ITE233 Final Project Report

Project Title: IoT Home Automation System

Course: ITE233 - Introduction to Internet of Things

Section 1

Term 3-2021

Course Teacher: Atikom Srivallop

Group 8 (Members):

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

Table of contents

Abstract……3
Feasibility Study……...3
Analysis, design, and structured chart……..…7
Implementation and outcome……13
Summary……75
References……..77
.

Abstract
In this study, a practical framework for putting home automation practices into practice is proposed. Depending on the user's demands and interests, mobile devices like sensors and actuators are used as a user interface in a range of home automation activities. The components of the devices can communicate with one another utilizing ZigBee WiFi protocols over a web gateway. The power and data transmission capabilities are limited to short ranges. While implementing automation techniques, students in academic laboratories frequently experience connection issues, which can lead to component loss through misconnection and complicate circuit debugging. An effective technique to accomplish many automation applications is with AUTODESK TINKERCAD. An effective technique to accomplish many automation applications is with AUTODESK TINKERCAD. The use of a garage car parking lot application as an example of how to use home automation applications for keypad-based door unlocking with IR-based appliance control, and the conclusion of the paper explains our design for the iHOCS environment includes a PIR/ultrasonic motion/gas/LDR sensor, dc motor, light bulbs, and ESP32 camera that generate data at intervals set by the user who uses the interior appliances of the home through motion, ambient light, smoke detector, PIR, Ultrasonic, and LDR sensor. The component connection and programming logic have been shown, along with results, using the recorded step-by-step video links for each application.

Feasibility Study

For the elderly and the disabled - Emphasizes enabling older persons and people with disabilities to stay safe and at ease at home. Seniors and people with disabilities who choose to remain in their own homes rather than visit a nursing facility are increasingly able to do so thanks to home automation. Although it is geared toward elderly people and individuals with disabilities, this sector employs a lot of the same technology and apparatus as home automation for security, entertainment, and energy conservation.

What is the problem? and other issues like these are addressed in a feasibility study, which is a high-level summary of the whole procedure. Exists a workable solution to the stated issue? Is the issue really worth addressing? Once the issue is properly identified, a feasibility study is carried out. A feasibility study is required to evaluate the technical, operational, and economic aspects of the proposed system in order to establish its viability. The management will have a comprehensive understanding of the planned system thanks to a thorough feasibility assessment.

To make sure that the project is flexible and does not encounter any significant obstacles, the following feasibilities are taken into account. The following are included in a feasibility study:

The three pillars of feasibility are: technical, economic, and operational.

In this stage, we examine the viability of every system that has been suggested and select the most viable one. The following three primary considerations form the basis of the feasibility study.

Technical Possibilities/Feasibility

In this phase, we examine the technological viability of the suggested systems. i.e., whether or not all of the technologies needed to construct the system are easily accessible. The project's technical feasibility assesses if the organization has the requisite technology and expertise to complete it, as well as how these should be acquired. Because of the following reasons, the system may be practical:

• The system can be developed using all available technology.
• This system can be expanded further since it is too versatile.
• The accuracy, usability, dependability, and data security of this system can all be guaranteed.
• This system can respond to queries right away.

Because all the technology required for our project is readily available, it is theoretically possible.

Financial viability
Due to the fact that it doesn't need any more funding and can be finished in six months, this project is totally doable economically. This process involves determining which plan is most cost-effective. We contrast the new system's financial gains with the investment. Only when the financial gains outweigh the investments and expenses is the new system economically viable. Economic viability establishes whether or not the project aim can be achieved within the resources allotted to it. It must decide if processing the full project is worthwhile or whether the advantages of the new system outweigh the expenditures. Benefits must match or surpass expenditures in terms of money. In this case, we need to take into account:

• The price of performing a thorough system investigation.
• The price of hardware and software for the type of application under consideration.
• The programming tools.
• The price of upkeep, etc.

Due to the relatively low development costs compared to the application's financial advantages, our proposal is commercially viable/feasible.

Functionality in Operation

The best operationally practicable solution is determined by which of the suggested systems consumes the fewest operational resources, such as people, time, etc. at this stage. Additionally, the solution must be operationally doable. Operational Feasibility assesses whether the proposed solution met user objectives and could be integrated into the functioning of the existing system.

• Since the techniques of processing and display may satisfy all user needs, the clients have fully embraced them.

• The system's planning and development included input from the clients.
• Under all conditions, the suggested system won't lead to any issues.

Due to the fulfillment of the time and people criteria, our project is operationally possible. We are a four-person team, and we spent 1.5 working months on this project.

Analysis of viability/feasibility

This project can be completed using inexpensive electrical and software technology, making it technically, financially, and operationally feasible.

Financial viability
This idea is built cheaply by using an Android phone and a few inexpensive electronic parts including the Raspberry Pi computer, camera modules, sensors of PIR/ultrasound/gas/LDR, dc motor, and light bulbs, and relay switches.

Technical viability
This idea is built on embedded systems and wireless technologies, both of which are relatively modern in terms of technology. Technology therefore tremendously benefits it.
Operational Suitability: This program will have a very straightforward user interface that is practically accessible to those who have never used an Android phone. Physically disabled persons may find it advantageous to be able to operate household equipment with the touch of a button. It is therefore feasible.

Low-cost operations: A lot of factors, including the price of the hardware and software, affect how much an IoT deployment will cost. The software for an automated smart home system is not very expensive. It would take about 80 man-hours to create a simple android app to flip a switch on and off. The program would need to be hosted by a server and IoT hardware, such as a board, controller, switches, and relays. The main distinction between Raspberry Pi and Arduino boards is that Raspberry Pi boards are microprocessors, whilst Arduino boards are microcontrollers. We simply need a board that supported bit pins for smart home automation, which the Arduino board does. Programmatically, the Raspberry Pi pins are managed.Our design for sensor-based home automation uses an ESP8266 Wi-Fi board as the gateway for wider network coverage and extended communication. The system connects to the Internet via this gateway and communicates with all appliances and devices in the home.

The communication protocols used are Wi-Fi, which is one of the primary operating standards for home automation technology, TCP/IP, and HTTPS/IP.To strengthen the level of security, the SVM is used to classify some features in the home before sending an alert to the user about the detected motion. Our design for the iHOCS environment which is the sensor-based home automation includes PIR/ultrasonic motion/gas/LDR sensor, dc motor, light bulbs and ESP32 camera that generate data at intervals set by the user. The generated data need to be stored for monitoring of the home, analysis, and for future prediction. The cloud infrastructure is needed for real-time storage of the data, processing, communication, and monitoring in a smart home. Therefore, in our work, a cloud computing module is used to store data generated, display it graphically, and process it which we will explain in

complete detail in the implementation, and outcome, using structured charts, analysis, and design.

Applications for home automation are many and user-specific. A keypad-based door unlocking system is modeled in this paper application utilized in smart homes, which will provide security for items at homes, schools, and workplaces. A 4X4 keypad input device for inputting the Personal Identification Number (PIN) and a display unit in the form of a Liquid Crystal Display (LCD) for informational visual display are both included in the security system. Additionally, it has a servo motor that acts as a switch to lock and unlock the door as well as an Arduino that has been programmed to interpret the input data and conduct the necessary action. The security system, which is situated at every door, collects the PIN as a person enters it and searches its database for a match. The door opens if the saved PIN and the captured PIN are compatible; if not, the incorrect password is shown on the LCD and the door remains closed.

Managing parking spots is one of the difficulties that cities and municipalities must deal with in the second application. Car owners look for parking spaces for roughly 100 hours a year on average. Automation offers creative Parking Solutions for necessities in traffic management because of this. reduced traffic loads and environmental effects increased the attractiveness of communities with a higher standard of living, and improved occupancy and profitability of multistory parking lots and car parks. When the parking lot is full, this system will show the available automobiles that need to be parked. No more space will be seen. The available automobiles that need to be parked will be visible once the vehicle passes through the departure point once again. We make use of the Led associated with a car for blynk app to know if a car is parked in a certain slot. The Arduino Uno, Ethernet Shield, Blynk App, and IR sensors make up the system. In order to find a vehicle, sensors are employed. When a car is seen in a certain slot, the associated led on the app turns bright.

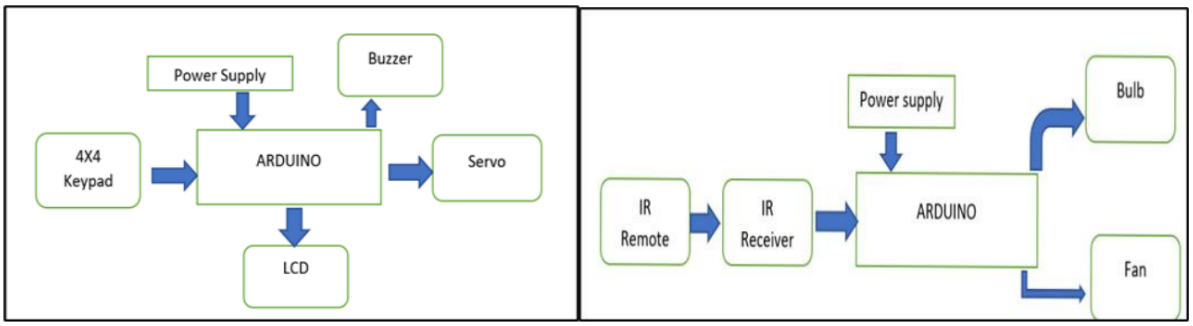
The final application makes use of the smoke detectors, LDR, PIR, and ultrasonic sensors on the microcontroller board. A smoke detector will detect gas if it detects any leaks. Gas detection sensors offer information that triggers the alarm circuit and sounds the alert. The entry door will automatically open if an ultrasonic sensor picks up something. There is a person in front of the door. The fan is turned on and human activity is detected by the PIR sensor. Additionally, there is a manual fan. LDR is used to automatically turn on the light when someone is home at night; however, when it is daylight, it is quickly turned off. These features for the previous application appeared to perform well and as intended.

The final application is therefore the setup of iHOCS involves electrical sockets and other basic home appliances to be controlled, such as light bulbs, television, cooling, and heating apparatus. As stated earlier, it also measures environmental conditions around the house. Lastly, it detects movement and captures an image of the person detected. The sockets and home appliances are connected to the relay board module, and the relay module is connected to the ESP8266 for the Android application to receive and send commands. An LDR sensor is used to measure the light levels, and the values are displayed on the screen of the user's smartphone.LDRs (light-dependent resistors) are used to detect light levels, for example, in automatic security lights.

Their resistance decreases as the light intensity increases: in the dark and at low light levels, the resistance of an LDR is high and little current can flow through it The home is enhanced with greater security and safety of the occupants. A PIR/ultrasound motion/gas/LDR sensor is used to detect movement in the house, and once detected, a notification is generated on the Android mobile application to keep track of the graphs. The SVM does a background check before sending an alert to the user and ringing the alarm. The cloud storage embedded with the mobile application stores all the data generated from all the sensors. The sensors are interfaced with the Wi-Fi module for wireless transmission to the cloud platform via the mobile application.

Analysis, design, and structured chart

1. IR remote controls are used to operate domestic appliances as part of a home security system, while keypads are used to access doors.



Block design for a home security system with a keypad and an IR remote control is shown in Figure 1 above.

The components used in this application are listed in a table along with their respective quantities. Figure 2 of the list was produced by the TinkerCAD circuit platform.

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

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.

Figure 2: Components utilized in this application are listed.

The idea, logic, and execution of this simulation are simple to understand and carry out. Open the door first by entering the right PIN on the keypad and stating whether you did so properly or not. A piezo buzzer is installed to indicate that the PIN was entered wrongly when the wrong password is typed. We use an IR remote after unlocking the door to control different home appliances by pressing the remote's buttons.

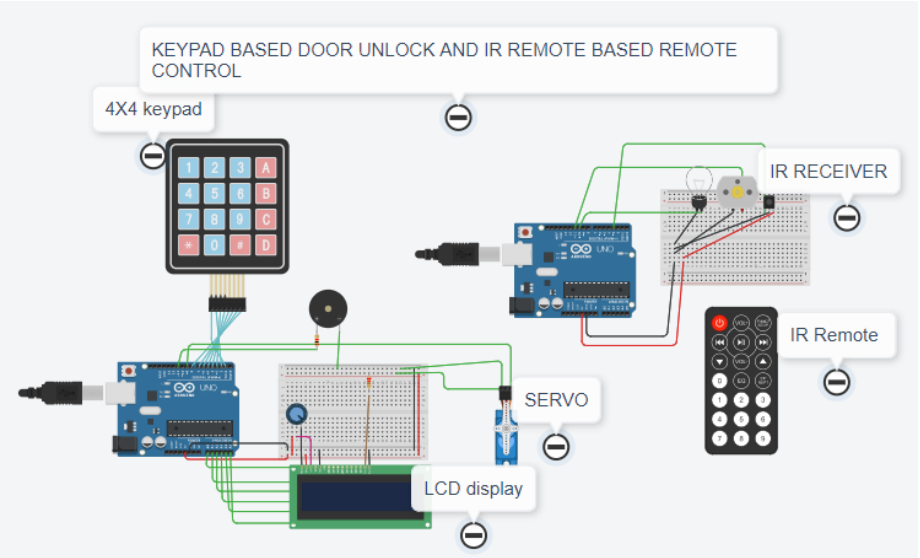
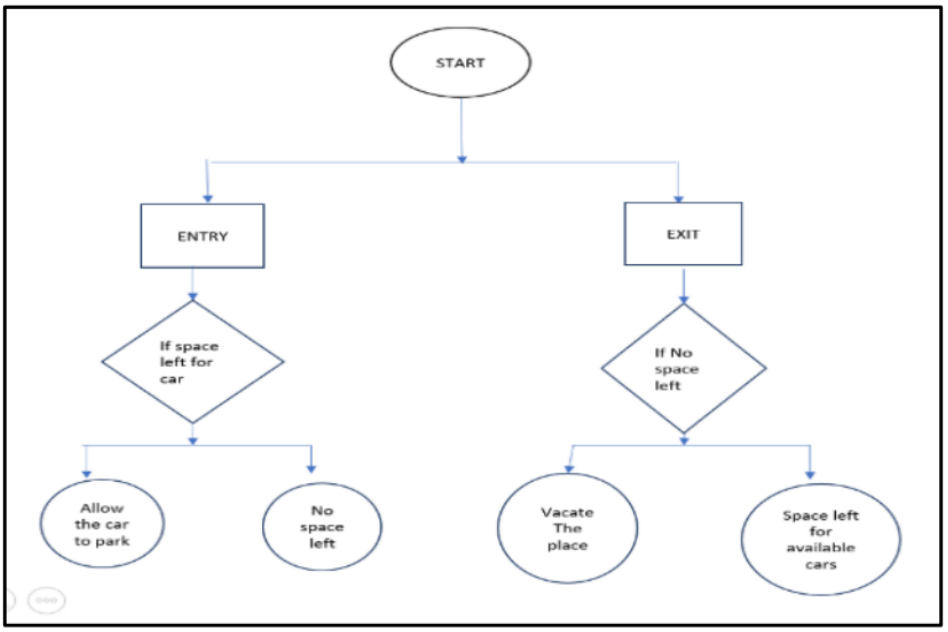


Figure 3: The circuit along with the components

2. Automated Car parking system



The flowchart for a garage parking lot is shown in Figure 4

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.

Figure 5: A list of the parts utilized in this application.

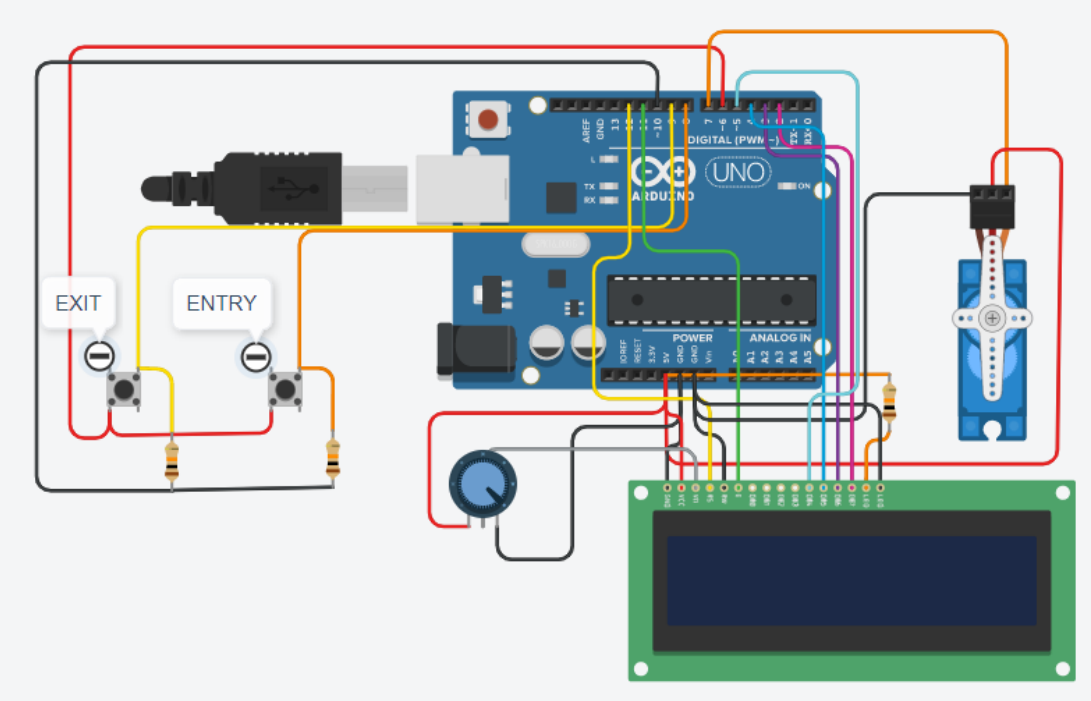


Figure 6: The Tinkercad simulation circuit along with the components

Two push buttons serve as the entry and exit points for this circuit. The servo motor will rotate if we press the push button, allowing the automobile to enter the parking lot. There won't be any spaces available in the parking lot is full. At that point, pressing the push button would prevent the servo motor from rotating, preventing the automobile from entering the parking lot. We need to press the exit push button in order to leave the parking lot. It will indicate that there is room for the available automobile if all the cars have departed the parking lot.

3. Sensor-Based Smart Home Automation

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 app
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 ohms, 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.

Figure 7: List of components used in this application

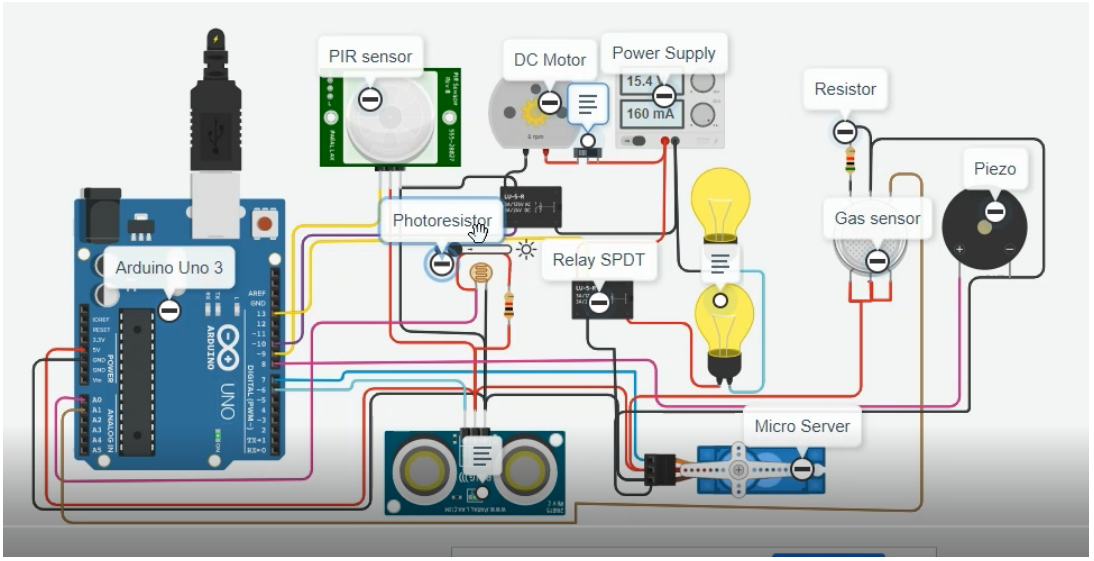


Figure8: The circuit along with the components in tinkercad

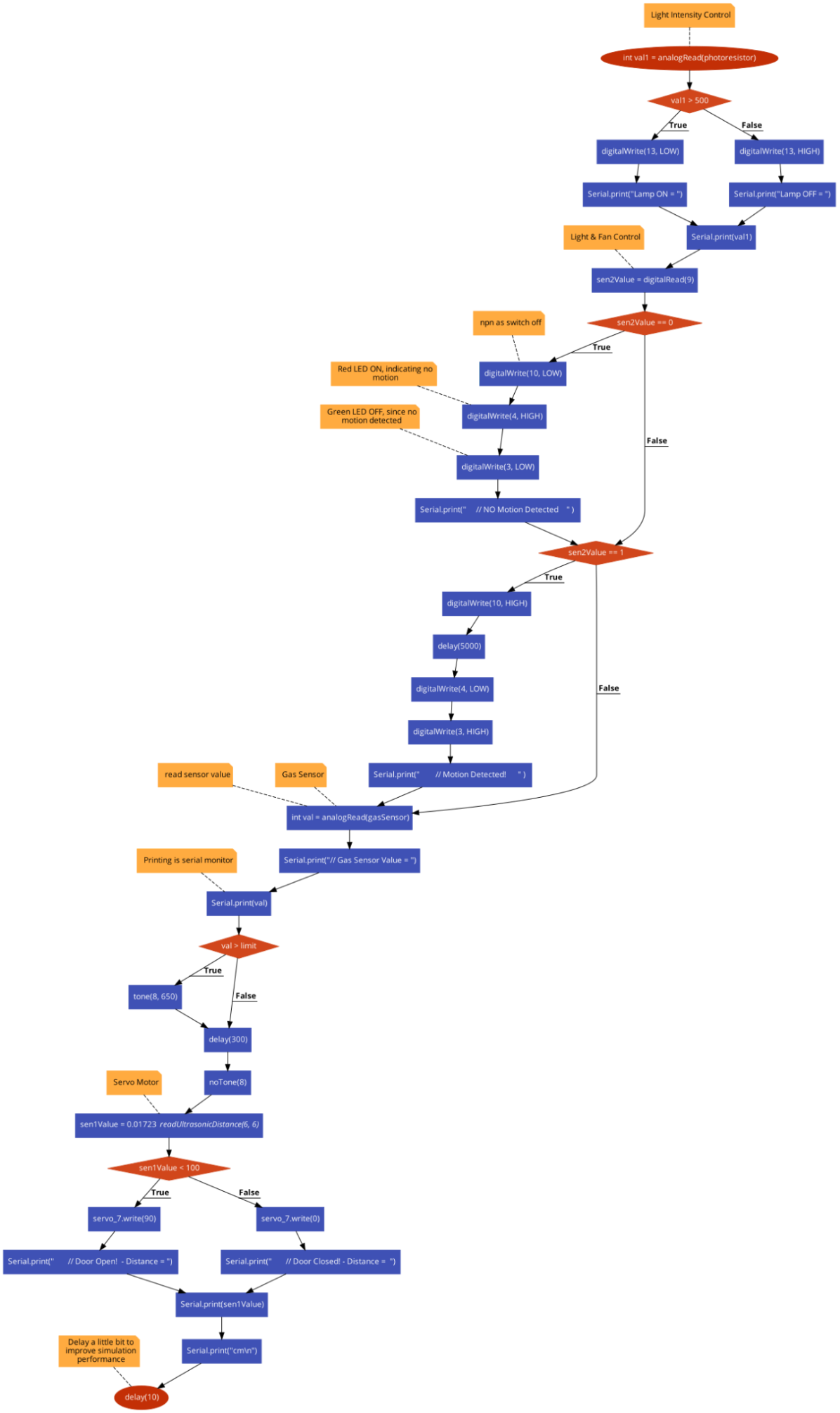

Light Intensity Control
int val1 = analogRead(photoresistor)
val1 > 500
True
False
digitalWrite(13, LOW)
digitalWrite(13, HIGH)
Serial.print("Lamp ON = ")
Serial.print("Lamp OFF = ")
Light & Fan Control
Serial.print(val1)
sen2Value = digitalRead(9)
sen2Value == 0
npn as switch off
True
Red LED ON, indicating no motion
digitalWrite(10, LOW)
digitalWrite(4, HIGH)
Green LED OFF, since no motion detected
digitalWrite(3, LOW)
False
Serial.print(" // NO Motion Detected ")
sen2Value == 1
True
digitalWrite(10, HIGH)
delay(5000)
digitalWrite(4, LOW)
digitalWrite(3, HIGH)
False
read sensor value
Gas Sensor
Serial.print(" // Motion Detected! ")
int val = analogRead(gasSensor)
Printing is serial monitor
Serial.print("// Gas Sensor Value = ")
Serial.print(val)
val > limit
True
False
tone(8, 650)
delay(300)
Servo Motor
noTone(8)
sen1Value = 0.01723 readUltrasonicDistance(6, 6)
sen1Value < 100
True
False
servo_7.write(90)
servo_7.write(0)
Serial.print(" // Door Open! - Distance = ")
Serial.print(" // Door Closed! - Distance = ")
Serial.print(sen1Value)
Delay a little bit to improve simulation performance
Serial.print("cm\n")
delay(10)

Figure 9: Flow diagram

Among the sensors on the microcontroller board are a smoke detector, a PIR sensor, an ultrasonic sensor, and an LDR sensor. If there is a gas leak, it is discovered using a smoke detector. If gas is detected in the sensor data being acquired, the Alarm circuit will activate. If someone approaches the front door, it will open on its own utilizing ultrasonic technology. The PIR, which also monitors humidity, turns on the fan. A manual fan is also included. By turning on the bulb upon nighttime entry and turning it off at departure, LDR automates light management in homes.

Implementation (coding) and outcome.

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

1. The actions taken in this door unlocking automation are represented by the outputs. Starting the simulation, entering the password, using the right password to open the door while the servo turns the LCDs, using the erroneous password to close the door, controlling the bulb and motor with an IR remote, and using the remote to switch off the light and motor. Figure 6's initial stage conducts the simulation after which an LCD welcome message is shown. When the simulation begins, the servo will move to its initial condition of closing the door, after which the LCD will show the input a password. Figurine 7 When the correct password is entered, the door will open as visible on the LCD and the servo will rotate as a warning; nevertheless, if the incorrect PIN is entered, the incorrect password will be shown on the LCD.

   Figure-8 After opening the door, you may enter and use the IR remote to access the home apps. The light will illuminate and the motor will revolve for button 2 when the IR remote's button 1 is depressed. The first step is to use the 4X4 keypad to input the password. Once you've done that, the LCD will show that the door is open and the servo will rotate to indicate that the door is opening. An erroneous password entry will cause the buzzer to sound when the button to close door # is pressed. Pressing the IR remote's buttons turns on and off lights, motors, and other household equipment. You can refer to the simulation step that was captured.

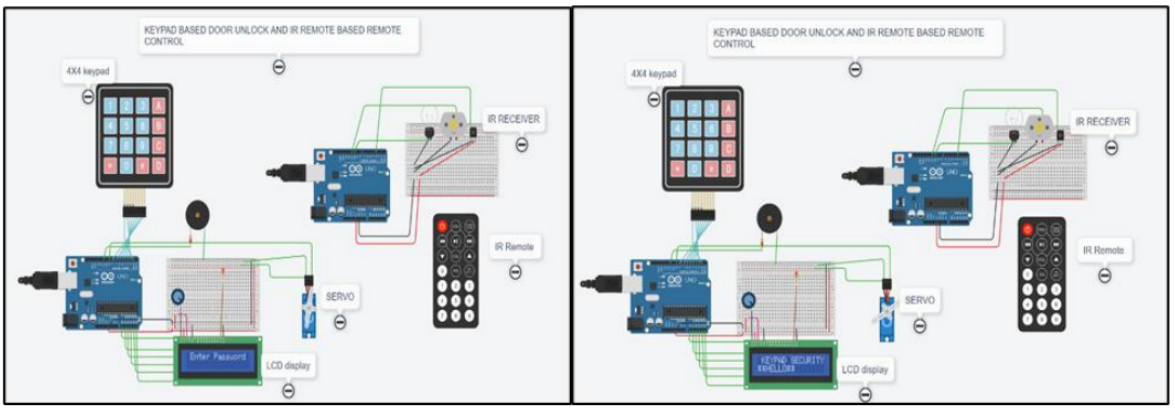


Figure-6 initial state of automation process

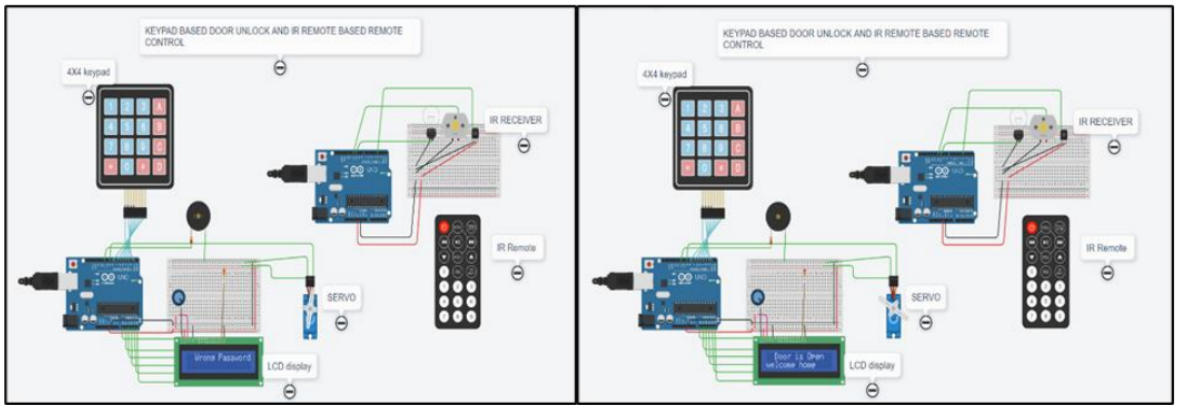


Figure-7After entering right password door opening displaying in LCD

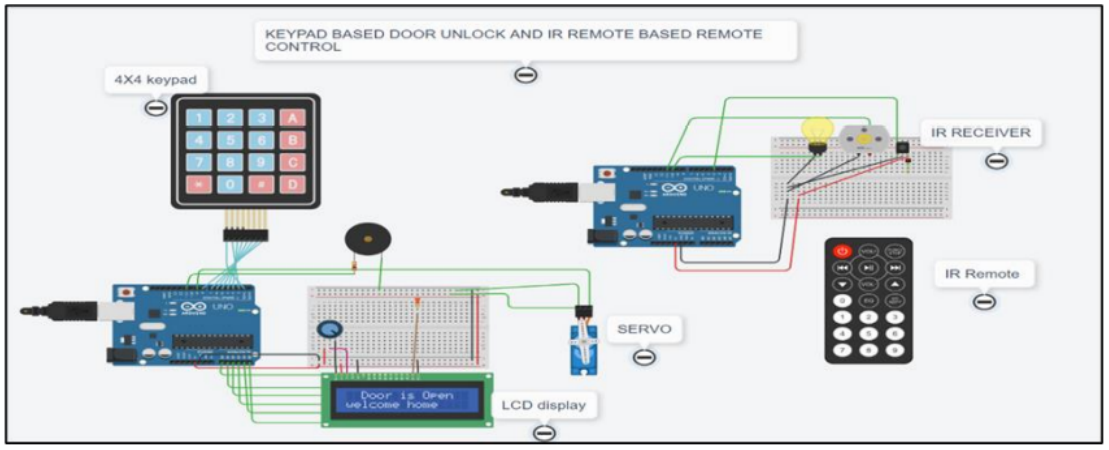


Figure-8 bulb and motor on when IR remote button pressed

The following screenshots show the coding used in tinkercad for the successful and complete simulation.

```
#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();

}
```


File Edit Format View Help

```
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();
 }
}
```

```
// 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)
}
```

The coding from a Tinkercad simulation for a keypad security system and an IR remote control for inside appliances is shown in Figure 31 below.

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. Figure 11 displays the block diagram.

Figure 11: System Block Diagram

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.

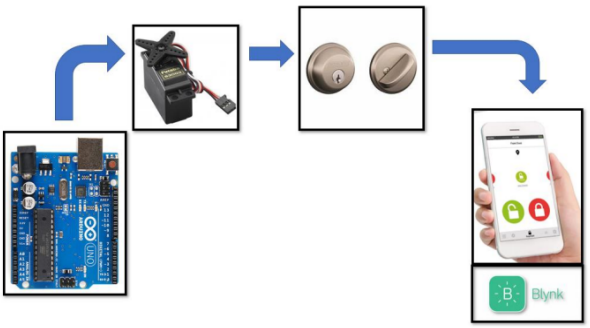


Figure 12: Project Sketch

Development of hardware.

Tinkercad website is used in this project to design and simulate the circuit. The complete design of the circuit using tinkercad is shown in Figure 31 below.

After the circuit simulation is complete, all of the components used in this project were installed and assembled. Figure 14 below shows the full connection of the circuit used in this project.

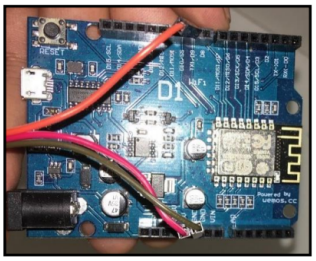


Figure 14: Circuit Connection

Automatic door closer and deadbolt were integrated together to complete the hardware. For this study, a deadbolt lock is required for users to install the IoT Door Lock system at their door. Another type of door lock is not suitable and possible to install. Figure 15 shows the installation of the deadbolt and servo motor.

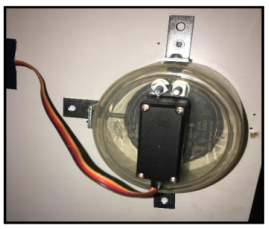


Figure 15: Installation of deadbolt

Other than that, an automatic door closer was installed at the door and also the other mechanical part. Figure 16 shows the installation of an automatic door closer to automatically close the door.

Figure 16: Installation of Automatic Door Closer

Software Development

This research is adopting a methods approach involving developing the control system using software such as Arduino IDE and Tinkercad. This software provides overall good programming customization and simulation of the system. Tinkercad is an open-source hardware initiative that makes electronics accessible as a creative material for anyone. The software tool, a community website, and services in the spirit of Processing and Arduino, foster a creative ecosystem that allows users to document their prototypes, share them with others, teach electronics in a classroom, and layout and manufacture professional PCBs.

Blynk Development

Blynk has been used as the app for the IoT monitoring system in this study. The Blynk application can be download through the app store for Apple users or the play store for android users. The setup on Arduino WeMos was very well documented and the library briefly described the features that are possible. It is easy to use with minimal software knowledge. Numerous features were added. Frequent updates to the Blynk app. Support for multiple devices with different modes of communication. The Blynk application can be download through AppStore for Apple users or the play store for android users. After downloading the application, the user must create an account to use the entertainment through the applications. After successfully creating an account, the home sides of the applications appeared. Then, select the new project to start creating a system through the Blynk applications. Figure 17 shows the home sides of the apps.

Figure 17: Blynk Home

After creating the new project, select the controllers from the Widget Box. Every widget contains the energy balance. However, the user has only been given a 2000 amount of energy balance. If the user reaches the limits of the energy balance, the user may have some additional balance by purchase the energy balance through the application. Any Energy sale is final without any returns, refills, or account balance topping up with more Energy The button widget was used for this project to control the servo motor. Figure 18 shows the Widget Box of Blynk applications.

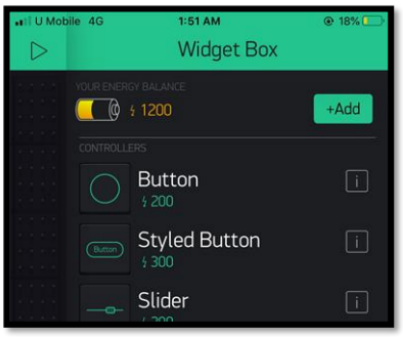


Figure 18: Blynk Widget Box

After that, the button that selected from the widget appeared in your project system. Figure 19 below shows the button from the Widget Box

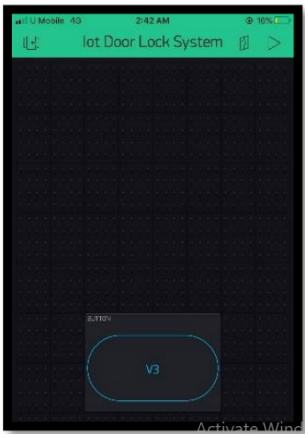


Figure 19: Widget Button

Every new project that has been created from the project contains an auth token. This auth token is provided for the user to put in their programming to connect the Blynk application with their controller. Auth Token is a unique identifier that is needed to connect your hardware to your smartphone. Every new project you create will have its own Auth Token. Click on the devices section and selected the required device as shown in Figure 20 below.

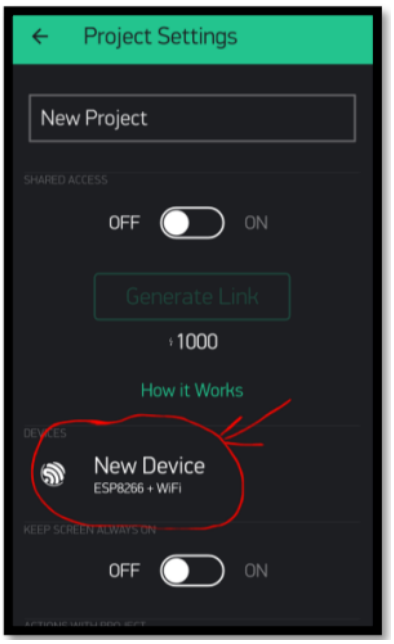


Figure 20: Device Section

After that, the user will get Auth Token automatically on their email after project creation. The user can also copy it manually. Figure 21 shows the Blynk auth token.

Figure 21: Blynk Auth Token

Programming to test run the servo motor can be found through the Blynk browser. Through the Blynk browser, they provided variable libraries for the user to run their project. This example shows how value can be pushed from Arduino to the Blynk App. Figure 12 below shows the Blynk browser.

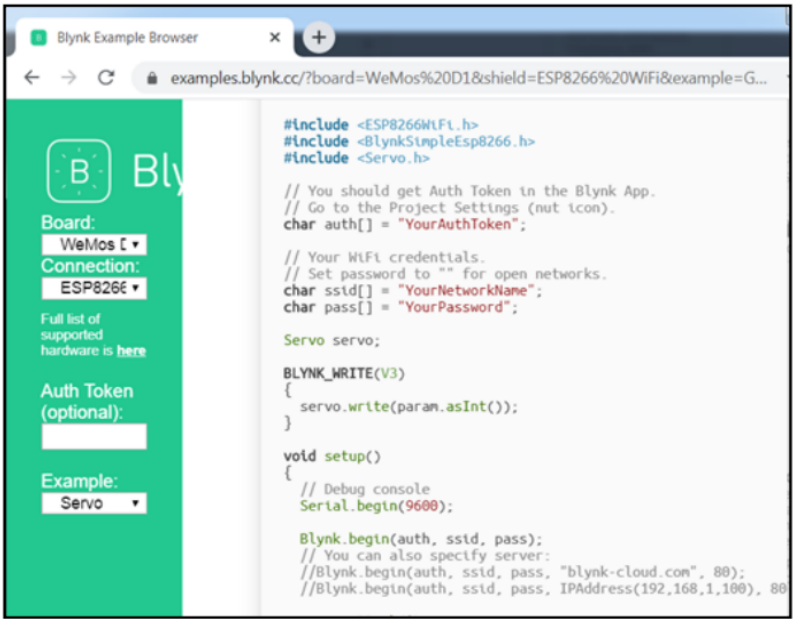


Figure 22: browser of blynk

RESULTS AND DISCUSSION

Figure 23 displays the whole hardware configuration. Inside the door is where the door lock is stored. The servo motor, which is here driven by a 9V battery, has been used to show the door lock. A smartphone's hotspot is used to connect the Arduino WeMos to the internet. By pressing the button, this system operates instantly, and the door will lock on its own.

Full hardware setup is shown in Figure 23.

The data are gathered by performing a durability test on the system count from 0 to 1000 for the data analysis in this project. There are 5 criteria that make up the durability condition. Table 1 displays the conditions for the durability test and the outcome for this durability.

Real-time durability conditions are shown in Table 1 below.

No.	Type of condition
5	Excellent condition
4	Very good condition
3	Good condition
2	Satisfactory condition
1	Poor condition

The servo motor responds to every form of condition state to carry out the load. It is due to the various servo motor types and possible loads that they can support. The servo motor will respond to the user clicking the button on the Blynk app to lock or unlock the door during this test run.

The flawless state demonstrates that the test is conducted in real-time and there is no overshoot. The very good condition then reveals a slight overshoot that replies 0.5 seconds later than in real-time. The excellent condition then displays the ensuing overshoot, which results in a response that is one second late in real-time. Not to mention, the successful condition displays the outcome of the overshoot that caused a response to be delayed by 2 seconds in real-time. Finally, a bad condition displays a response that is 3 to 5 seconds behind real-time.

The servo motor steadily overshoots on the 20th run because the source supply, an AC/DC adaptor that delivers 4.5V for the Arduino, is insufficient to power the servo motor. The servo motor begins to operate continuously until the 200th after the source supply is switched out for a 9V battery. Figure 24 below displays the outcome of the durability test count by 0-200 in real-time.

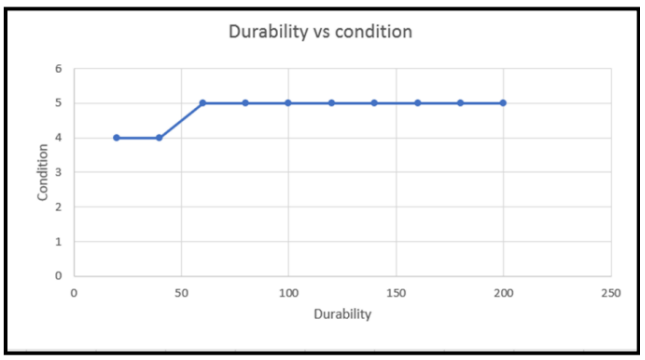


Figure 24 shows a real-time graph of a durability test from 0-200.

With Arduino WeMos and Blynk, the servo motor functions without any overshoots and with a count delay of 200 to 400. Figure 25 below displays the outcome of the durability test with a count range of 200 to 400 in real-time.

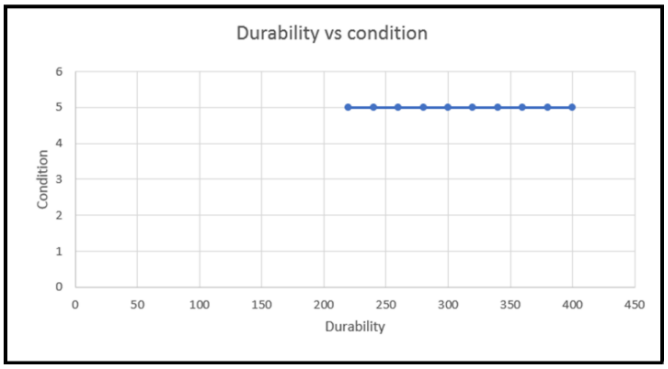


Figure 25: Real-time graph of a durability test from 200 to 400.

The servo motor operates flawlessly with Arduino WeMos and Blynk, with no overshoots and no lagging by 400 to 600 counts. The results of the durability test count from 400 to 600 are shown in real-time in Figure 26 below.

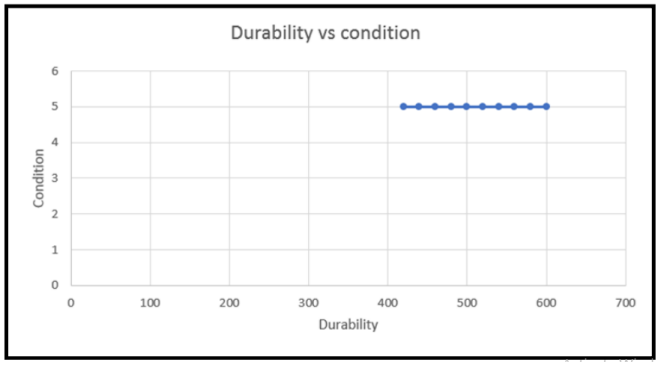


Figure 26: Real-time graph of a durability test from 400 to 600.

With Arduino WeMos and Blynk, the servo motor runs without any overshoots and without trailing by 600 to 1000 counts. Figure 27 below displays the outcome of the durability test count from 600 to 1000 in real-time.

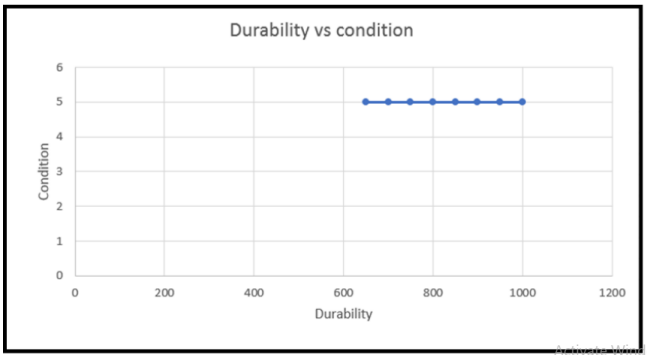


Figure 27: Real-time graph of a durability test from 600 to 1000.

The servo motor operates flawlessly with Arduino WeMos and Blynk, with no overshoots and no lagging by 600 to 1000 counts. The results of the durability test with a count range of 200 to 400 are shown in real-time in Figure 28 below.

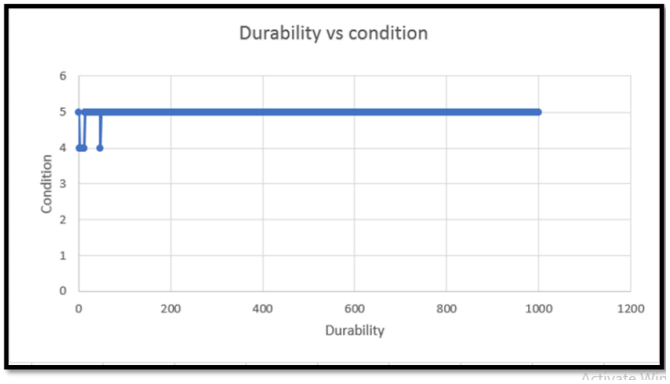


Figure 28: Real-time durability test graph from 0-1000.

There are many different servo motors on the market, and each one has a unique purpose. The servo motor MG996r that was utilized to control them is fairly powerful. The servo does not overshoot when traveling at a high speed. Before being disassembled, every AC servo motor will have its alignment examined and both statically and dynamically recorded. When the servo motor is serviced and rebuilt, this process guarantees accurate alignment. The servo motor must be aligned properly in order to function properly, as improper alignment will prevent the servo motor from operating at all. A second round of all the original tests is performed as part of the final servo motor repair testing to confirm the issue has been resolved. All servo motors are operated by drives that were produced or installed in matching servo motor and feedback configurations at the factory. A load was applied to this servo during testing, and it ran continuously for 20 minutes without experiencing any overshoot because of the load. The servo motor's test run is seen in Figure 29.

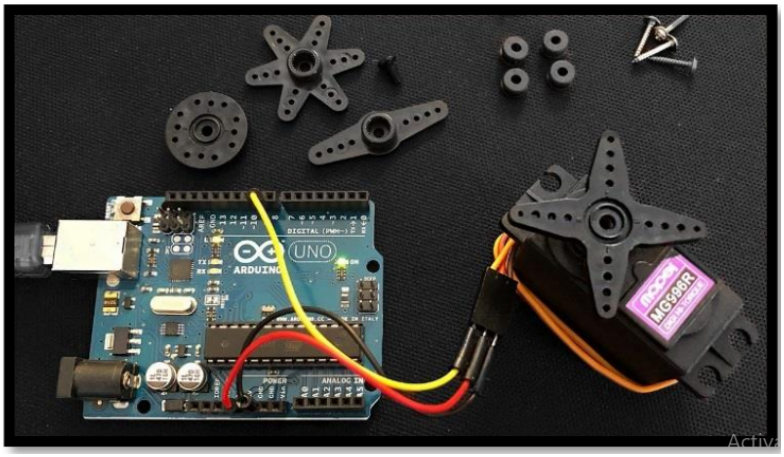


Figure 29: Servo motor testing

If the Arduino Wemos source is sufficient and suitable, adding the Blynk will allow for continuous control without overshoot. The servo motor will overshoot as a result of the incompatible source. The test run for the servo motor using the Blynk program is shown in Figure 20.

Figure 30: Run the Blynk program to test the servo motor.

A straightforward servo sweep program from the Arduino IDE examples is used to test the servo, which activates and rotates when the button widget on the app is pressed. After pressing the button, the servo turns continually until the button is released, at which point it comes to a halt. This is accomplished by setting virtual pins to 0°–180°, which causes the virtual pin to execute a certain sort of code (in this example, the servo angle code).

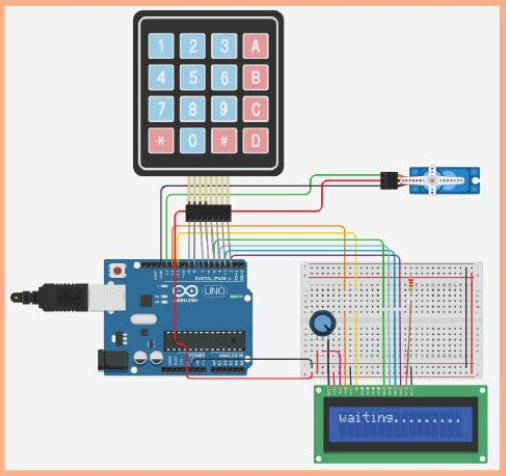
CONCLUSION

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.

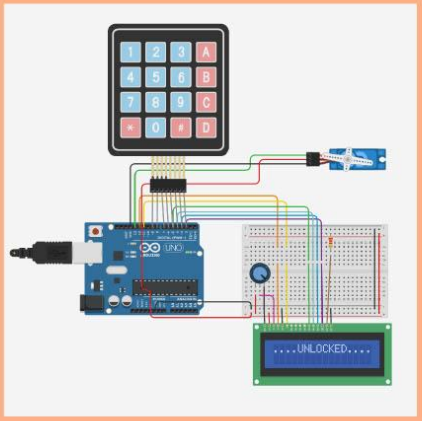
The goals of this project have been accomplished. By utilizing the Blynk application framework, this project seeks to construct an IoT monitoring system. This project consists of an Internet of Things (IoT) system that was constructed using an Arduino Wemos D1 as the microcontroller and a Blynk monitoring 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.

Screenshots of the security system's operation and results using Tinkercad are shown in Figure 31:

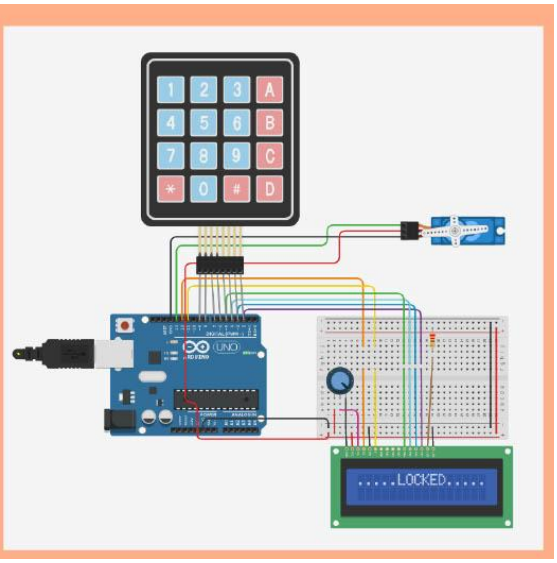
It then waits for the user to type in the password



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.



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";

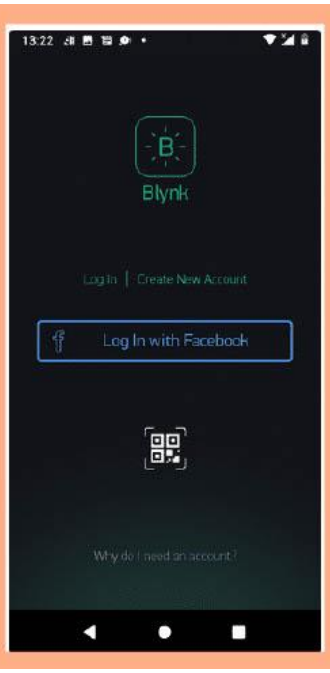
6. select NodeMCU as the device and WiFi as the connection type;

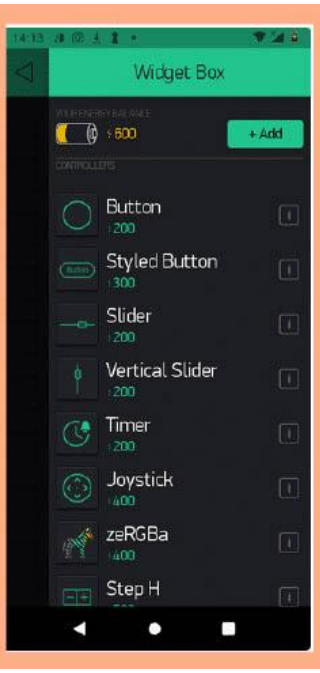
7. hit the create button at the bottom.

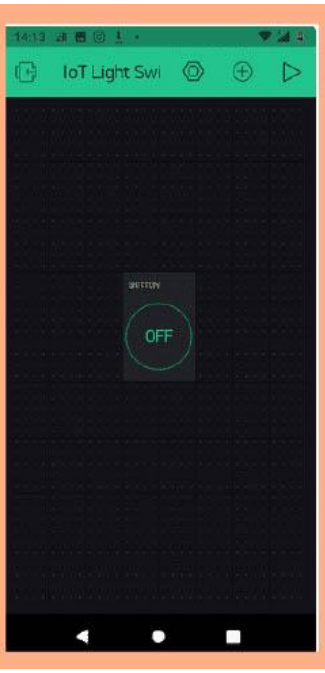
8. Add a button by tapping the "+" symbol.

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.







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.
```

```
 *************************************************************
  This example runs directly on NodeMCU.

  Note: This requires ESP8266 support package:
    https://github.com/esp8266/Arduino

  Please be sure to select the right NodeMCU module
  in the Tools -> Board menu!

  For advanced settings please follow ESP examples :
   - ESP8266_Standalone_Manual_IP.ino
   - ESP8266_Standalone_SmartConfig.ino
   - ESP8266_Standalone_SSL.ino

  Change WiFi ssid, pass, and Blynk auth token to run :)
  Feel free to apply it to any other example. It's simple!
 *************************************************************/
```

```
 *************************************************************

/* 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(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), 8080);
}

void loop()
{
  Blynk.run();
}

```

- **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

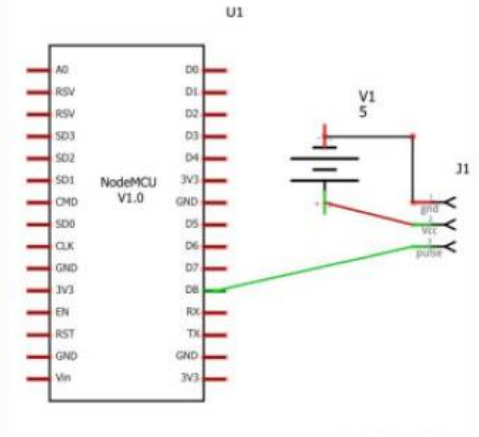
Software:

- Arduino IDE
- BLYNK app

Step 2: Circuit and Connections

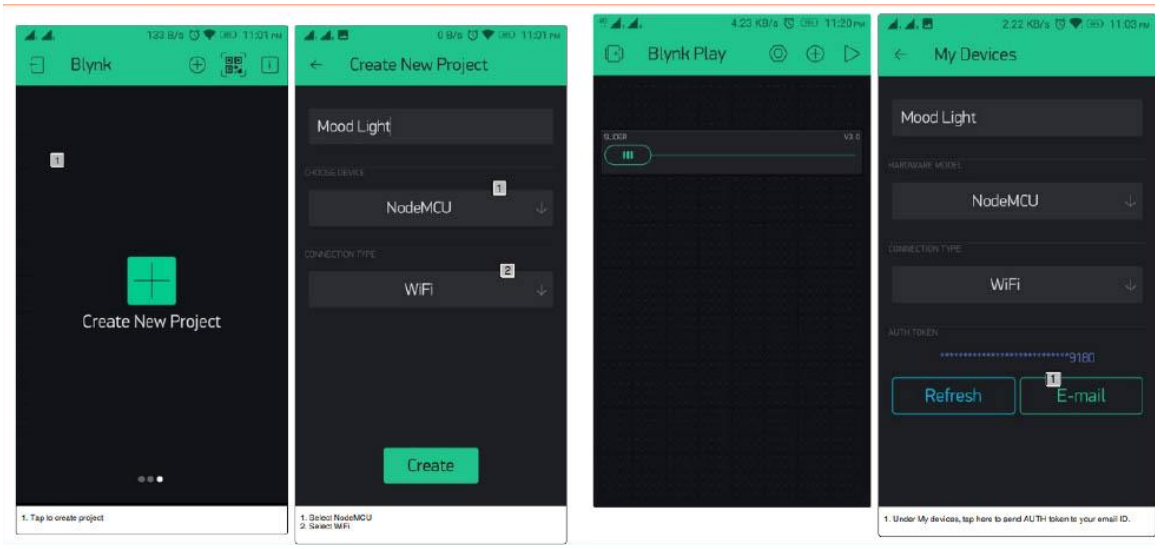
Connections:

- 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).



Step 4: Time to Code:

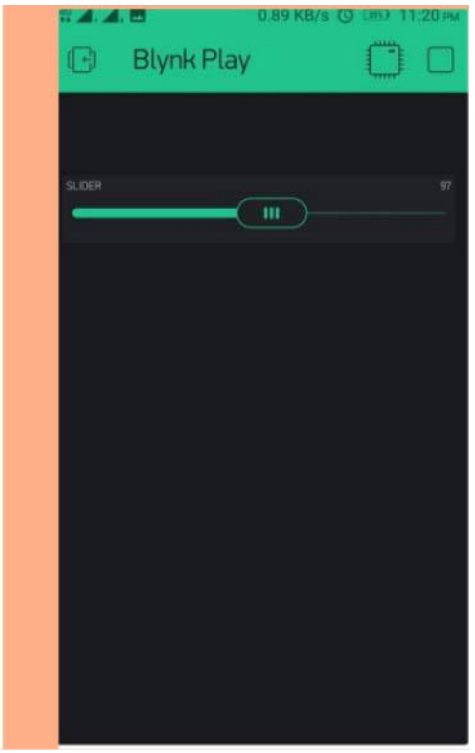
We only need to add our AUTH ID(noted above), Wifi SSID and Password in the code and upload it to NodeMCU using Arduino IDE. In the code, we do not need to setup our pins this is already done by BLYNK in their library.

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.

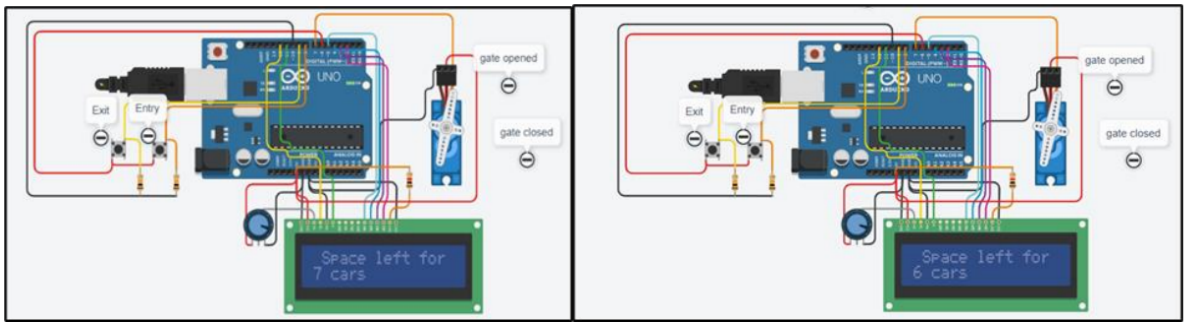
The aforementioned pictures show how to operate an IR remote control for a light bulb using the Blynk app, a relay, and a NodeMCU ESP8266.

2. Car parking system:

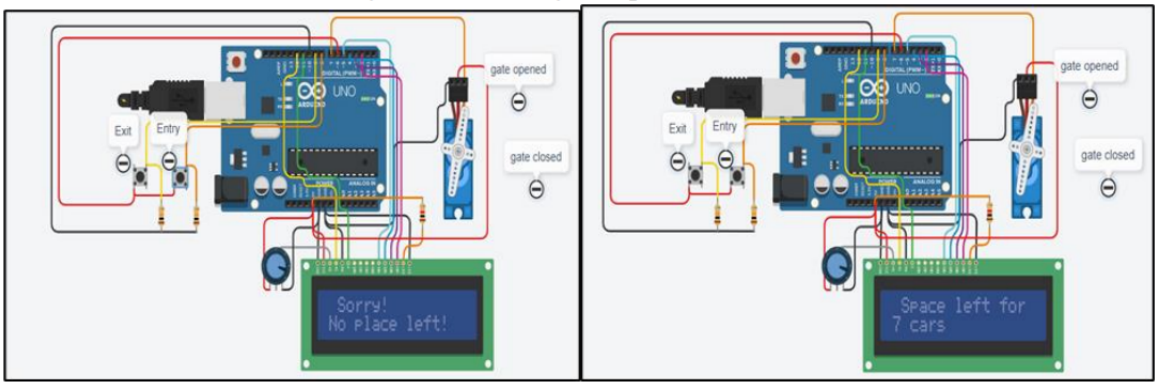
Figure 18 displays When a car arrives, the count on the LCD lowers from initial printing of 7 spaces remaining for automobiles. In Figure-19, When there is no room left after all the cars

have been parked, the space that is still available is shown. Implementing an automated parking lot operation in a garage is simple.

Once the automobile is parked, two pushbuttons are used to signal arriving and outgoing cars, and an LCD display will indicate how much space is left once the car has been parked. Similar to when a car is removed from a parking area, more room becomes available. To prevent the misplacing of autos in the lot, this application can be used at locations where heavy vehicles are parked. The following is a screenshot of the successful simulation of the car parking system.



The pictures shows the space left for the care



The pictures show after filling all spaces

The following screenshots are the coding programmed on tinkercad for the successful simulation.

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)

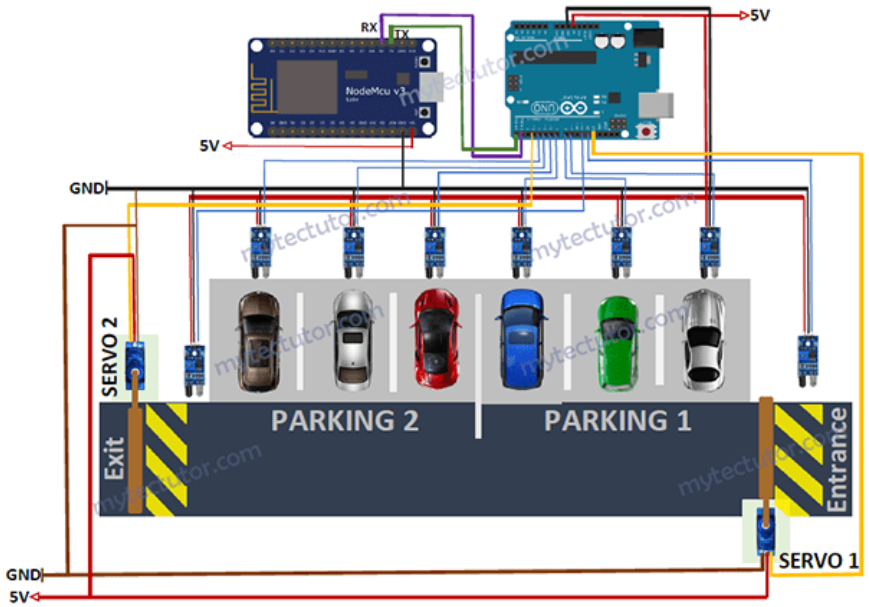
```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

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

Finding a parking spot in busy places like residential neighborhoods, business buildings, schools, and other institutions is a typical difficulty that may be resolved with the use of a smart auto parking system. I'll show how an ESP8266 NodeMCU, Arduino, and Blynk server may be used to create a smart auto parking system in this passage.

In order to know which parking spaces are available in a specific place, this technology enables automobile drivers to monitor parking slots on a mobile device like a smartphone. The general flow of traffic is improved since a vehicle won't waste time driving to a full parking lot, preventing the buildup of automobiles owing to a shortage of parking spaces.

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.

The smart automobile parking system employing Arduino and ESP8266 NodeMCU is schematically shown in the following image. I separated the parking space into two lots, each of which had three parking spaces. IR sensors are connected to the Arduino's digital pins 4,5,6,7,8, and 9 in these slots. At the entrance and exit of the parking lot, two more IR sensors are positioned and connected to the Arduino using digital pins 11 and 12. To move the toll gates at the entry and exit, two servo motors that are connected to Arduino digital pins 3 and 13 are employed.



Through the serial ports TR and RX of the ESP8266 NodeMCU linked to Pins 0 and 2 of the Arduino Uno, respectively, communication between the Arduino board and NodeMCU is made possible. Make careful you utilize the appropriate serial pins if you're using a different Arduino board.

Considering the power supply. A separate 5V power supply for the servo motors, Arduino board, and ESP8266 NodeMCU are required. Additionally, check to see that each component is securely grounded.

Regarding, the operation of the smart parking system. When there is a vacant space in the

parking lot, the IR sensor at the entry toll gate detects a coming automobile and opens automatically; however, if the lot is filled, the gate will not open.

Esp8266 NodeMCU permits the connection between the sensors in the parking lot and any mobile device via a network cloud service; in this example, I have chosen Blynk server. A straightforward mobile app is created using Blynk server to show the driver the occupied and empty spaces in the parking lot.

Code for an ESP8266 NodeMCU and Arduino smart car parking system.

The project's code is broken up into two sections: one one controls the Arduino board, while the second section manages the ESP8266 NodeMCU's connection to the Blynk server. However, the Arduino IDE was used to create both of these code samples.

The Arduino Board's source code

This code mostly comprises declarations for servo motors and IR sensors, as well as rules for managing how the entry and exit gates open and close in response to the state of the parking lot. For serial connection with the ESP8266 NodeMCU, the SoftwareSerial.h library is supplied.

```
#include <SoftwareSerial.h>
#include <Servo.h>

Servo myservo1;  // create servo object to control a servo
Servo myservo2;

SoftwareSerial nodemcu(0,1);

int parking1_slot1_ir_s = 4; // parking slot1 infrared sensor connected with pin number 4 of arduino
int parking1_slot2_ir_s = 5;
int parking1_slot3_ir_s = 6;

int parking2_slot1_ir_s = 7;
int parking2_slot2_ir_s = 8;
int parking2_slot3_ir_s = 9;

int entrance_gate = 11;
int exit_gate = 12;

int pos1 = 90;    // variable to store the servo position(entrance gate)
int pos2 = 90;    // exit gate

String sensor1;
String sensor2;
String sensor3;
String sensor4;
String sensor5;
String sensor6;


String cdata =""; // complete data, consisting of sensors values

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

```
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()
{

plslot1();
plslot2();
plslot3();

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 plslot1() // 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 plslot2() // parking 1 slot2
{
  if( digitalRead(parking1_slot2_ir_s) == LOW)
```

```
void p1slot2() // parking 1 slot2
{
  if( digitalRead(parking1_slot2_ir_s) == LOW)
  {
  sensor2 = "255";
  delay(200);
  }
if( digitalRead(parking1_slot2_ir_s) == HIGH)
  {
  sensor2 = "0";
 delay(200);
  }
}


void p1slot3() // parking 1 slot3
{
  if( digitalRead(parking1_slot3_ir_s) == LOW)
  {
  sensor3 = "255";
  delay(200);
  }
if( digitalRead(parking1_slot3_ir_s) == HIGH)
  {
  sensor3 = "0";
 delay(200);
  }
}


// now for parking 2

void p2slot1() // parking 1 slot3
{
  if( digitalRead(parking2_slot1_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
          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
        }
      }


}
```

Code for the NodeMCU ESP8266.

With the help of this code, the ESP8266 NodeMCU can communicate with the Blynk server and Arduino. In order for the device to access the Blynk server, the proper WiFi SSID, password, and 'AuthToken' must be entered.

Application for Blynk.
Using an Android or iOS device, this is utilized to remotely check on the condition of the parking lot. Simple LEDs are used in the application's design to indicate an occupied slot and an empty slot, as seen in the example below.

In the aforementioned illustration, parking 1 has three occupied spots whereas parking 2 has one vacant slot in the middle.

3. Simulating a house using sensors
The final application makes use of the smoke detectors, LDR, PIR, and ultrasonic sensors on the microcontroller board. A smoke detector will detect gas if it detects any leaks. Gas detection sensors offer information that triggers the alarm circuit and sounds the alert. The entry door will automatically open if an ultrasonic sensor picks up something. The PIR sensor activates the fan and looks for human activity when someone is in front of the door. Additionally, there is a manual fan. LDR is used to automatically turn on the light when someone is home at night; however, when it is daylight, it is quickly turned off. These features for the previous application appeared to perform well and as intended. Bulb turns on as light intensity drops here, and vice versa. The result section offers a general description of each phase of the simulation from Tinkercad. The coding for simulation in Tinkercad is displayed in the next screenshots.

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);
}

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
```

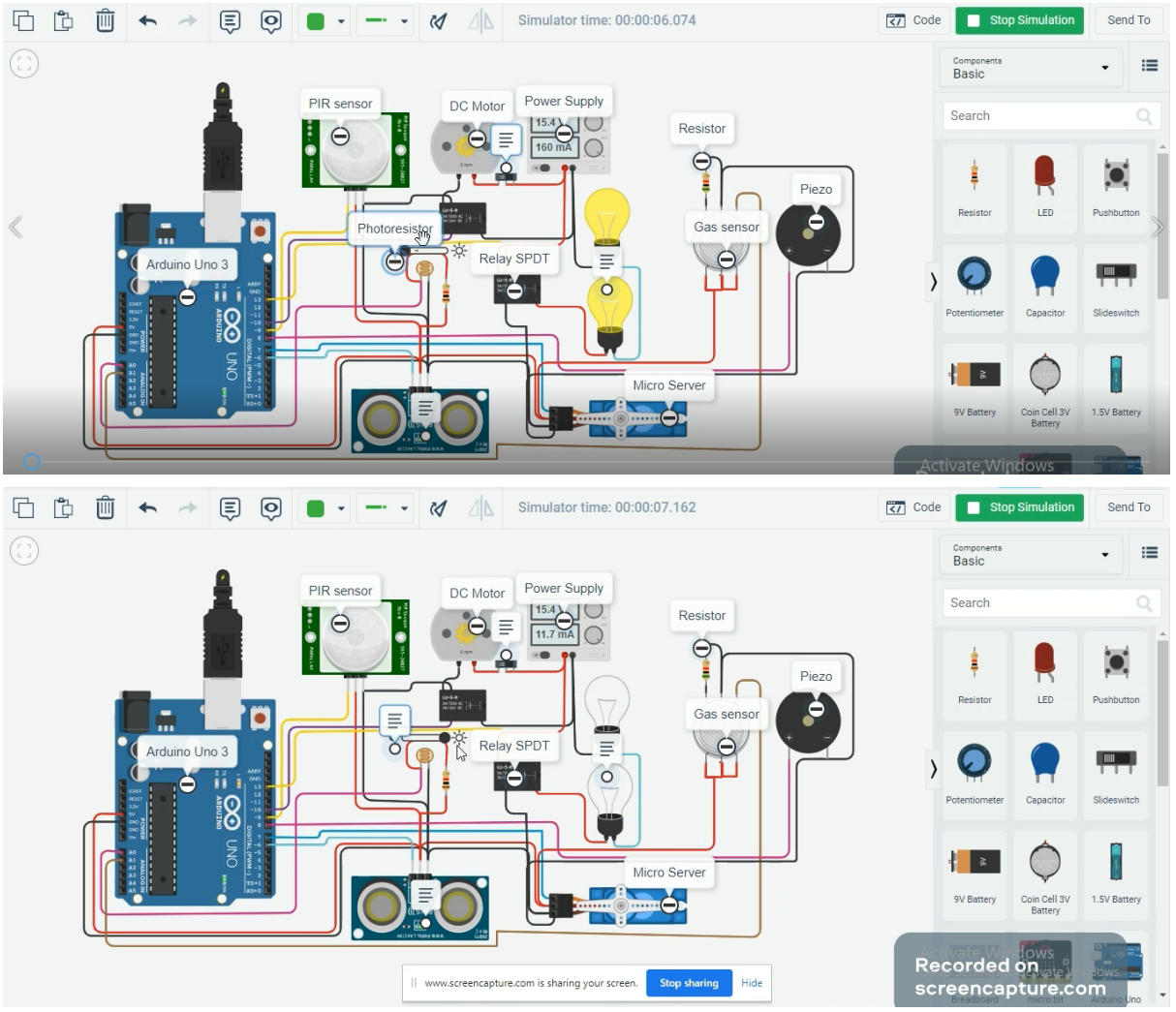
Coding (Block)

```
// 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)
}
```

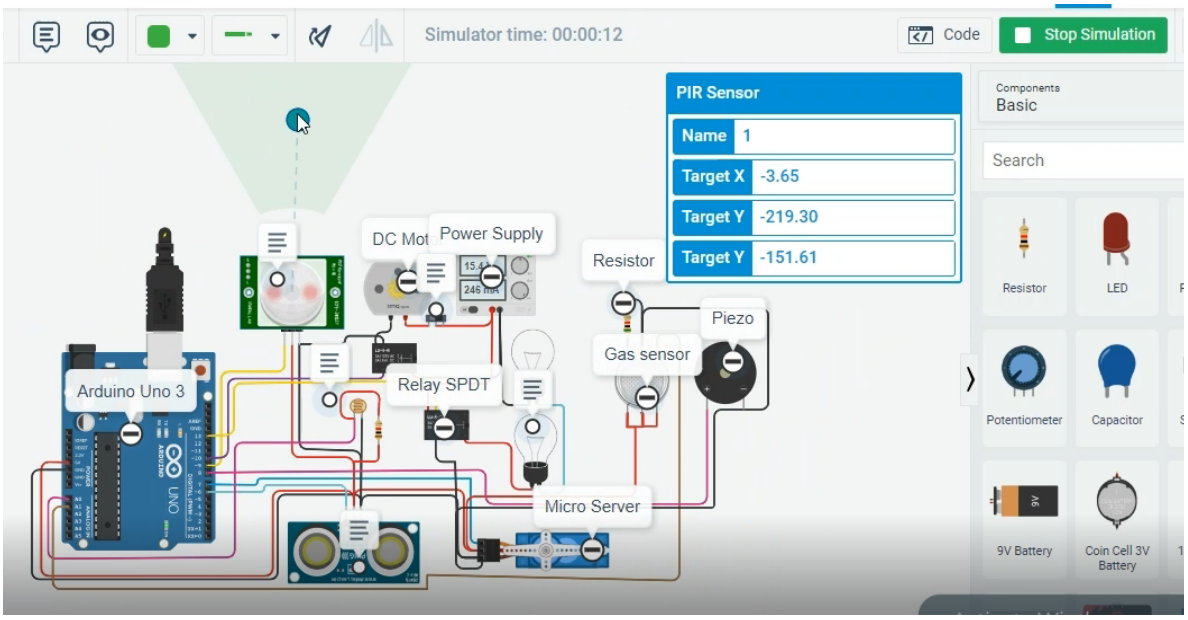
3. Sensor-based home automation system

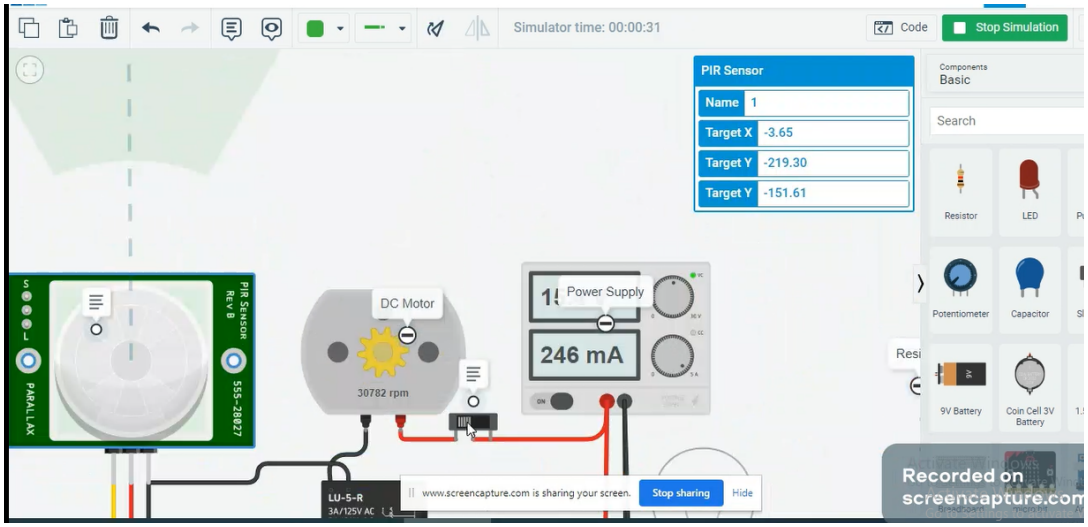
The final application makes use of the smoke detectors, LDR, PIR, and ultrasonic sensors on the microcontroller board. A smoke detector will detect gas if it detects any leaks. Gas detection sensors offer information that triggers the alarm circuit and sounds the alert. The entry door will automatically open if an ultrasonic sensor picks up something. The PIR sensor activates the fan and looks for human activity when someone is in front of the door. Additionally, there is a manual fan. LDR is used to automatically turn on the light when someone is home at night; however, when it is daylight, it is quickly turned off. These features for the previous application appeared to perform well and as intended.

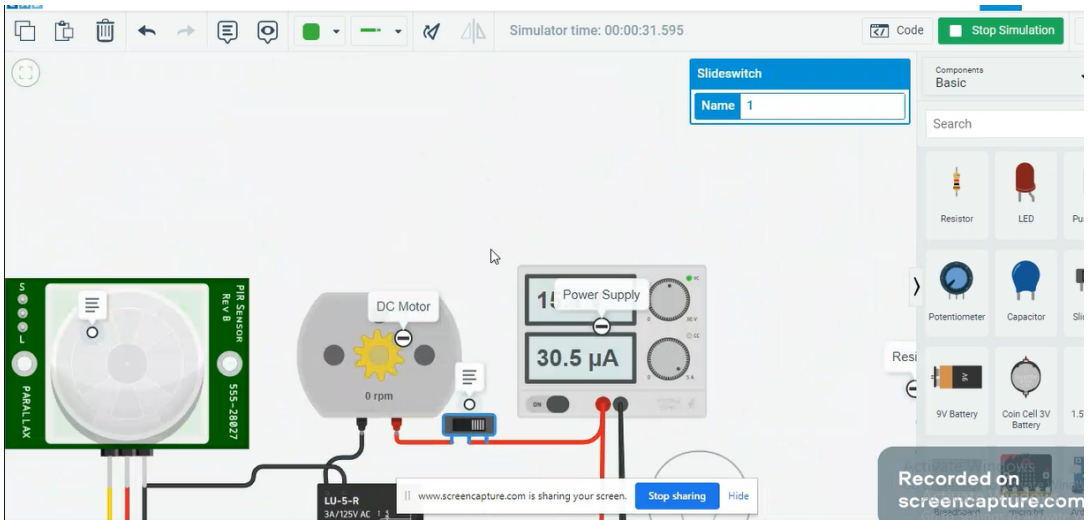
Bulb turns on as light intensity drops here, and vice versa.



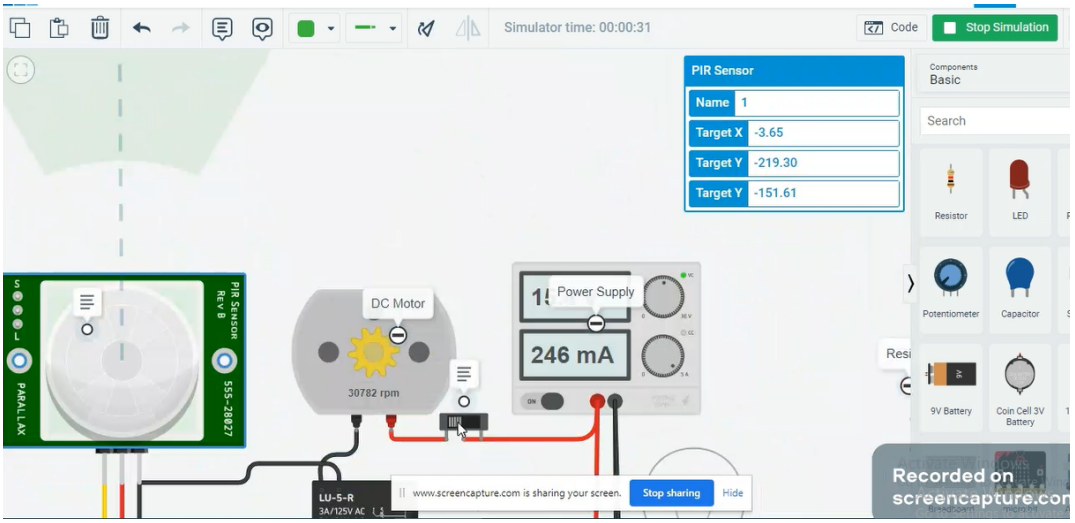
If someone enters the home, the fan will be turned on automatically. Also, we can control it manually with the switch.

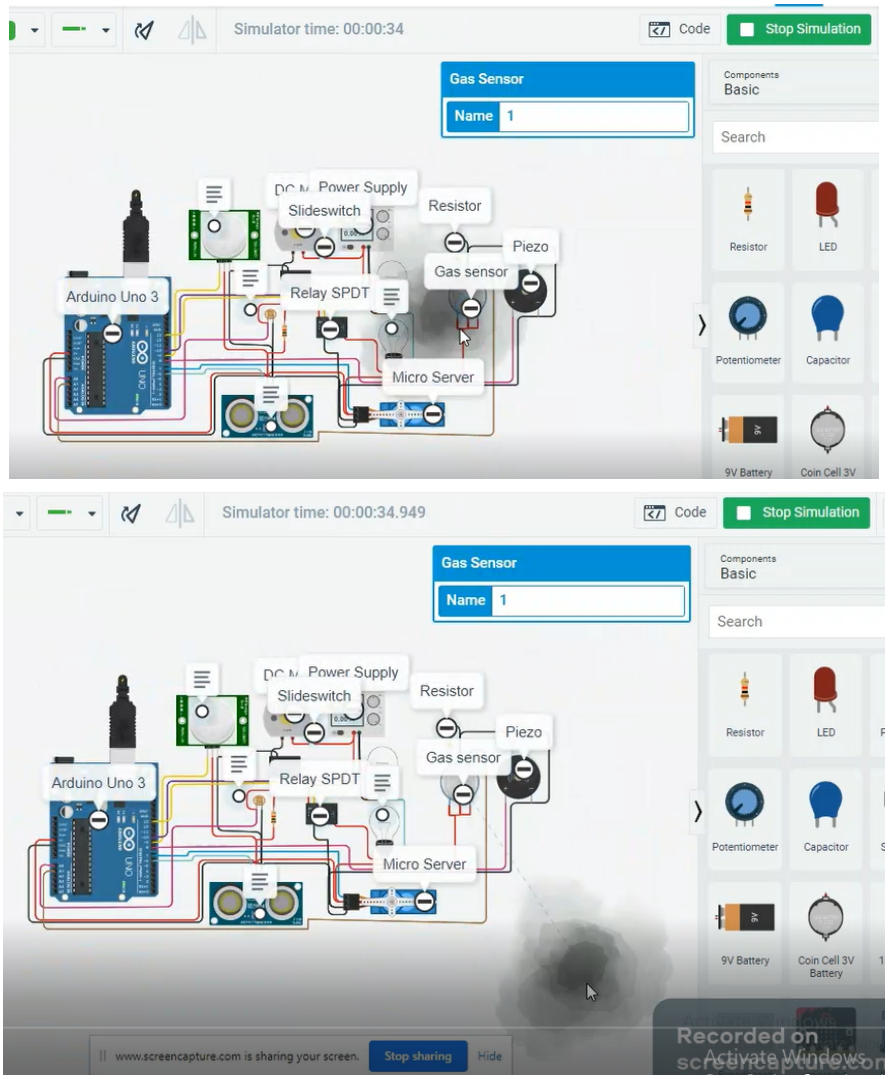




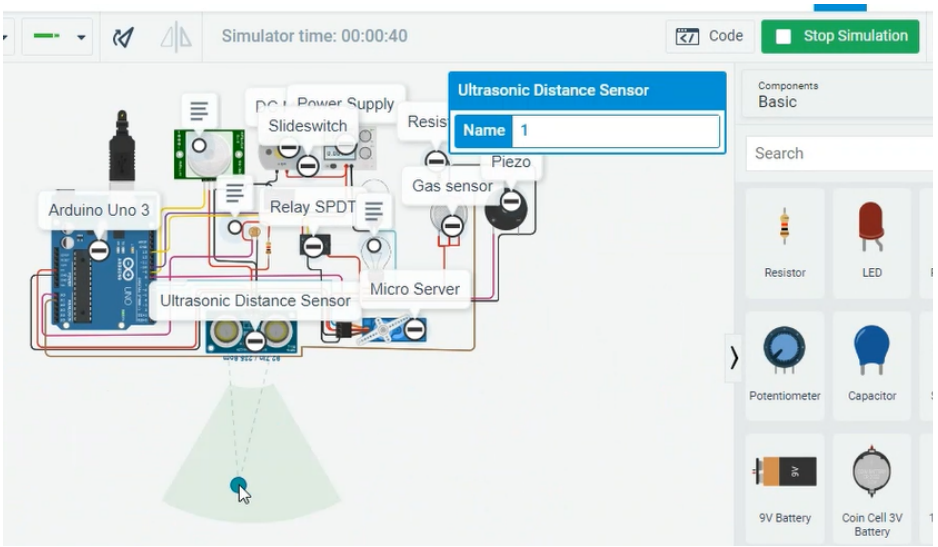


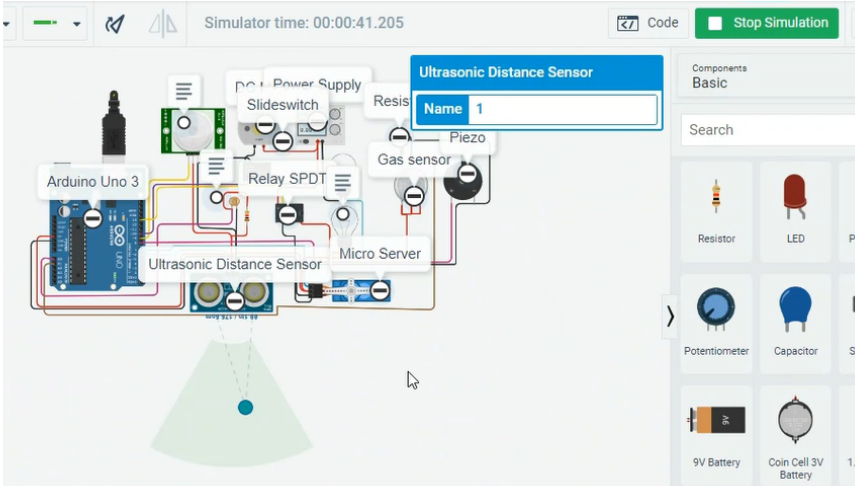
If there's any leakage of LPG gas, the gas sensor detects it and information is received as a result of detection by the gas sensor. The information is then transmitted to the alarm circuit to activate the buzzer and the buzzer rings to make an alarming sound until the gas is no longer nearer to the gas sensor.





The ultrasonic sensor is set up on top of the main door. If anyone comes nearer the door, the door automatically opens.





An overview of the proposed design.

For remote home control, pleasant and healthy living, energy efficiency, safety, security, and social advantages, smart home automation is an IoT application field. Homeowners now have complete remote control over their dwellings, appliances, ambient conditions, and activities taking on there and nearby, regardless of where they are in the world at the moment. For the intelligent control of a house, a number of commercial devices like Amazon Echo, Google Nest Hub, Wink Hub2, Samsung SmartThings, and Apple HomeKit have been created, tested, deployed, and used. Interoperability, data security, data analytics, communication security, and the home continue to be major obstacles for smart home automation. We offer an intelligent home automation system, which we have called the intelligent home control and security system, to solve some of these current issues in smart home automation (iHOCS). The system's description and general functions are provided in the next sections.

The iHOCS System

The iHOCS system is an intelligent home automation system built on the Internet of Things (IoT) that can detect motion and dangerous gases within the home and operate basic home appliances. By offering convenience, security, and safety for residents of homes, our iHoCS is intended to contribute to the field of home automation in the IoT. We present a system that classifies photos gathered from motions within the house and utilizes the support vector machine algorithm as a machine learning tool to differentiate between typical occupants or their pets and an intruder. Through a mobile application built on the Android operating system, the user in our system oversees, manages, and monitors the home. The intelligent device module, the communication and gateway module, the management and decision module, the cloud computing module, the presentation module, and the security module are the six components that the iHOCS combines to provide its functionality and presentation. In the subsections that follow, each of them is covered in turn.

Intelligent Device Module

The system's intelligent module defines the smart home appliances, such as gas/PIR/ultrasound/LDR sensors, DC motors, lightbulbs, and so on. The household appliances have a level of integration that allows them to respond to orders in an intelligent way. In the house, the sensors are utilized for data collection, monitoring, and detection. They include motion sensors for intruder detection and gas, smoke, and fire detectors; light sensors for measuring and monitoring environmental factors; the fan, or dc motor, operates when someone is close to the PIR sensor but can also be manually turned off by turning off the dc motor's slide-switch; and a smart camera for taking pictures of an intruder.

A smoke detector is used to find a gas leak if there is one. The Alarm circuit will turn on if gas is identified in the sensor data that is being gathered. The front door will automatically open using ultrasonic technology if someone is in front of it. The fan is activated by the PIR, which also measures humidity. Additionally, there is a manual fan. LDR automates light control in residences by turning on the bulb at nighttime entry and shutting it off at departure.

Gateway and communication module.

The interaction and communication between the sensors and gadgets in the house and the outside world depends on this module. Gateways provide connections and communication between local and external networks in a smart home setting. To interact with one another and to communicate with the user, the gadgets and sensors in the house are linked to a gateway. For expanded network coverage and communication, our solution employs an ESP8266 Wi-Fi board as the gateway. Through this gateway, the system connects to the Internet and communicates with every appliance and gadget in the house. Wi-Fi, one of the main operational standards for home automation technology, TCP/IP, and HTTPS/IP are utilized as the communication protocols.

The Module for Management and Decision.

Along with the conventional control of appliances, the iHOCS system offers safety and security. Prior to notifying the user about the observed motion, the SVM is used to categorize various features in the house in order to increase security.

We provide SVM machine learning techniques for categorizing and extracting distinctive characteristics from a picture taken by a camera in a smart home setting. Before raising an alert, the system will be able to differentiate between inhabitants and invaders thanks to the classification algorithms. We suggest a support vector machine to effectively categorize our photographs in order to do this.

A supervised, linear machine learning approach for classification and regression issues is known as a support vector machine. SVMs may be used for image classification, face identification, motion detection, handwriting recognition, text recognition, and other tasks. SVM is a machine learning method that is used to categorize data that can be separated linearly or non-linearly. It operates by identifying the optimum separating hyperplane for data discrimination. For analysis and prediction with little data, the support vector machine is quick, precise, and dependable. This serves as the basis for using it in our investigation. In our research, the SVM is employed to automatically extract characteristics from the photos and categorize them in order to identify people or items in the house. Human facial characteristics including eye color, face shape, skin tone, and nose shape are among the distinctive traits taken into account for image processing. The motion sensor picks up movement in our selected home environment and sends a signal to the camera to take a photo of the subject. The machine learning technique is used to analyze the identified image to confirm its identification before the system triggers a security alarm because there are several individuals in the home, each with their own distinctive traits.

We provide a case in which the user is away from home, but there are still people living there or the home security system has been turned on at night. Movements would still be picked up in the home and its surrounds in these two scenarios. The only movement would be picked up by a motion sensor, which could then notify the user. The mobility of the legal house inhabitants would cause these frequent notifications to be false alarms most of the time, which would be bothersome. By utilizing machine learning techniques, the system is improved in its ability to appropriately categorize photographs and determine if there is an intruder present or whether the motions seen were those of normal house residents.This module categorizes and analyzes the image's characteristics in order to alert the user and preserve the intruder's image.

Cloud Computing Module

A smart home includes portals and applications that provide quick and affordable ways to track, connect, and manage anything from anywhere at any time [48]. The PIR/ultrasonic motion/gas/LDR sensor, dc motor, lightbulbs, and ESP32 camera in our design for the iHOCS environment all produce data at user-configurable intervals. For home monitoring, analysis, and future forecasting, the generated data must be kept. For real-time data storage, processing, communication, and monitoring in a smart home, the cloud infrastructure is required. As a result, in our work, data created is stored, shown graphically, and processed using a cloud computing module.

Module for Presentation and Control.

The interface for remote home control is described in the presentation module. The user may choose the preferred appliance to operate or monitor using the mobile application. This module also includes a graphic representation of the output produced by the home's sensors as well as panels and tabs for carrying out various functions. To allow the user to view any changes in the state of the house and surroundings, the iHOCS system shows light bulbs being switched on and off as well as the output of the PIR motion/LDR/gas/ultrasound

motion sensor on the mobile application screen.

Module for Presentation and Control.

The interface for remote home control is described in the presentation module. The user may choose the preferred appliance to operate or monitor using the mobile application. This module also includes a graphic representation of the output produced by the home's sensors as well as panels and tabs for carrying out various functions. To allow the user to view any changes in the state of the house and surroundings, the iHOCS system shows light bulbs being switched on and off as well as the output of the PIR motion/LDR/gas/ultrasound motion sensor on the mobile application screen.

Architectural Design

The iHOCS system's architectural layout is seen in Figure 1 below. The system contains the user, household appliances, sensors, a Wi-Fi module, and the cloud platform, as shown in the architectural design. The ESP8266 Wi-Fi module serves as both a microcontroller and a communication device. The user, the home, and its equipment all communicate with each other wirelessly in this work. Data is gathered from the sensors and relay board by the ESP8266 and sent to the user through the Internet. An Android smartphone application, which interfaces with the microcontroller, allows the user to send and receive instructions and data from and to their house (ESP8266). The system's functional features include home monitoring, control, and security. Figure 2 uses a flowchart to depict the progression of procedures and actions inside the iHOCS environment.

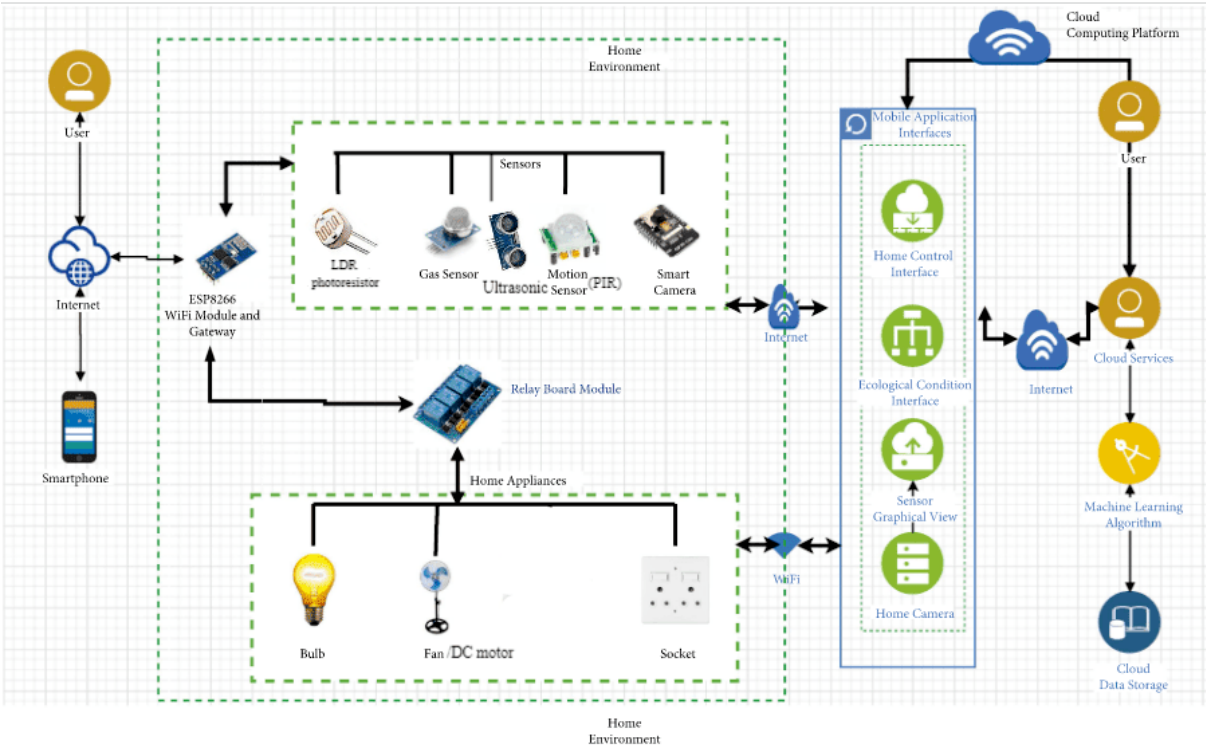


Figure 1: The iHOCS architecture.

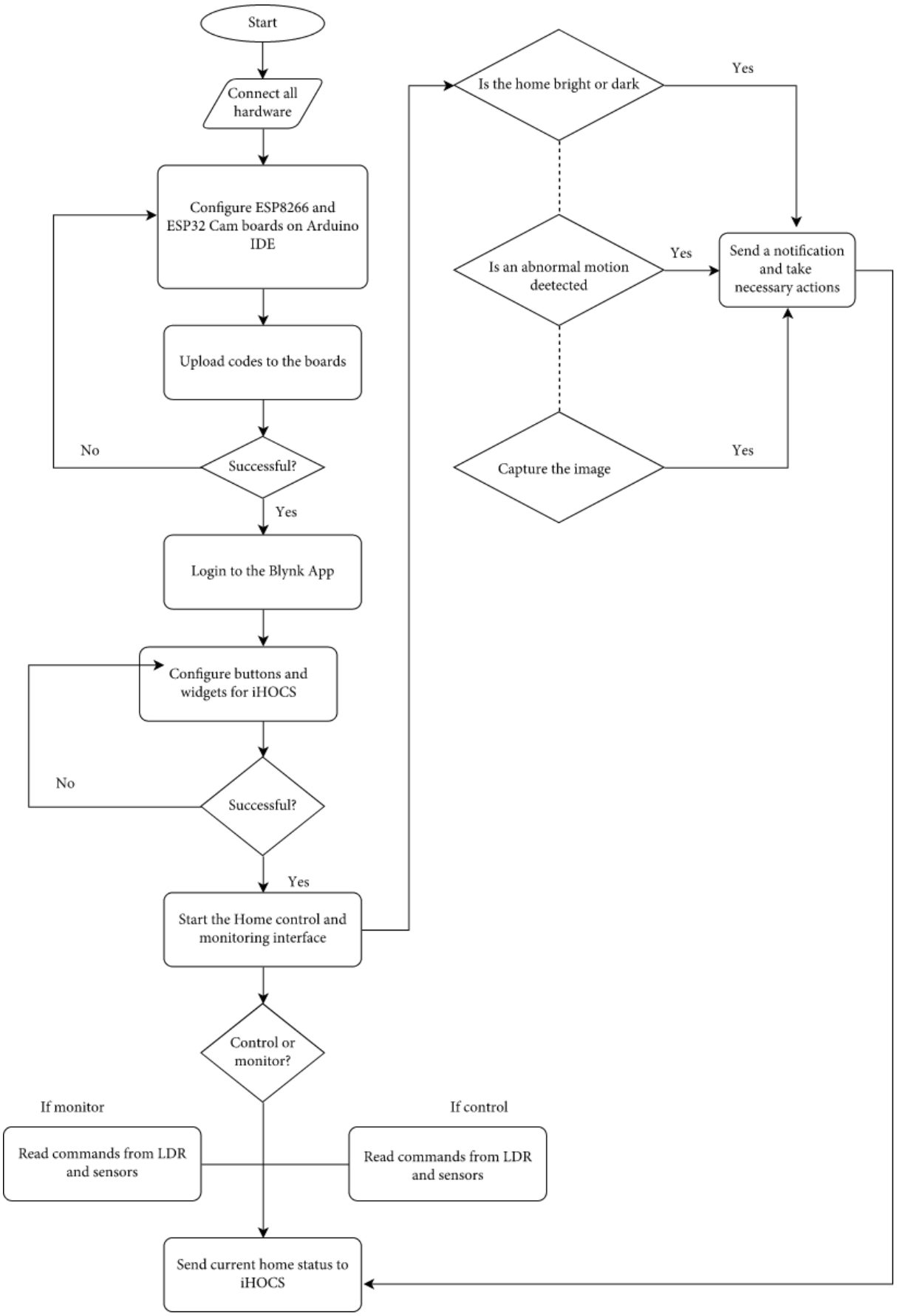


Figure 2 : Flow diagram of iHOCS.

The iHOCS configuration includes electrical outlets and other standard household equipment that may be controlled, including lightbulbs, televisions, air conditioning, and heating systems. It also measures the surrounding environmental factors, as was already mentioned. Finally, it notices movement and takes a picture of the subject. The relay board module is linked to the ESP8266, and the relay board module is connected to the plugs and household appliances so that the Android application may receive and transmit commands.

The user's smartphone's screen displays the results of the LDR sensor's measurement of the light levels. LDRs (light-dependent resistors) are used, for instance, in automatic security lighting, to detect light levels. The resistance of an LDR is high in the dark and at low light levels, and little current may pass through it. Their resistance reduces as the light intensity rises. The house is improved with increased security and occupant safety.

In order to monitor the graphs, a notice is issued on the Android mobile application when movement in the home is detected by a PIR/ultrasound motion/gas/LDR sensor. Prior to alerting the user and sounding the alarm, the SVM runs a background check. All of the data produced by all of the sensors is stored in the cloud storage that is integrated with the mobile application. For wireless transmission to the cloud platform through the mobile application, the sensors are interfaced with the Wi-Fi module.

Results and Discussion

In order to accomplish the required intelligent home control, monitoring, and security system, our iHOCS is configured utilizing IoT hardware and software tools, as illustrated in the architectural design. The ESP8266 Wi-Fi board, the HC-SR501 PIR motion sensor, the 5 V four-channel relay module, breadboards, LEDs, LDR, an ultrasonic sensor, and an ESP32 camera were used to create a prototype of the iHOCS system. Below is a detailed explanation of each component. The user can carry out the following standard tasks after successfully registering and configuring the iHOCS Android mobile application:
(1) Management of household electronics.
(2) Keep an eye on the surroundings at home (light levels).
(3) Automatically turn lights ON and OFF depending on the state of the house.
(4) Movement detection within the house.
(5)Receiving alerts about the house
(6)Check out the sensor data in graphical form.

PIR Motion Sensor i.e. HC-SR501

An IoT motion sensor called the HC-SR501 is utilized in microcontroller-based prototypes. It can operate in an ultralow-voltage working mode and is sensitive, extremely dependable, and reliable. The wide-range connection and user-friendly interface of this motion sensor are some of its other characteristics. It may be utilized in security systems, automated garage doors, dryers, automatic washing machines, automatic lights, and alarm systems since it can detect an intruder's movement. Three pins make up this PIR motion sensor (VCC, output, and ground). Any DC voltage between 4.5 and 12 volts may readily power an integrated voltage regulator. In our line of business, the HC-SR501 PIR motion sensor is mostly utilized to find unlawful movement within the house. To capture the image, it also transmits a signal to the

ESP32-CAM module. Before the system sounds the alarm, the SVM algorithm can identify the characteristics and determine whether the movement is the result of an intruder or not.

Wi-Fi Development Board i.e. ESP8266

The TCP/IP protocol stack built into the ESP8266's Wi-Fi Internet development allows for communication with Wi-Fi networks. In order to enable direct connections with other devices, it may also be utilized to create a local network of its own. With on-board processing and storage for integrating with sensors and other IoT devices, it is affordable.ESP-12s Wi-Fi module with exceptional antenna performance was employed in this study. 11 digital input and output pins as well as analog pins make up the features. In our work, we link sensors to the ESP8266 as a Wi-Fi module and development board, and we utilize a 5 V four-channel relay module to activate electrical appliances.

5 V Four-Channel Relay Module

A board used to regulate high voltage and high current loads is called a four-channel relay module. It is made to interface with sensors or a microcontroller board simply. For switching and isolating components, it features 5 V relay channels. The relay board contains an LED indication on each channel to show when the relay is turned ON. When the corresponding relay channel is activated, the LED indications get brighter. In our work, LEDs that function as a fan, lamp, socket, and an additional unit for an additional appliance are powered by a 5 V four-channel relay module.

ESP32-CAM

A low-cost development board for the Internet of Things (IoT) that has a Wi-Fi camera for image capture and video streaming is called the ESP32-Cam. An integrated PCB antenna, a camera, a micro-SD card connection, and Wi-Fi and Bluetooth connectivity are all features of the ESP32-CAM system on chip module. A smart home automation prototype uses the ESP32 to assure security. Because the ESP32-Cam lacks a USB port, it is utilized here using an FTDI programmer. Therefore, in order for the ESP32-CAM module to receive instructions for the picture capture, code created on the Arduino IDE is uploaded through the FTDI.

Software Components

Only a predetermined set of preconfigured instructions, methods, and programs may be used to operate and monitor the house, its gadgets, appliances, and sensors. Specific software programs are required for the correct operation and communication of our iHOCS system. The Arduino IDE and Blynk software are the main software packages used for configuring, programming, designing, and developing the iHOCS system.

Arduino IDE

For use with Windows, Linux, and MAC operating systems, the Arduino integrated development environment is a cross-platform application based on the C and C++ programming languages. Code that may be uploaded to boards is created using the Arduino IDE. In IoT prototyping, it is used to setup microcontroller boards. The Arduino IDE is used to create code for iHOCS, which is then uploaded to the ESP8266 and ESP32-Cam boards to control sensors and household appliances.

Blynk Application

Both the iOS and Android operating systems are supported by the cross-platform Blynk application. It is a tool for creating user interfaces for projects related to the Internet of Things and its applications. It is used to plan and create a mobile application that can manage hardware, show, save, and visualize sensor data. The Blynk platform is made up of three main parts: the Blynk app, which lets users design their own project interfaces using the widgets that are already available; the Blynk server, which acts as a channel of communication between various hardware platforms and smartphones; and the Blynk libraries, which allow for communication between the hardware platforms, the server, and the processes of incoming and outgoing commands. The graphical user interface, widgets, and graphical reading interfaces of our iHOCS system are designed using the Blynk program. The Blynk app's cloud computing capabilities are also used in our work as a cloud database for the real-time archiving of sensor data and associated parameters.

Results

The functionality, configuration, and setup of each component used in the design and development of the iHOCS system are all described in depth in this part. Additionally, the training dataset for machine learning picture categorization is displayed. The iHOCS system manages electrical household appliances (including lights, fans, sockets, and one other electrical appliance), assesses indoor environmental conditions (light intensity), detects motion and records images, sending the image to the user via a mobile application on an Android-based smartphone whenever and wherever the user chooses.

The Arduino IDE was used to create the configuration code, and when it had been successfully compiled, the programs were uploaded to the boards via a USB wire. The service set identification (SSID) and password for the home network were supplied during programming in order to ensure reliable connectivity. Additionally supplied was the authentication code produced during the mobile application setup in Blynk. To turn appliances ON or OFF, orders were sent via an Android-based smartphone. As stated in the

design, information obtained from home sensors is saved in a cloud database before being shown on the mobile application's interface. Because the user can view the trend of data directly on the mobile application page without having to shut or minimize the home control mobile application, the system has an advantage over earlier works that used ThingsSpeak and other web-based apps.

Prototype Implementation:

The ESP32-CAM board, an FDTI serial converter, breadboards to connect the various hardware parts to one another, an HC-SR501 PIR sensor, a number of jumper cables, an Arduino board, LEDs, resistors (10k, 1k, and 220 ohms), an NPN transistor, and a 5 V four-channel relay module are all integrated into the prototype implementation. On the breadboards, linkages were created using jumper wires (male to male, female to female, and male to a female) allowing both indirect communication between the various hardware components and direct communication. The Wi-Fi module and microcontroller were both implemented using the ESP8266. It links the different hardware parts to a gateway and wirelessly connects the household appliances (represented by LEDs) and sensors to the Internet (microcontroller).

PIR motion sensors were also linked to the microcontroller board using the proper jumper cables and pins, as well as the output units of the 5 V four-channel relay board to the ESP8266 board's designated pins. To signify our electrical equipment, several LEDs were employed (bulb, fan, socket, and an outlet for additional home appliances). According to the instruction from the smartphone, the LED lights switch ON or OFF. Figure 6(a) displays the command to turn on all four appliances in the house, and Figure 6(b) displays the LED lights that correspond to that instruction. Three appliances were turned on and one was turned off in Figures 7(a) and 7(b).

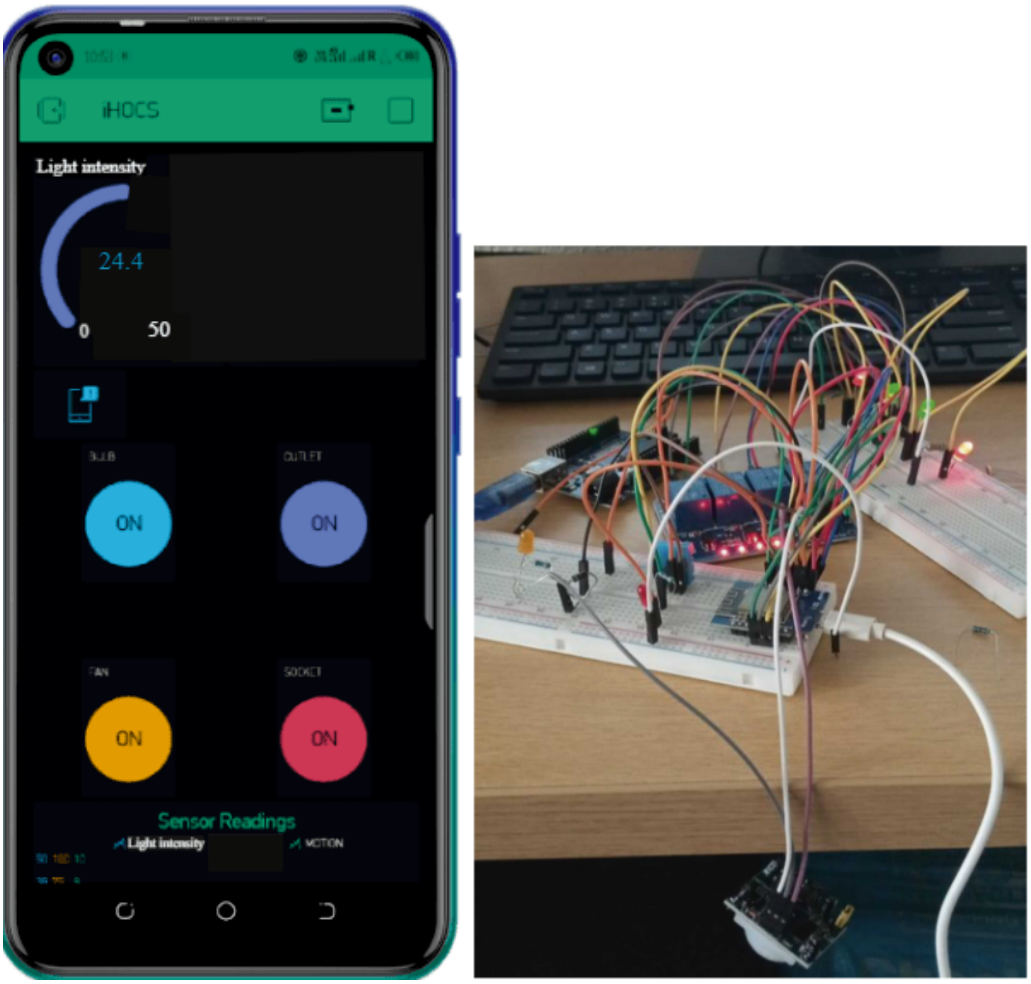


Figure 6
(a) All appliances ON
(b) Corresponding light indicators.

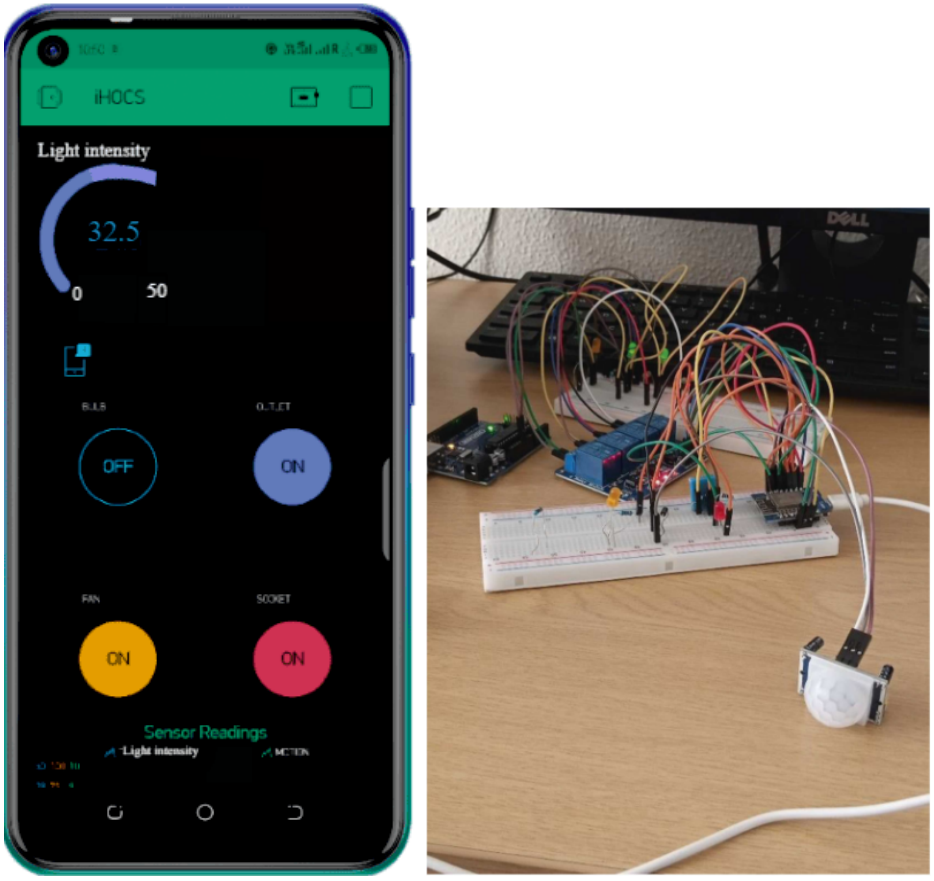


Figure 7
(a) Three appliances ON. (b)Corresponding three LED indicators ON.

Figures 8(a) and 8(b) show the mobile application screen's display of the values from the light sensors throughout the house (b). The mobile application will scan the LDR sensor and display the results every five seconds. Thanks to the cloud computing service and real-time cloud database provided by the Blynk platform, the LDR can gather the home's light intensity readings and display them visually on the iHOCS application's interface. Figure 8(a) shows the screen's reduced graphical sensor readings, whereas Figure 8(b) shows the full-screen graphical representation of the measurements.

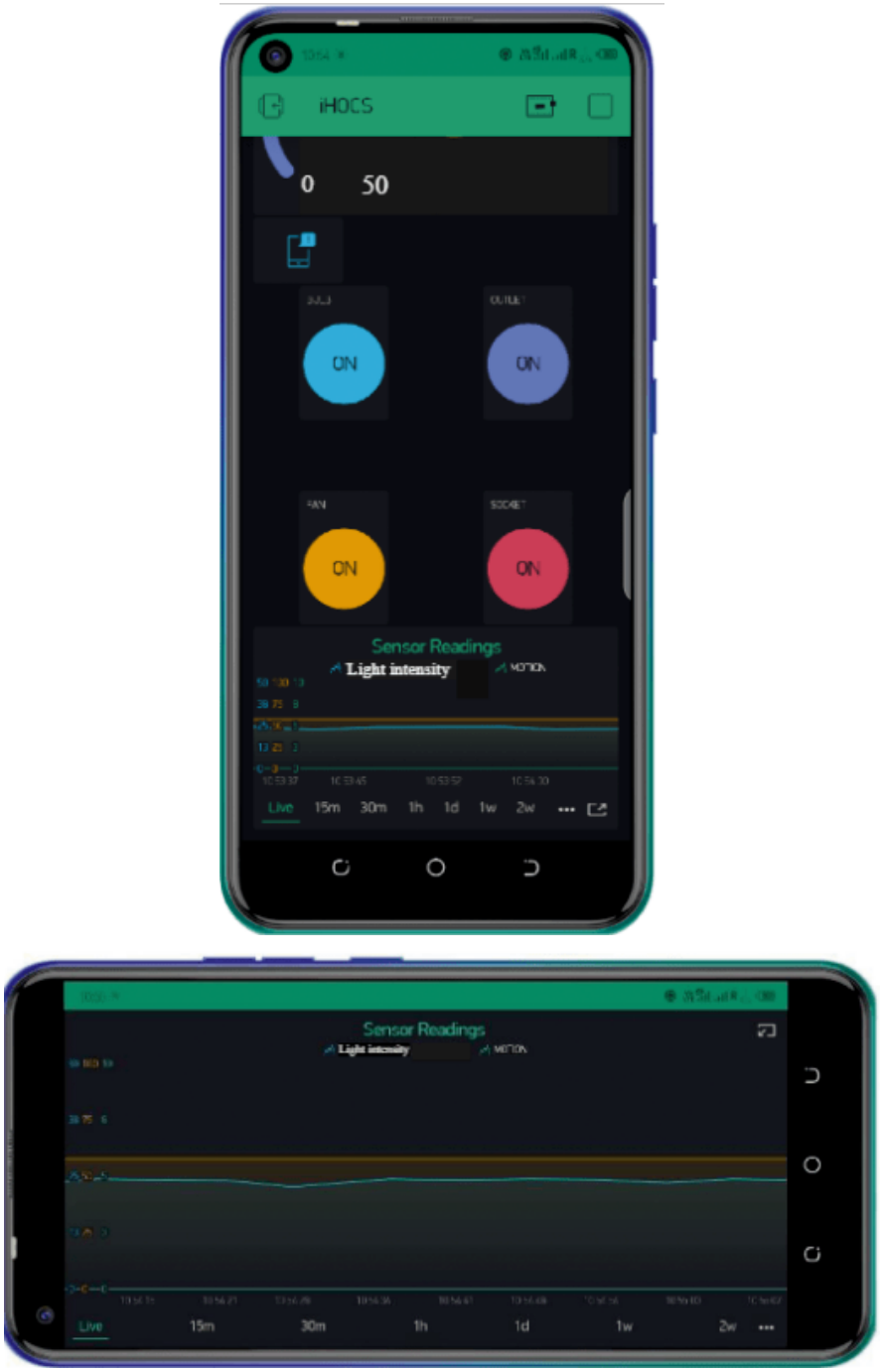


Figure 8

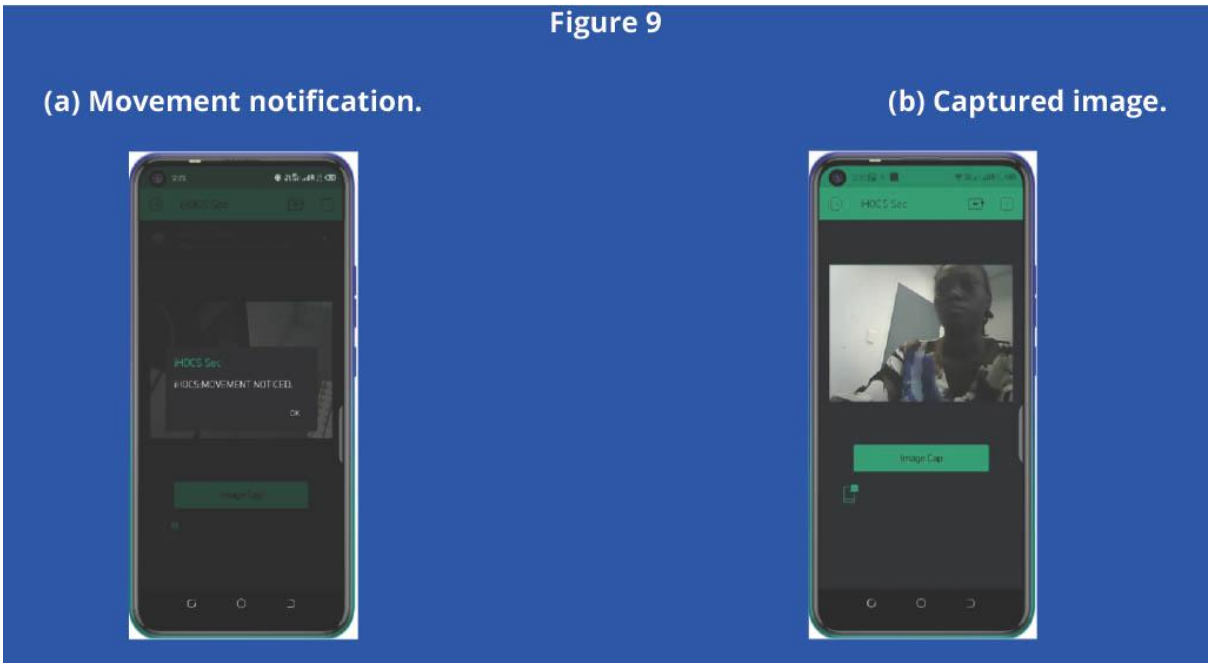
(a) In-app graphical readings. (b) Full screen of graphical readings.

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 temperature and humidity values. The motion sensor detects movements and delivers a notice

to the user via the mobile application (Figure 9(a)). ; Along with the temperature and humidity readings, a real-time graphical representation of the motion sensor data is also shown (Figures 8(a) and 8(b)). Figure 9(a) shows the message that the user received when a 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. Even in the dark, the ESP2-CAM can take a picture because of its built-in light.

Figure 9

(a) Movement notification. **(b) Captured image.**



The recommended system performs a number of energy-saving and home management functions. The environment's intelligent response to the light switch-on is shown in Figure 10. By prompting the user to turn on the light when it is dark or turn it off when it is bright, the device promotes energy saving. This enhances the home's energy management and conservation.

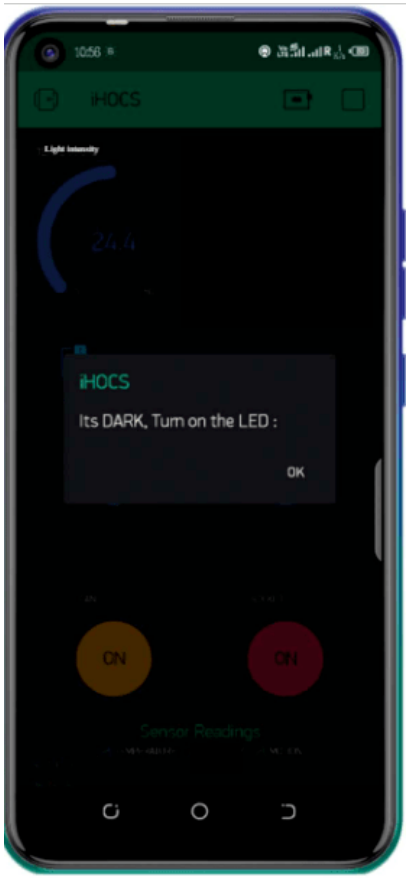


Figure 10 : Intelligent home response.

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 (temperature and humidity)
(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 technology also provides ease by enabling remote control of the home from any place, as is highlighted with smart home automation systems. As a result, the house may be observed and managed remotely from any place. For safety, it is also necessary to monitor the surroundings close to the house. For instance, if an elderly, disabled, or young person is left alone at home, the user may check the amount of light in the house from a distance and help the resident turn on or off the lights. This encourages a healthy lifestyle.

The security of the house and the suggested SVM algorithm for user notifications are additional benefits of iHOCS. False warnings and pointless notifications might be avoided with the use of an SVM algorithm. The user will only be informed when the system identifies a picture as one of an intruder, even if all images sensed by the system will still be recorded for later use.

The intelligent home automation system described in this study is based on IoT technology, cloud computing, and a machine learning algorithm. An Android-based mobile application that is part of the home automation system enables both remote and in-person management of the house. The system manages electrical household equipment, keeps an eye on the outside world with the help of temperature, humidity, and light sensors, and keeps the house secure with the help of a motion sensor and an IoT camera. Intelligent judgments are made by the system to automatically turn ON or OFF lights and to allow the user to view and choose whether to save a photo of a person taken by a camera.

On the Blynk app, for gas detection:

Lab 17 MQ-2 Gas Sensor with Blynk App and Nodemcu ESP8266.

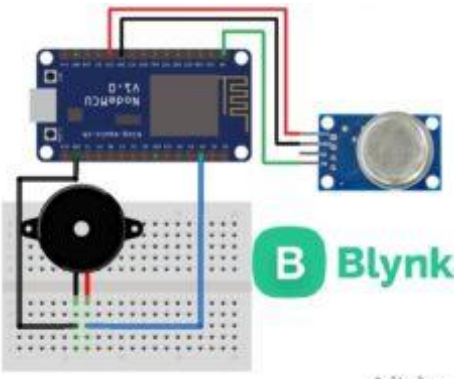
The Gas Sensor (MQ2) module is helpful for detecting gas leaks (home and industry). The H2, LPG, CH4, CO, alcohol, smoke, or propane may all be detected using this device. Measurements may be made as quickly as feasible because to its high sensitivity and quick reaction time. A potentiometer can be used to modify the sensor's sensitivity. The sensor is really protected by an Anti-explosion network, which is two layers of tiny stainless steel mesh. As we are sensing combustible gases, it assures that the heating element inside the sensor won't result in an explosion.

Additionally, it shields the sensor from damage and removes suspended particles, allowing only gaseous components to pass through the chamber. A copper-plated clamping ring secures the mesh to the remainder of the body.

Required Elements:

one NodeMCU, a single MQ-2 gas sensor, one buzzer, Wire jumpers and Breadboard.

Connections:

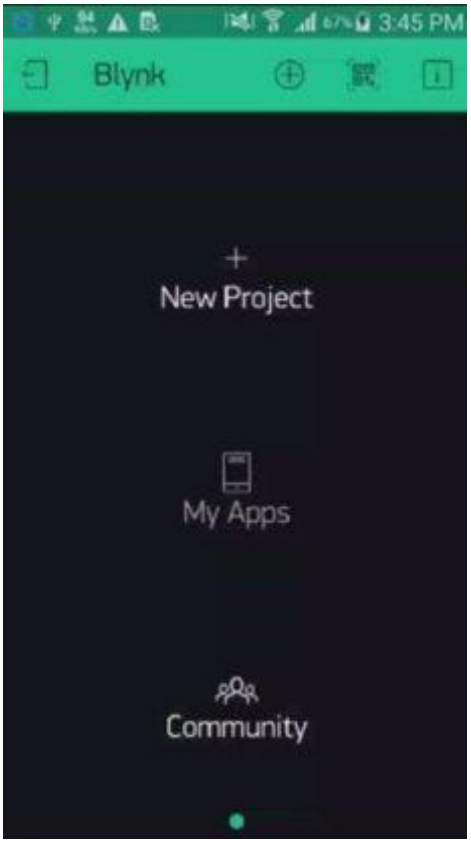


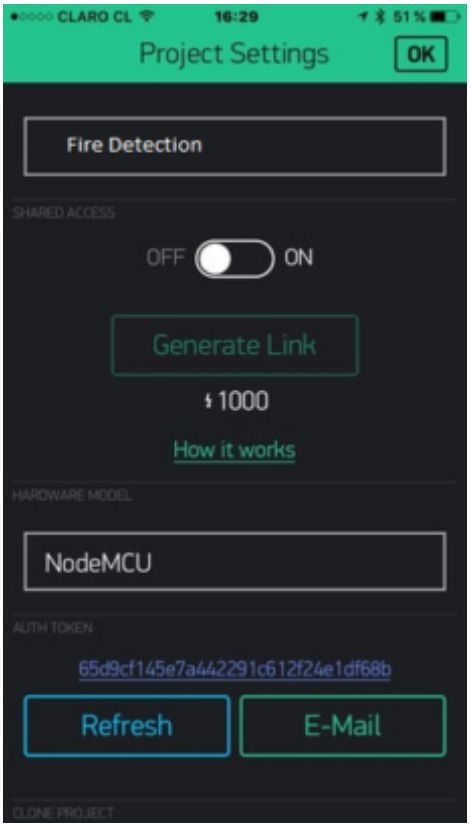
- The 3V pin of NodeMCU is linked to the VCC pin of MQ-2.
- The GND pins of MQ-2 and NodeMCU are linked together.
- The MQ-2's A0 pin is connected to the NodeMCU's A0 pin.
- The D2 pin of the NodeMCU is linked to the positive (+) pin of the Buzzer.
- Buzzer's GND (-) pin is connected to NodeMCU's GND pin.

Create a project using the Blynk applications after setting up the app. obeying a command. Log in to your Blynk app first, then follow the steps.

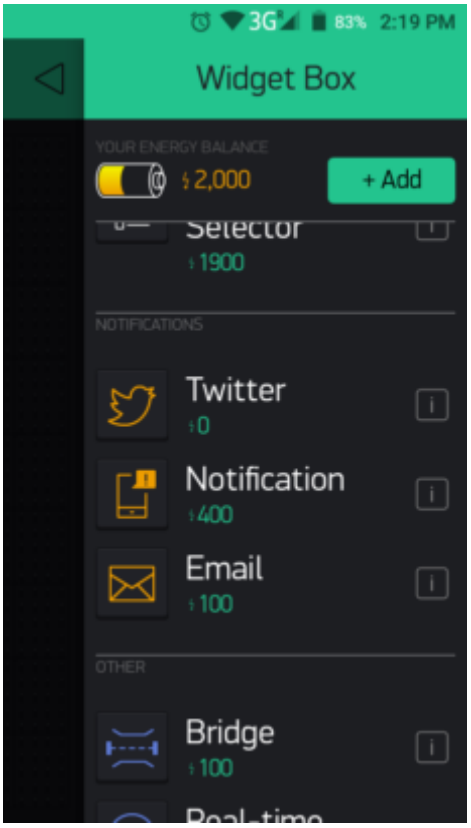
Create A New Project in Step 1.

Step2- Select Your project name and Board then Press Create.
Step3 -This is a very important step Hear you get your AUTH TOKEN it send to your email address. Press OK.
Step4-Now go to Widget Box And select Notification.
Step5-Now go to notification setting and apply the following settings.
Step6- now your screen will look like that.

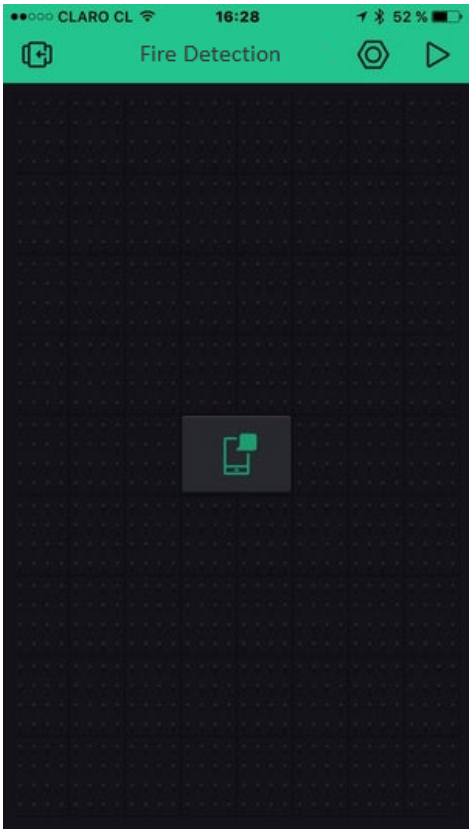



CLARO CL
16:29
51 %
Project Settings
OK
Fire Detection
SHARED ACCESS
OFF
ON
Generate Link
1000
How it works
HARDWARE MODEL
NodeMCU
AUTH TOKEN
65d9cf145e7a442291c612f24e1df68b
Refresh
E-Mail
CLONE PROJECT


3:53 PM
DHT Monito
Auth Token was sent to:
sarfulHassan@Gmail.com
You can also find it in Project Settings
OK
Don't show again


83% 2:19 PM
Widget Box
YOUR ENERGY BALANCE
2,000
+ Add
1900
NOTIFICATIONS
Twitter
0
Notification
400
Email
100
OTHER
Bridge
100


CLARO CL
16:29
52 %
Notifications Settings
OK
NOTIFY WHEN HARDWARE GOES OFFLINE
OFF
ON
PRIORITY
LOW
HIGH
Delete
+ 400



Arduino Code:
Upload the following code into the NodeMCU:

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
// You should get Auth Token in the Blynk App.
char auth[] = "Your auth token";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "your wifi SSID";
char pass[] = "your password";
int buzzer = D2;
int smokeA0 = A0;
// Your threshold value. You might need to change it.
int sensorThres = 600;
void setup() {
 pinMode(buzzer, OUTPUT);
 pinMode(smokeA0, INPUT);
 Serial.begin(9600);
}
void loop() {
 int analogSensor = analogRead(smokeA0);
```

```
 Serial.print("Pin A0: ");
 Serial.println(analogSensor);
 // Checks if it has reached the threshold value
 if (analogSensor > sensorThres)
 {
   tone(buzzer, 1000, 200);
   Blynk.notify("Alert: Fire in the House");
 }
 else
 {
   noTone(buzzer);
 }
 delay(100);
}
```

Code Definition:

The serial port that will be utilized for the Blynk Debug later in the background is first defined by the BLYNK PRINT Serial. All the functionalities required to run the NodeMCU over the internet will then be used by the header file. The header is then included to allow the Blynk app to communicate over wifi with the NodeMCU (Esp8266).

The authentication token is then included into the code. This step is necessary for security so that it can be confirmed that the hardware linked to the PC is speaking to the right Blynk Project and its user. After that, a connection between the hardware and the Blynk app via your network must be established using your network credentials (SSID and password).

The buzzer and smokeA0 pins of the NodeMCU, which are linked to the NodeMCU's pin D2 and A0, respectively, respectively, respectively, respectively, by the Positive (+) and A0 pins of the MQ-2, are then defined.

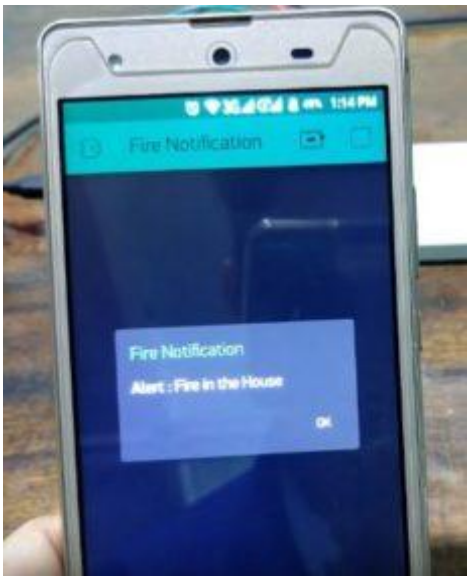
The sensor threshold value, "sensorThres," is then specified. The buzzer or alert sounds if the sensor's output value, which is related to how much gas is present nearby, surpasses this value. This has to be set with the sensor's output value in a typical atmosphere in mind.

The buzzer pin and smokeA0 pin's pin modes are then set to output and input, respectively, in the void setup(). In addition, a baud rate of 9600 bits per second is initialized for the serial communication between the PC and the NodeMCU. Additionally, the Blynk app and node mcu initiate communication using blynk.begin. The value analogously detected from the MQ-2 (A0) will be kept in the integer variable analogSensor created in the void loop() function. The Serial monitor will then display this value.

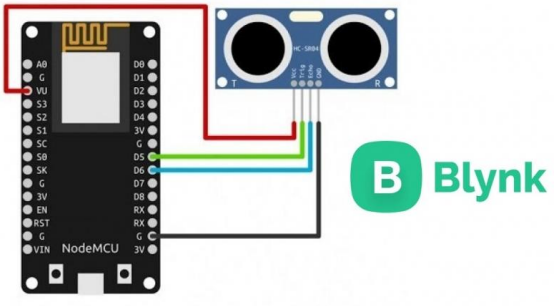
The buzzer will sound if the analogSensor value read from the MQ-2 is larger than the threshold "sensorthres," according to an if statement that follows. The Pin number, Frequency, which is measured in hertz (cycles per second) and sets the pitch of the noise emitted, and Duration, which controls how long the tone plays, are the three arguments

required by the tone function that controls the buzzer. The Blynk app is used to inform the user of messages under these circumstances as well.

Result:
When a combustible gas (or a gas from the list) is detected by the MQ-2 after the code has been uploaded and the connections have been created, the buzzer ought to sound. Additionally, the Blynk app notifies the user.



Ultrasonic Sensor Interface Nodemcu ESP8266-Lab 14 in Blynk App.



Introduction: We will study the HC-SR04, also known as an ultrasonic sensor, in this section. We will also learn how to connect it to the NodeMCU. Additionally, how to use the HC-SR04 to measure distance.

An ultrasonic sensor is a gadget that uses sound waves to gauge an object's distance. It uses a sound wave at a specified frequency to measure distance before waiting for the sound wave to return. The distance between the sensor and the item may be determined by keeping track of how long it takes for the sound wave to produce and bounce back.

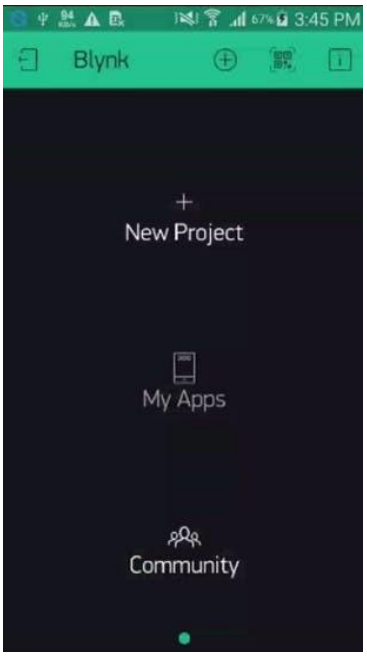
Required parts include a NodeMCU, an HC-SR04 ultrasonic sensor, a USB cable, and jumper wires.

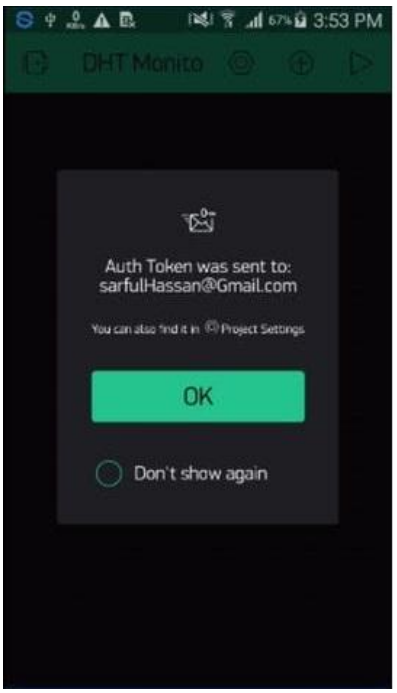
Connections: The Ultrasonic sensor's VCC pin is linked to the NodeMCU's VU pin, and its GND pin is connected to the NodeMCU's GND pin.

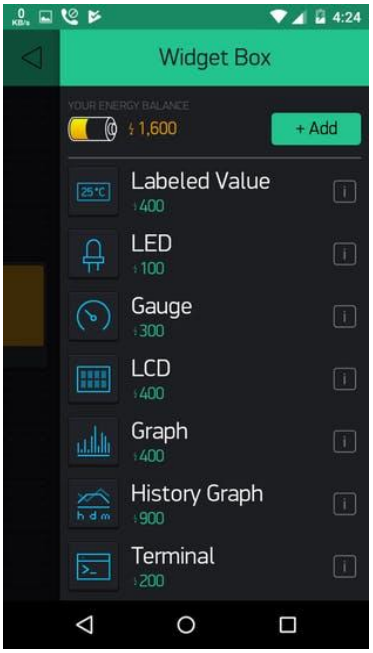
The NodeMCU's D5 pin is used for the Ultrasonic sensor's TRIG pin, while its D6 pin is used for the Ultrasonic sensor's ECHO pin.

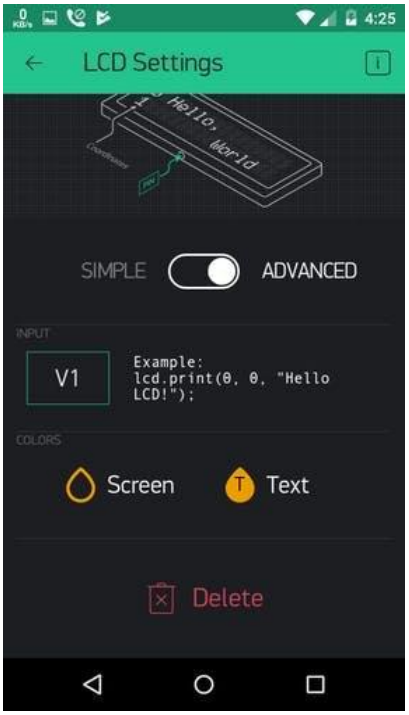
Configuring the Blynk App:
With Blynk applications, build a project while according to instructions
Log in to your Blynk app first, then follow the steps.

Create A New Project in Step 1. Step 2: Choose your project's name and board (I chose distance finder as my project name and NodeMCU as my board), then click Create. The third and most crucial stage is where we obtain your AUTH TOKEN, which is sent to your email address. Step 4- Go to the Widget Box and choose LCD. Step 5- Go to the LCD Settings and alter the input variable to "V1" and the colors to "Text".




3:53 PM
DHT Monito
Auth Token was sent to:
sarfulHassan@Gmail.com
You can also find it in Project Settings
OK
Don't show again


4:24
Widget Box
YOUR ENERGY BALANCE
1,600
+ Add
Labeled Value
400
LED
100
Gauge
300
LCD
400
Graph
400
History Graph
900
Terminal
200



Arduino Code:

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
// You should get Auth Token in the Blynk App.
char auth[] = "Your auth token";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "your wifi SSID";
char pass[] = "your password";

WidgetLCD lcd(V1);
#define TRIGGERPIN D5
#define ECHOPIN  D6
void setup()
{
 // Debug console
 Serial.begin(9600);
pinMode(TRIGGERPIN, OUTPUT);
  pinMode(ECHOPIN, INPUT);
  Blynk.begin(auth, ssid, pass);
   lcd.clear(); //Use it to clear the LCD Widget
 lcd.print(0, 0, "Distance in cm");
}
void loop()
{
```

```
 long duration, distance;
 digitalWrite(TRIGGERPIN, LOW);
 delayMicroseconds(2);
   digitalWrite(TRIGGERPIN, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIGGERPIN, LOW);
 duration = pulseIn(ECHOPIN, HIGH);
 distance = (duration/2) / 29.1;
 Serial.print(distance);
 Serial.println("Cm");
 lcd.clear();
 lcd.print(0, 0, "Distance in cm"); // use: (position X: 0-15, position Y: 0-1, "Message you want to print")
 lcd.print(7, 1, distance);
 Blynk.run();
 delay(1000);
}
```

Code Definition.

The serial port that will be utilized for the Blynk Debug later in the background is first defined by the BLYNK PRINT Serial. The NodeMCU may then be operated via the internet by using the functionalities provided by the header file "ESP8266Wifi." Then, for the Blynk app to function with the NodeMCU (Esp8266) through WiFi, the header "BlynkSimpleEsp8266" is inserted.

It is next necessary to establish security by inserting the authentication token into the code. This ensures that the hardware linked to the PC is speaking to the right Blynk Project and its user. After that, a connection between the hardware and the Blynk app via your network must be established using your network credentials (SSID and password).

WidgetLCD lcd(v1) is used to define an lcd widget in the blink app and define it to variable v1. The NodeMCU's D5 and D6 pins are then specified as constants with integer data types, trigPin, and echoPin, respectively.

The pin modes for trigPin and echoPin are configured in the void setup() method. Additionally, the serial communication between the PC and NodeMCU is initialized at a baud rate of 9600 bits per second. To initiate communication with the Blynk app, use Blynk.begin.
The trigPin is initially cleared in the void loop() method by being set to LOW for two milliseconds, then HIGH for ten milliseconds to broadcast a ten millisecond sound wave trigger pulse, and finally back to LOW.

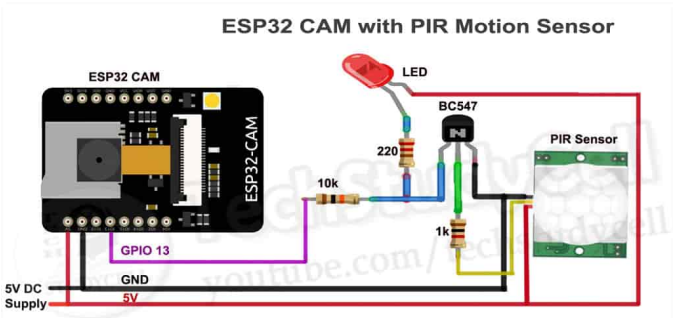
The pulse calculates the amount of time needed to strike an item and then return. This time is recorded in the duration variables of the function that calculates how long it took for the echoPin to receive a HIGH following the transmission of the sound wave trigger pulse.
After clearing all of the LCD's displayed graphics with the lcd.clear() command and printing our message at position (0,0) on the LCD using the lcd.prnt(0,0,"Distance in cm") command, Then, to show our measured distance on an LCD, we use lcd.print(7,1,distance) (7,1). The Blynk app and the NodeMCU are in constant contact thanks to the Blynk.run()> method.

Result
The centimeters between the ultrasonic sensor and the item in front of it would be displayed on the monitor and the Blynk LCD widget.

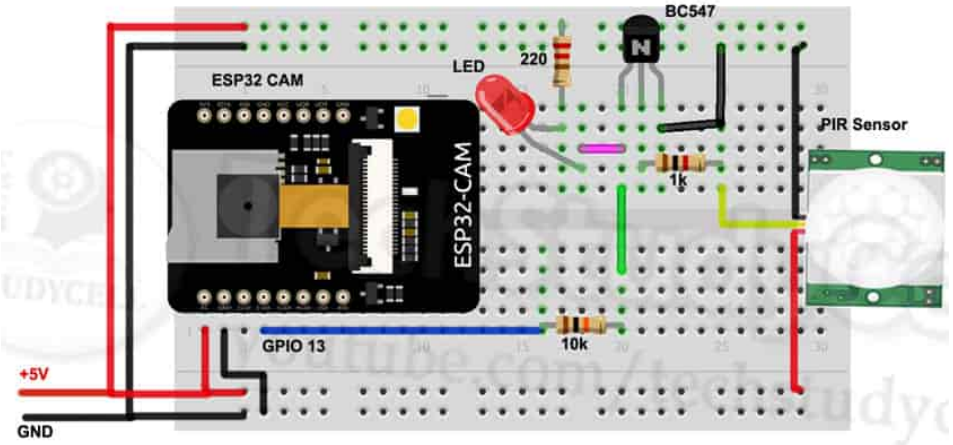
Blynk Security Camera with Motion Sensor and Notification by ESP32CAM.

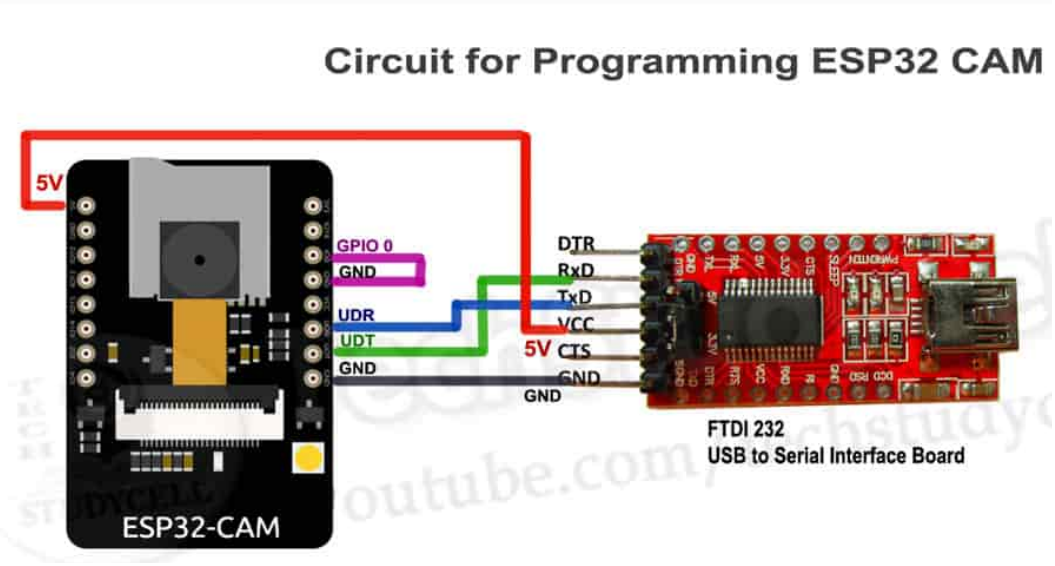
With the help of an ESP32 CAM, a PIR motion sensor, and the Blynk app, I created a DIY home surveillance system for this project. The ESP32-CAM Motion Sensor Security Camera will send a notice to a smartphone with a photo whenever any motion is detected by the PIR sensor. A basic circuit is used. Therefore, with the ESP32 CAM board and PIR motion sensor, you can create this motion sensor security camera with ease.

ESP32-CAM (AI Thinker), PIR Motion Sensor Module, BC547 NPN Transistor, 220 ohm, 1 k, and 10 k Resistor, LED, FTDI 232 USB to Serial Interface board, and 5 volt DC supply are the components needed for this ESP32CAM Blynk project.



Schematic for the breadboard used in this ESP32CAM Blynk project.





This is the ESP32-CAM motion sensor camera's breadboard schematic. This will assist you in designing the breadboard-based ESP32CAM PIR motion sensor camera circuit.
I used an FTDI232 USB to Serial interface card to program the ESP32CAM. According to the aforementioned circuit, I connected the ESP32CAM and FTDI232.
We must connect GPIO 0 to the ESP32CAM's GND pin before uploading the code.

Please verify the following configuration before uploading the code to the ESP32CAM:
Refresh your preferences -> Additional boards Manager URLs: http://arduino.esp8266.com/stable/package esp8266com index.json and https://dl.espressif.com/dl/package esp32 index.json
Setting the Board:
"ESP32 Wrover Module" board.
"921600" for upload speed
"80MHz" is the flash frequency.
"QIO" Flash Mode.
"Hue APP (3MB No OTA/1MB SPIFFS)" is the partition scheme.
"None" is the core debug level.
COM Depends on Your System, Port
When uploading the sketch, GPIO 0 has to be linked to the GND pin.
Press the ESP32 CAM on-board after attaching GPIO 0 to the GND pin.
Press the RESET button to start the board flashing.

This ESP32CAM Blynk project has the Blynk App configured.

How to install the Blynk App:
the Blynk App and launch the project.
Select the "+" button at the top.
From the Widget Box, choose the Image Gallery Widget.
(Pin-V1 setting) (Function: Display Image.)
From the Widget Box, pick the Styled Button.
(Setting: GP14 Pin, PUSH Mode.)
From the Widget Box, choose Notification.
(Function: get the alert.)

I tried the circuit after setting up the Blynk app.Connect your smartphone to the same WiFi network as this security camera circuit by providing 5V DC to it. You should now receive a

notice on your phone if the PIR sensor picks up any motion. Then, to take the photo, click the "Take Picture" button. Due to the ESP32-CAM's built-in LED's ability to provide enough light, the camera can also snap photographs in complete darkness.

The code : // Select camera model

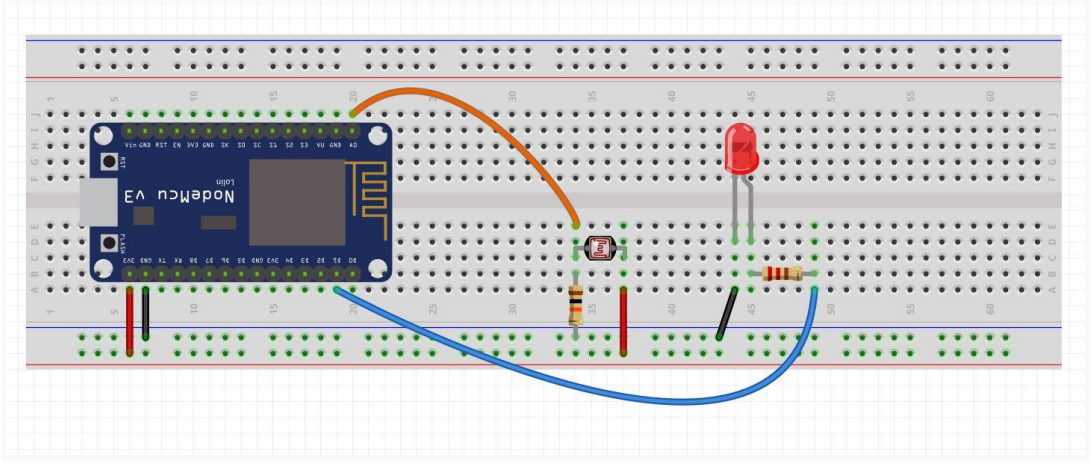
```
#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

#define PIR 13
#define PHOTO 14
#define LED 4

const char* ssid = "WIFI NAME";
const char* password = "WIFI PASSWORD";
char auth[] = "AUTH TOKEN";  //sent by Blynk

String local_IP;

void startCameraServer();

void takePhoto()
{
  digitalWrite(LED, HIGH);
  delay(200);
  uint32_t randomNum = random(50000);
  Serial.println("http://"+local_IP+"/capture?_cb="+ (String)randomNum);
  Blynk.setProperty(V1, "urls", "http://"+local_IP+"/capture?_cb="+(String)randomNum);
  digitalWrite(LED, LOW);
  delay(1000);
}

void setup() {
  Serial.begin(115200);
  pinMode(LED,OUTPUT);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
```

```
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality
//                     for larger pre-allocated frame buffer.
if(psramFound()){
  config.frame_size = FRAMESIZE_UXGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
  Serial.printf("Camera init failed with error 0x%x", err);
  return;
}

sensor_t * s = esp_camera_sensor_get();
// initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
  s->set_vflip(s, 1); // flip it back
  s->set_brightness(s, 1); // up the brightness just a bit
  s->set_saturation(s, -2); // lower the saturation
}
// drop down frame size for higher initial frame rate
s->set_framesize(s, FRAMESIZE_QVGA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  local_IP = WiFi.localIP().toString();
  Serial.println("' to connect");
  Blynk.begin(auth, ssid, password);
}

void loop() {
  // put your main code here, to run repeatedly:
  Blynk.run();
  if(digitalRead(PIR) == LOW){
  Serial.println("Send Notification");
  Blynk.notify("Intruder Detected...");
  Serial.println("Capture Photo");
  takePhoto();
  delay(3000);
  }
  if(digitalRead(PHOTO) == HIGH){
  Serial.println("Capture Photo");
  takePhoto();
  }
}
```

Using IOT Blynk app and coding for it to measure light intensity levels of LDR and NodeMCU:

Schematic:

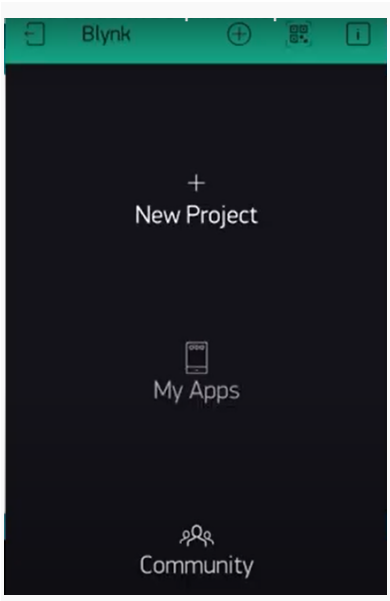


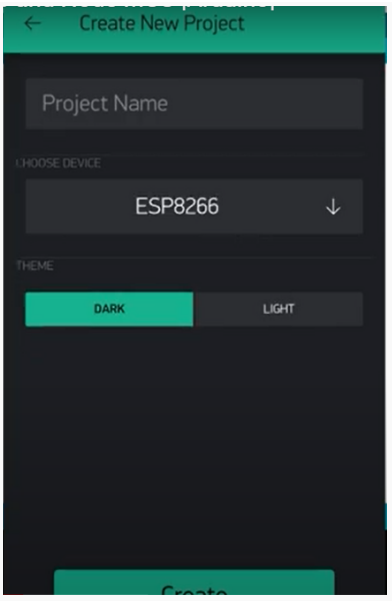
Code for using the blynk app:

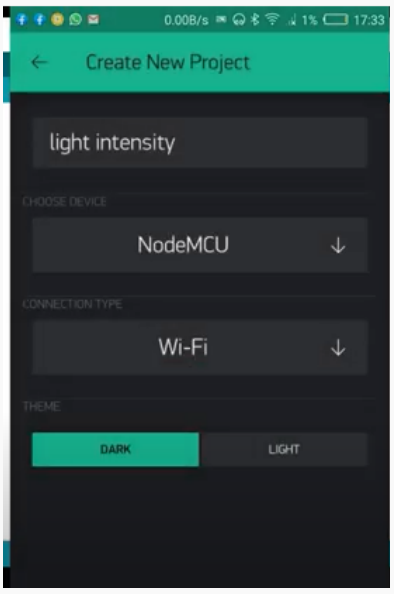
```
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#define led D1
#define sensor A0
char auth[] = "PF1zb_GCKcm6uL713uhw99U-iG7eVTmU";
char ssid[] = "voidloop";
char pass[] = "shivam1234";

BlynkTimer timer;


void setup()
{
  // Debug console
  pinMode(led, OUTPUT);
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
```

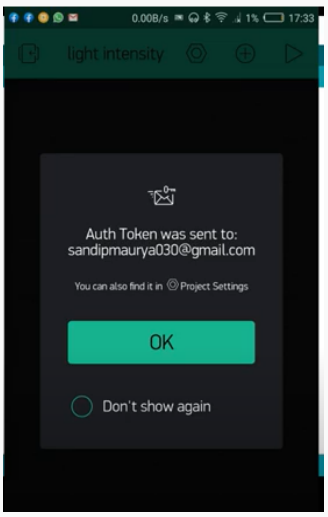
```
void setup()
{
  // Debug console
  pinMode(led, OUTPUT);
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(500L, sendSensor);

}

void loop()
{
  Blynk.run();
  timer.run();
}

void sendSensor()
{
  int LDR = analogRead(sensor);
    if(LDR <150)
  {
```

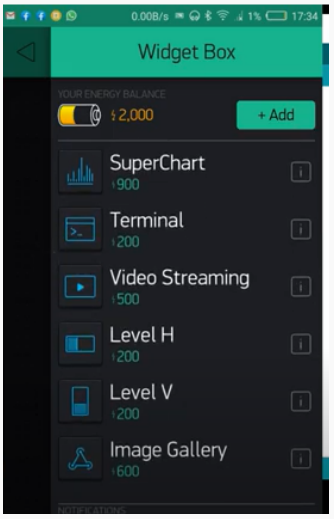
```
}

void sendSensor()
{
  int LDR = analogRead(sensor);
    if(LDR <150)
  {
    digitalWrite(led, HIGH);
    Blynk.notify("Light ON");
  }
  else
  {
    digitalWrite(led, LOW);

  }

  Blynk.virtualWrite(V5, LDR);

}
```


Blynk
New Project
My Apps
Community

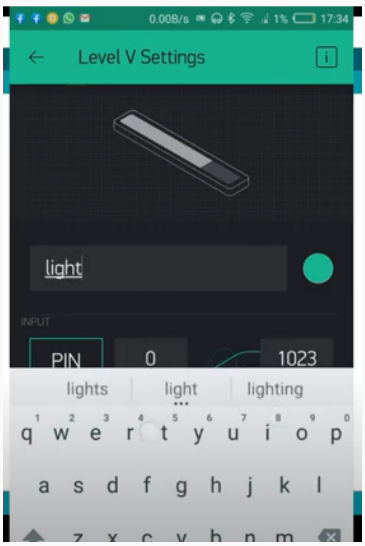

Create New Project
Project Name
CHOOSE DEVICE
ESP8266
THEME
DARK
LIGHT

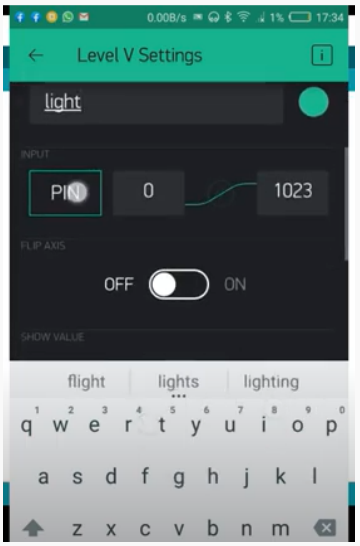

Create New Project
light intensity
CHOOSE DEVICE
NodeMCU
CONNECTION TYPE
Wi-Fi
THEME
DARK
LIGHT

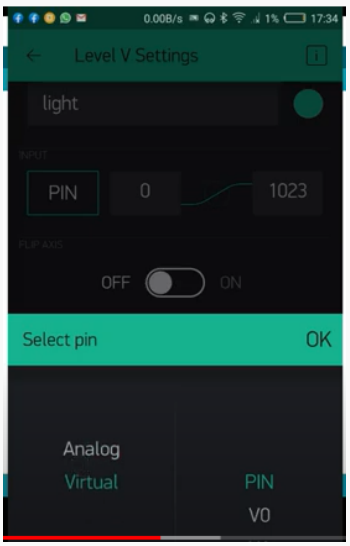

light intensity
Auth Token was sent to:
sandipmaurya030@gmail.com
You can also find it in Project Settings
OK
Don't show again

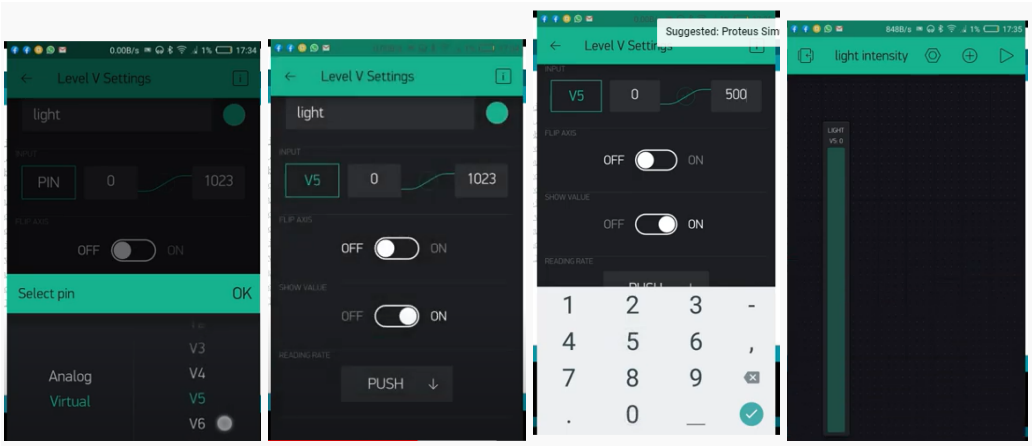

Widget Box
2,000
+ Add
SuperChart
900
Terminal
200
Video Streaming
500
Level H
200
Level V
200
Image Gallery
600

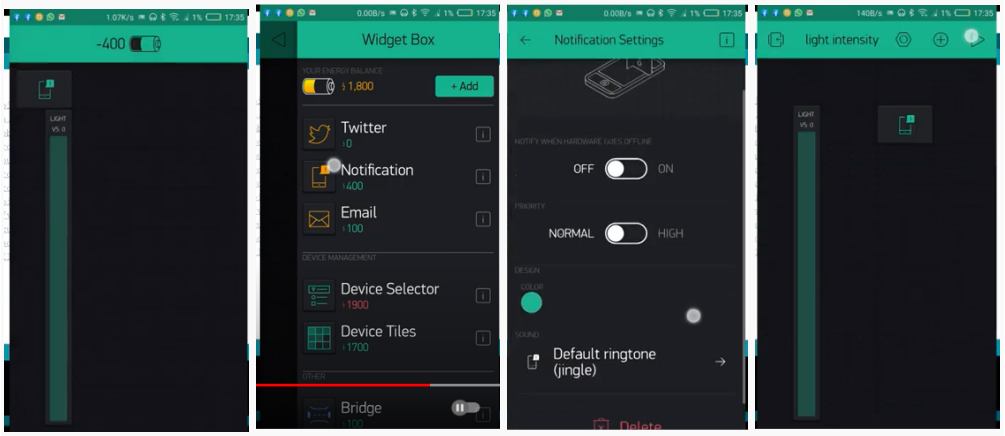

light intensity

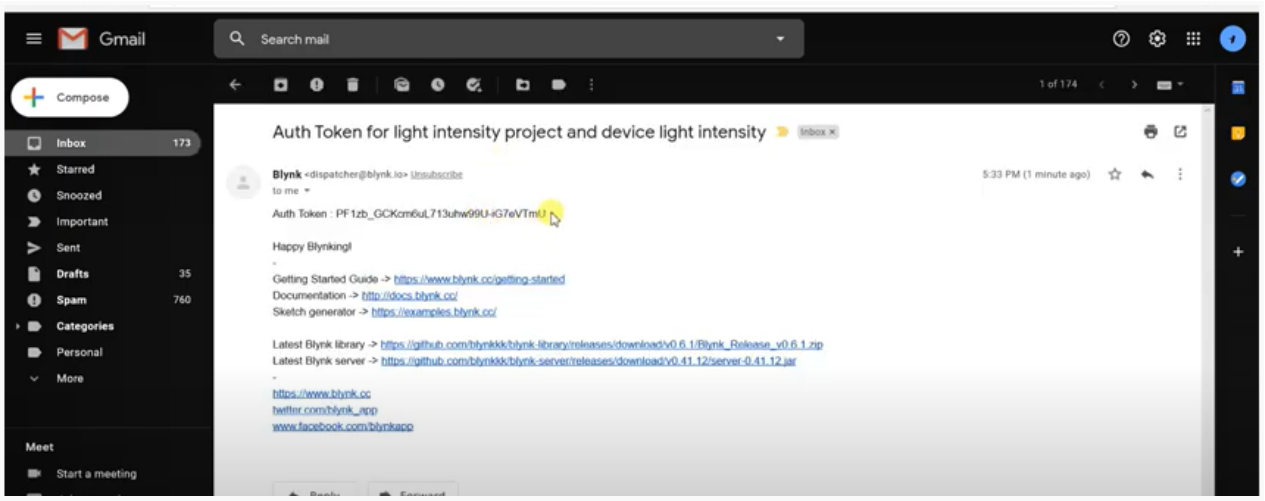

Level V Settings
light
INPUT
PIN
0
1023

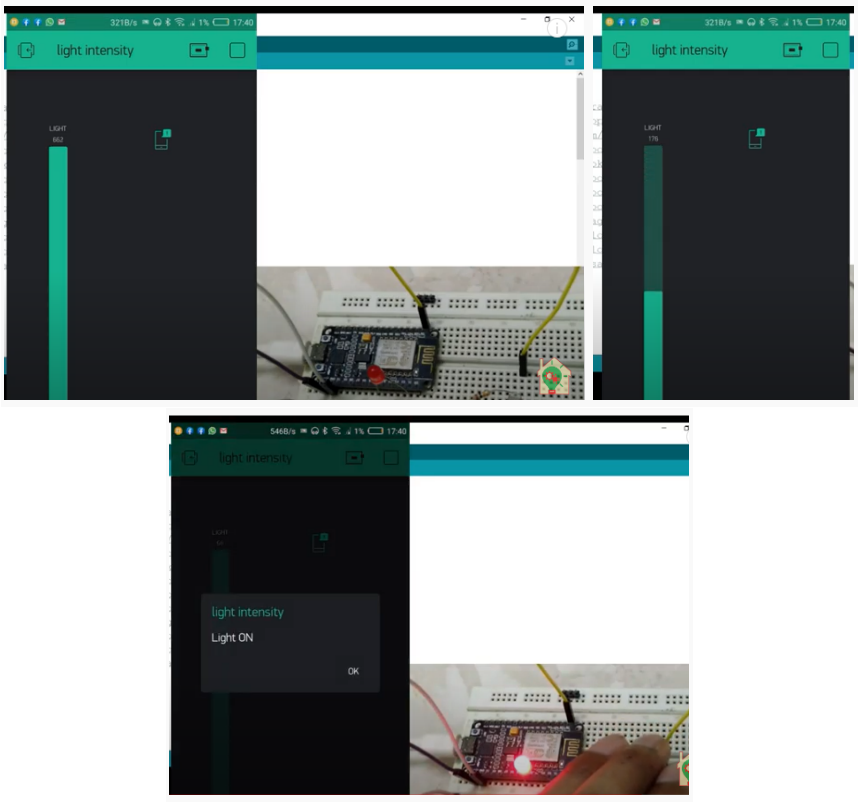

Level V Settings
light
INPUT
PIN
0
1023
FLIP AXIS
OFF
ON
SHOW VALUE


Level V Settings
light
INPUT
PIN
0
1023
FLIP AXIS
OFF
ON
Select pin
OK
Analog
Virtual
PIN
V0


Level V Settings
light
PIN
0
1023
OFF
ON
Select pin
OK
V3
V4
V5
V6
Analog
Virtual
Level V Settings
light
V5
0
1023
OFF
ON
OFF
ON
OFF
ON
PUSH
Suggested: Proteus Sim
Level V Settings
V5
0
500
OFF
ON
OFF
ON
1
2
3
-
4
5
6
,
7
8
9
.
0
light intensity


-400
Widget Box
+ Add
Twitter
Notification
Email
Device Selector
Device Tiles
Bridge
Notification Settings
OFF
ON
NORMAL
HIGH
Default ringtone
(jingle)
Delete
light intensity


Gmail
Search mail
Compose
Inbox 173
Starred
Snoozed
Important
Sent
Drafts 35
Spam 760
Categories
Personal
More
Meet
Start a meeting
1 of 174
Auth Token for light intensity project and device light intensity
Inbox
Blynk <dispatcher@blynk.io> Unsubscribe
to me
5:33 PM (1 minute ago)
Auth Token : PF1zb_GCKcm6uL713uhw99U-tG7eVTmU
Happy Blynking!
Getting Started Guide -> https://www.blynk.cc/getting-started
Documentation -> http://docs.blynk.cc/
Sketch generator -> https://examples.blynk.cc/
Latest Blynk library -> https://github.com/blynkkk/blynk-library/releases/download/v0.6.1/Blynk_Release_v0.6.1.zip
Latest Blynk server -> https://github.com/blynkkk/blynk-server/releases/download/v0.41.12/server-0.41.12.jar
https://www.blynk.cc
twitter.com/blynk_app
www.facebook.com/blynkapp



Info on coding and the result.

LDR light intensity sensor, as shown. We started a brand-new undertaking called light intensity. On a node pin, it is connected. It is a node MCU analog pin. Led is also attached to D's first pin. It will notify NodeMCU and the LED when the light intensity is too low by sending light intensity data to them. See some programming for the Arduino. I've already developed Arduino code. There is a blink library, and a sensor and a led are linked to an odd number on D. I entered the SSID and password in the code. A blink timer is included. Every half-second, it will call the transmit sensor function. Here, the sensor is being sent. The LDR variable exists. It is kept in a sensor read the log.

LEDs will illuminate if LDR is less than 150. Led will be switched off if not. A notice will be sent to the blink app if the led glows. And the blink app will receive their value via virtual number v5. Click build after selecting the hardware "Node MCU" for a wifi connection. The email has received an authentication token. Next, we'll choose component Level V from the widget box and enlarge it. Then change the title to "light" and choose "v5" for the pin. The v5 Leverage is reported as 500. We return to widget Bix to take the notice and adjust its location as indicated by the output screenshots.

We then press "play." I then receive the login code when I check my e-mail. The token number should be copied from the email and then pasted into char auth as necessary. I'll then merely make sure the board is in good shape. I returned to Arduino and chose the Arduino Nano board to pick the Node MCU 1.0 port, which is COM5. Okay, we'll compile, upload, and push the button to complete the upload. The board will be restricted. Display light is around 656 to 646, or lower, as you can see. If we covered the sensor, we could see the level

being downloaded. The led is glowing if it is totally covered. Here, the notification light turns on. You can see the level is high if I remove it. I'll cover this once more. You can tell whether the light is on because it will glow. Therefore, the function is proven to be true that the light bulb/led turns on as the light intensity of the LDR sensor drops here, and vice versa.

Summary

Issues in implementing the ideas into the project using Tinkercad

Tinkercad's official team has chosen to discontinue the ESP8266 module from Tinkercad Circuits because certain necessary security upgrades have significantly changed how the module connects to the internet. As a result, without an ESP8266 module, we were unable to link Tinkercad to the Blynk app. However, we provided an explanation of how the applications would be implemented, coded, and produced if we utilized the necessary elements in a way that was proved to be accurate when in practical practice using data from reliable research studies as sources.

Comments on the usage of the virtual platform.

It has highlighted the difficulties users have during the simulation process and the virtual platform for simulating automated operations. The simulation results are used in this study three to illustrate home automation tasks. In a security system, fan and light bulb controls are actuated by an IR remote, and servo movement is controlled by a secret PIN entered into the input keypad. Second, automated parking indicates the number of vehicles and available space in the garage. The block idea used in the virtual platform allows for easy programming and tremendous connectivity flexibility. The third application includes house interior automation features including temperature-based fan control, PIR motion-based LED on-off, ambient intensity adjustment of a lightbulb, and ultrasonic door opening.

Further studies:

Additionally, we intend to improve the system such that the user only sees captured photographs of an intruder. At the same time, other acquired images are saved in real-time to the system's cloud-based database. This will lift the restriction on the number of photographs that may be loaded into the mobile application. In order to improve the PIR sensor's ability to distinguish between an animal, the occupant, and an intruder, machine learning will also be applied. As a result, the user will receive fewer notifications from the PIR sensor. The machine learning method will also be used to categorize discovered photographs so that it can deal with a bigger data collection of invaders and several occupants.

Also, we plan to enhance the system such that the user will only receive captured images of an intruder, and other captured images will be saved in real-time into the cloud-based database of the system. This will remove the limitation on the encumbrance of a load of images on the mobile application. In addition, machine learning will also be used to enhance

the PIR sensor's performance to differentiate between an animal, the occupant, and an intruder. This will reduce the notification from the PIR sensor to the user. The machine learning algorithm will also be applied to classify detected images to work with several occupants and a larger data set of intruders.

In the future, we want to widen our horizons by incorporating machine learning into a mobile application that would categorize photographs taken by the camera and inform the user of the precise nature of the object that was caught. Additionally, we want to provide iOS compatibility to the mobile app. The method described in this paper may also be used to improve the security systems in big communities, such as smart cities, office buildings, hotels, malls, and academic environments. Prediction is said to be made simple using machine learning. To forecast weather and house conditions, machine learning may also be used in the environmental module of smart home automation.

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-switch-using-nodemcu-and-the-blynk-app/setup-blynk-on-your-smartphone
2. https://www.instructables.com/Servo-Motor-Controlled-With-BLYNK-Over-WiFi/
3. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-switch-using-nodemcu-and-the-blynk-app/the-code
4. https://easyelectronicsproject.com/esp32-projects/esp32cam-blynk-security-camera/
5. https://create.arduino.cc/projecthub/nolan-mathews/connect-to-blynk-using-esp8266-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-blynk-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_using_IOT__A_case_study_to_benefit_people_with_motion_disability__ijariie15836.pdf
12. https://www.youtube.com/watch?v=ej06oRWQ9j4&t=3s
13. https://blynk.hackster.io/KaustubhAgarwal/smart-parking-bdfa99
14. https://mytectutor.com/smart-car-parking-system-using-arduino-esp8266-nodemcu-and-blynk-server/
15. https://microcontrollerslab.com/control-esp8266-nodemcu-outputs-blynk-app-arduino-ide/
16. https://iopscience.iop.org/article/10.1088/1742-6596/2117/1/012021/pdf
17. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-switch-using-nodemcu-and-the-blynk-app/setup-blynk-on-your-smartphone
18. https://www.instructables.com/Servo-Motor-Controlled-With-BLYNK-Over-WiFi/
19. https://reacoda.gitbook.io/molemi-iot/introducing-the-nodemcu/a-light-bulb-switch-using-nodemcu-and-the-blynk-app/the-code
20. https://www.researchgate.net/publication/357748641_The_HOME_AUTOMATION_USING_IOT?enrichId=rgreq-e7589b112ee65e0cd3eab7eb213fa4dd-XXX&enrichSource=Y292ZXJQYWdlOzM1Nzc0ODY0MTtBUzoxMTExMDg0MTUyOTYzMDcyQDE2NDE5MTQ1MTM1MDA%3D&el=1_x_2&_esc=publicationCoverPdf
21. https://ijariie.com/AdminUploadPdf/An_efficient_home_automation_system_using_IOT__A_case_study_to_benefit_people_with_motion_disability__ijariie15836.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-5450a97b8d1a
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

The End