
- BLYNK app

Step 2: Circuit and Connections

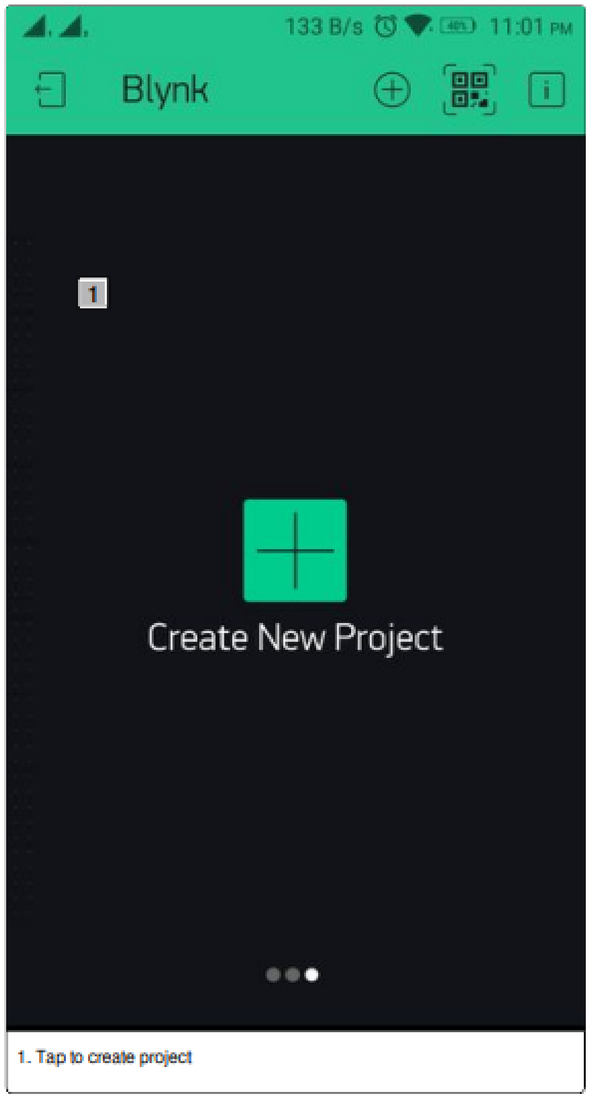
Connections:

- D8 pin of NodeMCU connects to Command pin of Servo Motor.
- 3V3 PIN of NodeMCU connects to Power PIN of Servo motor (usually RED one).
- GND PIN of NodeMCU connects to GND of servo motor (usually BLACK one).

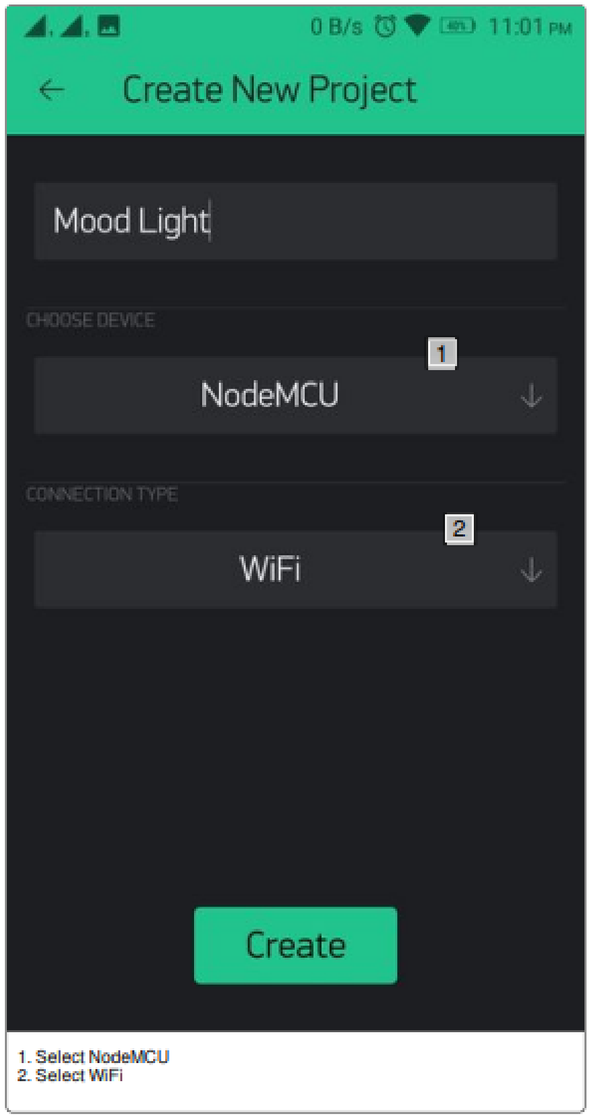


Step 3: BLYNK App Setup on Phone

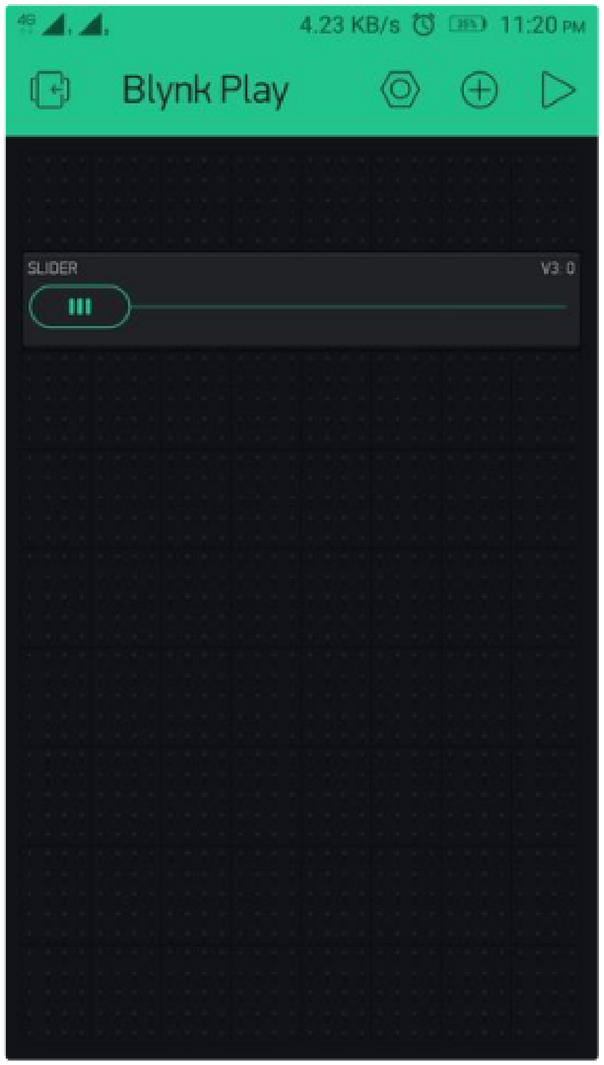
- Create a New Project in BLYNK app.
- Write Project Name and Select ESP8266 or NodeMCU from dropdown.
- An AUTH token will be sent to your registered email, note this down.
- Tap on the screen and add a SLIDER widget on the screen.
- Tap on the Widget, select Digital PIN 8 and Start value must be 0 and End Value must be 180 (make sure to these values must not less than 0 and greater than 180 or you might break your servo).

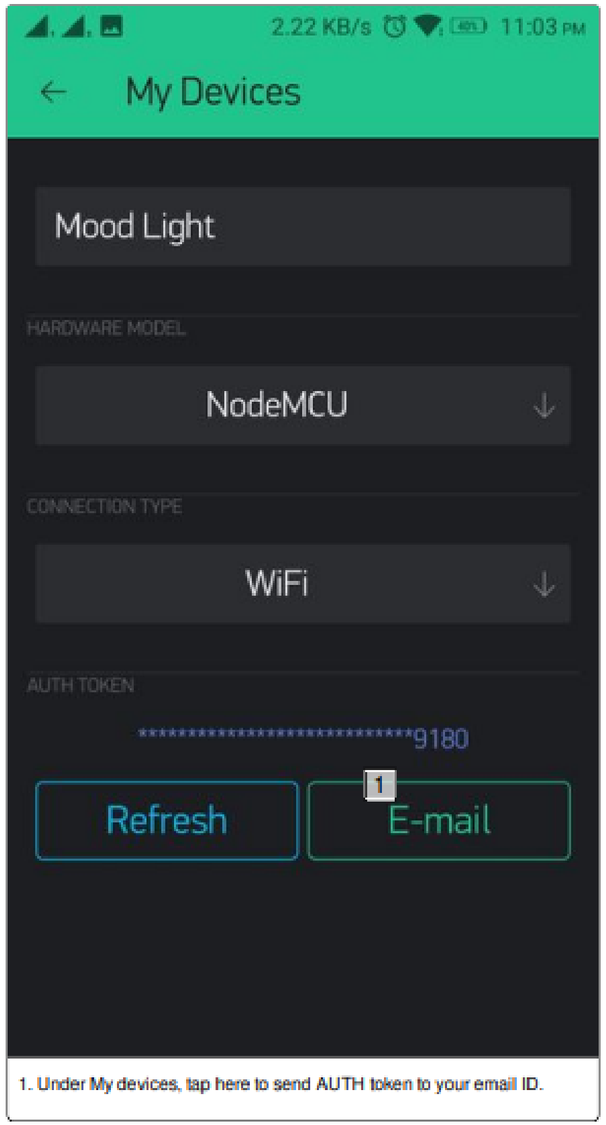


1. Tap to create project



1. Select NodeMCU
2. Select WiFi





1. Under My devices, tap here to send AUTH token to your email ID.

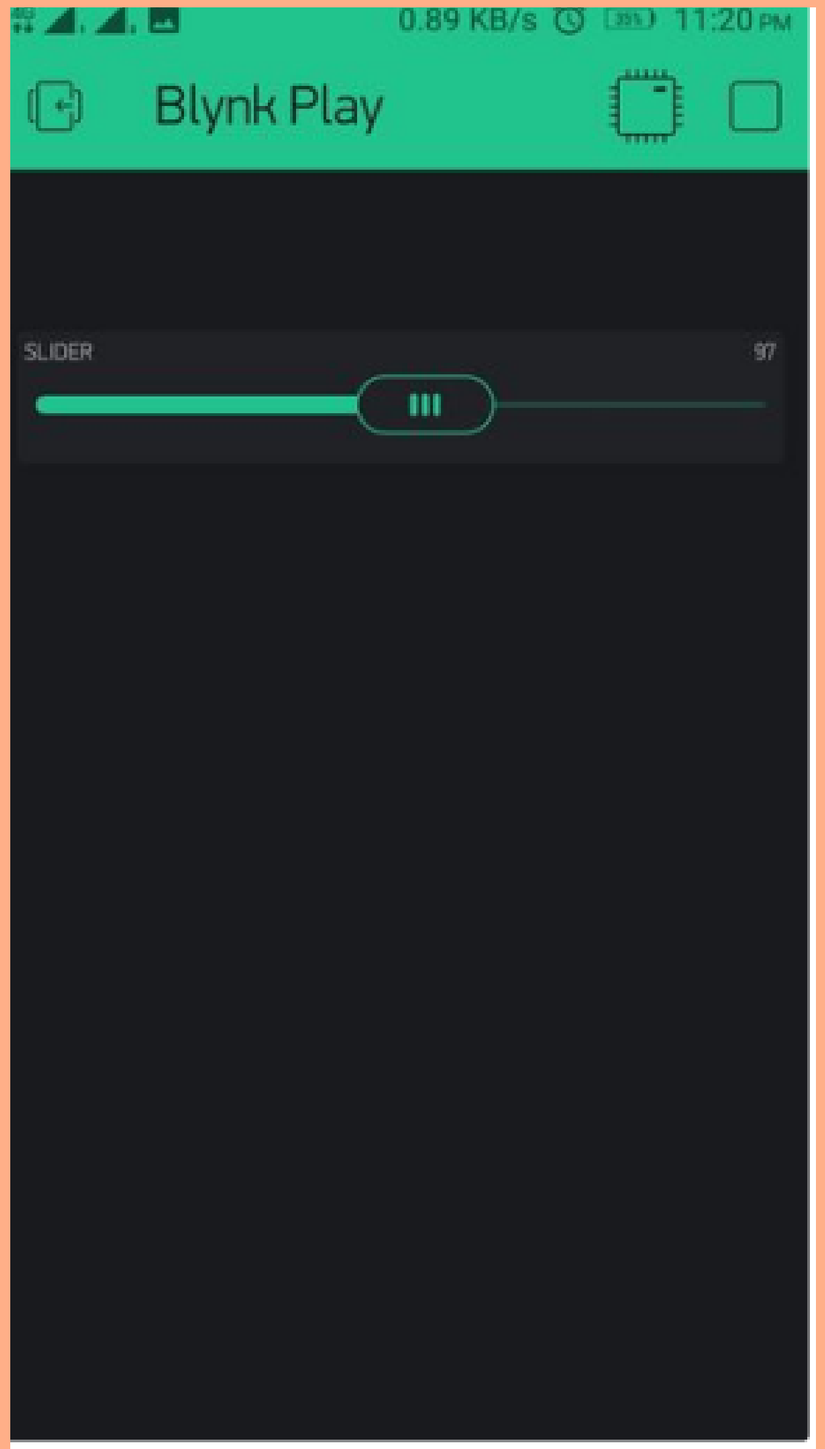
Step 4: Time to Code:

We only need to add our AUTH ID(noted above), Wifi SSID and Password in the code and upload it to NodeMCU using Arduino IDE. In the code, we do not need to setup our pins this is already done by BLYNK in their library.

Code:

```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial



#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.

// Go to the Project Settings (nut icon).

char auth[] = "YourAuthToken";

// Your WiFi credentials.

// Set password to "" for open networks.

char ssid[] = "YourNetworkName";

char pass[] = "YourPassword";

void setup() {

// Debug console Serial.begin(115200);

Blynk.begin(auth, ssid, pass);

// You can also specify server:

//Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);

//Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);

}

void loop()

{

Blynk.run(); // You can inject your own code or combine it with other sketches.

}
```

If you do not have BLYNKEsp8266 library, download the BLYNKEsp library from Sketch menu --> Include Library -->Manage Libraries. Search for the BLYNK and install it.



Step 5: Upload Your Code to NodeMCU

Upload the code to your NodeMCU or ESP8266 and device will be displayed online on your BLYNK app. Click Play button on top right of your app and slide over the slider to rotate Servo as much you need.

Overall sources mostly utilised for the idea:

- https://tatiuc.edu.my/ijset/index.php/ijset/article/download/93/59
- https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-switch-using-nodemcu-and-the-blynk-app)

.....

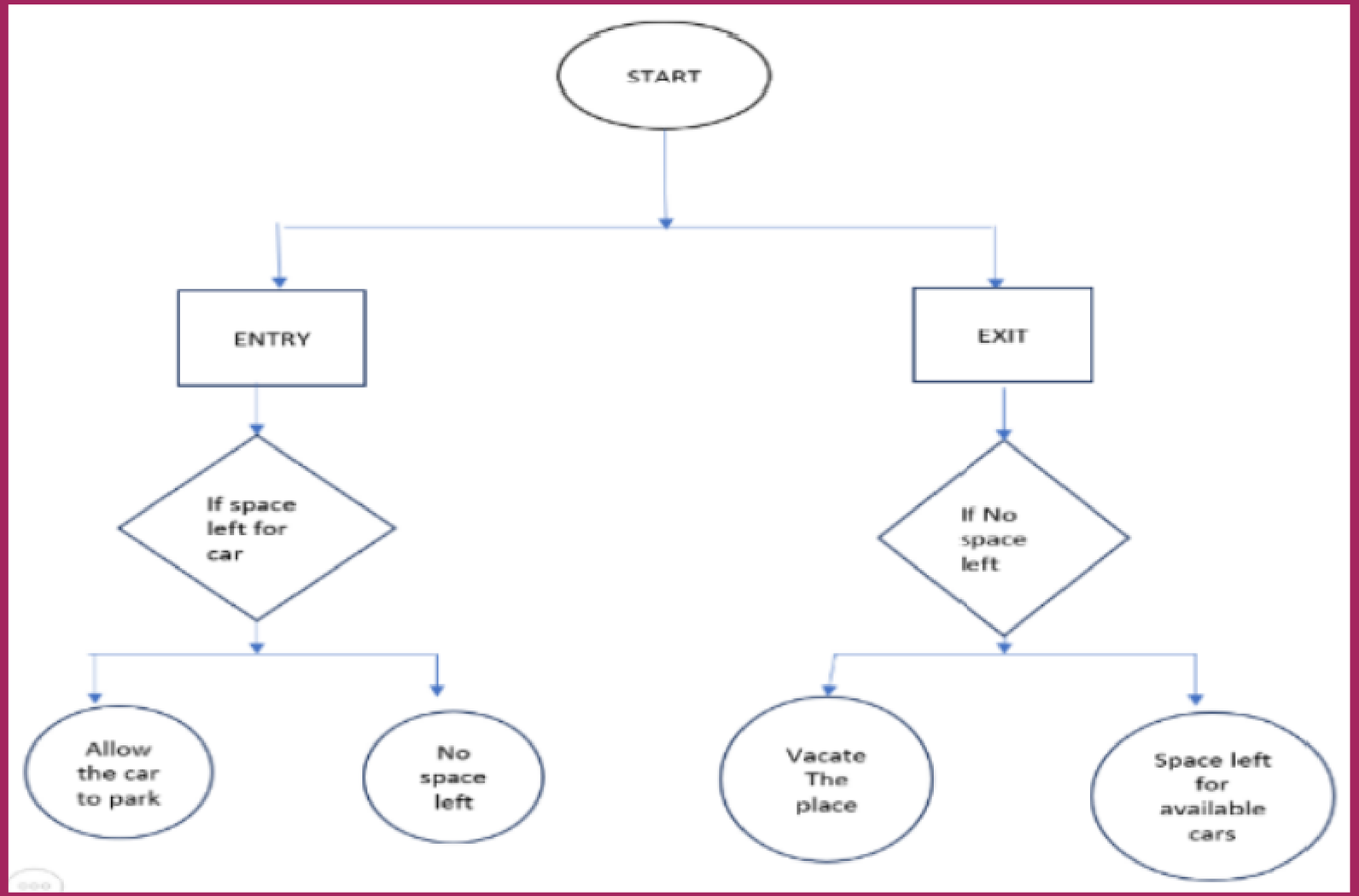
2.Automated Car parking system

List of components used in this application

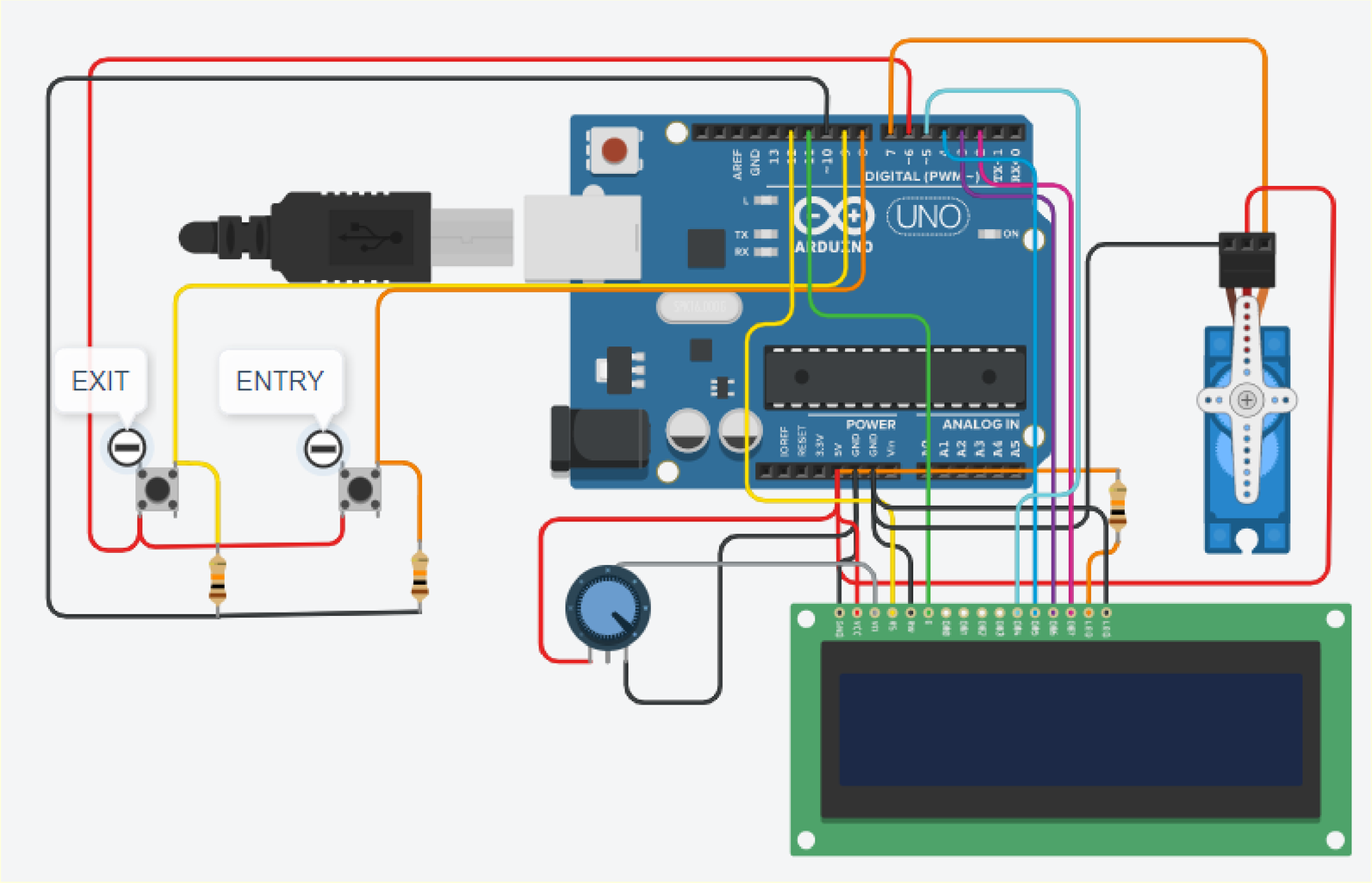
Name	Quantity	Component
Button	2	Pushbutton
R1, R2, R3	3	10 kΩ Resistor
SERVO1	1	servo motor
U2	1	LCD 16 x 2
U1	1	Arduino Uno R3
Rpot1	1	250 kΩ Potentiometer

Some other components are also in the list as they were used: ESP8266 NodeMCU, Blynk server, connecting the WiFi device is an Arduino board based on the ATMega328 AVR microcontroller, ESP8266 NodeMCU WiFi module, simple LEDs more than one & IR sensors. Some other components are also in the list as they were used: ESP8266 NodeMCU, Blynk server, connecting the WiFi device is an Arduino board based on the ATMega328 AVR microcontroller, ESP8266 NodeMCU WiFi module, simple LEDs more than one & IR sensors.

Flow diagram of a garage parking lot.



The circuit along with the components



Tinkercad simulation video with explanation of the function:

Coding(Text)

```
#include <Servo.h>
#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);//connected to RS,EN,D4,D5,D6,D7 of LCD
display respectively
Servo myservo;  // create servo object to control a servo

#define ServoM    7        //Connected to the servo motor.
#define Exit      9         //Pin connected to the EXIT sensor.
#define In        8         //Pin connected to the IN sensor.
#define Pwr  6              //Extra power pin for sensors
(Don't connect servo's power to this!)
#define Gnd  10             //Extra groung pin for sensors
(Don't connect servo's power to this!)
#define BarLow    90      //Low position of the barrier.
#define BarUp     177       //Up position of the barrier.
#define CAPACITY  7         //Capacity of the parking lot.



void setup(){
  myservo.attach(ServoM);          // attaches the servo.
```

```
void setup(){
  myservo.attach(ServoM);           // attaches the servo.
  lcd.begin(16,2);
  //lcd.print("Space left for");
  pinMode(Gnd, OUTPUT);
  pinMode(Pwr, OUTPUT);
  pinMode(Exit, INPUT);            // set "EXIT" sensor pin to input
  pinMode(In, INPUT);              // set "IN" sensor pin to input
  digitalWrite(Gnd, LOW);
  digitalWrite(Pwr, HIGH);
  myservo.write(BarLow);           //Barrier in the low position
//  delay(1000);
}

int  Available= 7;                      // Number of places available.

//=============================================================
void loop(){
if (Available == 1){
  lcd.clear();
```

```
//============================
void loop(){
if (Available == 1){
  lcd.clear();
lcd.setCursor(1,0);
lcd.print("Space left for");
lcd.setCursor(0,1);
lcd.print(Available);
lcd.print(" car");
}else{
if (Available >= 1){
lcd.clear();
lcd.setCursor(1,0);
lcd.print("Space left for");
lcd.setCursor(0,1);
lcd.print(Available);
lcd.print(" cars");
}else{
  lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("Sorry!");
```

```
}else{
if (Available >= 1){
lcd.clear();
lcd.setCursor(1,0);
lcd.print("Space left for");
lcd.setCursor(0,1);
lcd.print(Available);
lcd.print(" cars");
}else{
  lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("Sorry!");
    lcd.setCursor(0,1);
    lcd.print("No place left!");
}
}


if(digitalRead(In)==1)
{
  if(Available != 0){
```

```
if(digitalRead(In)==1)
{
  if(Available != 0){
    Available--;
    myservo.write(BarUp);
    delay(3000);
    myservo.write(BarLow);
    }
  }
if(digitalRead(Exit)==1)
{
  if(Available != CAPACITY){
    Available++;
    myservo.write(BarUp);
    delay(3000);
    myservo.write(BarLow);
    }
}
  delay(20);
}
```

Coding(Block)

```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

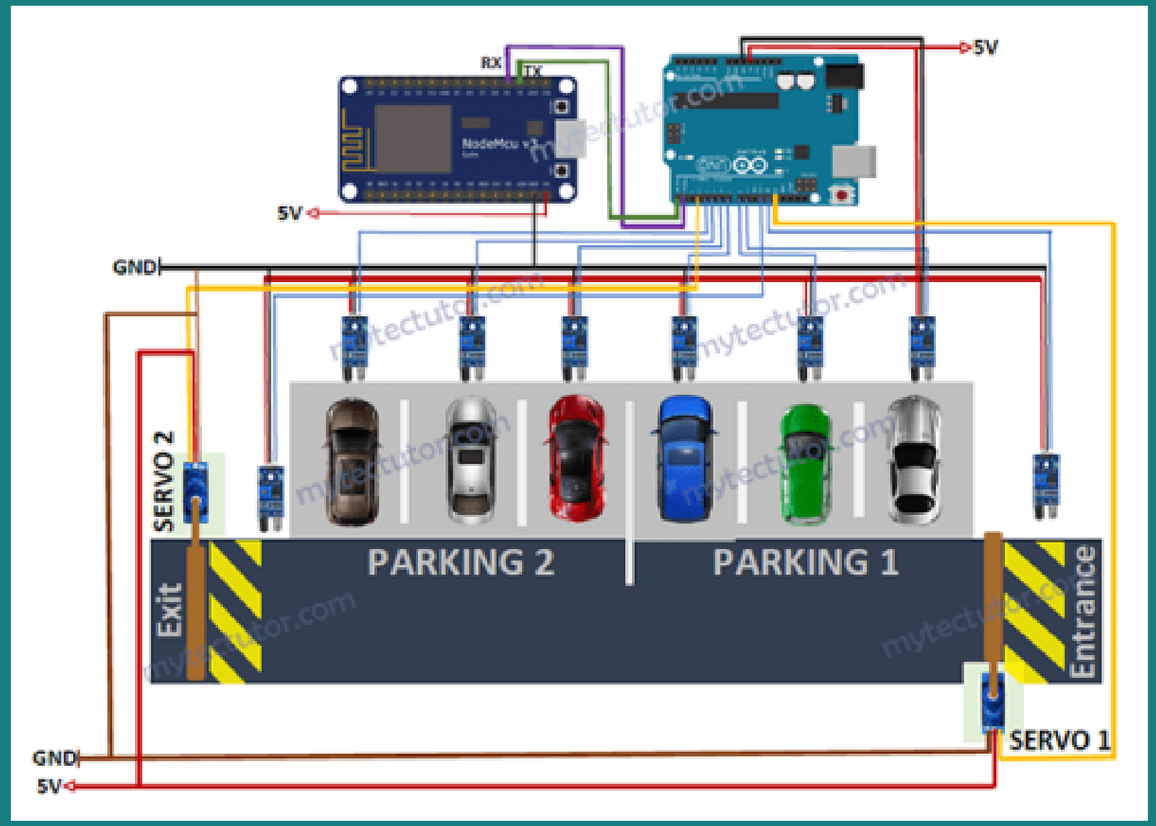
Using Blynk to represent Data (source: https://mytectutor.com/smart-car-parking-system-using-arduino-esp8266-nodemcu-and-blynk-server/)

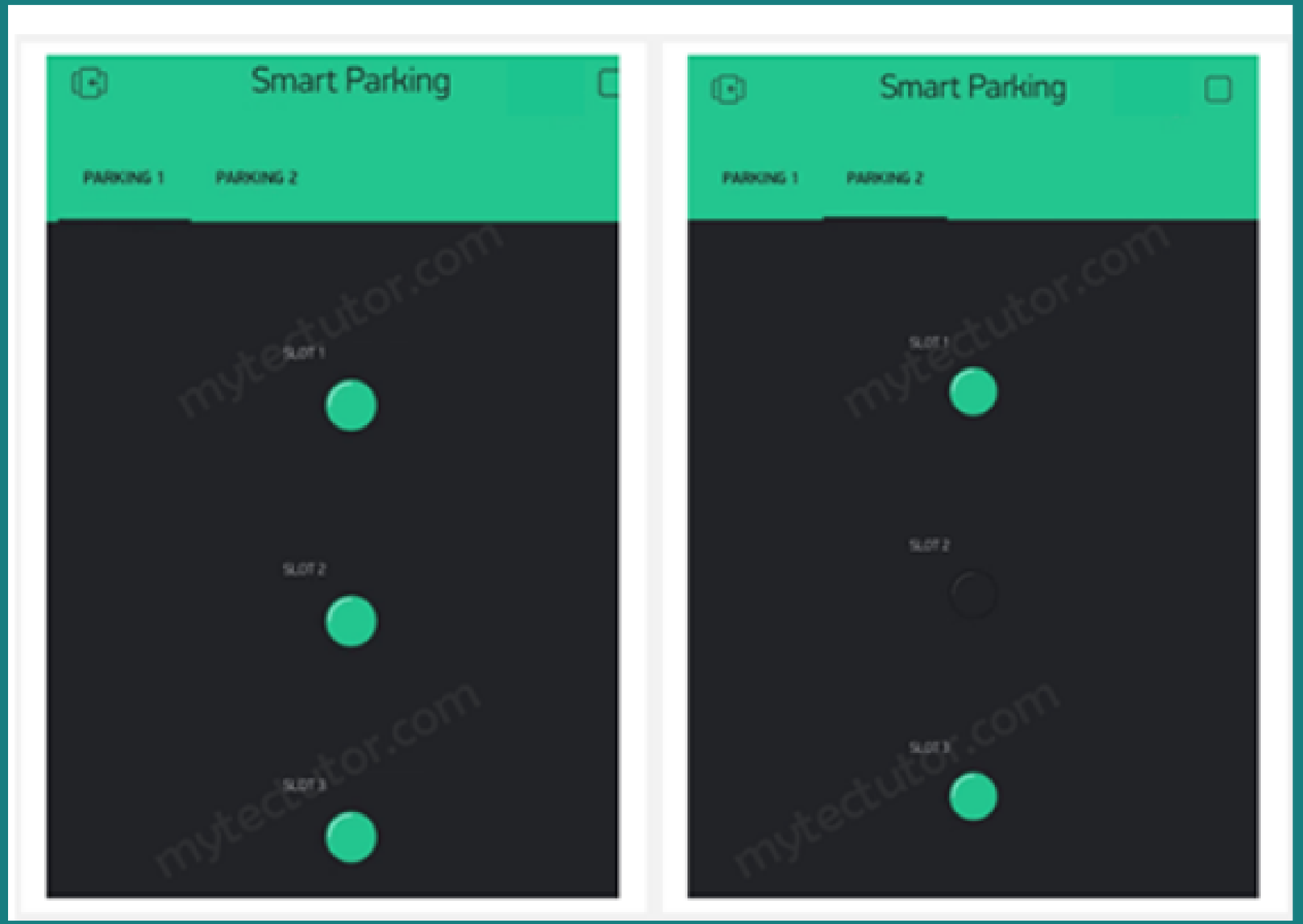
- A smart auto parking system requires hardware components. Servo motors are employed to replicate the opening and closing of the toll gates at the parking lot entrance and exit.

- In this instance, IR sensors are used to determine if a car has taken up a certain spot. These sensors send a signal to the microcontroller.

- The main control device for receiving signals from the IR sensors and motors and connecting the parking lot to the WiFi device is an Arduino board based on the ATMega328 AVR microcontroller.

- The parking lot will be connected to a cloud service using an ESP8266 NodeMCU WiFi module so that we may access the parking lot online using a mobile phone application.




Smart Parking
PARKING 1
PARKING 2
SLOT 1
SLOT 2
SLOT 3
Smart Parking
PARKING 1
PARKING 2
SLOT 1
SLOT 2
SLOT 3

Coding for the blynk

```
void loop()
{

    //------light intensity control------//
//-----------------------------------------------------------------
   int val1 = analogRead(LDR);
  if (val1 > 500)
       {
       digitalWrite(13, LOW);
    Serial.print("Bulb ON = ");
    Serial.print(val1);
       }
  else
       {
       digitalWrite(13, HIGH);
     Serial.print("Bulb OFF = ");
    Serial.print(val1);
       }
```

```
//-----------------------------------------------------------------
       //------ light & fan control --------//
//-----------------------------------------------------------------
  sen2Value = digitalRead(9);
  if (sen2Value == 0)
       {
       digitalWrite(10, LOW); //npn as switch OFF
       digitalWrite(4, HIGH); // Red LED ON,indicating no motion
       digitalWrite(3, LOW); //Green LED OFF, since no Motion detected
    Serial.print("     || NO Motion Detected    " );
       }

  if (sen2Value == 1)
       {
       digitalWrite(10, HIGH);//npn as switch ON
    delay(5000);
       digitalWrite(4, LOW); // RED LED OFF
       digitalWrite(3, HIGH);//GREEN LED ON , indicating motion detected
     Serial.print("        || Motion Detected!      " );
       }
```

```
void setup()
{
Serial.begin(9600);
nodemcu.begin(9600);

pinMode(parking1_slot1_ir_s, INPUT);
pinMode(parking1_slot2_ir_s, INPUT);
pinMode(parking1_slot3_ir_s, INPUT);

pinMode(parking2_slot1_ir_s, INPUT);
pinMode(parking2_slot2_ir_s, INPUT);
pinMode(parking2_slot3_ir_s, INPUT);

pinMode(entrance_gate, INPUT);
pinMode(exit_gate, INPUT);

myservo1.attach(13);  // attaches the servo on pin 9 to the servo object
myservo2.attach(3);


}

void loop()
{

p1slot1();
p1slot2();
p1slot3();

p2slot1();
p2slot2();
p2slot3();

gates();
//conditions();
```

```
   cdata = cdata + sensor1 +"," + sensor2 + ","+ sensor3 +","+ sensor4 + "," + sensor5 + "," + sensor6 +",";
   // comma will be used a delimeter
   Serial.println(cdata);
   nodemcu.println(cdata);
   delay(6000); // 100 milli seconds
   cdata = "";
digitalWrite(parking1_slot1_ir_s, HIGH);
digitalWrite(parking1_slot2_ir_s, HIGH);
digitalWrite(parking1_slot3_ir_s, HIGH);

digitalWrite(parking2_slot1_ir_s, HIGH);
digitalWrite(parking2_slot2_ir_s, HIGH);
digitalWrite(parking2_slot3_ir_s, HIGH);

digitalWrite(entrance_gate, HIGH);
digitalWrite(exit_gate, HIGH);
}


void p1slot1() // parkng 1 slot1
{
  if( digitalRead(parking1_slot1_ir_s) == LOW)
  {
  sensor1 = "255";
 delay(200);
  }
if( digitalRead(parking1_slot1_ir_s) == HIGH)
{
  sensor1 = "0";
 delay(200);
}

}

void p1slot2() // parking 1 slot2
{
  if( digitalRead(parking1_slot2_ir_s) == LOW)
```

```
// now for parking 2

void p2slot1() // parking 1 slot3
{
  if( digitalRead(parking2_slot1_ir_s) == LOW)
  {
  sensor4 = "255";
  delay(200);
  }
if( digitalRead(parking2_slot1_ir_s) == HIGH)
  {
  sensor4 = "0";
 delay(200);
  }
}


void p2slot2() // parking 1 slot3
{
  if( digitalRead(parking2_slot2_ir_s) == LOW)
  {
  sensor5 = "255";
  delay(200);
  }
if( digitalRead(parking2_slot2_ir_s) == HIGH)
  {
  sensor5 = "0";
 delay(200);
  }
}


void p2slot3() // parking 1 slot3
{
  if( digitalRead(parking2_slot3_ir_s) == LOW)
  {
```

```
// now for parking 2

void p2slot1() // parking 1 slot3
{
  if( digitalRead(parking2_slot1_ir_s) == LOW)
  {
  sensor4 = "255";
  delay(200);
  }
if( digitalRead(parking2_slot1_ir_s) == HIGH)
  {
  sensor4 = "0";
 delay(200);
  }
}


void p2slot2() // parking 1 slot3
{
  if( digitalRead(parking2_slot2_ir_s) == LOW)
  {
  sensor5 = "255";
  delay(200);
  }
if( digitalRead(parking2_slot2_ir_s) == HIGH)
  {
  sensor5 = "0";
 delay(200);
  }
}


void p2slot3() // parking 1 slot3
{
  if( digitalRead(parking2_slot3_ir_s) == LOW)
  {
```

```
void p2slot3() // parking 1 slot3
{
  if( digitalRead(parking2_slot3_ir_s) == LOW)
  {
  sensor6 = "255";
  delay(200);
  }
if( digitalRead(parking2_slot3_ir_s) == HIGH)
  {
  sensor6 = "0";
 delay(200);
  }
}

// for the gates

void gates()
{
  if (digitalRead(exit_gate) == LOW)
      {
        for (pos2 = 90; pos2 <= 180 ; pos2 += 1) { // goes from 0 degrees to 180 degrees
          // in steps of 1 degree
          myservo2.write(pos2);              // tell servo to go to position in variable 'pos'
          delay(15);                        // waits 15ms for the servo to reach the position
        }
          delay(1000);
        for (pos2 = 180; pos2 >= 90; pos2 -= 1) { // goes from 180 degrees to 0 degrees
          myservo2.write(pos2);              // tell servo to go to position in variable 'pos'
          delay(15);                        // waits 15ms for the servo to reach the position
        }
      }

  if (((digitalRead(entrance_gate) == LOW)) && (( digitalRead(parking1_slot1_ir_s) == HIGH) ||
  ( digitalRead(parking1_slot2_ir_s) == HIGH) || ( digitalRead(parking1_slot3_ir_s) == HIGH) ||
  ( digitalRead(parking2_slot1_ir_s) == HIGH) || ( digitalRead(parking2_slot2_ir_s) == HIGH)||
  ( digitalRead(parking2_slot3_ir_s) == HIGH)))
```

```
void gates()
{
  if (digitalRead(exit_gate) == LOW)
      {
        for (pos2 = 90; pos2 <= 180 ; pos2 += 1) { // goes from 0 degrees to 180 degrees
          // in steps of 1 degree
          myservo2.write(pos2);              // tell servo to go to position in variable 'pos'
          delay(15);                        // waits 15ms for the servo to reach the position
        }
          delay(1000);
        for (pos2 = 180; pos2 >= 90; pos2 -= 1) { // goes from 180 degrees to 0 degrees
```

```
void gates()
{
  if (digitalRead(exit_gate) == LOW)
      {
        for (pos2 = 90; pos2 <= 180 ; pos2 += 1) { // goes from 0 degrees to 180 degrees
          // in steps of 1 degree
          myservo2.write(pos2);              // tell servo to go to position in variable 'pos'
          delay(15);                         // waits 15ms for the servo to reach the position
        }
          delay(1000);
        for (pos2 = 180; pos2 >= 90; pos2 -= 1) { // goes from 180 degrees to 0 degrees
          myservo2.write(pos2);              // tell servo to go to position in variable 'pos'
          delay(15);                         // waits 15ms for the servo to reach the position
        }
      }

  if (((digitalRead(entrance_gate) == LOW)) && (( digitalRead(parking1_slot1_ir_s) == HIGH) ||
  ( digitalRead(parking1_slot2_ir_s) == HIGH) || ( digitalRead(parking1_slot3_ir_s) == HIGH) ||
  ( digitalRead(parking2_slot1_ir_s) == HIGH) || ( digitalRead(parking2_slot2_ir_s) == HIGH)||
  ( digitalRead(parking2_slot3_ir_s) == HIGH)))
      {
        for (pos1 = 0; pos1 <= 90 ; pos1 += 1) { // goes from 0 degrees to 180 degrees
          // in steps of 1 degree
          myservo1.write(pos1);              // tell servo to go to position in variable 'pos'
          delay(15);                         // waits 15ms for the servo to reach the position
        }
          delay(1000);
        for (pos1 = 90; pos1 >= 0; pos1 -= 1) { // goes from 180 degrees to 0 degrees
          myservo1.write(pos1);              // tell servo to go to position in variable 'pos'
          delay(15);                         // waits 15ms for the servo to reach the position
        }
      }


}
```

Sources mostly used for the idea:

1.https://mytectutor.com/smart-car-parking-system-using-arduino-esp8266-nodemcu-and-blynk-server/

2.https://blynk.hackster.io/KaustubhAgarwal/smart-parking-bdfa99

Sources mostly used for the idea:

1. https://mytectutor.com/smart-car-parking-system-using-arduino-esp8266-nodemcu-and-blynk-server/
2. https://blynk.hackster.io/KaustubhAgarwal/smart-parking-bdfa99

3. Sensor-Based Smart Home Automation

List of components used in this application

Name	Quantity	Component
M1	1	DC Motor
Power supply	1	Power supply
PE11	1	Piezo buzzer
SMOKE DETECTOR MQ6	1	Gas Sensor
Slide Switch	1	Slide Switch

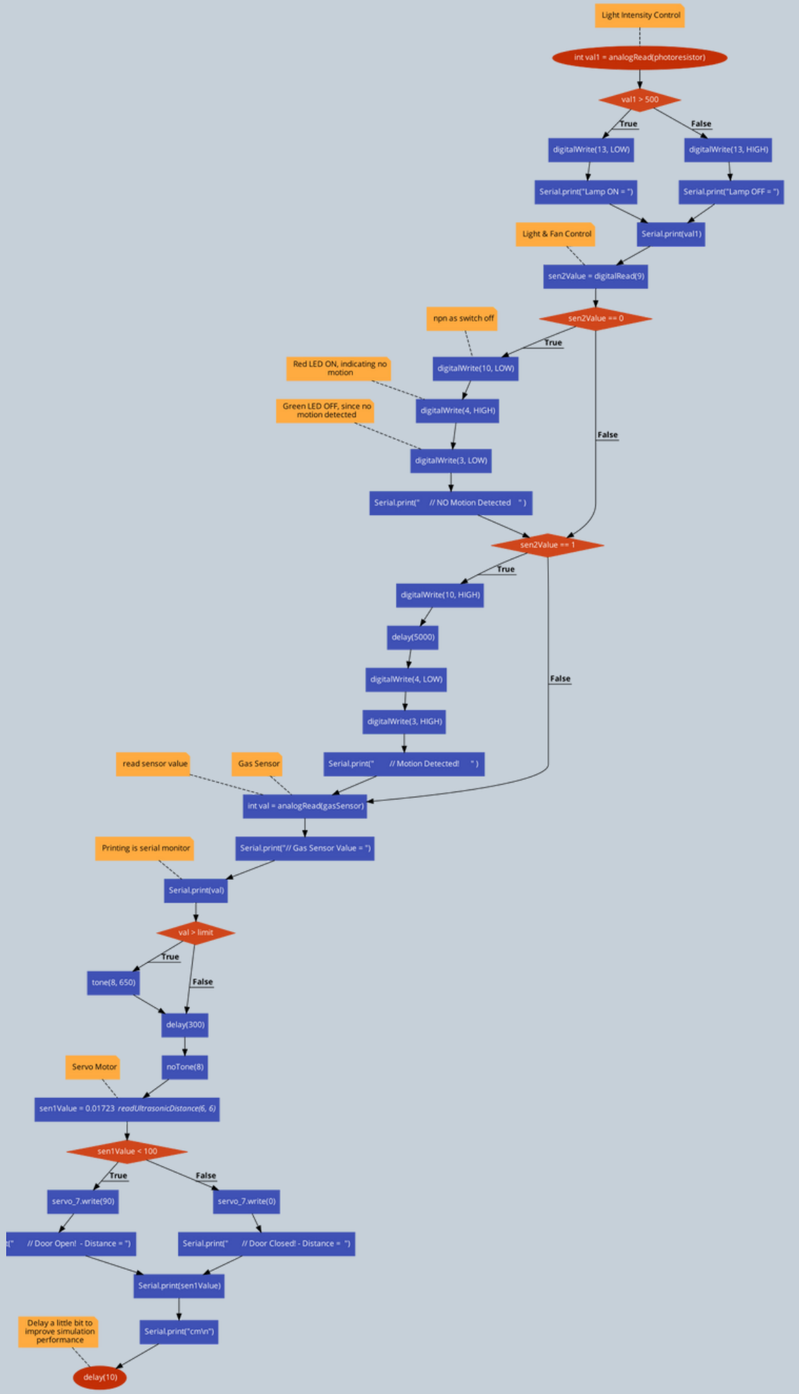
Name	Quantity	Component
PING1	1	Ultrasonic Distance Sensor
R1, R2	2	1 kΩ Resistor
SERVO1	1	servo motor
L1, L2	1	Light bulb
U1	1	Arduino Uno R3
LDR	1	Photoresistor
Relay SPDT	2	Relay SPDT i.e. Relay Board of 2 channel
PIR1	1	PIR Sensor

Name	Quantity	Component
ESP8266	1	Wi-Fi Development Board
Relay	1	5V Four-Channel Relay Module i.e. for practical while working with blynk
DC supply	1	5 volt DC supply

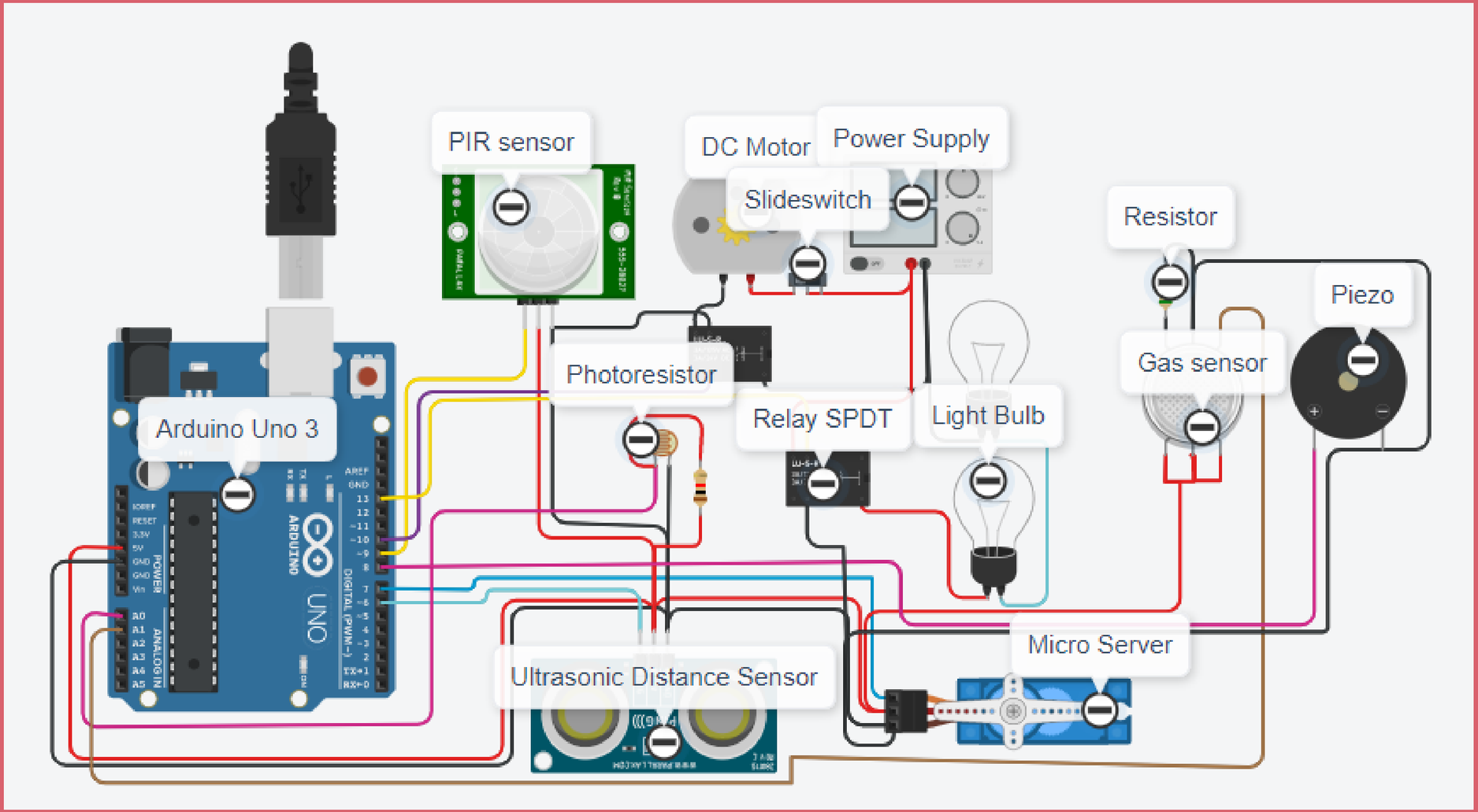
Some other components that were used :

PIR Motion Sensor i.e. HC-SR501, ESP32-CAM, Blynk app, 3 LED, BC547 NPN Transistor, 220 ohm, 1 k, and 10 k Resistor, FTDI 232 USB to Serial Interface board, a NodeMCU, an HC-SR04 ultrasonic sensor, a USB cable, one NodeMCU, a single MQ-2 wire jumpers and Breadboard.

Flow diagram Sensor-Based Smart Home Automation.



The circuit along with the components



Tinkercad simulation video with explanation of the function:

Coding (Text)

```
#include <Servo.h>

int output1Value = 0;
int sen1Value = 0;
int sen2Value = 0;
int const gas_sensor = A1;
int const LDR = A0;
int limit = 400;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);  // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
// Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  // Reads the echo pin, and returns the sound wave travel time in microseconds
  return pulseIn(echoPin, HIGH);
```

```
  // Reads the echo pin, and returns the sound wave travel time in microseconds
  return pulseIn(echoPin, HIGH);
}

Servo servo_7;

void setup()
{
   Serial.begin(9600);           //initialize serial communication
  pinMode(A0, INPUT);            //LDR
  pinMode(A1,INPUT);             //gas sensor
  pinMode(13, OUTPUT);           //connected to relay
  servo_7.attach(7, 500, 2500); //servo motor

  pinMode(8,OUTPUT);             //signal to piezo buzzer
  pinMode(9, INPUT);             //signal to PIR
  pinMode(10, OUTPUT);           //signal to npn as switch
  pinMode(4, OUTPUT);            //Red LED
  pinMode(3, OUTPUT);            //Green LED

}
```

```
void loop()
{

     //------light intensity control------//
//-------------------------------------------------------------
    int val1 = analogRead(LDR);
  if (val1 > 500)
        {
        digitalWrite(13, LOW);
    Serial.print("Bulb ON = ");
    Serial.print(val1);
        }
  else
        {
        digitalWrite(13, HIGH);
     Serial.print("Bulb OFF = ");
    Serial.print(val1);
        }
```

```
//-------------------------------------------------------------
        //------ light & fan control --------//
//-------------------------------------------------------------
  sen2Value = digitalRead(9);
  if (sen2Value == 0)
        {
        digitalWrite(10, LOW); //npn as switch OFF
        digitalWrite(4, HIGH); // Red LED ON,indicating no motion
        digitalWrite(3, LOW); //Green LED OFF, since no Motion detected
    Serial.print("     || NO Motion Detected    " );
        }

  if (sen2Value == 1)
        {
        digitalWrite(10, HIGH);//npn as switch ON
    delay(5000);
        digitalWrite(4, LOW); // RED LED OFF
        digitalWrite(3, HIGH);//GREEN LED ON , indicating motion detected
     Serial.print("        || Motion Detected!      " );
        }
```

```
//-------------------------------------------------------------
       // ------- Gas Sensor --------//
//-------------------------------------------------------------
int val = analogRead(gas_sensor);      //read sensor value
  Serial.print("|| Gas Sensor Value = ");
  Serial.print(val);                                  //Printing in serial monitor
//val = map(val, 300, 750, 0, 100);
  if (val > limit)
        {
        tone(8, 650);
        }
        delay(300);
        noTone(8);

 //-------------------------------------------------------------
      //-------  servo motor  ---------//
 //-------------------------------------------------------------
  sen1Value = 0.01723 * readUltrasonicDistance(6, 6);

  if (sen1Value < 100)
        {
```

```
//-------------------------------------------------------------
     //-------  servo motor  ---------//
//-------------------------------------------------------------
 sen1Value = 0.01723 * readUltrasonicDistance(6, 6);

 if (sen1Value < 100)
       {
       servo_7.write(90);
   Serial.print("         || Door Open!   ; Distance = ");
   Serial.print(sen1Value);
  Serial.print("\n");

       }
 else
       {
       servo_7.write(0);
   Serial.print("         || Door Closed! ; Distance =  ");
   Serial.print(sen1Value);
   Serial.print("\n");
 }
 delay(10); // Delay a little bit to improve simulation performance
```

Coding (Block)

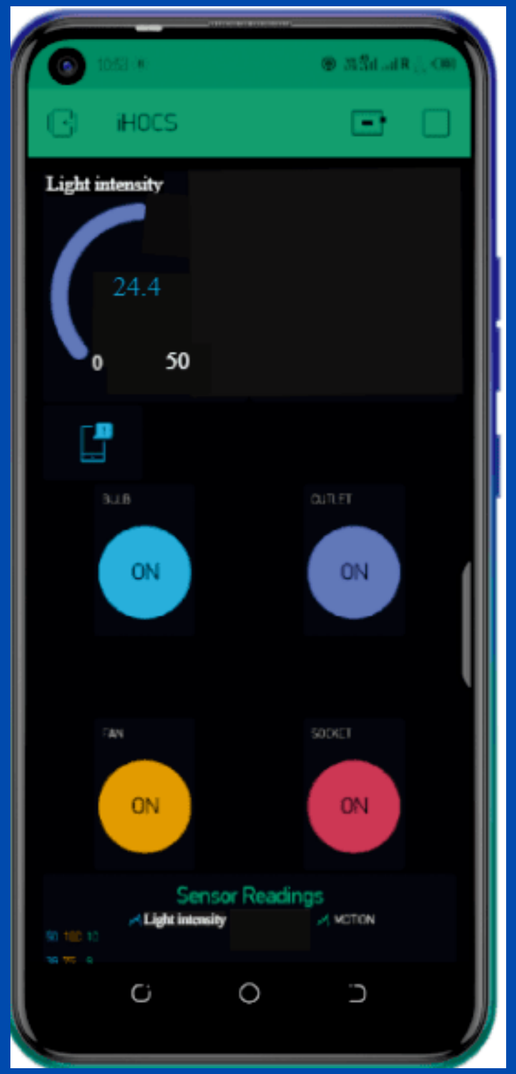
```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Blynk

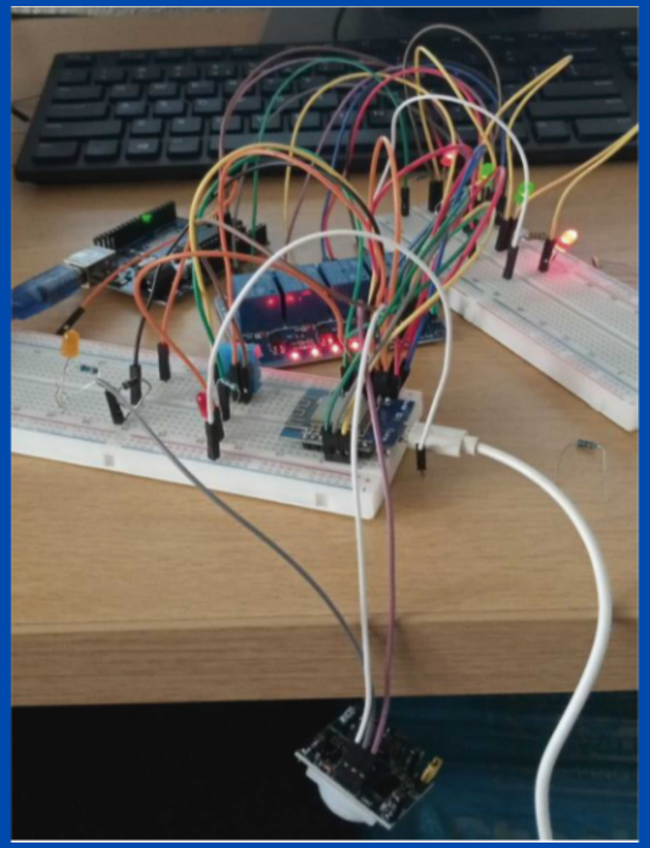
- The lights of the LEDs turn ON or OFF as specified by the command from the smartphone.
- Figure 6(a) shows the command to switch ON all four appliances in the home, while Figure 6(b) shows the corresponding lights from the LEDs.
- In Figures 7(a) and 7(b), three appliances were switched ON, and one appliance was turned OFF.
- Figures 8(a) and 8 demonstrate how the values of the light levels detected around the home are shown on the mobile application screen (b).
- Every five seconds, a reading from the LDR sensor will be captured and shown by the mobile application.
- The LDR sensor is able to collect the home's light levels measurements and present them graphically on the iHOCS application's interface thanks to the cloud computing service and real-time cloud database rendered through the Blynk platform.

Figure 6

(a) All appliances ON.



(b) Corresponding light indicators.



- Figure 8(a) displays the screen's minimized graphical sensor readings, whereas Figure 8(b) displays the readings' full-screen graphical display.
- The iHOCS system, as previously mentioned, also guarantees the safety and security of the house.
- The system's home security module is prototyped using the ESP32-CAM board and a PIR motion sensor.
- The ESP32-CAM is used to take pictures of the item, person, or animal caught, while the motion sensor is used to detect movement inside the home.
- Figures 8(a) and 8(b) show a real-time graphical representation of the motion sensor readings with the light level values.
- The motion sensor detects movements and delivers a notice to the user via the mobile application (Figure 9(a)).
- Figure 9(a) shows the message that the user received when motion was detected inside the home's premises prior to the image being taken.

- Our system is improved by being built to allow the user to quickly capture the image of the person so that the user may respond pro-actively, even if our system automatically records all photographs of the individual responsible for the movement.

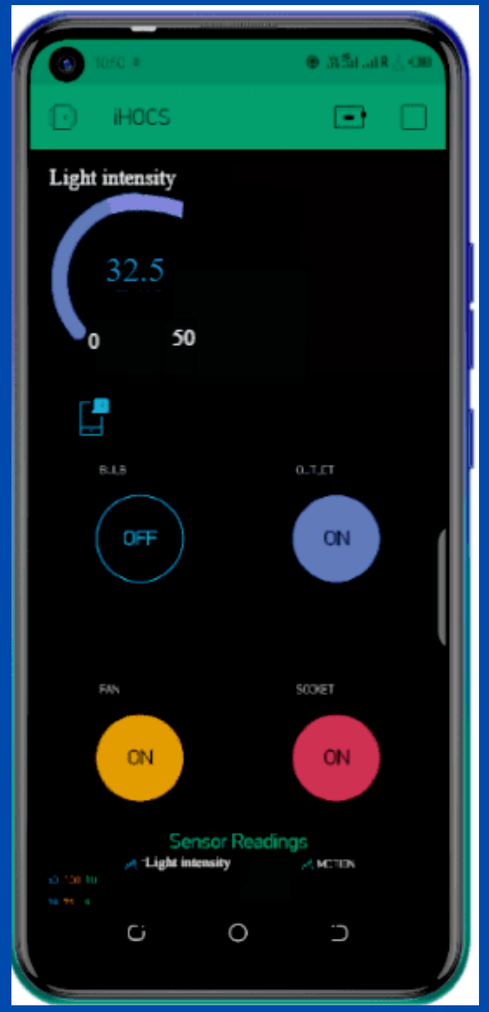
- If the user is inside the building, they can choose to ignore the notice, or if they are outside, they can see what is happening at home right away.

- Figure 9(b) showed the image of the individual inside the house that was taken when the motion sensor picked up movement. The ESP2-CAM is equipped with a light that enables it to take pictures in low light.

- The suggested system carries out several home control and energy-saving tasks. Figure 10 depicts the environment's intelligent reaction to the light switch-on. The technology helps with energy conservation by reminding the user to either switch on the light when it is dark or off when it is bright. This improves energy management and conservation in the house.

Figure 7

(a) Three appliances ON.



(b) Corresponding three LED indicators ON

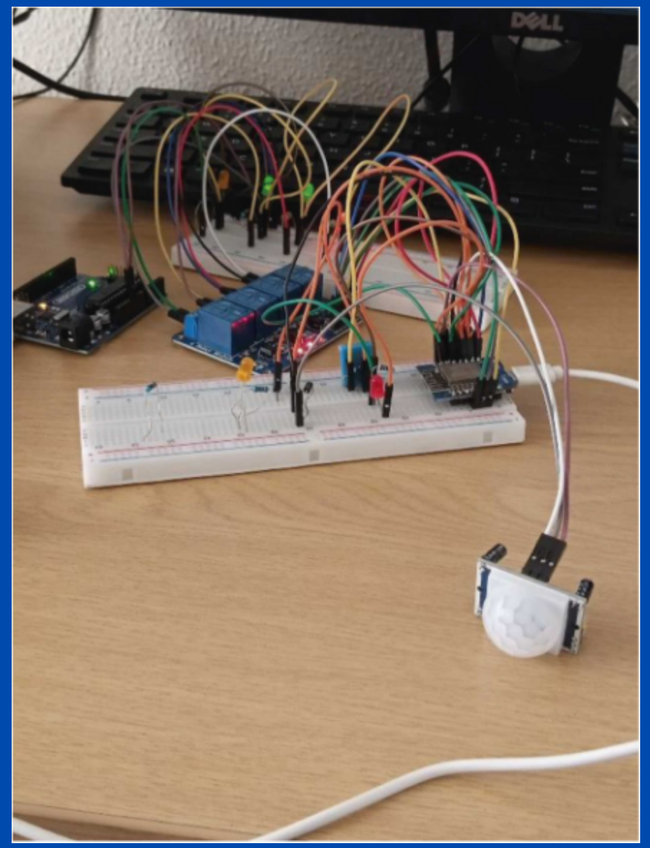
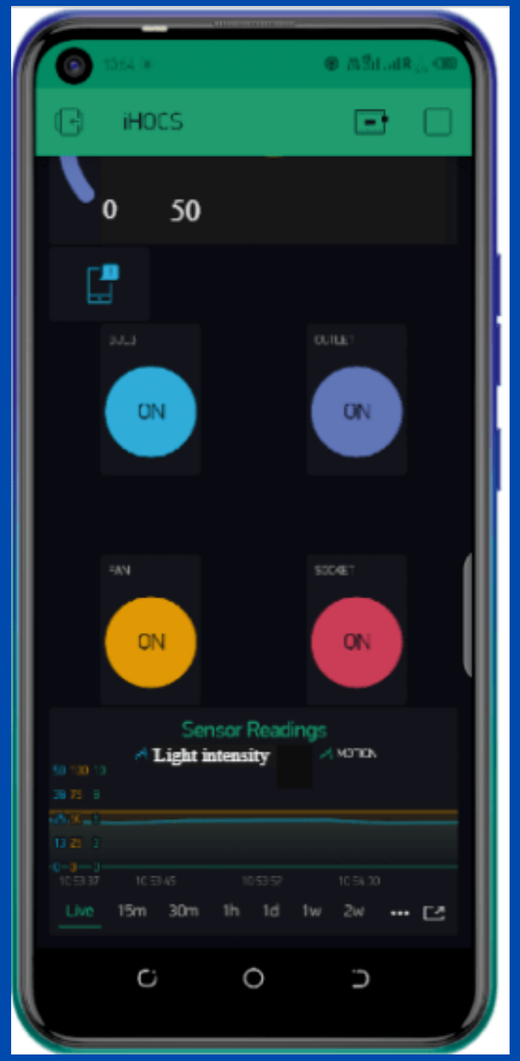


Figure 8

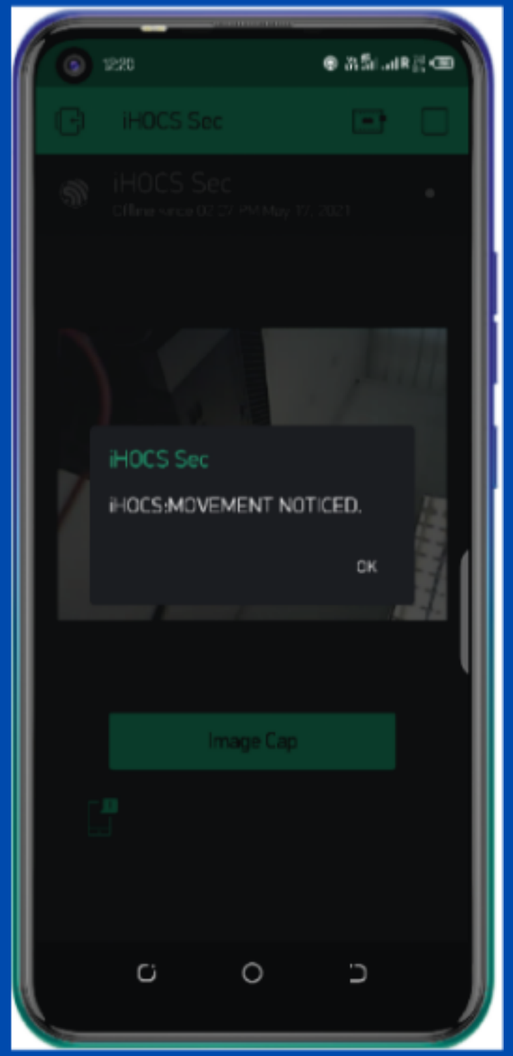
(a) In-app graphical readings.



(b) Full screen of graphical readings.

Figure 9

(a) Movement notification.



(b) Captured image.

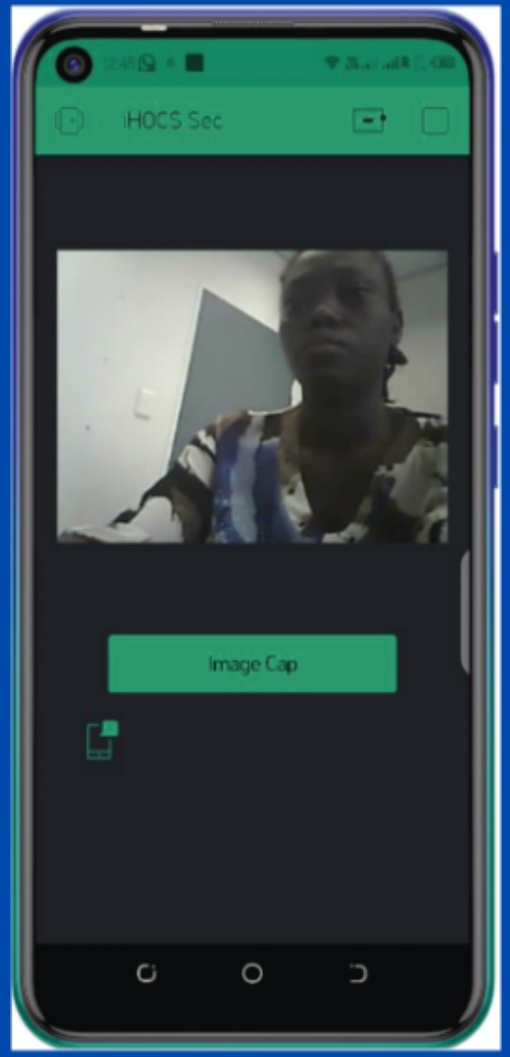
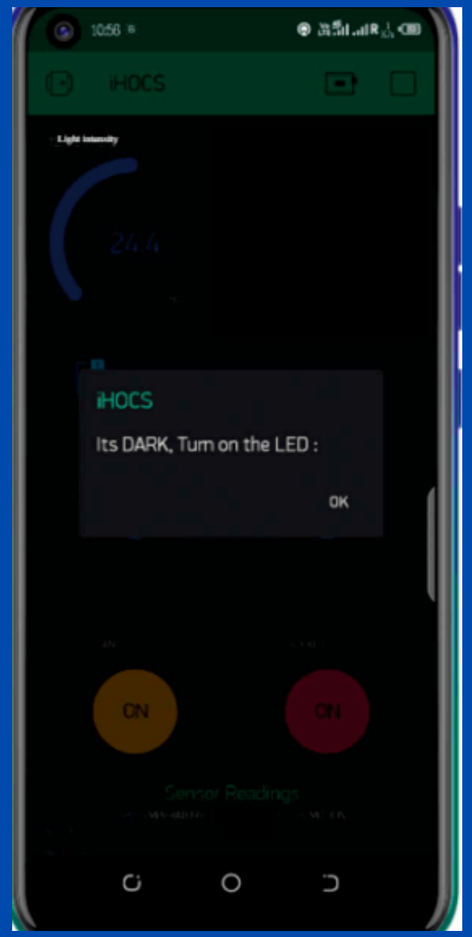


Figure 10

Intelligent home response.



After successful registration and configuration of the iHOCS Android mobile application, the user can perform the following basic functions:

(1) Control of electrical home appliances
(2) Monitor home environmental conditions (light levels)
(3) Automatic switch lights ON and OFF, based on the home condition
(4) Detection of movement in the home
(5) Receiving notifications about the home
(6) View graphical data of home sensors.

(The codes for all the three simulation for the project has been explained in detail in case of using blynk and tinkercad in the final report)

Sources mainly utilised for the idea:

1. https://easyelectronicsproject.com/esp32-projects/esp32cam-blynk-security-camera/
2. https://mindstormengg.com/nodemcu-esp8266-lab-17-mq-2-gas-sensor-with-blynk-app/
3. https://www.hindawi.com/journals/scn/2021/9928254/#abstract
4. https://www.instructables.com/Ultimate-Automatic-Lighting-System-Using-Arduino-L/
5. https://arvindunimap.wordpress.com/how-to-use-esp32-blynk-with-led-ldr-lm35/
6. https://doi.org/10.1155/2021/9928254

Virtual platform difficulties and future possibilities.

- At the moment, the tinker platform simulates electrical, electronics, and Boolean components, as well as Arduino and ATTINY as programable modules.
- It makes it simpler to grasp the properties of components, power supplies, and display devices.
- Debugging and short-circuit scenarios benefit from the added simulation tools.
- TinkerCAD might expand IoT simulation capabilities in addition to the current features in order to release new automation applications.
- During the simulation phase, we encountered several challenges with merging two Arduinos and using various motor control controller boards.

- In the future, we intend to widen our horizons by incorporating machine learning into a mobile application to identify photographs obtained by the camera and tell the user of the specific identification of the photographed object.
- We also want to make the mobile application available on the iOS platform.
- The technique given in this paper may also be used to security systems in big communities such as smart cities, office areas, hotels, malls, and university settings to improve the security of the unique environment.
- It is also claimed that machine learning makes prediction easier.
- Machine learning may also be used to anticipate weather and house conditions in the environmental module of smart home automation.

References:

1. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-swit ch-using-nodemcu-and-the-blynk-app/setup-blynk-on-your-smartphone
2. https://www.instructables.com/Servo-Motor-Controlled-With-BLYNK-Over-WiF i/
3. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-swit ch-using-nodemcu-and-the-blynk-app/the-code
4. https://easyelectronicsproject.com/esp32-projects/esp32cam-blynk-security-came ra/
5. https://create.arduino.cc/projecthub/nolan-mathews/connect-to-blynk-using-esp8 266-as-arduino-uno-wifi-shield-m1-46a453
6. https://tatiuc.edu.my/ijset/index.php/ijset/article/view/93
7. https://mindstormengg.com/nodemcu-esp8266-lab-17-mq-2-gas-sensor-with-blyn k-app/
8. https://create.arduino.cc/projecthub/techno_z/program-your-arduino-from-your raspberry-pi-3407d4
9. https://www.hindawi.com/journals/scn/2021/9928254/#abstract
10. https://roboindia.com/tutorials/pir-motion-sensor-nodemcu-blynk/
11.https://ijariie.com/AdminUploadPdf/An_efficient_home_automation_system_usi ng_IOT__A_case_study_to_benefit_people_with_motion_disability__ijariie1583 6.pdf
12. https://blynk.hackster.io/KaustubhAgarwal/smart-parking-bdfa99
13. https://mytectutor.com/smart-car-parking-system-using-arduino-esp8266-nodem cu-and-blynk-server/
14. https://microcontrollerslab.com/control-esp8266-nodemcu-outputs-blynk-app-ar duino-ide/
15. https://iopscience.iop.org/article/10.1088/1742-6596/2117/1/012021/pdf
16. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-swit ch-using-nodemcu-and-the-blynk-app/setup-blynk-on-your-smartphone
17. https://www.instructables.com/Servo-Motor-Controlled-With-BLYNK-Over-WiF i/

18. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-swit ch-using-nodemcu-and-the-blynk-app/the-code
19. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-swit ch-using- nodemcu-and-the-blynk-app/the-code
20. https://www.researchgate.net/publication/357748641_The_HOME_AUTOMATI ON_USING_IOT?enrichId=rgreq-e7589b112ee65e0cd3eab7eb213fa4dd-XXX&enrichSource=Y292ZXJQYWdlOzM1Nzc0ODY0MTtBUzoxMTExMDg0MTUyOTYzMDcyQDE2NDE5MTQ1MTM1MDA%3D&el=1_x_2&_esc=publicationCo verPdf
21. https://ijariie.com/AdminUploadPdf/An_efficient_home_automation_system_using_IOT__A_case_study_to_benefit_people_with_motion_disability__ijariie1583 6.pdf
22. http://www.aui.ma/sse-capstone-repository/pdf/spring-2017/Brainyhab%20Affordable%20Wireless%20Smart%20Home%20System.pdf
23. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3884923
24. https://www.pcbway.com/project/sponsor/IoT_Home_Automation_using_Blynk_NodeMCU_ESP8266_edcfd442.html
25. https://medium.com/@1730103/automatic-door-lock-system-in-arduino-uno-545 0a97b8d1a
26. https://iotdesignpro.com/projects/control-arduino-remotely-using-blynk-app
27. https://www.pcbway.com/project/sponsor/IoT_Home_Automation_using_Blynk_NodeMCU_ESP8266_edcfd442.html

Thank you!