1. Configuring ESP32

Components:

- DHT11 Sensor: For temperature and humidity.
- BH1750 Sensor: For light intensity detection.
- Relay: To control the lamp.
- Connect Sensors DHT11, BH1750, Relay to ESP32

2. Developing Control Logic: Program Automation Tasks Based on Sensor Input

Automation for the lamp: The automation is based on the light intensity from the BH1750. When the light level is below a threshold, the ESP32 will turn on the lamp. When the light intensity is above this threshold, it turns off the lamp.

3. Creating a Mobile App with MIT App Inventor

- 1. Create a User Interface by adding buttons and labels.
- 2. Add BLE Component to allow BLE connection to the app.

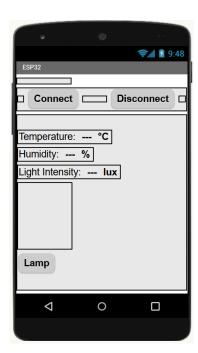
Application Logic:

1. Connect to the ESP32:

Scan for BLE devices and connect to the ESP32. When the ESP32 is connected, the app will show connected status.

2. Control Lamp:

When the Button for the lamp is pressed, sends a BLE command to change the lamp's state to ON or OFF



3. Display Sensor Data:

When BLE command is received from ESP32, display the current temperature, humidity, and light intensity.

MIT App Inventor blocks:

```
initialize global Service_UUID to df9b6ad3-8c86-4c75-8036-9ed2f039a3b0

initialize global RX_char_UUID to de663d68-1e27-4f2a-9bd3-7f4be4e9119c

initialize global TX_char_UUID to de663d68-1e27-4f2a-9bd3-7f4be4e9119c

initialize global TX_char_UUID to de6c3d68-1e27-4f2a-9bd3-7f4be4e9119c

initialize global TX_char_UUID to de6c7c69-1f45-46c4-9c3b-7f3ac73ea2a7

initialize global TX_char_UUID3 to de6c7c69-1f45-46c4-9c3b-7f3ac73ea2a7

when Screen1 Initialize

do call BluetoothLE2 StartScanning

when Connect AfterPicking

do call BluetoothLE2 Connect
   index Connect BluetoothLE2 DeviceFound

do set Connect ElementsFromString to BluetoothLE2 DeviceList

when BluetoothLE2 Connected

do set Connect ElementsFromString

when Disconnect Cick

do call BluetoothLE2 Disconnect
   set Ucin BluetoothLE2 StartScanning
```

```
when LED .Click
do 👩 if BluetoothLE2 🔻 IsDeviceConnected 🔻
   then if LED . BackgroundColor = V
         then set LED . BackgroundColor to
               call BluetoothLE2 .WriteStrings
                                  serviceUuid
                                              get global service_UUID *
                                              get (global RX_char_UUID *
                                      utf16 ( false v
                    LED v BackgroundColor v = v
         then set LED . BackgroundColor to
               call BluetoothLE2 .WriteStrings
                                           get global service_UUID •
                                              get global RX_char_UUID
                                      utf16 | false v
                                              * B *
```

```
when Clock1 • Timer

do O if BluetoothLE2 • SegisterForStrings serviceUpid characteristicUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid characteristicUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid characteristicUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • RegisterForStrings serviceUpid opt global service_UpiD • call BluetoothLE2 • call BluetoothLE2 •
```

4. Cloud Integration

- 1. Link ESP32 to Arduino IoT Cloud.
- 2. Define variables for Temperature, Humidity, Light Intensity, and Lamp.
- 3. Use the Arduino IoT Cloud dashboard to control the lamp and monitor sensor data remotely.

