PRACTICAL AND INNOVATIVE APPLICATIONS OF IOT & IOT NETWORKS SMART HOME

Abstract

Home automation has received a lot of attention of late in the IoT/M2M context. Basic applications of the automated home include remote media control, heating control, lighting control, including low power landscape lighting control, and appliance control. Sensed homes, as examples of smart space, are seen as "next-step/next-generation" applications. Smart meters and energy, also fit in this category. Telehealth (e.g., assisted living and in-home m-health services) also can be captured under this set of applications; security and emergency services also can be included here.

Authors

Dr. Pravin Latane
Department of Electronics and Communication Engineering
Sinhgad Institute of Technology
Lonavala, India.
pclatane@gmail.com

Dr. Vilas Ubale
Department of Electronics and Communication Engineering
Amrutvahini College of Engineering
Sangamner,India.
vilasubale1978@gmail.com

TITLE: To Design and implement IoT system for the applications like: Home Automatiomation

I. THEORY

In the present era, there is a strong emphasis on enhancing comfort, and one effective solution is the integration of IoT (Internet of Things) devices. These devices enable us to streamline our lives by offering remote control over various aspects. For instance, household appliances, door locks, and machinery can be conveniently managed through smartphones or web servers. A practical approach to achieving this is by creating an Android application using MIT App Inventor.

The process involves programming a NodeMCU, which serves as the central hub, to establish an HTTP web server. This server enables the control of home appliances. The communication between the NodeMCU and the Android app primarily employs the HTTP GET method.

To determine the functionality of the web server, a straightforward method is to open a web browser and utilize specific URLs. These URLs are configured to trigger actions such as turning lights on or off. This provides a seamless way to interact with and monitor the connected devices.

http://172.128.10.40/gpio/1
http://172.128.10.40:/gpio/0

The NodeMCU's IP address is 172.128.10.40. You can conveniently find the IP address of your NodeMCU by examining the serial monitor. Once you upload the code in the Arduino IDE and run it, the serial monitor will show the device's IP address. This approach validates the proper functioning of the web server.

MIT App Inventor functions as an open-source web platform designed for creating Android applications. While it was initially created by Google, it is currently managed by the Massachusetts Institute of Technology (MIT). This software serves as an inclusive choice, even for beginners, to develop Android applications. Its intuitive nature relies on a visual interface that allows users to easily build applications using a drag-and-drop technique, all of which can be used on Android devices.

Subsequent to the design phase within MIT App Inventor, the application can be acquired on an Android phone through a QR code, or alternatively, its APK can be downloaded onto a computer for later installation on a smartphone. Once this is accomplished, the next step involves establishing a connection between the application and the ESP8266 module, facilitating the control of various home appliances.

II. HOW RELAY WORKS?

A relay serves as an electrical switch that is activated through an electrical signal. It proves invaluable in scenarios where multiple circuits need to be managed using a single input. By utilizing a relay, it becomes feasible to control the operation of an electrical

circuit, toggling it on or off. The functioning of a relay hinges on a low-level current that is capable of controlling the switching of a higher magnitude current.

Typically featuring five terminals, a relay's configuration is as follows:

In the absence of voltage applied to the coil, the COM (common) terminal establishes a connection with the NC (normally closed) terminal.

When voltage is applied to the coil, it generates an electromagnetic field that draws the armature towards it. This movement results in the connection between the COM terminal and the NO (normally open) terminal.

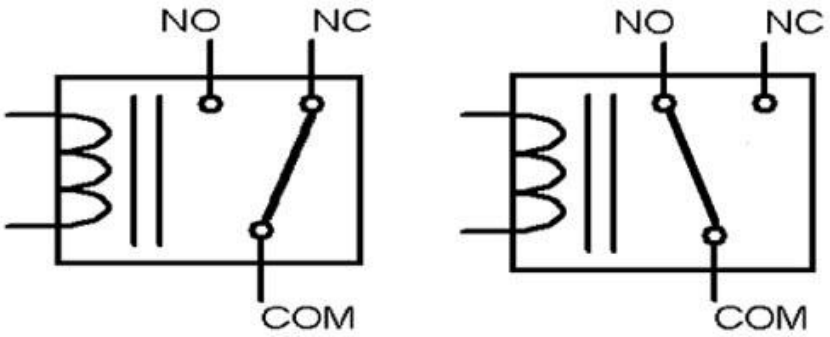

Figure 1: Relay Diagram

1. **Relay Driver:** In the construction of a concise driver circuit, a combination of a Transistor, Diode, and Resistor is utilized to configure the relay. The Transistor amplifies the current, the Resistor offers the required bias for the transistor, and the Diode comes into play when the transistor is deactivated to prevent any backward current flow. It's worth highlighting that a 5V Relay module is employed within this arrangement.

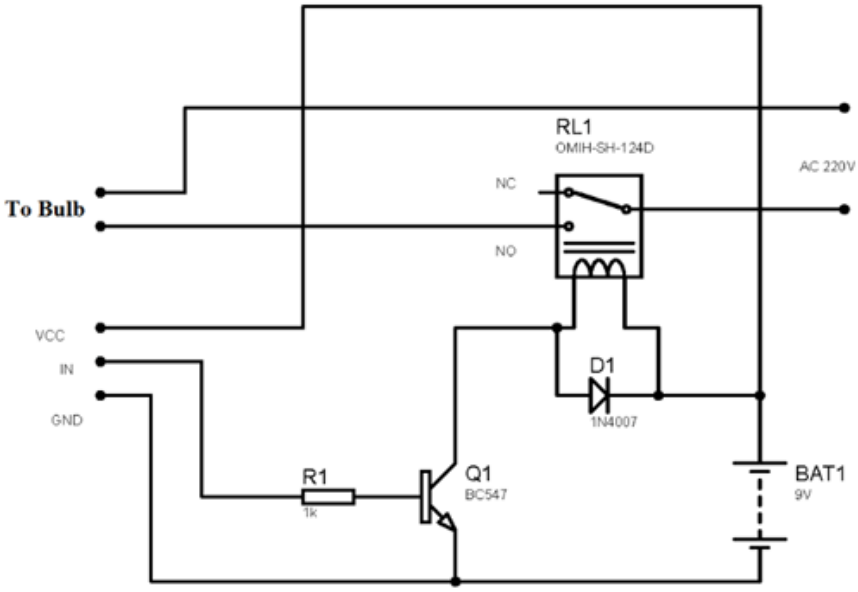

Figure 2: Relay Interfacing diagram

III. COMPONENTS REQUIRED

- ESP8266 (Node MCU)
- Lamp
- 5V Relay
- Breadboard
- Connecting Wires

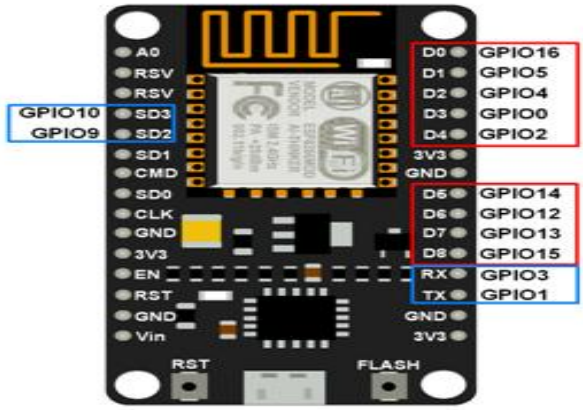

Figure 3: Node McU

1. **Circuit Diagram:** In this experiment, we are going to log and monitor data over internet using MIT app Inventer IoT server. And we can view the logged data over time on Mobile app. System is made using ESP8266 WiFi module and Relay module. ESP8266 WiFi chip reads the current date and sends it to MIT app inventer server for live monitoring.

   In this experiment, relay module can be used with esp8266 or Nodemcu and program from Arduino IDE, the mobile data has used to send data to MIT app and it was really productive, results are quite good.

2. **Connection Diagram**

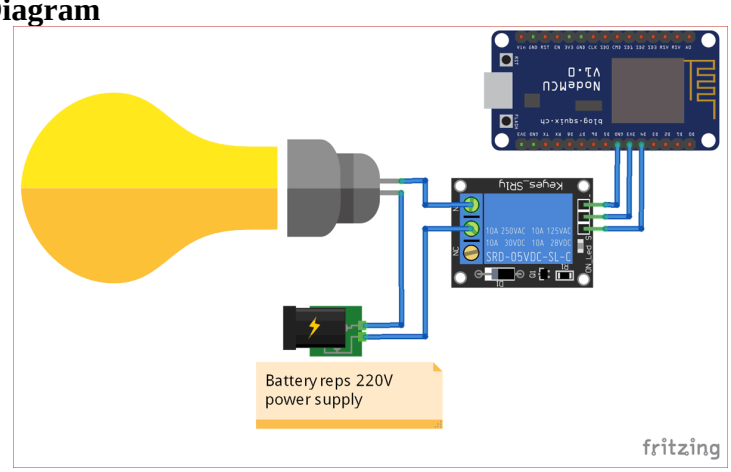

Figure 4: Interfacing of Relay Module with Node McU

IV. DEVELOP AN ANDROID APPLICATION USING MIT APP INVENTOR

To begin the procedure of creating an Android application for controlling lights, we will employ MIT App Inventor, following these subsequent steps:

Begin by accessing the MIT App Inventor's official website: http://ai2.appinventor.mit.edu/

Proceed by selecting the 'Create apps' option situated at the upper right-hand corner of the interface.

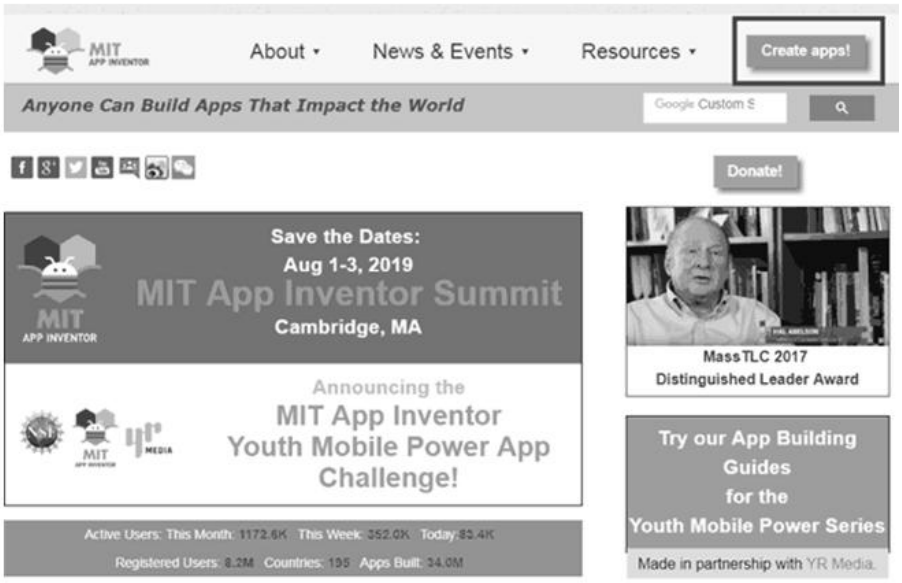

Figure 5: MIT App Inventor Website

Step 1: Moving forward, on the subsequent screen, navigate to 'Projects' and subsequently choose 'Start new project'.

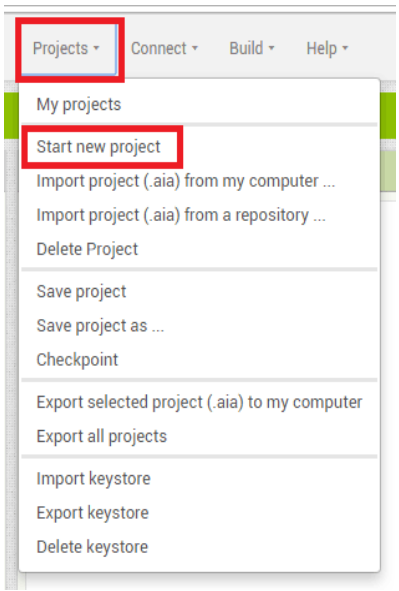

Figure 6: Start new project

Step 2: Afterward, pick the 'Button' component and drag and drop two buttons onto the main screen. To customize the buttons, you have the option to enter your desired names from the choices provided on the right side.

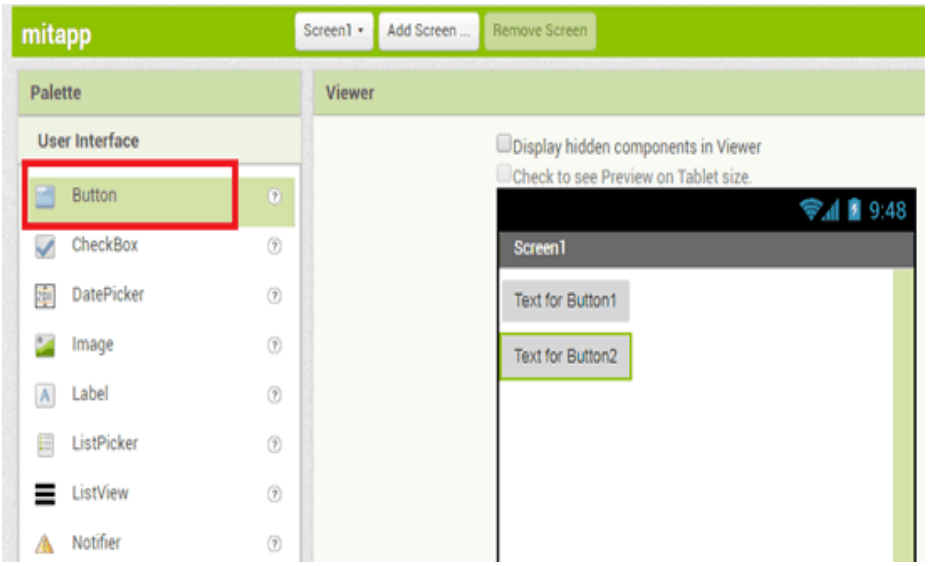

Figure 7: Add Buttons

Step 3: Continuing from there, navigate to the 'connectivity' segment and then drag and drop the web component onto the primary screen.

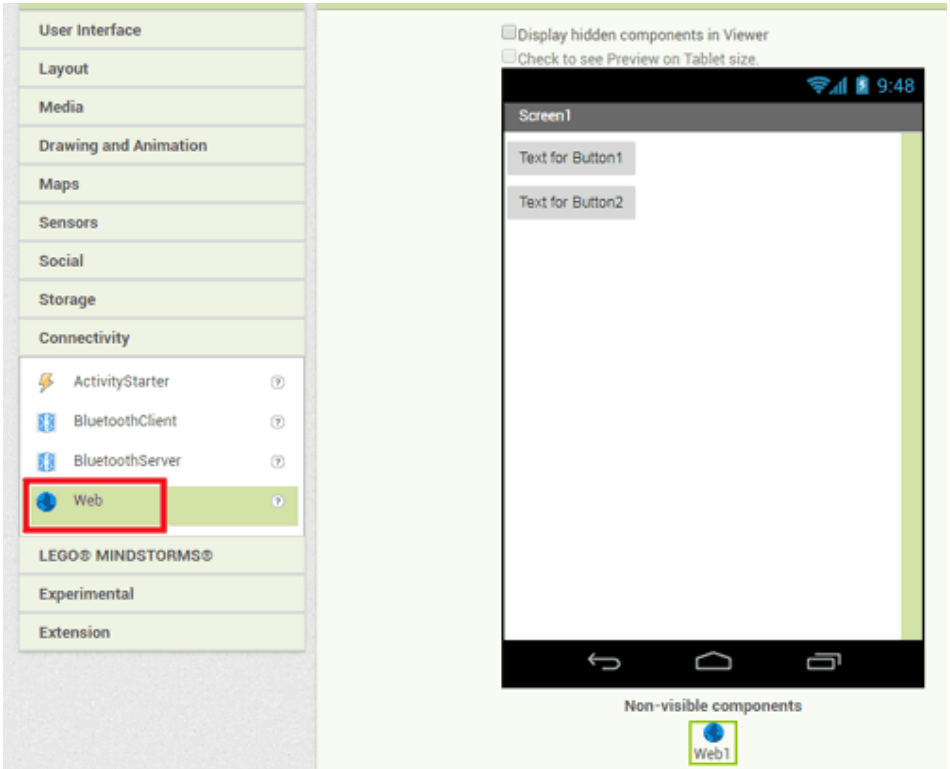

Figure 8: Add cloud Connectivity

Step 4: Now, access the 'Blocks' section to incorporate blocks into your application.

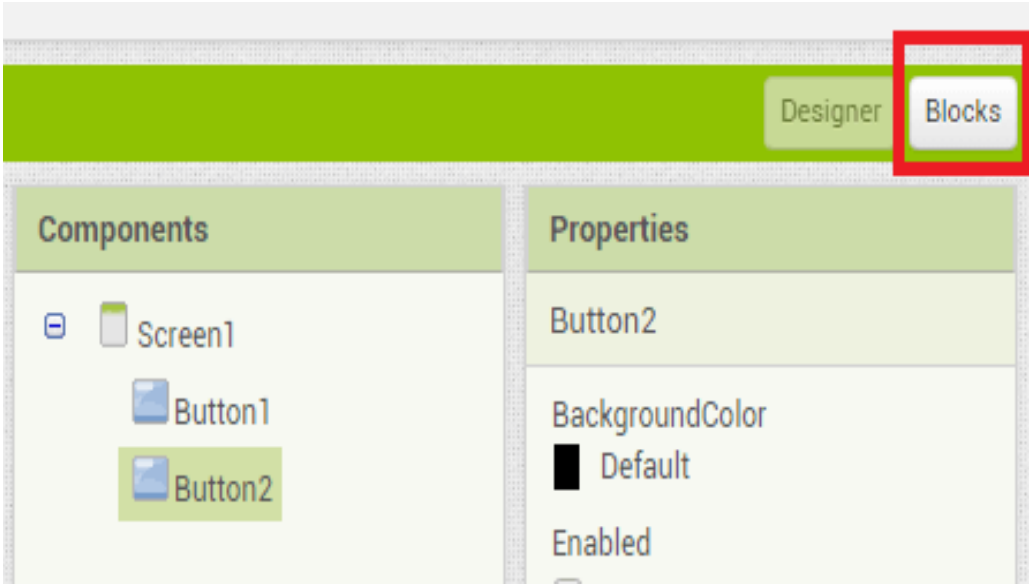

Figure 9: Block Code Programming

Step 5: Within the block's menu, select 'button1,' and then choose the highlighted red option.

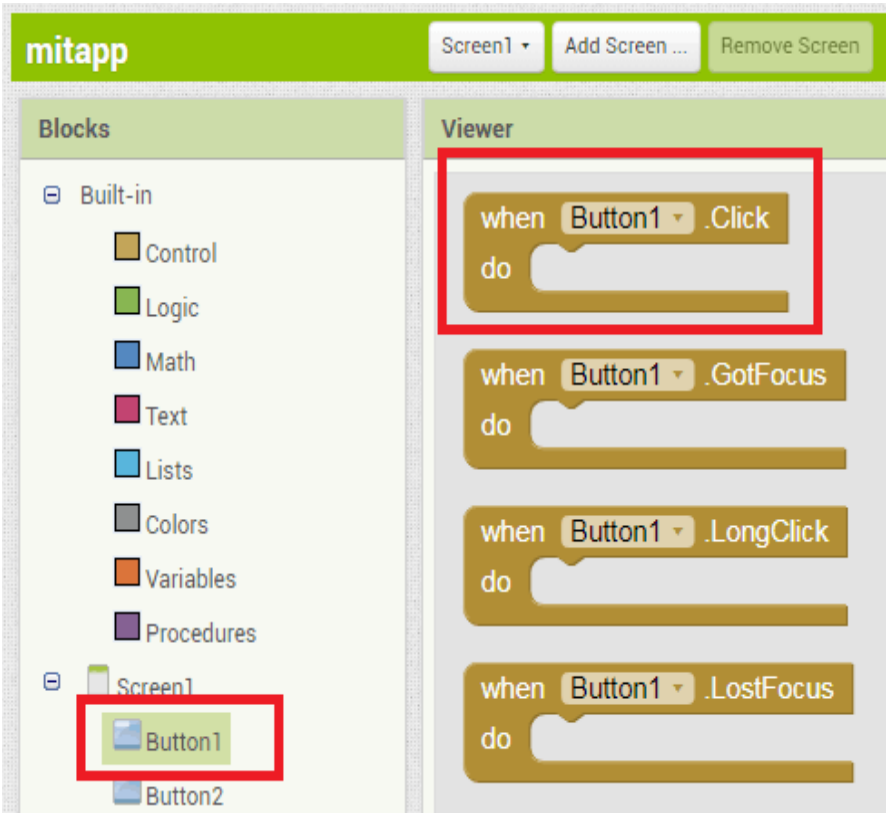

Figure 10: Block Coding

Step 6: Next, click on 'web1.' Scroll through and select the block that is marked with a red indicator.

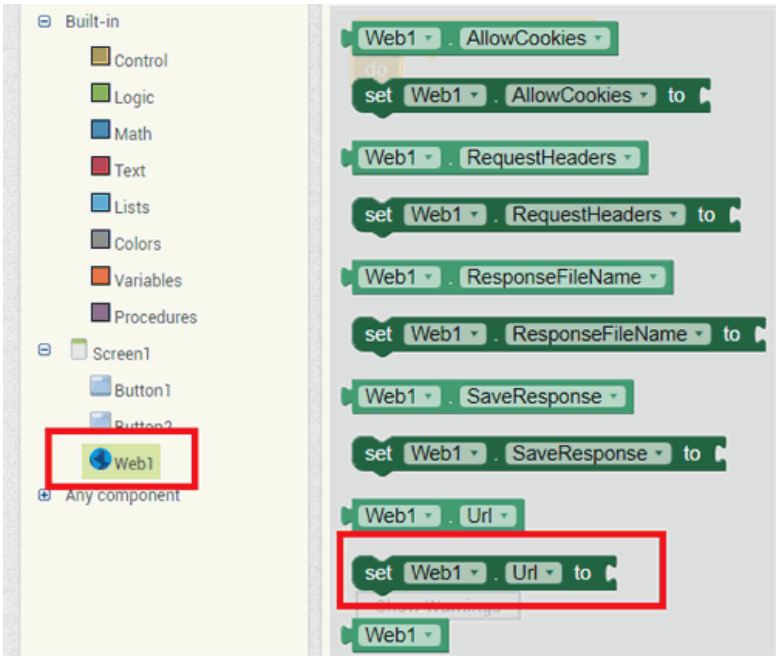

Figure 11: Adding Connectivity

Step 7: Now, go to the 'Text' menu and choose the first option. Input your preferred URL within the text menu.

Figure 12: Adding Text Message

Step 8: Next, click on 'web1' once more and proceed to select the option highlighted in red.

Figure 13: MIT App Inventor Logo

Step 9: Apply the identical procedure for 'Button2.'

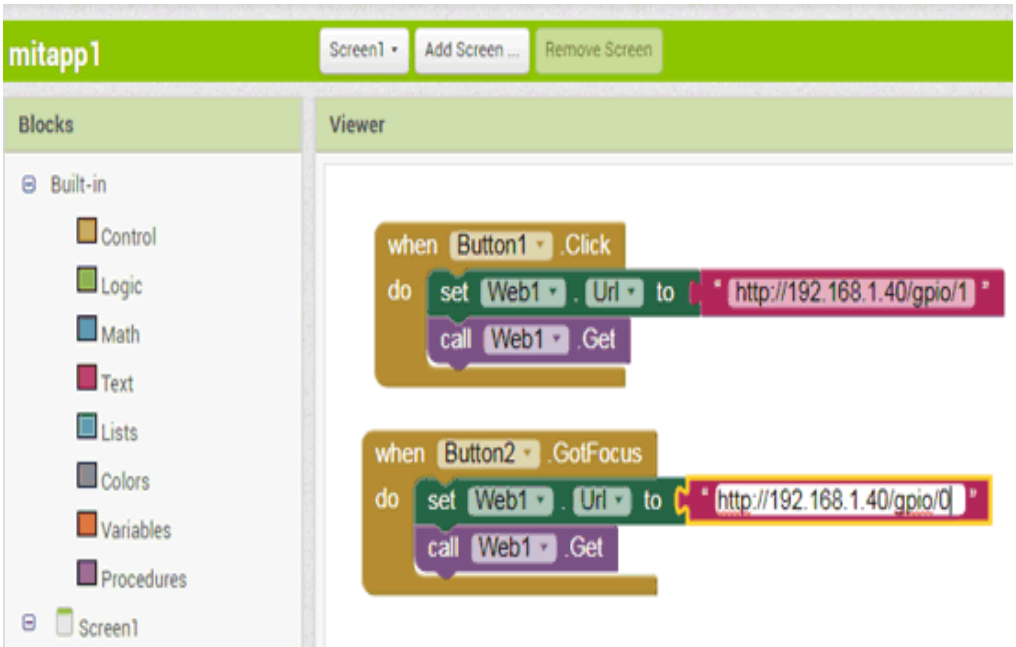

Figure 14: Final Block Code Program

Step 10: The application is now ready for downloading. Simply click on the 'Build' choice to obtain the APK file. You have two methods to get the app APK: either through the QR code or directly onto your computer. Following that, you can install the downloaded APK on your Android device.

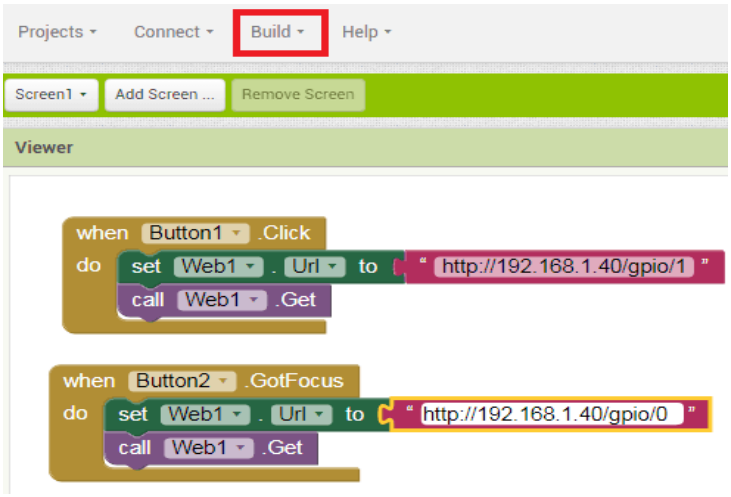

Figure 15: Genarate APK file

Step 11: Your application is now completely set up, providing you with the capability to manage the lighting using the included on-off buttons within the app.

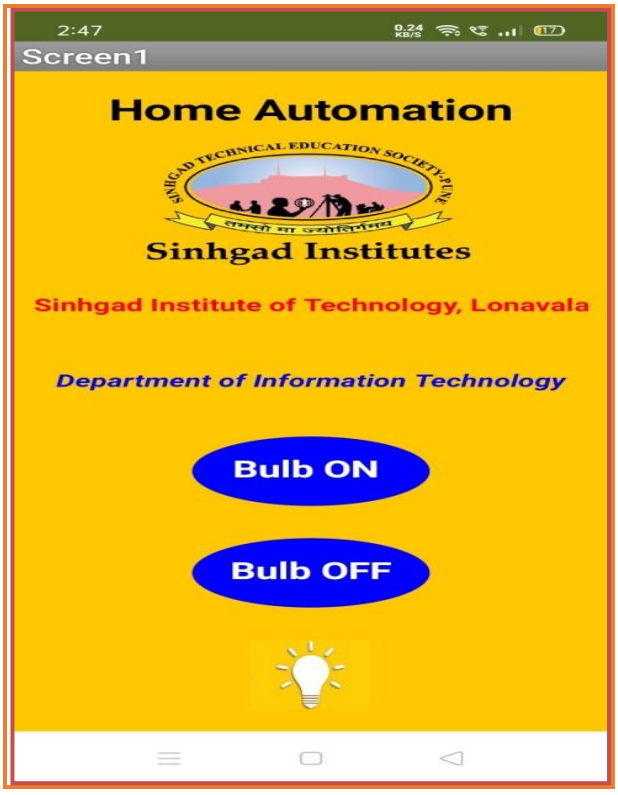

Figure 16: View of Mobile for Application Control

V. PROGRAM

```
#include <ESP8266WiFi.h>
 const char* ssid = "OPPO_pcl";
const char* password = "pclatane123";
WiFiServer server(80);
 void setup() {
  Serial.begin(115200); //Default Baud Rate for NodeMCU
  delay(10);
  pinMode(2, OUTPUT);  // Connect Relay to NodeMCU's D4 Pin
  digitalWrite(2, 0);
  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  // Start the server
  server.begin();
  Serial.println("Server started");
  // Print the IP address
  Serial.println(WiFi.localIP());
}
 void loop() {
  // Checking, if is client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

// Wait for data from client
  Serial.println("new client Joined");
  while(!client.available()){
  delay(1);
  }
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();
int val;
  if (req.indexOf("/gpio/0") != -1)
    val = 0;
  else if (req.indexOf("/gpio/1") != -1)
```

```
    val = 1;
  else {
    Serial.println("invalid request, try again");
    client.stop();
    return;   }
 String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nGPIO is now ";
  s += (val)?"high":"low";
  s += "</html>\n";
// Send the parameter to the client
client.print(s);
delay(1);
Serial.println("Client disconnected, try again");
}
// Set GPIO2 state according to the request
  digitalWrite(2, val);
  client.flush();
```

Figure 17: Home Automation Program

VI. CONCLUSION

After the study of this assignment, we are familiar with the home automation and use of MIT app Inventer to upload data on cloud.