A Project Report on

**HANDY HELPER**

**Submitted by**

Ramanuja Phani Vishnu Teja Kandalam

**Submitted to**

****

**as a part of**

**iB Hubs IoT Boot camp ’18, Visakhapatnam**

**Date: 31-05-2018**

**Acknowledgement**

We take the opportunity to express our reverence to all the trainers for thier guidance, inspiration and innovative technical discussions during this work.

We thank everybody for their constant guidance and support. They have been great sources of inspiration to us.

We would like to thank all our friends and especially our fellow participants for all the thoughtful and mind stimulating group interactions we had, which prompted us to think beyond the obvious.

**Abstract:**

The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enable these objects to connect and exchange data. It is a network in which physical objects can exchange data internally or with other connected machines. We can implement IoT using any development boards like raspberry pie, Octabrix, etc. The other major component we used is a joystick. It has many other applications in DIY electronics. In this project the user can control various home appliances using the joystick.

Our project **HANDY HELPER** uses two development boards, one as server with which appliances were connected and the other as client who sends requests to the server and has joystick connected to it. The main goal of **HANDY HELPER** is to minimize the human effort with low cost. This also helps us to control any electrical devices we wish and make out home a better place to live in.

# Introduction

In the past five years we have seen the growth of IoT around the globe. Everybody now are trying to adapt this technology into their domains. One such idea is this **HANDY HELPER** which is used in simplifying the day to day activities. We can use this simple, small gadget to control our appliances at home. This is also very helpful for the blind people to switch on/off any appliances at their home as they feel difficulty in finding the switches.

Out **HANDY HELPER** comes with a joystick + development board and a server board. The joystick is used to send the requests regarding the specific application to the server and the server reacts accordingly.

The applications we included in our project are:

1. Controlling a home appliance,

2. Finding your misplaced phone using the joystick and

3. Sounding the emergency buzzer in the case of any emergency.

We used a joystick to send the requests according to the user’s application. The development board used to interpret the values of the joystick is Octabrix. We took a development board called NodeMCU to act as server and respond accordingly. Both of these modules have ESP8266 Wi.Fi. S.o.C. as common. We used a special cloud platform called Blynk to accept requests from the joystick i.e. Octabrix and this helps in sending the notifications to the phone easily.

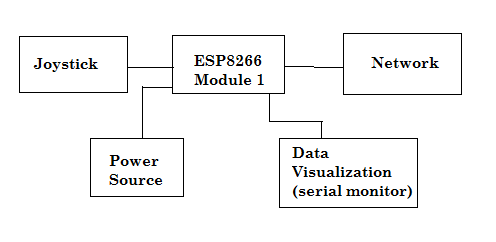
Now a day’s everyone were very busy in their personal and professional life so that they need a handy device by which they can control everything in and out of their home. Our **HANDY HELPER** helps them a lot, as they need not even get up from their chair or use their mobile phone to switch off their fans or lights.

# Approach:

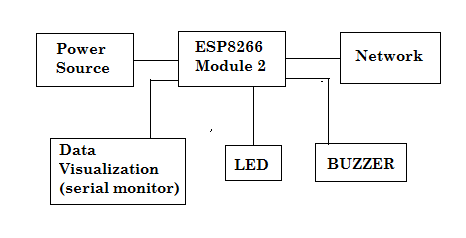
As you know that our project has two development kits. One is for the server and the other is for client. We made the server and the client to communicate with each other using the HTTP protocol which is using a GET request.

This module gives you the functional block diagrams of the server and the client.

The block diagram of the client is as follows:



The functional block diagram of the client module is as follows:



# Hardware and Software components used

The Hardware components include:

* Octabrix HDK ,
* Joystick module,
* Jumper wires,
* Buzzer and
* LED bulb.
* Octabrix HDK:

Octabrix ​is a Wi-Fi development board based on the famous and low-cost Wi-Fi System on Chip. ESP8266. ESP8266 is a wireless System on Chip that provides ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the perfect combination of lowest cost and minimal space requirement. Octabrix houses the ESP-12F version of ESP8266.

Octabrix has 10 digital input/output pins (all of which can be used as PWM outputs), a single analog input,​ a micro USB connection, a reset button, an on-board light sensor, programmable push button and LEDs. One of the interesting features of Octabrix is the user programmable RGB Pixel Ring​.

* Joystick module:

When we hear the word "Joystick" we think of game controllers. If we talk about electronics there are many useful applications of joystick. These type of module are mostly used in Arduino based DIY projects and Robot control.

This Joystick module typically provides Analog Outputs and the output voltages provided by this module keep changing according to the direction in which we move it. And we can get the direction of movement by interpreting these voltage changes using some microcontroller.

* Jumper wires:

A jumper wire is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normlly used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jumper wires arefitted by inserting their end connectors into the slots provided in a breadboard, the header connector of a circuit board,or a piece of test equipment

* Buzzer:

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as mouse click or keystroke.

* LED:

A light-emitting diode (LED) is a two-lead semiconductor light source. it is a p-n junction diode that emits light when activated. when a suitable current is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called as electroluminescence, and the color of the light is determined by the energy band gap of the semiconductor. LED's are typically small (less than 1 mm^2) and integrated optical components may be used to shape the radiation pattern.

Software components include:

* Arduino software (IDE),
* Blynk mobile application
* Arduino software:

Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world. These products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone.

We used the Arduino IDE to develop the code and it also suppers the feature of uploading the program to the development board easily.

* Blynk application:

Blynk is a free application for smart phones that can be downloaded from the App store for Apple or Play Store for Android. This application has a number of widgets that supports different IoT applications. Actually Blynk can be exactly defined as a server which can take requests from our module and send response to another client. When we start our project with Blynk, it sends a authentication token to our Google mail by using this token we can send the requests from our module to the Blynk server.

# Features of the project

The major features of our project are explained below:

1. Controlling the remote appliances:-

This is the major and important feature of the **HANDY HELPER** where we can control the devices in the remote location by just pushing the joystick. This feature helps a lot in decreasing the human effort to do the job. It also increases the efficiency of the work. For example lets us consider that you were busy in your kitchen and wanted to control the light of the bedroom. To do so either you have to go to the bedroom or you have to search for your mobile phone and do that. Instead of doing either of things you can just carry a handy joystick which can control everything with a single push.

In our project we also provided the LED which indicates the state of the remote appliance. That is if the remote appliance is in ON state then the LED before you turns ON. Or if the remote appliance in OFF state you can easily notice that by just seeing the LED on the board.

2. Phone notifications using the cloud platform:-

When you just misplaced your phone, it will be easy for you if you just try to send the notifications to your phone. The other major feature of this project is that it sends the notifications to your phone. To accomplish this we use a cloud platform called Blynk which is free and easy to use for the beginners. When you just push the joystick to your right then the client module sends the request to the Blynk server to send the notification to the registered phone via the Blynk app installed on it.

3. Emergency button:-

The joystick also comes with a button in the middle of it. We used this button to indicate that someone is in emergency. By pressing this button the serve gets a request to turn on the buzzer all the time when the button was pressed. So that by hearing the sound of the buzzer your neighbours or someone around your house can easily know that you are in danger and can help you. This helps a lot in the time of fire accidents and thefts.

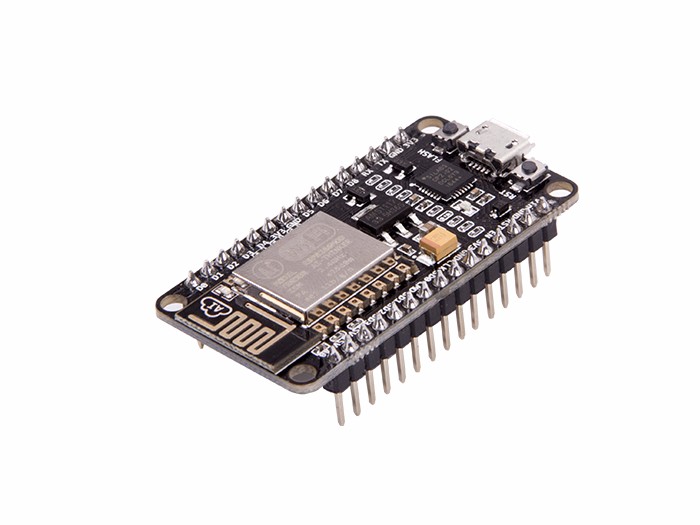
# Step by Step implementation

The major steps in building our project are:

Step 1: implementing the server module:-

To begin the process we must create the server and then connect the devices to those respective pins. In this project we used a development board called NodeMCU which has ESP 8266 as its System on Chip. The two devices are LED and a buzzer connected to 4 and 5 pins of NodeMCU respectively.

We then connect the NodeMCU to the network using the Wi.Fi. module provided on it namely ESP8266. It has its specific Arduino library called “ESP8266WiFi.h” which has many functions to connect and control the network options of the board. We must get an IP address to that server which plays an important role in the communication. The only duty of the server is to take the requests from client and then act accordingly. To do so we just take the request and check few conditions of the request and execute the specific tasks. A NodeMCU exactly looks like this:



Step 2: Implementing the client module:-

We used the IB Hubs Octabrix HDK as our client module. The client is connected with a joystick through which the user sends his request to the Octabrix. Then the Octabrix which is acting as client sends a request to the server which is NodeMCU using its IP address.

Step 2.1: Connecting the joystick to HDK:

This Joystick module typically provides analog outputs and the output voltages provided by this module keep changing according to the direction in which we move the stick. We can get the direction of movement by interpreting these voltage changes using some development board. This joystick module has two axes as you can see. They are X-axis and Y-axis. Each axis of JOYSTICK is mounted to a potentiometer or pot.

This analaog joystick is a device which is has 5 pins include a ground, +5V, x-axis, y-axis and the button pin. In this application we used the x-axis as our reference which indicates the movement of the joystick on x axis (horizontal path). The button pin is used to indicate the state of the button. The x axis pin is connected to the A0 pin of the HDK and the button pin is connected to the digital pin of HDK. An analog joystick typically looks like this:



Whenever the push button of the joystick was pressed then the buzzer will be activated / deactivated. If the joystick is moved in the left direction then then the HDK must request the server to turn on the LED. and when it is moved in the right direction it sends a notification to your phone which makes it easy for you to find it when misplaced.

Step 2.2: Preparing the HDK to be HTTP client:

We can create a web client or a HTTP client. Here we created a HTTP client. HTTP stands for Hyper Text Transfer Protocol which is most popular for client server communication. The Arduino has a special class for client called HTTPClient which is used for sending the requests through the server IP address and also receive the response from the server.

We use the Http.begin () method to pass the request to the server. The sample request to the sample server on the same network looks as follows:

<http://192.168.1.108/light/1>

In the above considered example URL the http specifies the http protocol, 192.168.1.108 specifies the IP address, the light specifies what the request is about and the 1 or 0 specifies the state. This request can be of any type, that is we can also write the request URL by using some variables and their values.

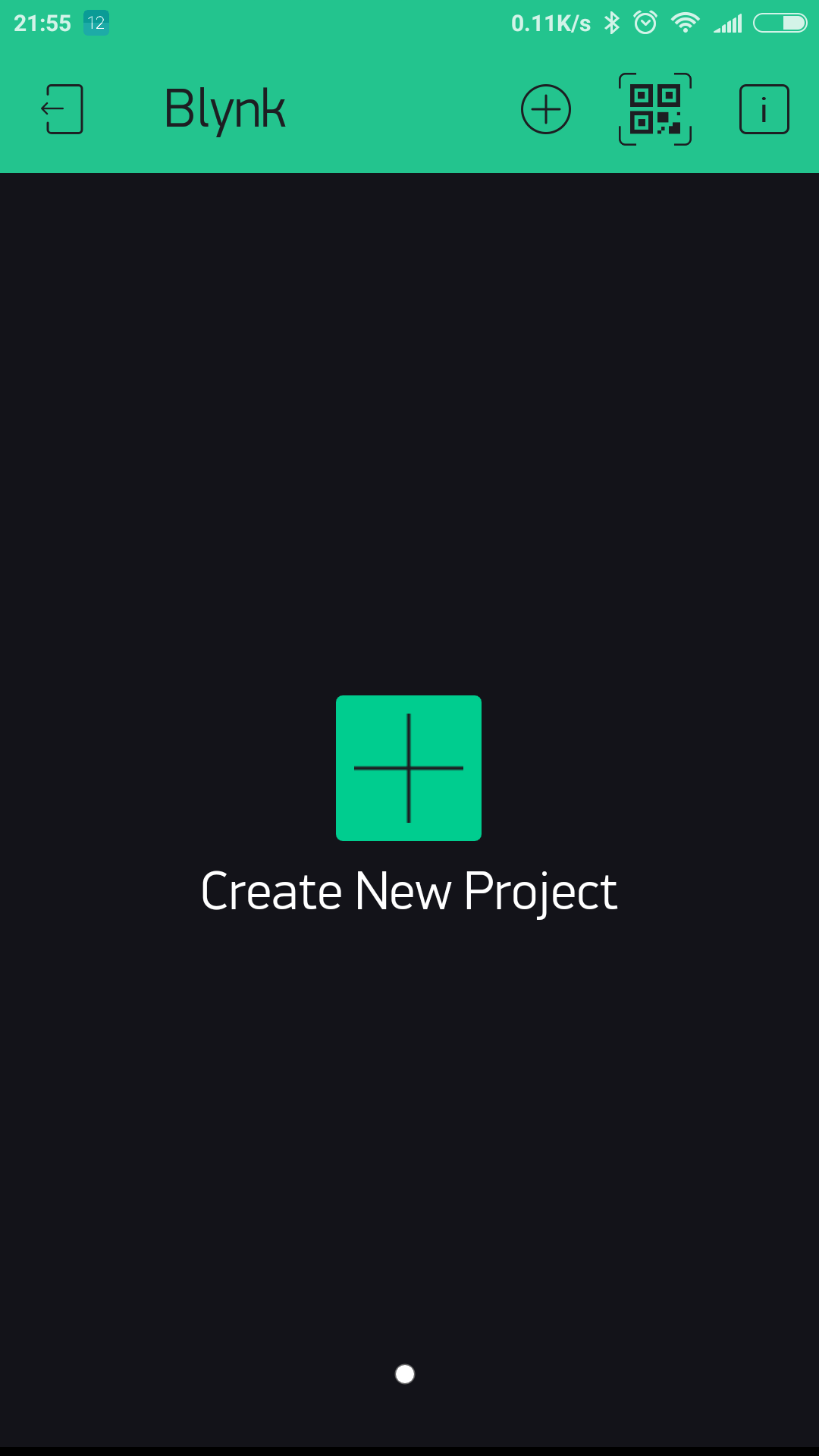
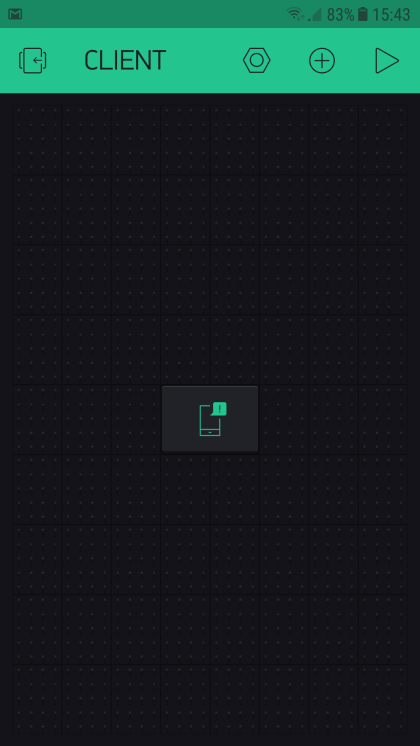
Now we can just check the value of the A0 pin to which the joystick x axis is connected and basing on that we can send the request to the server.

Step 3: Connecting the client to Blynk server:-

We use the Blynk server which was already discussed above in our project to send notifications to our phone. To enable this we have to install the Blynk app on our smart phone and then use the authentication token from the Blynk to connect the HDK to the Blynk server. In that new project we must add a widget called notification which is used to handle the notification requests you have made. This helps us in the situations like when we forget our phone somewhere and searching for it the entire home. We can make just move the joystick to its right this sends the request to the Blynk server to send a notification to your phone so that you can easily find it.

We can use the Blynk. begin () function to connect the module to the Wi.Fi network and also give the module a authentication token. This authentication token is used identify the module when a request is sent from it. In our case the Blynk server sends the response to the client by sending a notification to the phone.

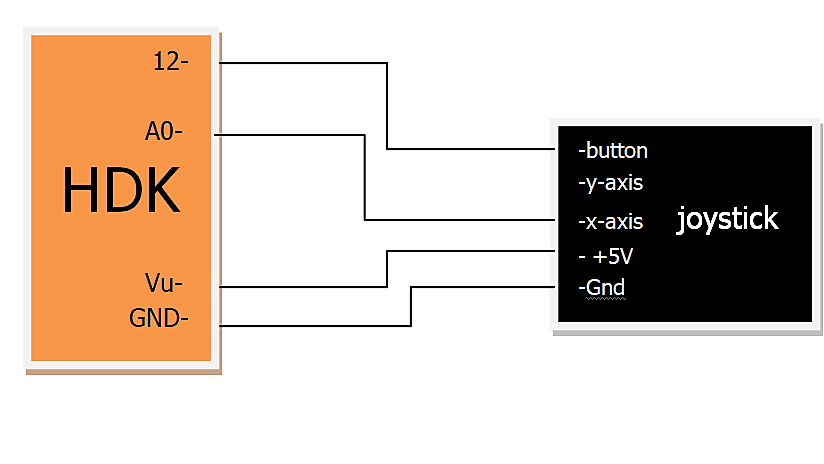
The screenshots of the Blynk app is as follows:

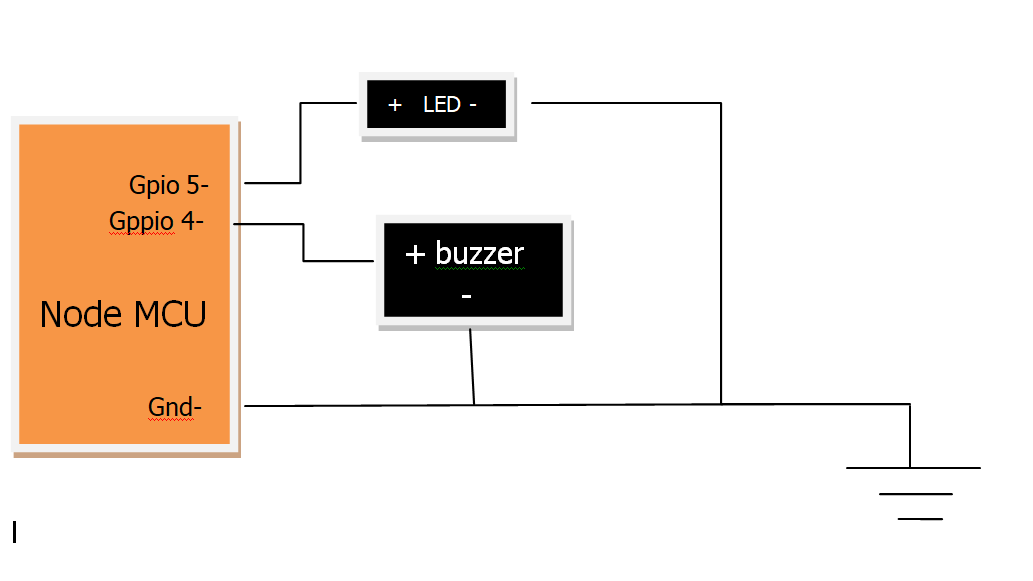
The left screenshot shows the first screen after you login with your gmail into Blynk. Then after clicking the New Project button we created a project called CLIENT and added a notification widget to it. This was in the right screenshot.

# Hardware circuit diagram

The hardware circuit diagram of the client module is as follows:



The hardware circuit diagram of the server module is as follows:



# Code

The operational code for the server module is as follows:

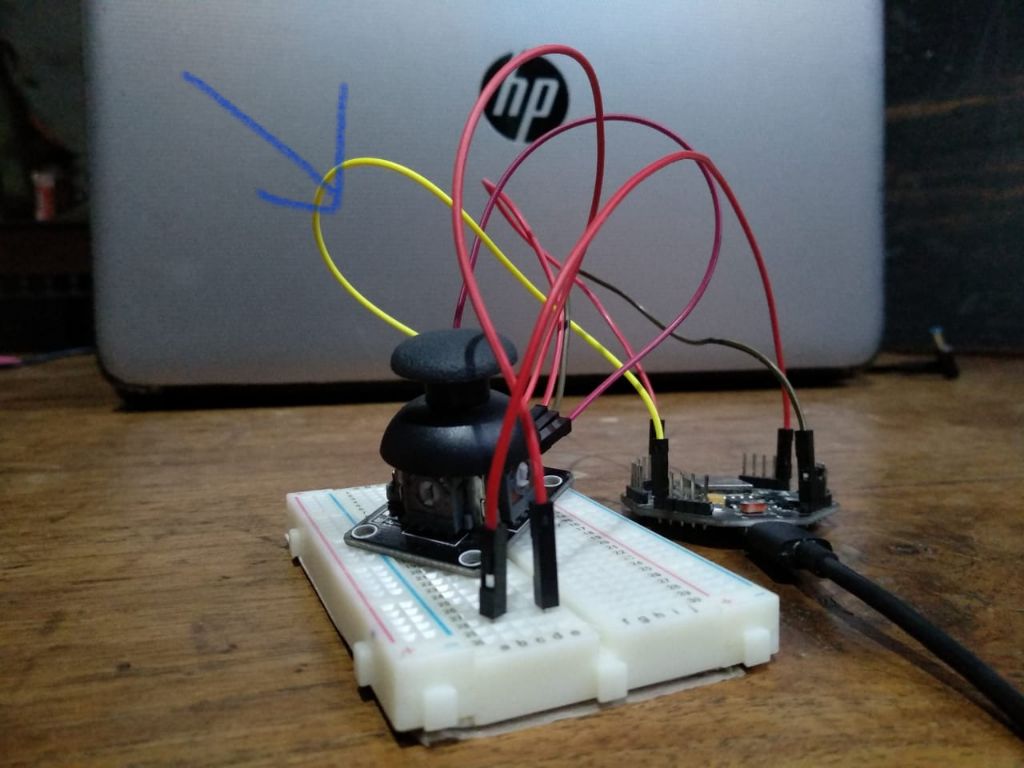
|  |
| --- |
| #include <ESP8266WiFi.h> // including the library  const char\* ssid = "K V MOHAN(SRC)"; //network name  const char\* password = "makemehappy"; //password  WiFiServer server(80); //creating an instance to WiFiServer  void setup() {  pinMode(4,OUTPUT); //declaring a digital pin for light  pinMode(05, OUTPUT); // a digital pin to connect buzzer  Serial.begin(115200); // turning on the Serial monitor  delay(10);  Serial.println();  Serial.println();  Serial.print("Connecting to ");  Serial.println(ssid);  WiFi.mode(WIFI\_STA);  WiFi.begin(ssid, password); // connecting to the network  while (WiFi.status() != WL\_CONNECTED) // loops till it gets connected  {  delay(500);  Serial.println(".");  }  Serial.println("");  Serial.println("WiFi connected");  server.begin(115200); // Start the server  Serial.println("Server started");  Serial.println(WiFi.localIP()); // printing the IP address of server  }  void loop()  {  while(WiFi.status() == WL\_CONNECTED)  {  WiFiClient client = server.available(); // taking the available client and storing in  //client  if (!client) {  return;  }  Serial.println("new client");  while(!client.available()) //waits till the client gets available  {  delay(1);  }  String req = client.readStringUntil('\r'); //reading the request from client  Serial.println(req);  client.flush(); //waits till the characters in buffer of client  // have been sent  int val,val2;  if (req.indexOf("/flash/0") != -1) // checking the request againest several  val = 0; // conditions  else if (req.indexOf("/flash/1") != -1)  val = 1;  else if (req.indexOf("/buzzer/0") != -1)  val2=0;  else if(req.indexOf("/buzzer/1") != -1)  val2=1;  else {  return;  }  digitalWrite(05, val); // writing the pins according to the values  digitalWrite(04,val2);  client.flush();  String s = "done ";  s+=req;  client.print(s); // sending the response to the client  delay(1);  }  Serial.println("wifi disconnected");  } |
|  |

The testing code of the client is:

|  |
| --- |
| #include <ESP8266WiFi.h> //importing the Wi.Fi library  #include <ESP8266HTTPClient.h> //importing the client libraries  #include <BlynkSimpleEsp8266.h> //importing the Blynk libraries  BlynkTimer timer; //creating a timer instance  const char\* ssid = "K V MOHAN(SRC)"; //network name  const char\* password = "makemehappy"; //password of the netowrk  char\* url;  void setup () {  pinMode(A0,INPUT); // pin to connect x axis of joystick  pinMode(13,OUTPUT); // pin to connect the button pin  Blynk.begin("8c8fb4e6b1b54da3b396e695af813c3a",ssid,password);  // connecting to Blynk server through the  // internet  timer.setInterval(2,hello); //setting the timer  Serial.begin(115200);  }  void loop() {  Blynk.run();  timer.run();  }  void hello()  {  int lv;  HTTPClient http; //Declare an object of class HTTPClient  if(analogRead(A0)==1024) //checking the position of the joystick  {  if(digitalRead(13)==0)  {  lv =1;  url="http://192.168.1.100/flash/1"; //preparing the respective url to send  //request  }  else if(digitalRead(13)==1)  {  lv=0;  url="http://192.168.1.100/flash/0";  }  }  else if(analogRead(A0)<=10)  {  Blynk.notify("I AM HERE");  }  else if(digitalRead(5) == 0)  {  url=”http://192.168.1.100/buzzer/1”;  }  else if(digitalRead(5) == 1)  {  url=”http://192.168.1.100/buzzer/0”;  }  digitalWrite(13,lv); //set the LED accordingly  http.begin(url); //sending the request  int httpCode = http.GET(); // Check the returning code  if (httpCode > 0) {  String payload = http.getString(); //Get the request response payload  Serial.println(payload); //Print the response payload  }  } |

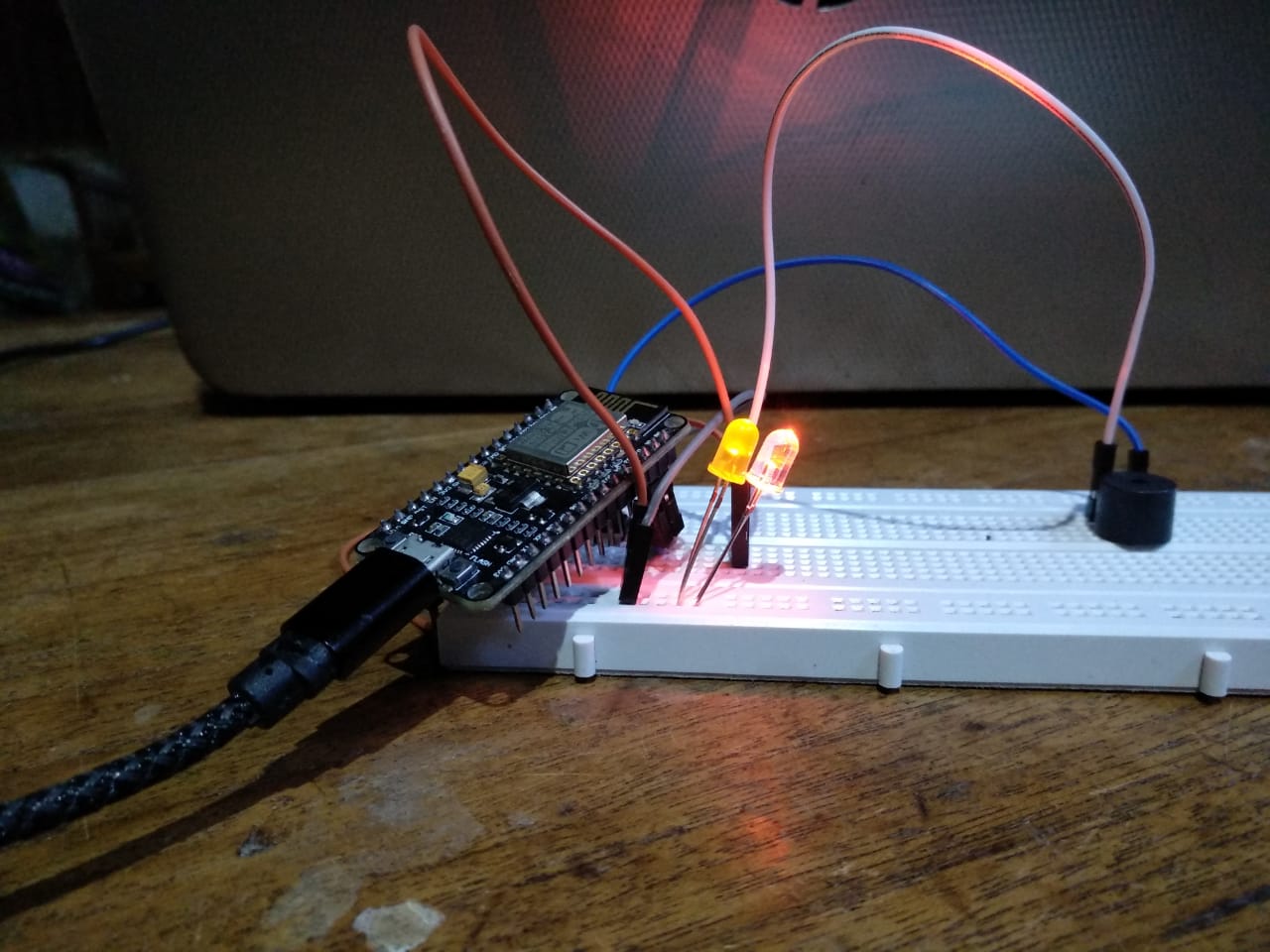
# Final Prototype

The image of the client module after connecting the joystick is as follows:

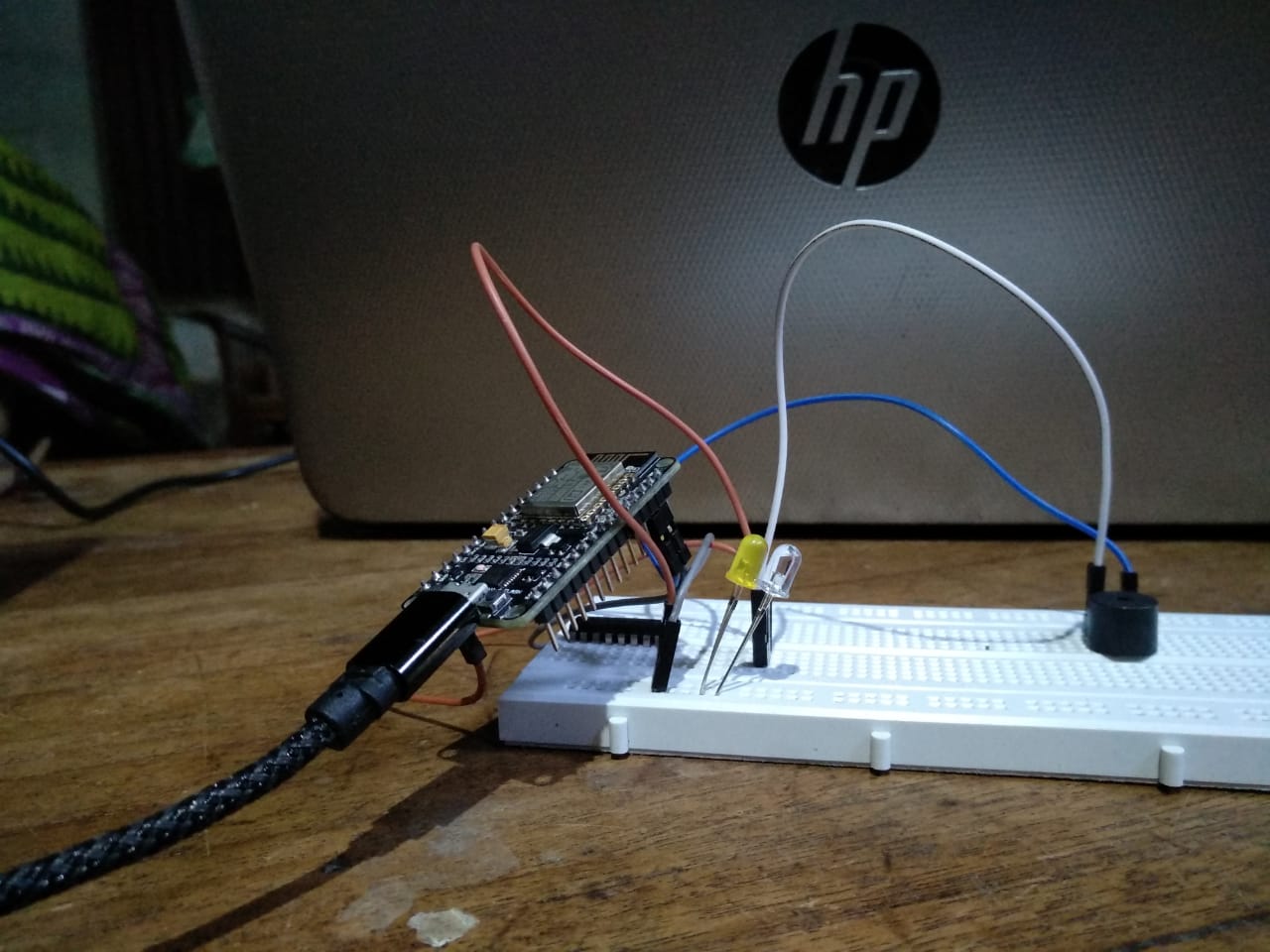


The photos of the server module are as follows:

The below photo shows the server with connections:



When the server lights up the LEDs when it gets a request from client and it is as follors:



# Future scope of the project

As our project is only limited to few specific constraints, it can be extended to some further enhancements as follows:

1. We used only a single axis of the joystick but we can make use of a total of 8 positions and a button of the joystick as a whole. This can be done with a multiplexer connected to the joystick, but this enhancement increases the circuitry and a little complexity in the code.

2. The other development that can be expected in our project is that the usage of the client and server on the different networks. As we know that out **HANDY HELPER** are limited by the network. This enhancement increases the cost of the project.

3. The major enhancement of the **HANDY HELPER** is taking them to each and every domain. This means we cannot limit the usage of a **HANDY HELPER** to a home but we canuse it in the other fields like agricultural fields, hospitals, schools factories and many other places to make thing more easier for us everywhere.

# Constraints for real time deployment

# Our project requires the following constraints for deploying it in the real time:

1. +5V power supply for both server and client,

2. The server and client must be connected to a common network and

3. It is advisable to place the server near the router to get requests faster from clients.

# References

<https://www.youtube.com/watch?v=qDq7vm-7Xs4&t=2s>

<https://randomnerdtutorials.com/esp8266-web-server/>

<https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/>