nodeMCU-esp8266 & Raspberry Pi Multi Irrigation (PART 1)

Introduction:

Project was requested by a college student seeking to learn and develop an IoT projects. This project consists of 2 nodeMCU esp8266 v1.0 and a raspberry pi zero w as it's processing units. Aim of the project is to water multiple (3) indoor plant beds all located at different regions of the apartment. Along with watering the plant beds we will also monitor temperature, light intensity, soil moisture levels, security camera and have a remote monitor & control website with user login. Bonus with telegram bot.

Hardware Components:

1.	Raspberry Pi Zero W	x1
2.	32GB Micro SD	x1
3.	Raspberry Pi Zero W Case	x1 (with camera mount)
4.	Amazon Basics micro USB	x1
5.	Raspberry Pi 5MP Camera	x1
6.	12V 5A Power Supply	x1
7.	nodeMCU esp8266 V1.0	x2
8.	Soil Moisture Sensor	x6
9.	LM35	x3
10.	LDR R10k	x4
11.	HCSR-04	x1
12.	12V Water Pump	x3 (can be switched with regular AC Pump (max 240v))
13.	5V Relay	x3
14.	74HC4067 Analog MUX	x1
15.	Wires	
16.	Prototype PCB 4x4	x2

Software Tools & Technologies:

All this tools/software are available for Windows, Mac and Linux

1. Raspbian Buster OS (you can also use lite version)
2. SSH
3. Python
4. MariaDB
5. Apache2
6. PHPMYadmin
7. PHP 7
8. Remote.it Service
9. Arduino CC
10. easyEDA

Basic Setup:

To setup your Raspberry Pi: RPi Setup

To setup ArduinoCC with nodeMCU: nodeMCU Setup

To setup MariaDB server on Raspberry Pi: Server Setup

Designing the circuit (step 1):

Before designing let us have a look at the nodeMCU esp8266, we notice in 'Figure – 01' that it has only one analog pin and we have 13 analog sensors. To connect so many sensors we will use an analog 16:1 multiplexer.

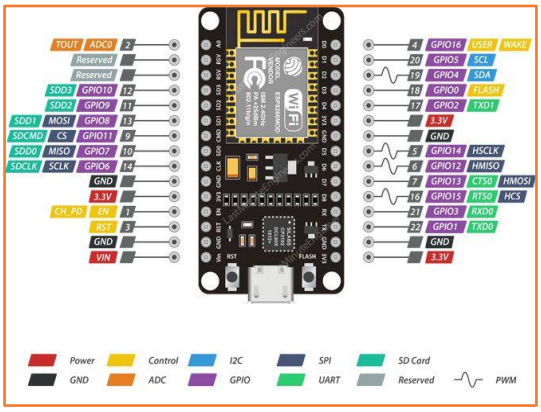
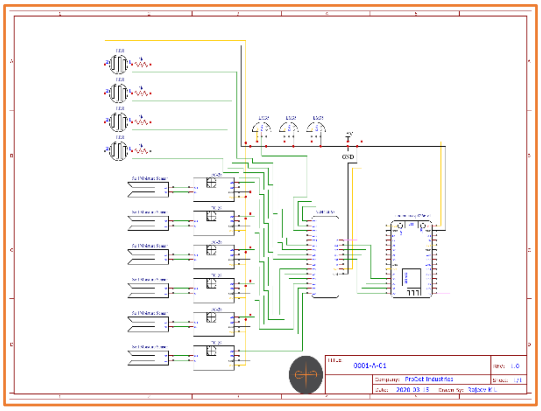


Figure - 01

We will design the circuit in easyEDA. Create a new project. In the libraries search for nodemcu, this should give you a list of existing designs. Select one and make the connections as shown in 'Circuit Design – 01'. Save the project and build the prototype PCB accordingly.



Circuit Design - 01

Programming the nodemcu (step 2):

Let us program the nodemcu to read the sensors through the multiplexer. Observe in the 'Code Snippet – 01', I have used 3 arrays to store values of different sensors. To read the values I am using a 'readMUX( )' function which modifies the control pins to access channel pins. The 3 loops are used to traverse through the MUX/DEMUX and read sensor values.

Upon executing this code without connecting any sensors, you should get all sensor values as zero. After connecting all the sensors with the MUX, new set of values will be visible in the serial monitor.

```
//Functions for reading sensor inputs
void read() {
    float temp[3];
    int moist[6];
    int ldr[4];
    int sensorValue;
    float temporary = (3.3/1024)*100;
    int sid1 = 1001;//sensor id for moisture sensor
    int sid2 = 2001;//sensor id for light sensor
    int sid3 = 3001;//sensor id for temperature sensor

    for (int i = 1; i < 7; i++) {
        moist[i-1] = readMUX(i);//reading moisture sensor value
        super[i-1][0] = (float)moist[i-1];//assigning value to the array
        super[i-1][1] = sid1++;
    }
    for (int i = 8; i < 12; i++) {
        ldr[i-8] = readMUX(i);//reading moisture sensor value
        super[i-2][0] = (float)ldr[i-8];//assigning value to the array
        super[i-2][1] = sid2++;
    }
    for (int i = 13; i < 16; i++) {
        sensorValue = readMUX(i);//reading moisture sensor value
        sensorValue *= temporary;//caliberating analog value to Celsius
        temp[i-13] = sensorValue;
        super[i-3][0] = (float)temp[i-13];//assigning value to the array
        super[i-3][1] = sid3++;
    }
}
```

Code Snippet - 01

Programming on Raspberry Pi (step 3):

In the raspberry pi, open the phpMyAdmin page to create a MariaDB schema. Create a table 'EventTable' with 4 attributes – Event ID, Sensor ID, Sensor Value, EventTime.

Now let us create a php script to accept the values from the nodemcu and insert the data to the table that we created. Create a new file named 'postEspData.php'. Initialize the server name, DB username and password. To accept values from a client application/device we will use the 'POST' request method. To verify that no random user can update data, we will use a key which is known only to our devices and server. Once all the necessary values are acquired, we connect to MariaDB. When the connection is established, we insert the acquired data into the 'EventTable'.

```php
<?php
    //
    $servername = "localhost";
    $username = "alpha";
    $password = "alpha";
    $dbname = "Project";
    $apiKeyInternal = "1qaz2wsx3edc";
    $apiKeyExternal = $SensorID = $SensorValue = "";

    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $apiKeyExternal = test_input($_POST["apikey"]);
        if ($apiKeyExternal == $apiKeyInternal) {
            $SensorID = test_input($_POST["SensorID"]);
            $SensorValue = test_input($_POST["SensorValue"]);

            $conn = new mysqli($servername,$username,$password,$dbname);
            if ($conn->connect_error) {
                die("connection failed" . $conn->connect_error);
            }
            $sql = "INSERT INTO `EventTable` (`Sensor ID`, `Sensor Value`) VALUES ($SensorID,$SensorValue)";
            if ($conn->query($sql) === TRUE) {
                echo "new record";
            }
            else {
                echo "Error: " . $sql . "<br>" . $conn->error;
            }
            $conn->close();
        }
        else {
            echo "Wrong API";
        }
    }
    else {
        echo "no data posted";
    }

    function test_input($data) {
        $data = trim($data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
        return $data;
    }
?>
```

Code Snippet - 02

Connecting the nodemcu to Server (step 4):

To connect we need to use WiFiClient and ESP8266WiFi libraries. Start the server connection with http.begin(serverName), define the type of content in the header. Append the 'Sensor ID', 'Sensor Value' and 'apiKey' to the httpRequestData. POST this request data to the server, if the upload is successful, you will get a response code of 200.

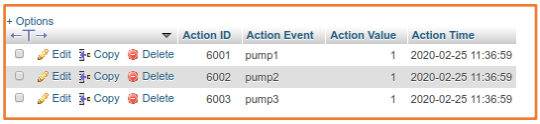
```
char serverName[100];
String apiKeyValue = "1qaz2wsx3edc";
char requestData[] = "apikey%s &SensorID%f &Sensorvalue%f";

void publish() {
    for (int i = 0; i < 13; i++) {
        HTTPClient http;
        http.begin(serverName);
        http.addHeader("Content-Type", "application/x-www-form-urlencoded");
            String httpRequestData = "apikey=" + apiKeyValue + "&SensorID="
            + (int)super[i][1] + "&SensorValue=" + (int)super[i][0];
            Serial.print("http request data:");
            Serial.println(httpRequestData);
            int httpResponseCode = http.POST(httpRequestData);
            if (httpResponseCode > 0) {
                Serial.print("Response Code:");
                Serial.println(httpResponseCode);
            }
            else {
                Serial.print("Error Code:");
                Serial.println(httpResponseCode);
            }
        http.end();
    }
}
```

Code Snippet - 03

Programming MariaDB (step 5):

Now we will write a procedure to identify low soil moisture levels and high moisture levels. With this we can control the state of the pumps.

First create a new table 'ActionTable' with 4 attributes – Action ID, Action Event, Action Value, Action Time. Add 3 records for 3 pumps, set Action Value as 0.

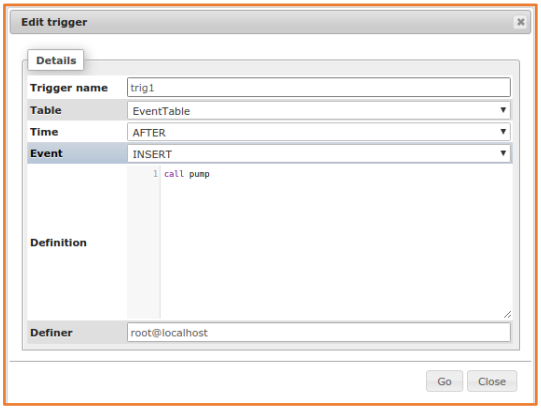


+ Options

	Action ID	Action Event	Action Value	Action Time
Edit Copy Delete	6001	pump1	1	2020-02-25 11:36:59
Edit Copy Delete	6002	pump2	1	2020-02-25 11:36:59
Edit Copy Delete	6003	pump3	1	2020-02-25 11:36:59

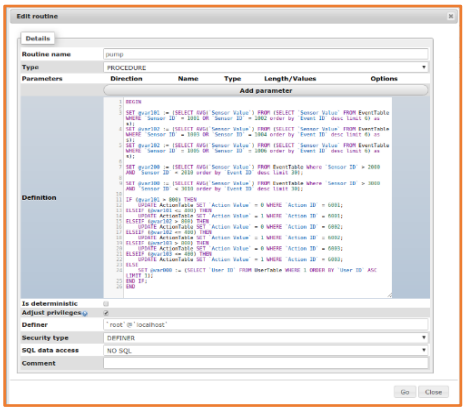
SQL - 01

Now go to 'EventTable' and create a trigger with 'Time' as 'AFTER' and 'Event' as 'INSERT'. In the trigger write 'call pump()'.



SQL – 02

Now let's create a routine with 'Type' as 'Procedure'. Select the average of last 3 readings of pair of sensors. Using this average value, we will update the 'ActionTable' accordingly. I have used lower limit as 400 (min 0) to switch ON the pump and a upper limit of 800 (max 1023) to switch OFF the pump.



SQL – 03

To verify if this procedure is working, insert values manually into the 'EventTable' and check the 'ActionTable' for changes.

This should complete our Main Objective of watering 3 separate soil beds when the soil moisture levels are low and automatically turn off after watering.

nodeMCU-esp8266 & Raspberry Pi Multi Irrigation (PART 2)

Introduction:

To polish this project to a product we need to add many small features which makes it more user friendly and easily accessible.

Features to be included:

1. Connect nodemcu to any Wi-Fi without changing the source code
2. Website to view all sensor status and control pumps with user login
3. Remote access for website (without website registration)
4. Security camera with real-time security alerts with Telegram Bot

Connect to any Wi-Fi:

Before we connect to Wi-Fi, we need to cover another feature called EEPROM. The nodemcu has 4MB of internal flash memory. To access this memory we use the EEPROM library. Initiate the memory size required by using EEPROM.begin(size_in_bytes). In our case the max size would be 150 bytes. We will use this memory to save details like Wi-Fi SSID, Wi-Fi Password and Server IP.

If you have used any smart devices, at setup we setup our home Wi-Fi and password into the device. To do this we need to create an internal server (hotspot) which can be connected by any external device.

```
96  void launchWeb() {
97      Serial.println("");
98      if (WiFi.status() == WL_CONNECTED)
99          Serial.println("WiFi connected");
00      Serial.print("Local IP: ");
01      Serial.println(WiFi.localIP());
02      Serial.print("SoftAP IP: ");
03      Serial.println(WiFi.softAPIP());
04      createWebServer();
05      // Start the server
06      server.begin();
07      Serial.println("Server started");
08  }
```

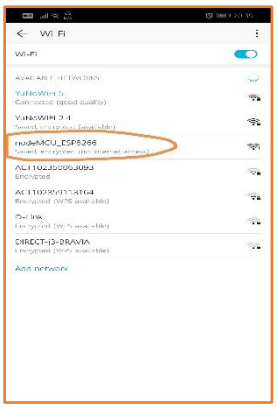
Code Snippet – 01

To make a hotspot, we need to create 3 functions: launchWeb(), createWebServer() and setupAP(). The launchWeb() will setup the internal IP to which we can connect using an external device. The createWebServer() hosts a webpage where we can input our Wi-Fi credentials and this data is written into the EEPROM. The launchWeb() creates the hotspot and also provides a feature to scan other Wi-Fi/hotspot.

After uploading the code observe the serial monitor, the Wi-Fi will fail to connect and it will create a new hotspot “nodeMCU_ESP8266” without any password.

```
Server IP:192.168.1.5.
Waiting for Wifi to connect
********************
Connect timed out, opening AP
Turning the HotSpot On

Local IP: (IP unset)
SoftAP IP: 192.168.4.1
Server started
scan done
2 networks found
1: YuNoWiFi 2.4 (-68)*
2: ACT102350063093 (-92)*

softap

Local IP: (IP unset)
SoftAP IP: 192.168.4.1
Server started
over

Waiting.
.....................................
```

Serial Monitor – 01



Browser – 01

Now connect to this hotspot and go to 192.168.4.1 in your phone browser. Type in your Wi-Fi SSID, password and Server IP (Raspberry Pi’s IP) and hit submit. After you hit submit, the webpage will fail to load and your connection to “nodeMCU_ESP8266” hotspot will be disconnected. In the serial monitor you will see that the values you entered in the webpage will be written to the EEPROM.

```
writing eeprom ssid:
Wrote: Y
Wrote: u
Wrote: N
Wrote: o
Wrote: W
Wrote: i
Wrote: F
Wrote: i
```

Serial Monitor – 02

```
writing eeprom pass:
Wrote: c
Wrote: a
Wrote: s
Wrote: i
Wrote: n
Wrote: o
Wrote:
Wrote: R
```

Serial Monitor – 03

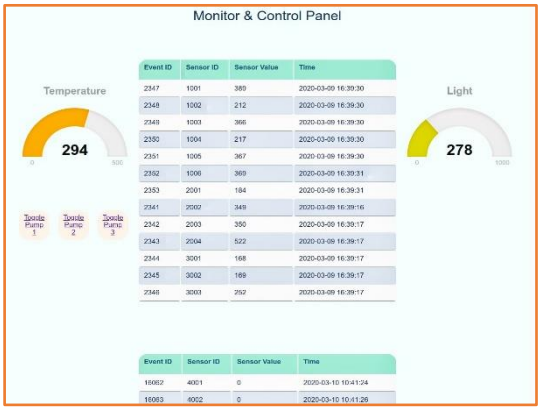
```
writing server IP:
Wrote:1
Wrote:9
Wrote:2
Wrote:.
Wrote:1
Wrote:6
Wrote:8
Wrote:.
```

Serial Monitor – 04

After writing all the credentials to the EEPROM, the nodemcu will reboot and connect to the newly set Wi-FI and Server IP. Immediately it will start reading the sensors and publish it.

Webpage to view sensor values and control pumps:

To view the sensor values in the webpage, we will create a php script which will fetch these details from the MariaDB. To make it easy on the php scripting we will create a 'VIEW' in MariaDB which filters the latest values of the sensors. Using this 'VIEW' we will display it on the webpage. The webpage code is written in the 'main.php' file. Now place this file inside /var/www/html in the Raspberry Pi. Open a terminal in Raspberry Pi and type 'ifconfig' this will give you the IP of Raspberry Pi. On your phone browser go to 'Raspberry Pi IP/main.php'. This will redirect you back to the login page, after login you will be able to see 3 tables and 2 meters.



Browser – 02

Remote Website:

To setup remote access to website, we need to setup a 'remote.it' service on the Raspberry Pi. Follow this link remote.it to setup the remote web server on RaspBerry Pi. Once the setup is done, login to your remote.it account and click on the device, in the pop-up you should see HTTP service active. Click on it and you will get a website link. This link can be accessed from any network.

Camera Security and Telegram Bot:

First step is to create a telegram account which is available for Android and Apple devices.

To create a telegram bot first search 'BotFather' in telegram app. Now type '/newbot' in the chat box. 'BotFather' will guide you into creating the bot. After Creating the bot, 'BotFather' will provide you a 'HTTP API' token. Make a copy of this. Now search your bot in telegram app, type 'hi'. This will be used to identify chat id. Now run the chatID.py in Raspberry Pi and type '/hi' in the telegram app. This will print a 'chat_ID' in the Raspberry Pi terminal. Copy this 'chat_ID' to the FinalCamera.py. Execute the FinalLauncherScript.sh to run this automatically.

Before we start recording the camera, we need to check the condition for entry/exit identified by the HCSR04 sensors from the MariaDB. When the conditions are true, the camera will record for 20 seconds with a framerate of 20fps. This will be sent to the telegram bot that you created earlier.