CONTEXT
-Introduction
-Materials
-Component Description
-Connection Layout (Diagram)
-Procedures Videos (Contained in Folder of this Project)
-Source Code
-MIT App UI designs & Block Code
-Problems & Solution

INTRODUCTION

Internet of Things describes the network of physical objects—a.k.a. "things"—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. The extensive set of applications for IoT devices is often divided into consumer, commercial, industrial, and infrastructure spaces
Wi-Fi is a family of wireless network protocols, based on the IEEE 802.11 family of standards, which are commonly used for local area networking of devices and Internet access, allowing nearby digital devices to exchange data by radio waves.

This project is focused on the use of Wifi to control the extension in turn making it a smart wireless extension that can be controlled anywhere in the world as long as it has Internet connectivity. This wireless extension uses “ROBOTICS CENTER” as SSID and “shut down” as the SSID Password, it was created & developed by Joel Omoroje.

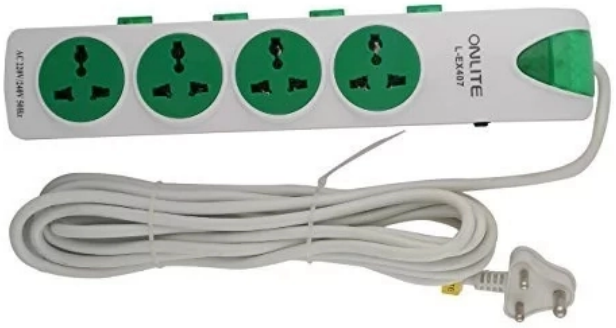
Materials:

- 1 Extension Box
- 1 AC to DC converter (conversion of 220V to 5v)
- 1 Relay
- 1 ESP32

Wireless Materials
-Electricity
-Internet access (wireless)

Component Description

Extension Box:

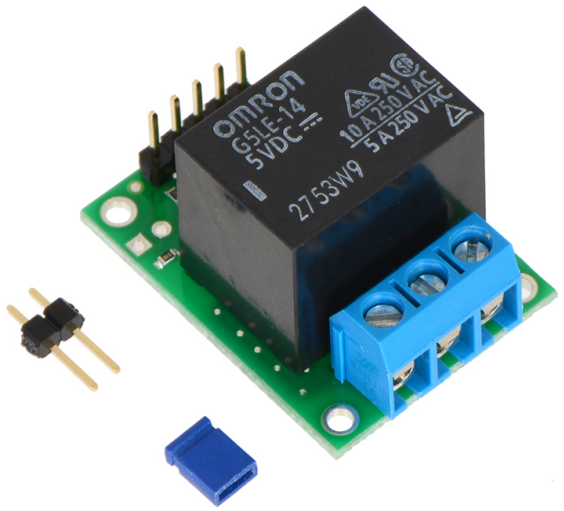


Extension box hosts various sockets.
Extension sockets are built in an extension box, which makes use of electricity in our homes, offices and various other places convenient. We would be constructing an electrical extension with the following equipment; screw-driver, nails and screws, wooden plank, flexible wire, etc.

AC to DC Converter:

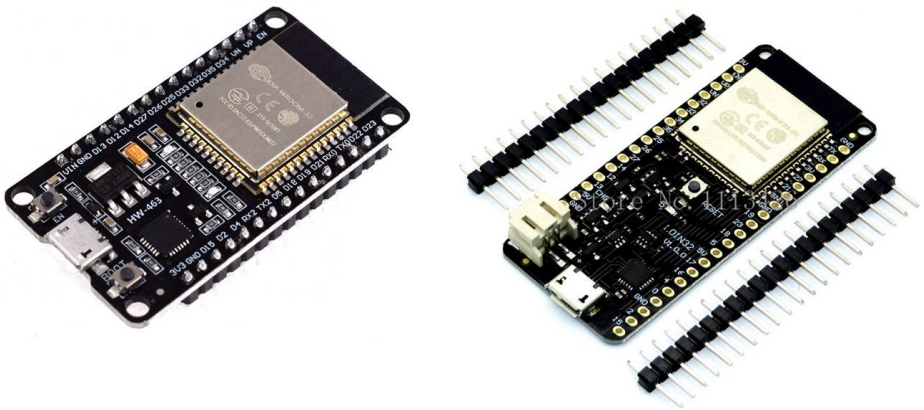
AC to DC Converters are one of the most important in power electronics because there are lot of real applications based on these conversions. The AC current to dc current conversion process is known as rectification. This rectifier converts AC supply into the DC supply at load end connection. Normally Transformers are used to adjust the AC source to get a step down transformer to reduce the voltage level to have better operation range for DC supply. Normally this AC to DC converter converts low voltage values.

Relay:



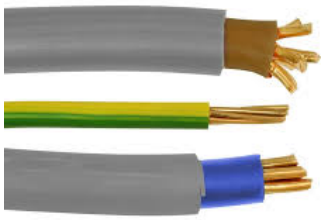
Relays are the switches which aim at closing and opening the circuits electronically as well as electromechanically. It controls the opening and closing of the circuit contacts of an electronic circuit. When the relay contact is open (NO), the relay isn't energize with the open contact. However, if it is closed (NC), the relay isn't energize given the closed contact. However, when energy (electricity or charge) is supplied, the states are prone to change.

ESP32:



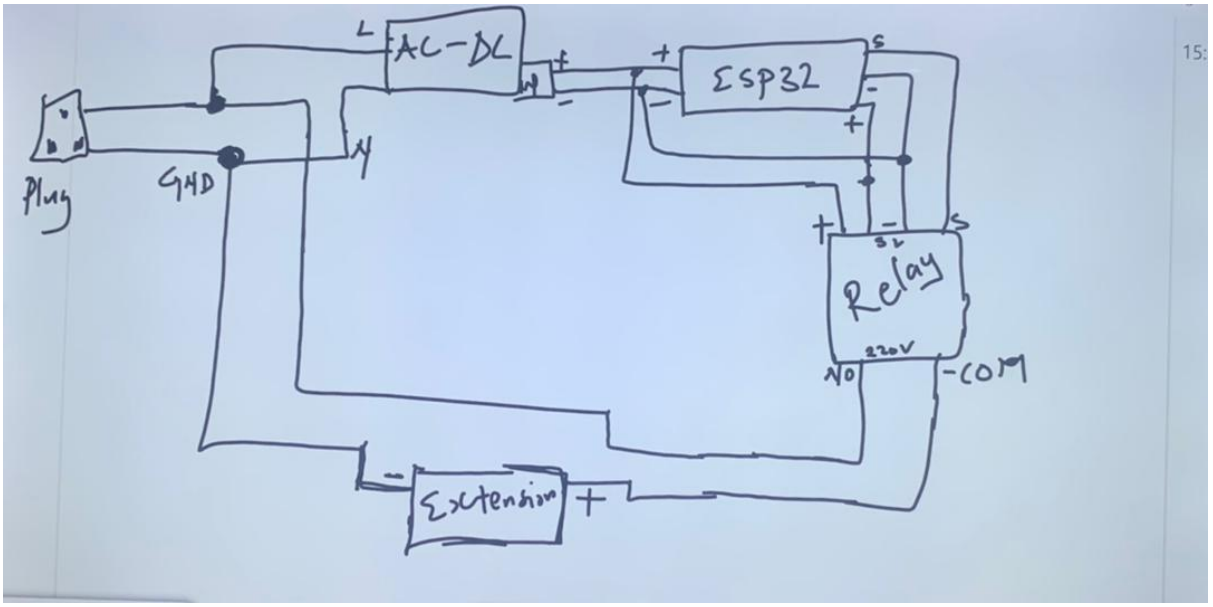
ESP32 is a WiFi-compatible microcontroller, but to that it adds support for Bluetooth low-energy (i.e BLE, BT4.0, Bluetooth Smart), and nearly 30 I/O pins. The ESP32's power and versatility will make it the foundation of IoT and connected projects for many years to come.

Wires:



A *wire* is a single usually cylindrical, flexible strand or rod of metal. *Wires* are used to bear mechanical loads or electricity and telecommunications signals.

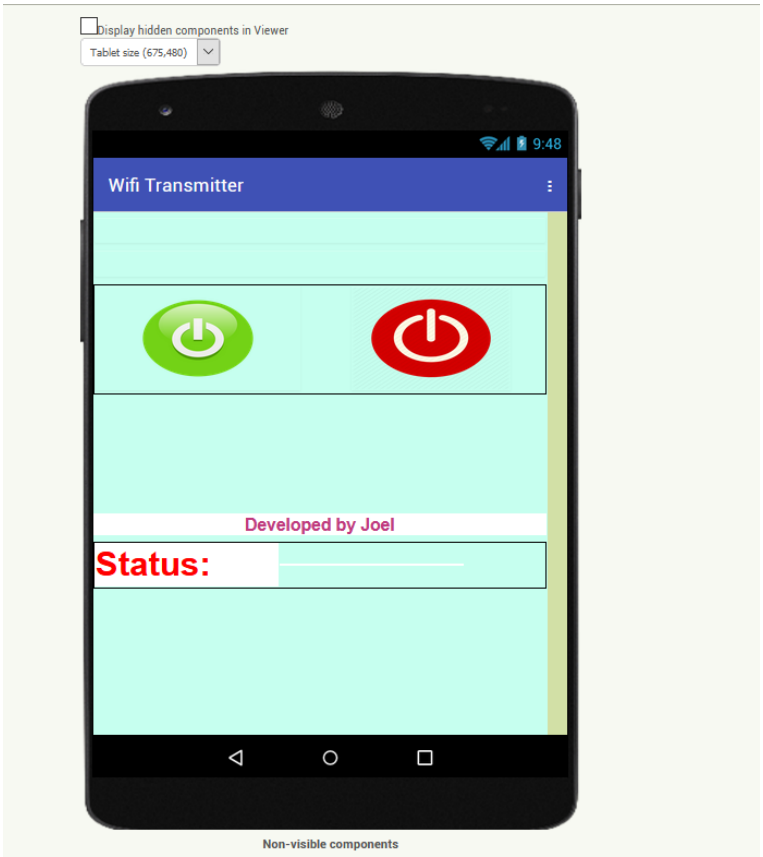
Connection Layout



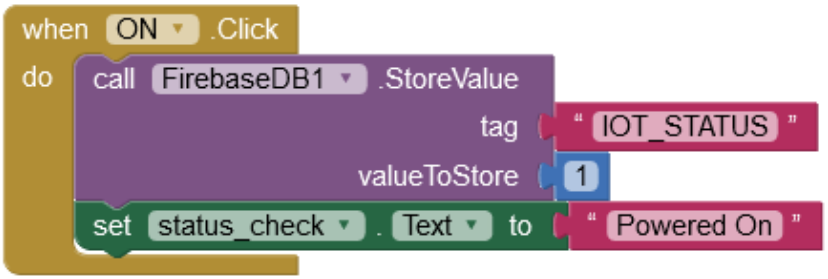
Source Code:

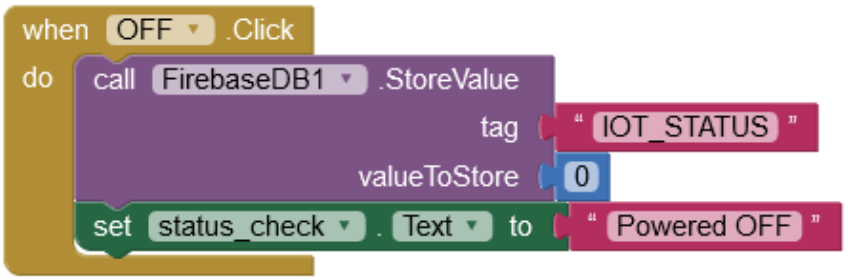
```
#include <WiFi.h>
#include <IOXhop_FirebaseESP32.h>

#define FIREBASE_HOST "wifi---esp32-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "GJPaMgFWnEAHFNANCtCS0sQmDscWdKqpB4mIWX6n"
#define WIFI_SSID "DEVELOPED BY JOEL"
#define WIFI_PASSWORD "******************"

void setup() {
  Serial.begin(115200);
        delay(1000);
        WiFi.begin(WIFI_SSID, WIFI_PASSWORD); //try to connect with wifi
        Serial.println("Connecting");
        while (WiFi.status() != WL_CONNECTED) {
        delay(500);
```

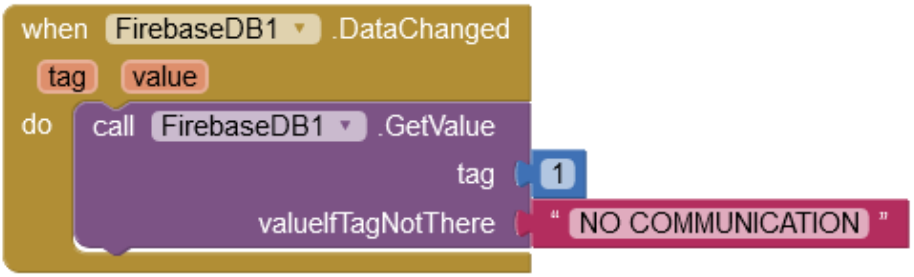
```
        }
        Serial.println("CONNECTED TO ");
        Serial.println(WIFI_SSID);
        Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
        Serial.println("starting Firebase");
        pinMode(19,OUTPUT);
}

void loop() {
String value  = Firebase.getString("IOT_STATUS");

  // set value
  Firebase.setString("esp32_firebase_status", "connected to firebase successfully");

  // get value
  Serial.print("IOT_STATUS: ");
  Serial.println(Firebase.getString("IOT_STATUS"));


//Validates the Input of IOT_STATUS and switches on the Extension
   if (value == "1"){
        digitalWrite(19,HIGH);
        Firebase.setString("extension", "extension is powered on");
        Serial.println(Firebase.getString("extension"));
        Serial.println("Powered on the extension");
        delay(500);
        }

//Validates the Input of IOT_STATUS and switches off the Extension
        else if (value == "0"){
        digitalWrite(19,LOW);
        Firebase.setString("extension", "update: extension is powered off");
        Serial.println(Firebase.getString("extension"));
        Serial.println("Powered off the extension");
        delay(500);
        }
//Error returned when values are neither 1 or 0
        else{
        Serial.println("ERROR FOUND NEITHER 1 or 0");
        }
}
```



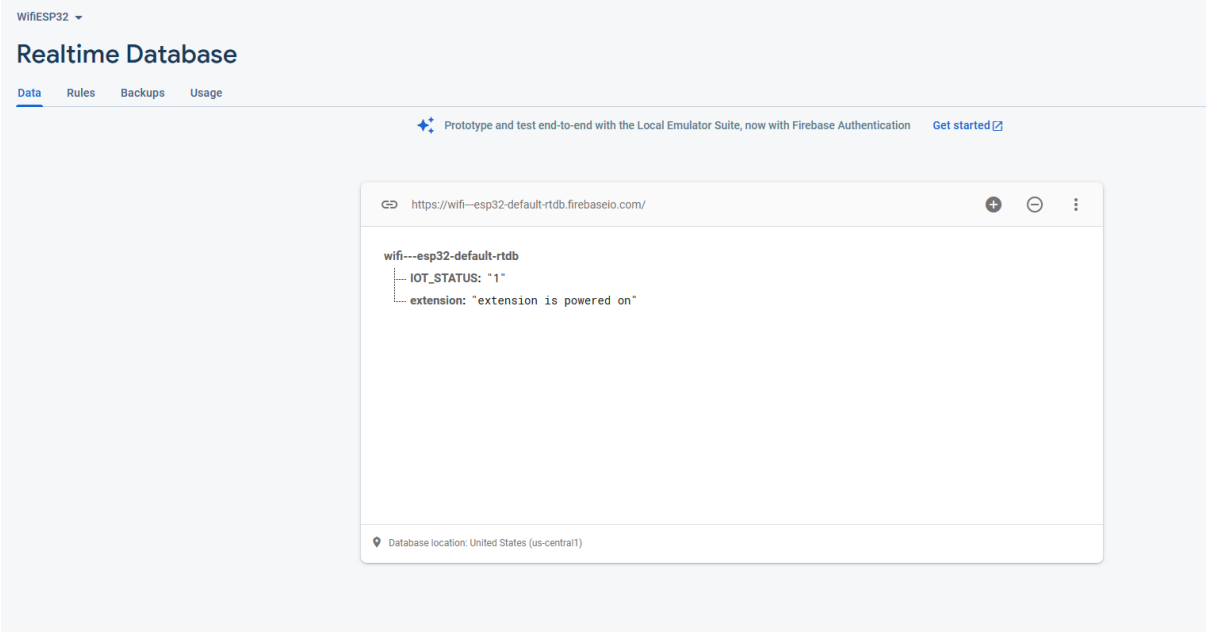
MIT App UI designs & Block Code


when ON .Click
do call FirebaseDB1 .StoreValue
tag " IOT_STATUS "
valueToStore 1
set status_check . Text to " Powered On "


when OFF .Click
do call FirebaseDB1 .StoreValue
tag " IOT_STATUS "
valueToStore 0
set status_check . Text to " Powered OFF "


when FirebaseDB1 .DataChanged
tag value
do call FirebaseDB1 .GetValue
tag 1
valueIfTagNotThere " NO COMMUNICATION "

Firebase Connection



Problems & Solution
While working on this project the problem encountered was installation of the Arduino Json library, i installed the 7.17.3 version which was way above the needed library so in order to have zero issues recreating this project i advise you installed 5.13.5 version of the Arduino JSON.

Also i had the issue of using two different firebase link on the MIT app inventor and the Arduino code, always cross check that and ensure both are using the right linking address.