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.

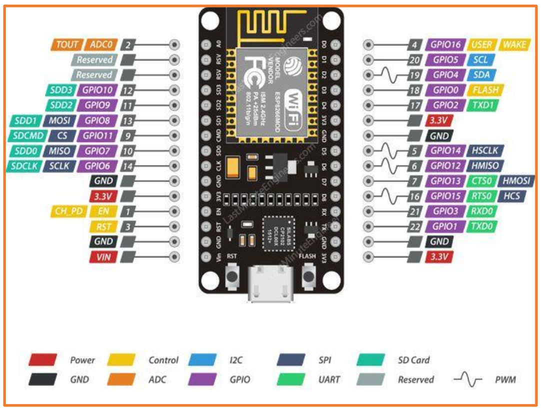
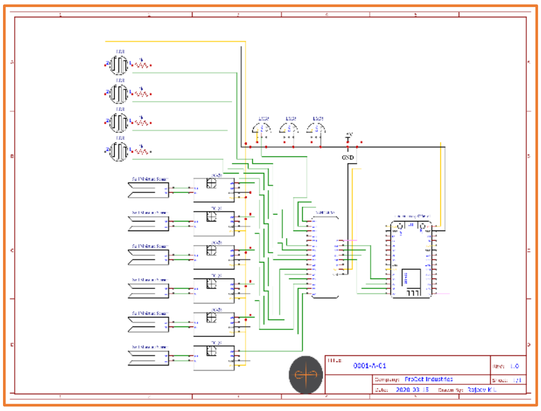
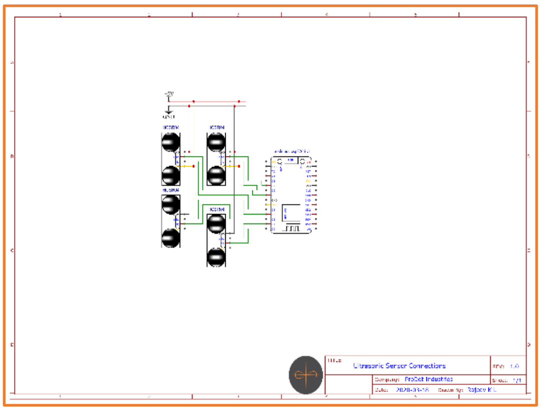


Figure - 01

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

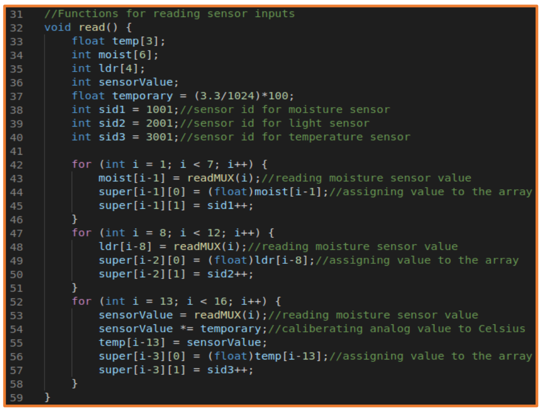


Circuit Design - 02

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.



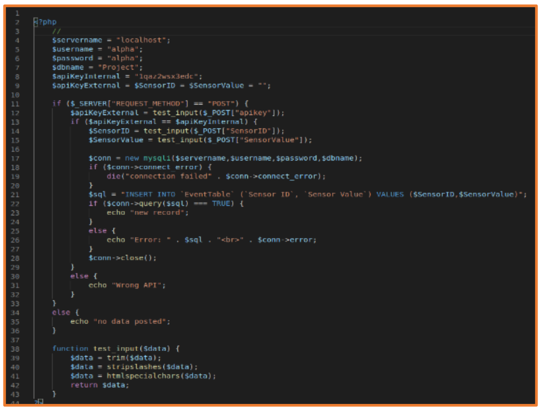
```
//Functions for reading sensor inputs
void read() {
    float temp[3];
    int moist[6];
    int ldr[4];
    int sensorValue;
    float temporary = (3.3/1024)*100;
    int sid1 = 1001;//sensor id for moisture sensor
    int sid2 = 2001;//sensor id for light sensor
    int sid3 = 3001;//sensor id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

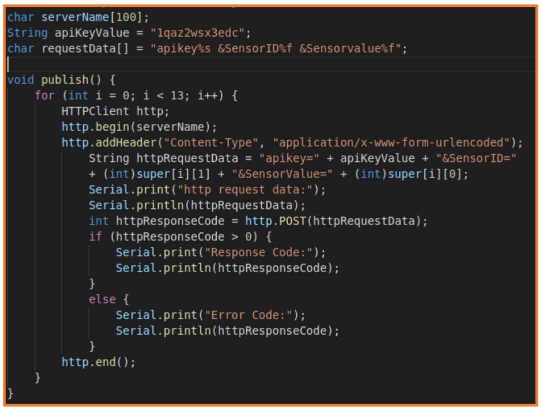


```
<?php
    //
    $servername = "localhost";
    $username = "alpha";
    $password = "alpha";
    $dbname = "Project";
    $apiKeyInternal = "1qaz2wsx3edc";
    $apiKeyExternal = $SensorID = $SensorValue = "";

    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.



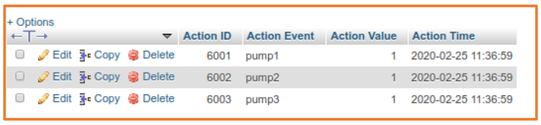
```
char serverName[100];
String apiKeyValue = "1qaz2wsx3edc";
char requestData[] = "apikey%s &SensorID%f &Sensorvalue%f";

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.

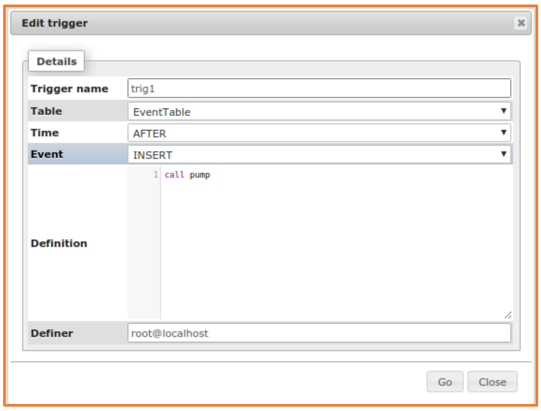
First create a new table 'ActionTable' with 4 attributes – Action ID, Action Event, Action Value, Action Time. Add 3 records for 3 pumps, set Action Value as 0.



←T→		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()'.



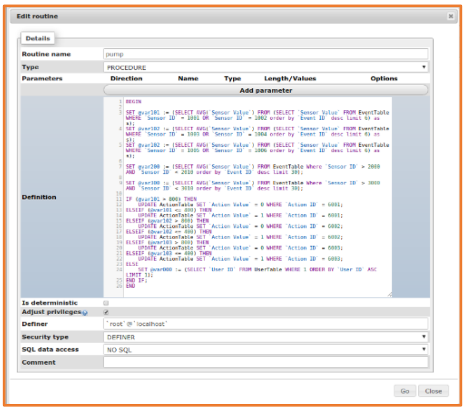
Trigger name: trig1
Table: EventTable
Time: AFTER
Event: INSERT
Definition:

```
call pump
```

Definer: root@localhost

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.



Routine name: pump
Type: PROCEDURE
Definition:

```
BEGIN

SET @var101 := (SELECT AVG(`Sensor Value`) FROM (SELECT `Sensor Value` FROM EventTable WHERE `Sensor ID` = 1001 OR `Sensor ID` = 1002 order by `Event ID` desc limit 6) as s);
SET @var102 := (SELECT AVG(`Sensor Value`) FROM (SELECT `Sensor Value` FROM EventTable WHERE `Sensor ID` = 1003 OR `Sensor ID` = 1004 order by `Event ID` desc limit 6) as s);
SET @var102 := (SELECT AVG(`Sensor Value`) FROM (SELECT `Sensor Value` FROM EventTable WHERE `Sensor ID` = 1005 OR `Sensor ID` = 1006 order by `Event ID` desc limit 6) as s);

SET @var200 := (SELECT AVG(`Sensor Value`) FROM EventTable Where `Sensor ID` > 2000 AND `Sensor ID` < 2010 order by `Event ID` desc limit 30);

SET @var300 := (SELECT AVG(`Sensor Value`) FROM EventTable Where `Sensor ID` > 3000 AND `Sensor ID` < 3010 order by `Event ID` desc limit 30);

IF (@var101 > 800) THEN
    UPDATE ActionTable SET `Action Value` = 0 WHERE `Action ID` = 6001;
ELSEIF (@var101 <= 400) THEN
    UPDATE ActionTable SET `Action Value` = 1 WHERE `Action ID` = 6001;
ELSEIF (@var102 > 800) THEN
    UPDATE ActionTable SET `Action Value` = 0 WHERE `Action ID` = 6002;
ELSEIF (@var102 <= 400) THEN
    UPDATE ActionTable SET `Action Value` = 1 WHERE `Action ID` = 6002;
ELSEIF (@var103 > 800) THEN
    UPDATE ActionTable SET `Action Value` = 0 WHERE `Action ID` = 6003;
ELSEIF (@var103 <= 400) THEN
    UPDATE ActionTable SET `Action Value` = 1 WHERE `Action ID` = 6003;
ELSE
    SET @var000 := (SELECT `User ID` FROM UserTable WHERE 1 ORDER BY `User ID` ASC LIMIT 1);
END IF;
END
```

Is deterministic: □
Adjust privileges: ☑
Definer: `root`@`localhost`
Security type: DEFINER
SQL data access: NO SQL
Comment:

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.

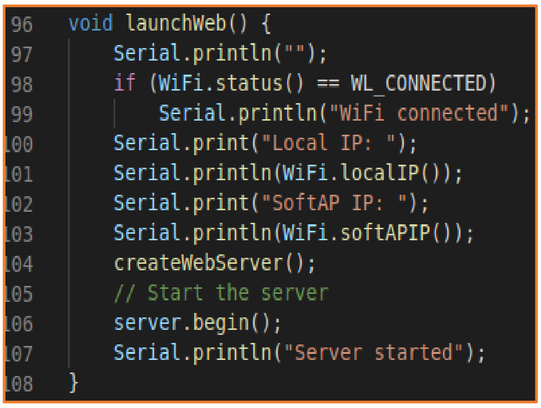
Features to be included:

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.

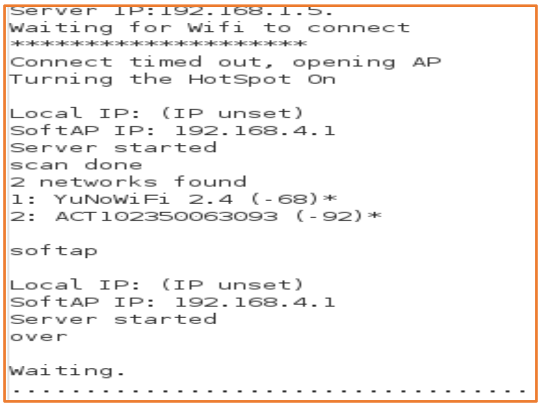


```
void launchWeb() {
    Serial.println("");
    if (WiFi.status() == WL_CONNECTED)
        Serial.println("WiFi connected");
    Serial.print("Local IP: ");
    Serial.println(WiFi.localIP());
    Serial.print("SoftAP IP: ");
    Serial.println(WiFi.softAPIP());
    createWebServer();
    // Start the server
    server.begin();
    Serial.println("Server started");
}
```

Code Snippet – 01

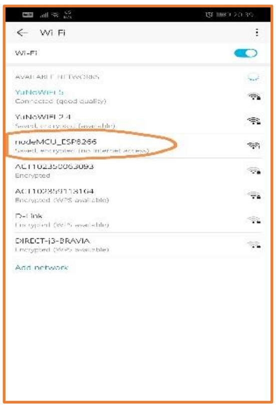
To make a hotspot, we need to create 3 functions: launchWeb(), createWebServer() and setupAP(). The launchWeb() will setup the internal IP to which we can connect using an external device. The createWebServer() hosts a webpage where we can input our Wi-Fi credentials and this data is written into the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



Wi Fi

Wi-Fi

AVAILABLE NETWORKS

YuNoWiFi 5
Connected (good quality)

YuNoWIFI 2.4
Saved, encrypted (available)

nodeMCU_ESP8266
Saved, encrypted (no internet access)

ACT102350063093
Encrypted

ACT102591113164
Encrypted (WPS available)

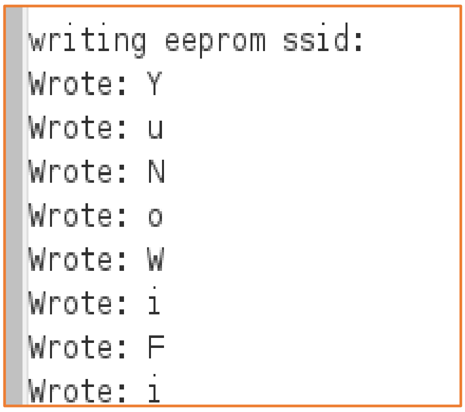
D-Link
Encrypted (WPS available)

DIRECT-j3-BRAVIA
Encrypted (WPS available)

Add network

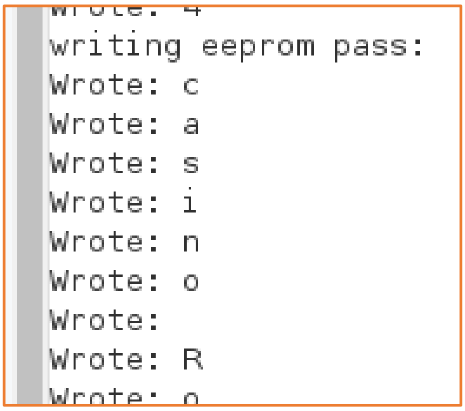
Browser – 01

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.



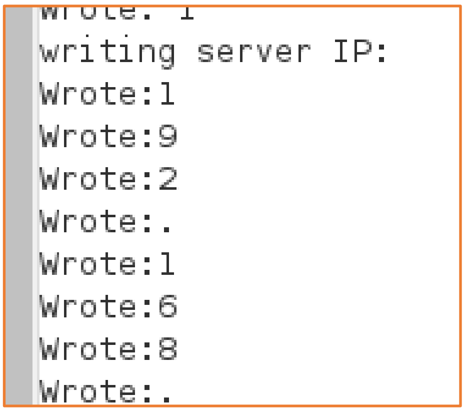
```
writing eeprom ssid:
Wrote: Y
Wrote: u
Wrote: N
Wrote: o
Wrote: W
Wrote: i
Wrote: F
Wrote: i
```

Serial Monitor – 02



```
Wrote: 4
writing eeprom pass:
Wrote: c
Wrote: a
Wrote: s
Wrote: i
Wrote: n
Wrote: o
Wrote:
Wrote: R
Wrote: o
```

Serial Monitor – 03



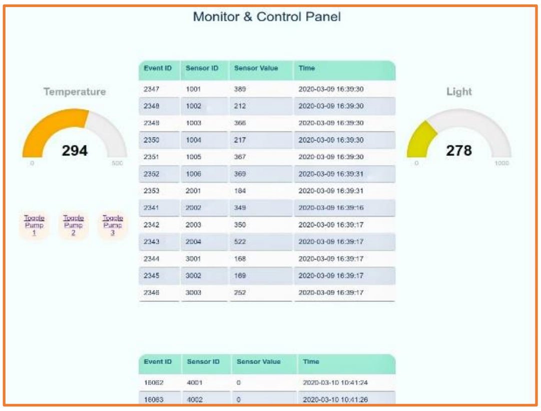
```
Wrote: 1
writing server IP:
Wrote:1
Wrote:9
Wrote:2
Wrote:.
Wrote:1
Wrote:6
Wrote:8
Wrote:.
```

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.



Monitor & Control Panel

Event ID	Sensor ID	Sensor Value	Time
2347	1001	389	2020-03-09 16:39:30
2348	1002	212	2020-03-09 16:39:30
2349	1003	366	2020-03-09 16:39:30
2350	1004	217	2020-03-09 16:39:30
2351	1005	367	2020-03-09 16:39:30
2352	1006	369	2020-03-09 16:39:31
2353	2001	184	2020-03-09 16:39:31
2341	2002	349	2020-03-09 16:39:16
2342	2003	350	2020-03-09 16:39:17
2343	2004	522	2020-03-09 16:39:17
2344	3001	168	2020-03-09 16:39:17
2345	3002	169	2020-03-09 16:39:17
2346	3003	252	2020-03-09 16:39:17

Event ID	Sensor ID	Sensor Value	Time
16062	4001	0	2020-03-10 10:41:24
16063	4002	0	2020-03-10 10:41:26

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.

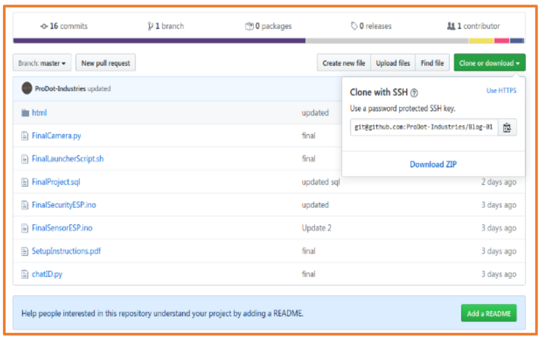
To create a telegram bot first search 'BotFather' in telegram app. Now type '/newbot' in the chat box. 'BotFather' will guide you into creating the bot. After Creating the bot, 'BotFather' will provide you a 'HTTP API' token. Make a copy of this. Now search your bot in telegram app, type 'hi'. This will 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.

nodeMCU-esp8266 & Raspberry Pi Multi Irrigation (PART 3)

Installing the Project:

Step 1: Download the project from GitHub



16 commits · 1 branch · 0 packages · 0 releases · 1 contributor

ProDot-Industries updated

html	updated	
FinalCamera.py	final	
FinalLauncherScript.sh	final	
FinalProject.sql	updated sql	2 days ago
FinalSecurityESP.ino	updated	3 days ago
FinalSensorESP.ino	Update 2	3 days ago
SetupInstructions.pdf	final	3 days ago
chatID.py	final	3 days ago

Clone with SSH ⓘ

Use a password protected SSH key.

```
git@github.com:ProDot-Industries/Blog-01
```

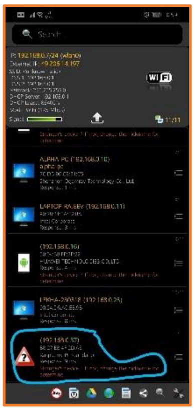
Help people interested in this repository understand your project by adding a README.

Browser – 01

Step 2: Flash the .ino files into the 2 nodeMCUs

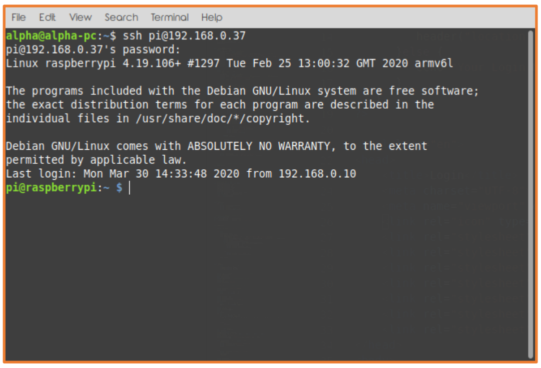
Step 3: Build the circuit as shown in ‘Circuit Connection – 1’ & ‘Circuit Connection – 2’ for both nodeMCUs.

Step 4: Boot the Raspberry Pi, and connect using ‘ssh’. To find the IP of Raspberry Pi, you can use an application called ‘Network Scanner’ (available for Android in Play Store)



Network Scanner - 01

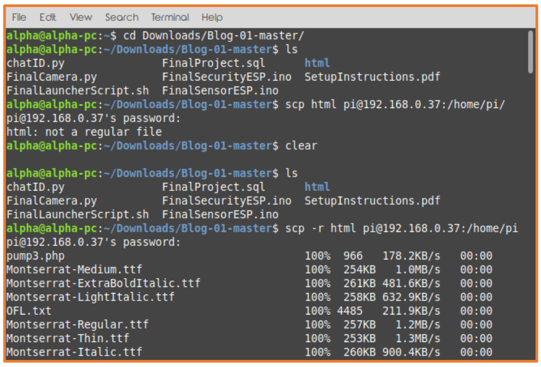
Open your terminal and type ‘ssh pi@192.168.0.37’ (ssh username@local-IP). It should log into the Raspberry Pi terminal



```
alpha@alpha-pc:~$ ssh pi@192.168.0.37
pi@192.168.0.37's password:
Linux raspberrypi 4.19.106+ #1297 Tue Feb 25 13:00:32 GMT 2020 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Mar 30 14:33:48 2020 from 192.168.0.10
pi@raspberrypi:~ $ |
```

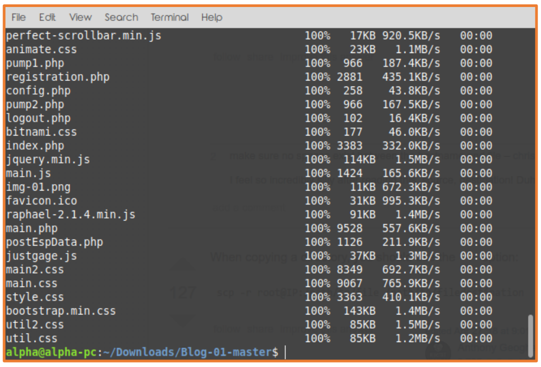
Terminal 1 - 01

Step 5: Now copy the ‘html’ folder from the downloaded github source to Raspberry Pi using ‘scp -r html pi@192.168.0.37:/home/pi/’. Do this in a different terminal



```
alpha@alpha-pc:~$ cd Downloads/Blog-01-master/
alpha@alpha-pc:~/Downloads/Blog-01-master$ ls
chatID.py               FinalProject.sql      html
FinalCamera.py          FinalSecurityESP.ino  SetupInstructions.pdf
FinalLauncherScript.sh  FinalSensorESP.ino
alpha@alpha-pc:~/Downloads/Blog-01-master$ scp html pi@192.168.0.37:/home/pi/
pi@192.168.0.37's password:
html: not a regular file
alpha@alpha-pc:~/Downloads/Blog-01-master$ clear

alpha@alpha-pc:~/Downloads/Blog-01-master$ ls
chatID.py               FinalProject.sql      html
FinalCamera.py          FinalSecurityESP.ino  SetupInstructions.pdf
FinalLauncherScript.sh  FinalSensorESP.ino
alpha@alpha-pc:~/Downloads/Blog-01-master$ scp -r html pi@192.168.0.37:/home/pi
pi@192.168.0.37's password:
pump3.php                                     100%  966   178.2KB/s   00:00
Montserrat-Medium.ttf                         100%  254KB   1.0MB/s   00:00
Montserrat-ExtraBoldItalic.ttf                100%  261KB 481.6KB/s   00:00
Montserrat-LightItalic.ttf                    100%  258KB 632.9KB/s   00:00
OFL.txt                                       100% 4485   211.9KB/s   00:00
Montserrat-Regular.ttf                        100%  257KB   1.2MB/s   00:00
Montserrat-Thin.ttf                           100%  253KB   1.3MB/s   00:00
Montserrat-Italic.ttf                         100%  260KB 900.4KB/s   00:00
```

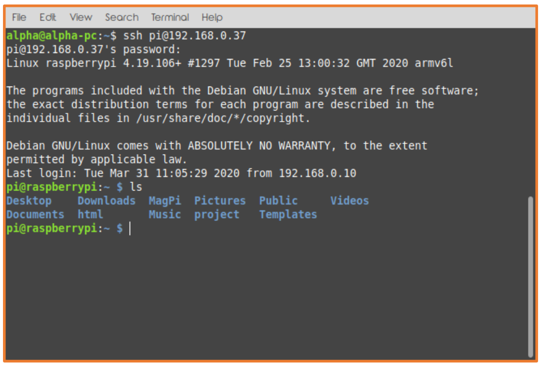
Terminal 2 - 01



```
perfect-scrollbar.min.js                      100%   17KB 920.5KB/s   00:00
animate.css                                   100%   23KB   1.1MB/s   00:00
pump1.php                                     100%  966   187.4KB/s   00:00
registration.php                              100% 2881   435.1KB/s   00:00
config.php                                    100%  258    43.8KB/s   00:00
pump2.php                                     100%  966   167.5KB/s   00:00
logout.php                                    100%  102    16.4KB/s   00:00
bitnami.css                                   100%  177    46.0KB/s   00:00
index.php                                     100% 3383   332.0KB/s   00:00
jquery.min.js                                 100%  114KB   1.5MB/s   00:00
main.js                                       100% 1424   165.6KB/s   00:00
img-01.png                                    100%   11KB 672.3KB/s   00:00
favicon.ico                                   100%   31KB 995.3KB/s   00:00
raphael-2.1.4.min.js                          100%   91KB   1.4MB/s   00:00
main.php                                      100% 9528   557.6KB/s   00:00
postEspData.php                               100% 1126   211.9KB/s   00:00
justgage.js                                   100%   37KB   1.3MB/s   00:00
main2.css                                     100% 8349   692.7KB/s   00:00
main.css                                      100% 9067   765.9KB/s   00:00
style.css                                     100% 3363   410.1KB/s   00:00
bootstrap.min.css                             100%  143KB   1.4MB/s   00:00
util2.css                                     100%   85KB   1.5MB/s   00:00
util.css                                      100%   85KB   1.2MB/s   00:00
alpha@alpha-pc:~/Downloads/Blog-01-master$ |
```

Terminal 2 - 02

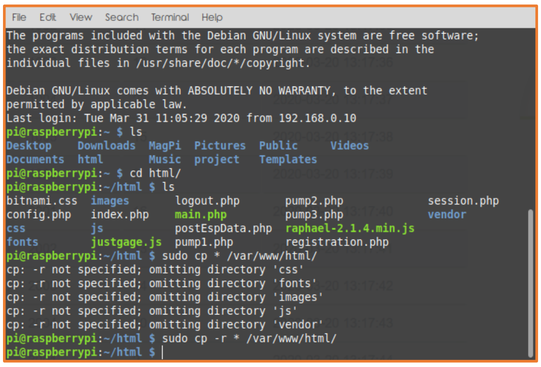
Step 6: To verify if the files have actually been copied, open the Raspberry Pi connected terminal and type 'ls', there should be a folder called 'html'.



```
alpha@alpha-pc:~$ ssh pi@192.168.0.37
pi@192.168.0.37's password:
Linux raspberrypi 4.19.106+ #1297 Tue Feb 25 13:00:32 GMT 2020 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 31 11:05:29 2020 from 192.168.0.10
pi@raspberrypi:~ $ ls
Desktop    Downloads  MagPi  Pictures  Public     Videos
Documents  html       Music  project   Templates
pi@raspberrypi:~ $ |
```

Terminal 1 - 02

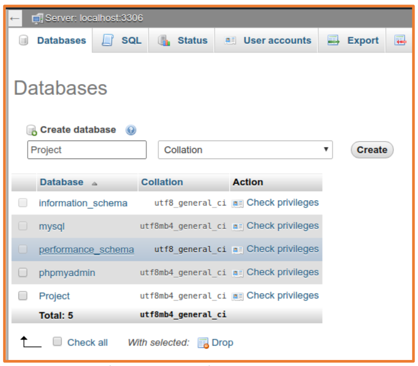
Step 7: Copy the contents of this folder to /var/www/html. Use 'sudo cp -r * /var/www/html' to copy. If you do not mention '-r' it will not copy the directories and its sub files/directories.



```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 31 11:05:29 2020 from 192.168.0.10
pi@raspberrypi:~ $ ls
Desktop    Downloads  MagPi  Pictures  Public     Videos
Documents  html       Music  project   Templates
pi@raspberrypi:~ $ cd html/
pi@raspberrypi:~/html $ ls
bitnami.css  images       logout.php       pump2.php             session.php
config.php   index.php    main.php         pump3.php             vendor
css          js           postEspData.php  raphael-2.1.4.min.js
fonts        justgage.js  pump1.php        registration.php
pi@raspberrypi:~/html $ sudo cp * /var/www/html/
cp: -r not specified; omitting directory 'css'
cp: -r not specified; omitting directory 'fonts'
cp: -r not specified; omitting directory 'images'
cp: -r not specified; omitting directory 'js'
cp: -r not specified; omitting directory 'vendor'
pi@raspberrypi:~/html $ sudo cp -r * /var/www/html/
pi@raspberrypi:~/html $ |
```

Terminal 1 - 03

Step 8: Create a new 'schema' in 'phpMyAdmin' of Raspberry Pi with the name 'Project'. To open 'phpMyAdmin', open your browser and type '192.168.0.37/phpmyadmin' (Raspberry Pi-IP/phpmyadmin).



Databases

Create database

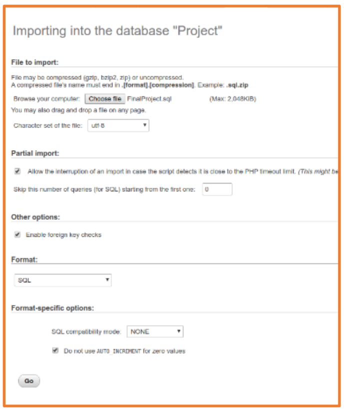
Project | Collation ▾ | Create

	Database	Collation	Action
□	information_schema	utf8_general_ci	Check privileges
□	mysql	utf8mb4_general_ci	Check privileges
□	performance_schema	utf8_general_ci	Check privileges
□	phpmyadmin	utf8mb4_general_ci	Check privileges
□	Project	utf8mb4_general_ci	Check privileges
	Total: 5	utf8mb4_general_ci	

□ Check all   With selected: Drop

phpMyAdmin - 01

Step 9: Import the .sql file included in the repository. After importing it will give a 'Import successfully finished' message.



Importing into the database "Project"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: Choose file FinalProject.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file: utf-8

Partial import:

☑ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be*

Skip this number of queries (for SQL) starting from the first one: 0

Other options:

☑ Enable foreign key checks

Format:

SQL

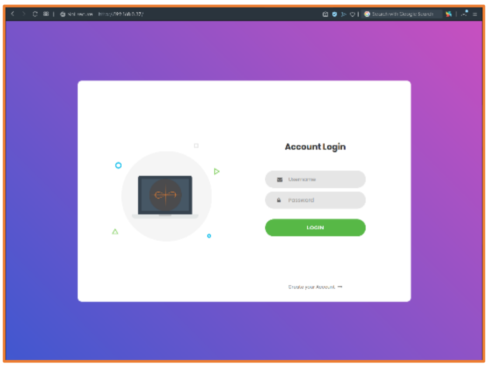
Format-specific options:

SQL compatibility mode: NONE

☑ Do not use `AUTO_INCREMENT` for zero values

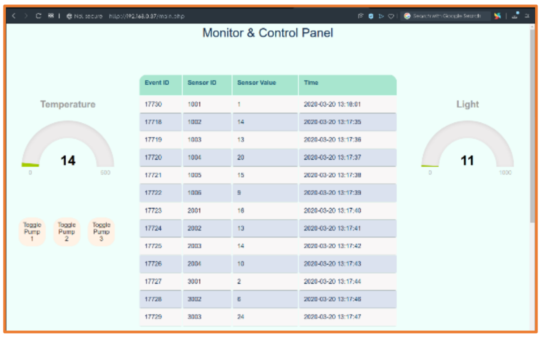
phpMyAdmin - 02

Step 10: In a new tab type ‘192.168.0.37’ (Raspberry Pi-IP). It should load a login page. Click on ‘Create Your Account ->’. If you move your cursor on the image you will see a cool animation.



Browser - 02

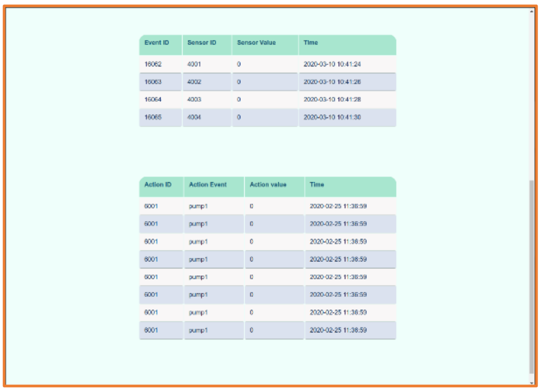
Step 11: Fill in the details to create your account. After submitting, it will take you back to the ‘login’ page. Fill in your username and password. This will take you to the ‘main’ page. You should see 3 tables, 2 measurement gauges and 4 buttons in this page.



Monitor & Control Panel

Event ID	Sensor ID	Sensor Value	Time
17730	1001	1	2020-03-20 13:18:01
17718	1002	14	2020-03-20 13:17:35
17719	1003	13	2020-03-20 13:17:36
17720	1004	20	2020-03-20 13:17:37
17721	1005	15	2020-03-20 13:17:38
17722	1006	9	2020-03-20 13:17:39
17723	2001	16	2020-03-20 13:17:40
17724	2002	13	2020-03-20 13:17:41
17725	2003	14	2020-03-20 13:17:42
17726	2004	10	2020-03-20 13:17:43
17727	3001	2	2020-03-20 13:17:44
17728	3002	6	2020-03-20 13:17:46
17729	3003	24	2020-03-20 13:17:47

Browser - 03



Event ID	Sensor ID	Sensor Value	Time
16062	4001	0	2020-03-10 10:41:24
16063	4002	0	2020-03-10 10:41:26
16064	4003	0	2020-03-10 10:41:28
16065	4004	0	2020-03-10 10:41:30

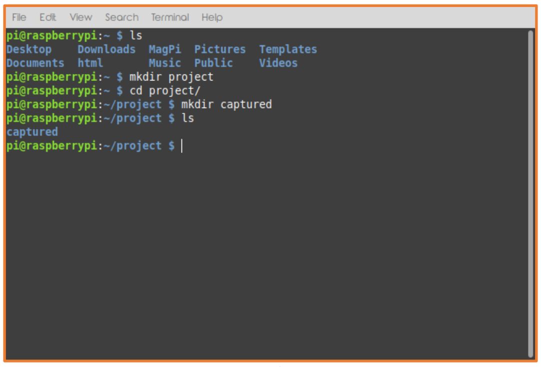
Action ID	Action Event	Action value	Time
6001	pump1	0	2020-02-25 11:36:59
6001	pump1	0	2020-02-25 11:36:59
6001	pump1	0	2020-02-25 11:36:59
6001	pump1	0	2020-02-25 11:36:59
6001	pump1	0	2020-02-25 11:36:59
6001	pump1	0	2020-02-25 11:36:59
6001	pump1	0	2020-02-25 11:36:59
6001	pump1	0	2020-02-25 11:36:59

Browser - 04

Step 12: This website is now accessible within your local network. To make it accessible anywhere by using the internet, we need to install a service provided by ‘remote.it’. You can follow their setup guide remote.it

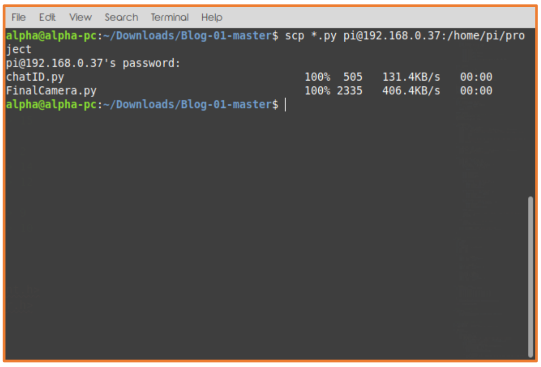
Step 13: To make sure the data from nodeMCU is being transferred to ‘MariaDB’ in Raspberry Pi, boot up both nodeMCUs with the complete circuit and check the serial monitor. Refer Part – 2 of this blog to setup ‘ssid’ & ‘password’ in nodeMCU.

Step 14: Once all that is finished, copy the ‘.py’ files to Raspberry Pi. Create a folder called ‘project’ and a sub folder called ‘captured’ in ‘home’ directory.



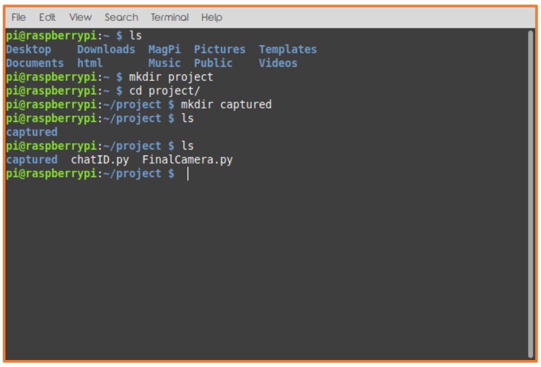
```
pi@raspberrypi:~ $ ls
Desktop    Downloads  MagPi  Pictures  Templates
Documents  html       Music  Public    Videos
pi@raspberrypi:~ $ mkdir project
pi@raspberrypi:~ $ cd project/
pi@raspberrypi:~/project $ mkdir captured
pi@raspberrypi:~/project $ ls
captured
pi@raspberrypi:~/project $ |
```

Terminal 1 - 04



```
alpha@alpha-pc:~/Downloads/Blog-01-master$ scp *.py pi@192.168.0.37:/home/pi/pro
ject
pi@192.168.0.37's password:
chatID.py                                      100%  505   131.4KB/s   00:00
FinalCamera.py                                 100% 2335   406.4KB/s   00:00
alpha@alpha-pc:~/Downloads/Blog-01-master$ |
```

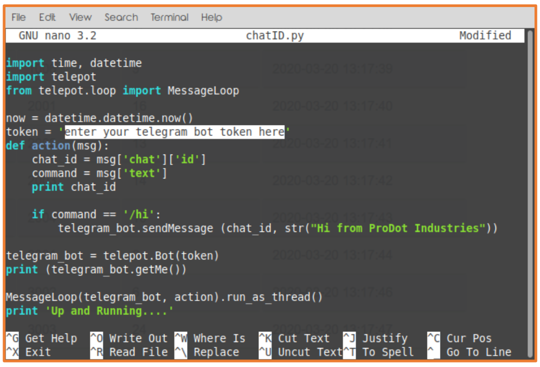
Terminal 2 - 03



```
pi@raspberrypi:~ $ ls
Desktop    Downloads  MagPi  Pictures  Templates
Documents  html       Music  Public    Videos
pi@raspberrypi:~ $ mkdir project
pi@raspberrypi:~ $ cd project/
pi@raspberrypi:~/project $ mkdir captured
pi@raspberrypi:~/project $ ls
captured
pi@raspberrypi:~/project $ ls
captured  chatID.py  FinalCamera.py
pi@raspberrypi:~/project $  |
```

Terminal 1 - 05

Step 15: Edit the ‘chatID.py’ using ‘nano chatID.py’ and enter your ‘Telegram Bot Token’. Run the python script. You can refer Part - 2 of this blog to create a telegram bot.

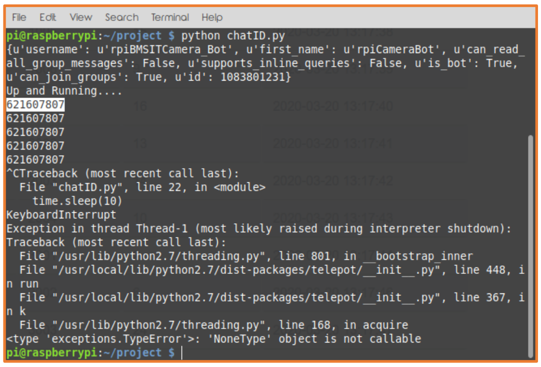


```
  GNU nano 3.2                     chatID.py                         Modified

import time, datetime
import telepot
from telepot.loop import MessageLoop

now = datetime.datetime.now()
token = 'enter your telegram bot token here'
def action(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    print chat_id

    if command == '/hi':
        telegram_bot.sendMessage (chat_id, str("Hi from ProDot Industries"))

telegram_bot = telepot.Bot(token)
print (telegram_bot.getMe())

MessageLoop(telegram_bot, action).run_as_thread()
print 'Up and Running....'

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Spell    ^_ Go To Line
```

Terminal 1 - 06

After entering your token within the quotes, press ‘ctrl+o’ to save the file, then press ‘ctrl+x’ to close the editor.

Step 16: Run this script using ‘python chatID.py’. In your ‘Telegram Bot’ type ‘/hi’. This will return a ‘chat ID’ in the terminal. Copy this ‘chat ID’. Press ‘ctrl+c’ to exit the script.

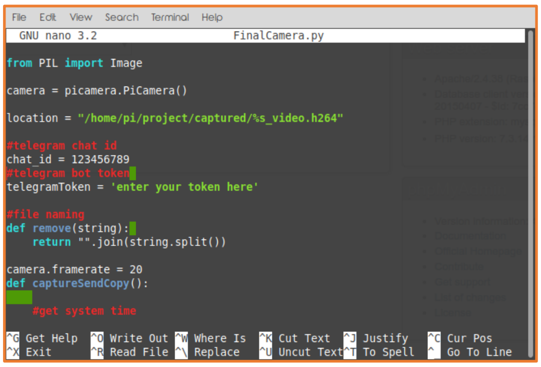


```
pi@raspberrypi:~/project $ python chatID.py
{u'username': u'rpiBMSITCamera_Bot', u'first_name': u'rpiCameraBot', u'can_read_
all_group_messages': False, u'supports_inline_queries': False, u'is_bot': True,
u'can_join_groups': True, u'id': 1083801231}
Up and Running....
621607807
621607807
621607807
621607807
621607807
^CTraceback (most recent call last):
  File "chatID.py", line 22, in <module>
    time.sleep(10)
KeyboardInterrupt
Exception in thread Thread-1 (most likely raised during interpreter shutdown):
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.py", line 801, in __bootstrap_inner
  File "/usr/local/lib/python2.7/dist-packages/telepot/__init__.py", line 448, i
n run
  File "/usr/local/lib/python2.7/dist-packages/telepot/__init__.py", line 367, i
n k
  File "/usr/lib/python2.7/threading.py", line 168, in acquire
<type 'exceptions.TypeError'>: 'NoneType' object is not callable
pi@raspberrypi:~/project $ |
```

Terminal 1 - 07

Telegram Bot - 01

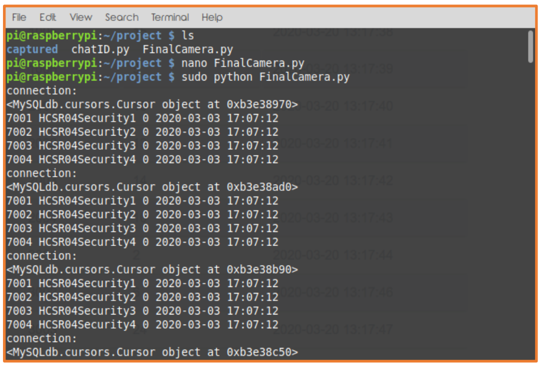
Step 17: Edit the ‘FinalCamera.py’, add the ‘chat ID’ that you copied earlier and add the ‘Telegram Bot Token’. Save the file with ‘ctrl+o’ and close the editor with ‘ctrl+x’.



```
  GNU nano 3.2                          FinalCamera.py

from PIL import Image

camera = picamera.PiCamera()

location = "/home/pi/project/captured/%s_video.h264"

#telegram chat id
chat_id = 123456789
#telegram bot token
telegramToken = 'enter your token here'

#file naming
def remove(string):
    return "".join(string.split())

camera.framerate = 20
def captureSendCopy():

    #get system time

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Terminal 1 - 08

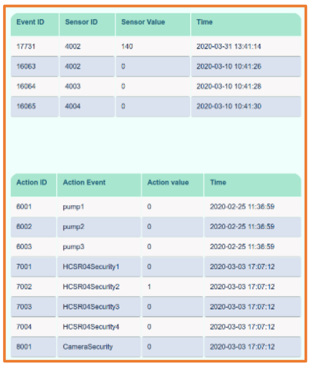
Step 18: Run the script with 'sudo python FinalCamera.py'. The script will self-terminate after executing 10,000 times. You can change this in the source.



```
pi@raspberrypi:~/project $ ls
captured  chatID.py  FinalCamera.py
pi@raspberrypi:~/project $ nano FinalCamera.py
pi@raspberrypi:~/project $ sudo python FinalCamera.py
connection:
<MySQLdb.cursors.Cursor object at 0xb3e38970>
7001 HCSR04Security1 0 2020-03-03 17:07:12
7002 HCSR04Security2 0 2020-03-03 17:07:12
7003 HCSR04Security3 0 2020-03-03 17:07:12
7004 HCSR04Security4 0 2020-03-03 17:07:12
connection:
<MySQLdb.cursors.Cursor object at 0xb3e38ad0>
7001 HCSR04Security1 0 2020-03-03 17:07:12
7002 HCSR04Security2 0 2020-03-03 17:07:12
7003 HCSR04Security3 0 2020-03-03 17:07:12
7004 HCSR04Security4 0 2020-03-03 17:07:12
connection:
<MySQLdb.cursors.Cursor object at 0xb3e38b90>
7001 HCSR04Security1 0 2020-03-03 17:07:12
7002 HCSR04Security2 0 2020-03-03 17:07:12
7003 HCSR04Security3 0 2020-03-03 17:07:12
7004 HCSR04Security4 0 2020-03-03 17:07:12
connection:
<MySQLdb.cursors.Cursor object at 0xb3e38c50>
```

Terminal 1 - 09

Step 19: You can see that the values for 'HCSR04' are all 0. Now hold your hand on top of any of the 'HCSR04' sensors. This will trigger a change in the 'MariaDB'.

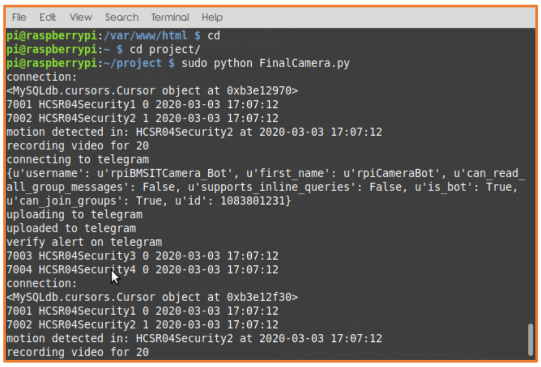


Event ID	Sensor ID	Sensor Value	Time
17731	4002	140	2020-03-31 13:41:14
16063	4002	0	2020-03-10 10:41:26
16064	4003	0	2020-03-10 10:41:28
16065	4004	0	2020-03-10 10:41:30

Action ID	Action Event	Action value	Time
6001	pump1	0	2020-02-25 11:36:59
6002	pump2	0	2020-02-25 11:36:59
6003	pump3	0	2020-02-25 11:36:59
7001	HCSR04Security1	0	2020-03-03 17:07:12
7002	HCSR04Security2	1	2020-03-03 17:07:12
7003	HCSR04Security3	0	2020-03-03 17:07:12
7004	HCSR04Security4	0	2020-03-03 17:07:12
8001	CameraSecurity	0	2020-03-03 17:07:12

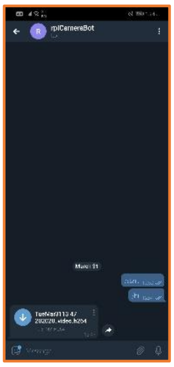
Browser - 05

You can observe the changes in the webpage (I have connected only 1 HCSR04). Now observe the python script that you started earlier. You can see that I have received a video file on my bot. You can also observe that the 'CameraSecurity' in the table becomes 1 when recording.



```
pi@raspberrypi:/var/www/html $ cd
pi@raspberrypi:~ $ cd project/
pi@raspberrypi:~/project $ sudo python FinalCamera.py
connection:
<MySQLdb.cursors.Cursor object at 0xb3e12970>
7001 HCSR04Security1 0 2020-03-03 17:07:12
7002 HCSR04Security2 1 2020-03-03 17:07:12
motion detected in: HCSR04Security2 at 2020-03-03 17:07:12
recording video for 20
connecting to telegram
{u'username': u'rpiBMSITCamera_Bot', u'first_name': u'rpiCameraBot', u'can_read_
all_group_messages': False, u'supports_inline_queries': False, u'is_bot': True,
u'can_join_groups': True, u'id': 1083801231}
uploading to telegram
uploaded to telegram
verify alert on telegram
7003 HCSR04Security3 0 2020-03-03 17:07:12
7004 HCSR04Security4 0 2020-03-03 17:07:12
connection:
<MySQLdb.cursors.Cursor object at 0xb3e12f30>
7001 HCSR04Security1 0 2020-03-03 17:07:12
7002 HCSR04Security2 1 2020-03-03 17:07:12
motion detected in: HCSR04Security2 at 2020-03-03 17:07:12
recording video for 20
```

Terminal 1 - 010

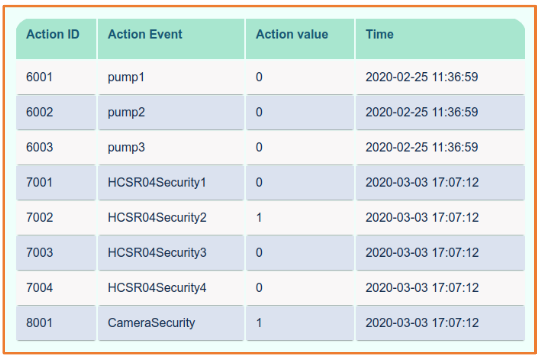


rpiCameraBot

March 31

TueMar3113 47 202020_video.h264

Telegram Bot - 02



Action ID	Action Event	Action value	Time
6001	pump1	0	2020-02-25 11:36:59
6002	pump2	0	2020-02-25 11:36:59
6003	pump3	0	2020-02-25 11:36:59
7001	HCSR04Security1	0	2020-03-03 17:07:12
7002	HCSR04Security2	1	2020-03-03 17:07:12
7003	HCSR04Security3	0	2020-03-03 17:07:12
7004	HCSR04Security4	0	2020-03-03 17:07:12
8001	CameraSecurity	1	2020-03-03 17:07:12

Browser - 06

Step 20: Copy the ‘FinalLauncherScript.sh’ to Raspberry Pi with ‘scp *.sh pi@192.168.0.37:/home/pi’.

Step 21: Edit ‘.bashrc’ to auto run the script. Add this line at the last of the script ‘./FinalLauncherScript.sh’. This will auto execute the python script at boot.