Team 4

2018101024 : Sartak Periwal
2018101039 : Aakash Dantre
2018111026 : Shivaan Sehgal
2018111034 : Sanskar Tibrewal

PROJECT DOCUMENTATION

WATER LEVEL MONITORING IN OVERHEAD TANKS

Part I : Developer Documentation

1. Introduction

1.1. Problem statement and scope :

To monitor the water level in an overhead tank and upload it on the server at a 25% mark.

Our project is expected to upload 1 on the server when there is water at 25% mark in the tank and upload 0 if the water is less than 25% mark.

As water scarcity is a big issue nowadays our project aims at pre alarming the students and the authority that the water level has gone too down(as 25% is too low for such a big hostel). Hence our basic motive behind the project was to pre alarming system for bakul for future use.

1.2. Purpose of the system :

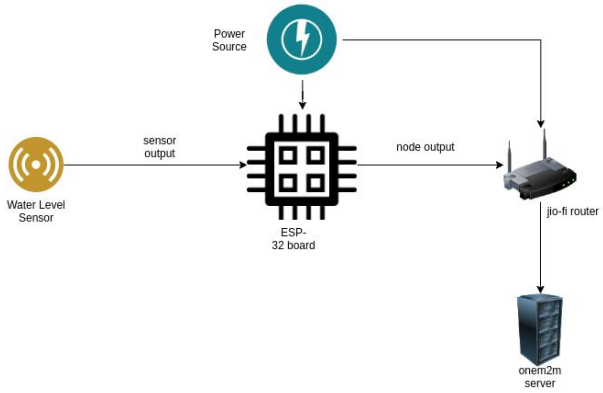
Collect data to help us know about the water consumption rate in Bakul over the year.

Act as an alarming system to indicate low water level(less than 25%) in the tank.

1.3. Overview of the system :

The major components of the system are as follows:-

- Sensor
- MCU
- JioFi
- Server



2. Design Documentation

2.1. System requirements :

1. **Sensor :**
    a. 5V power from VCC.ground voltage equal to or lesser than 0.1.
    b. Waterproofing (is also a requirement).
    c. The sensor must be placed at a height of exactly 75% from the top of the tank. (these are the point to be satisfied for the proper working of sensor component)
2. **MCU :**
    a. Power connection through micro USB power cable self.
    b. working charger(AC->DC) converter .
    c. Sensor input to a valid input pin to check to read the boolean output.
    d. proper internet connectivity with correct SSID and password.
3. **Server :**
    a. working server address.
4. **JioFi :**
    a. proper jio sim with a valid internet connection.
    b. AC power source.
    c. working charger for the jio-fi to get power supply.

2.2. System Specification :

1. **Microprocessor :** ESP 32
2. **Sensor :** FS-IR02
3. **Server :** One M2M

2.3. Stakeholders :

- **Dr. Aftab Husain**
- **Assigned TA**
- **Our working team**

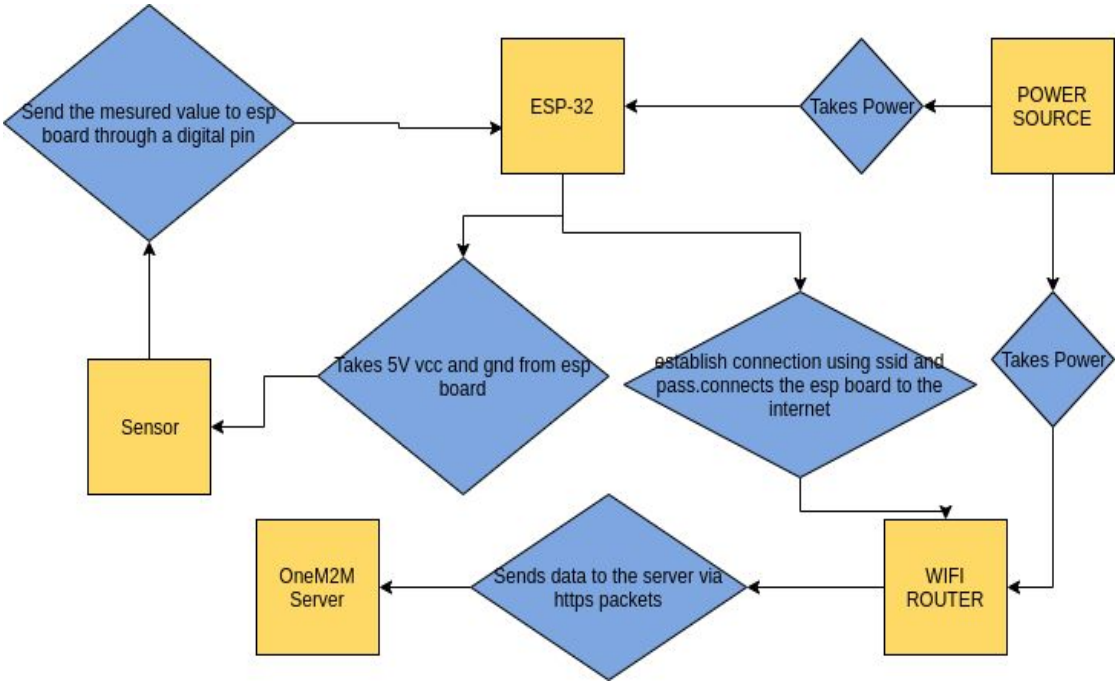
- **College Administration**
- **Hostel Committee**
- **Smart City Project under Hyderabad**

2.4 Design entities / Main components :

1. **Box:** IP65
2. **ESP 32 (Hardware) :**
    a. Wi-Fi - 802.11 b/g/n
    b. Bluetooth - v4.2 BR/EDR and BLE
    c. 2 × I²C interfaces
3. **ESP 32 (Software) :**
    a. HTTP
    b. JSON time library
    c. Thingspeak library
4. **Sensor :**
    a. Operating Voltage: 5V
    b. Output Current : 12 mA
    c. Working Temperature: 25 ~ 105 ℃
    d. Low level Output : < 0.1V
    e. High level Output : > 4.6 V
5. **USB Adapter:** To provide power to the node MCU
6. **AC/DC converter:** One m2m MicroUSB to connect to node MCU
7. **Server IP:** Click here to check data uploaded in the container.

2.5 Design Details :

1. **Entity interaction:** Various components of the system interact in following way :



2. **Conceptual Flow :**
   a. The sensor takes input from the water tank and gives it output to the node MCU.
   b. Node with the help of internet connection though jiofi sends the value to the server.
   c. Setup Code :

```
void setup()
{
        Serial.begin(115200);
        ledUpdate();
        connect_to_WIFI();
        pinMode(SENSOR_PIN, INPUT);
        pinMode(LED_R, OUTPUT);
        pinMode(LED_G,OUTPUT);
        Serial.println("Setup done");
}
```

d. Realtime Running Code :

```
void loop()
{
        if (WiFi.status() != WL_CONNECTED)
        {
                Serial.println("Connection lost.. trying to reconnect");
                ledFlag=0;ledUpdate();
                connect_to_WIFI();
        }
        int OUT = digitalRead(SENSOR_PIN);
        Serial.println(OUT);
        // convert it to a string
        String val = (String)OUT;
        val = "\"" + val + "\"";
        // Send data to OneM2M server
        createCI(server, "Team4_Water_Level_Monitoring_in_Overhead_Tanks", "node_1", val);
        delay(1000*300); // DO NOT CHANGE THIS LINE 10 min delay
}
```

e. createCI() - send the request to the server(with the give ip) to create a node in the descrived container.

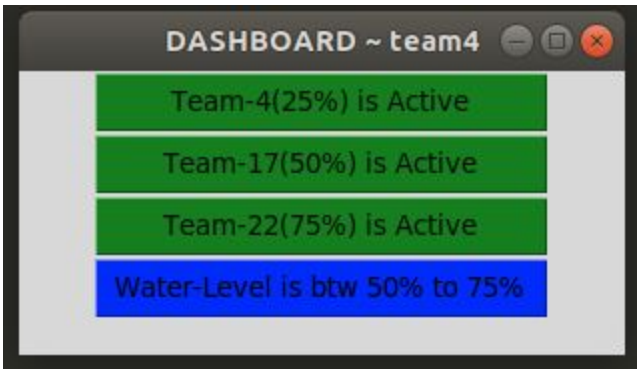
```
String createCI(String server, String ae, String cnt, String val)
{
        HTTPClient http;
        http.begin(server + ae + "/" + cnt + "/");
        http.addHeader("X-M2M-Origin", "admin:admin");
        http.addHeader("Content-Type", "application/json;ty=4");
        http.addHeader("Content-Length", "100");
        http.addHeader("Connection", "close");
        int code = http.POST("{\"m2m:cin\": {\"cnf\": \"text/plain:0\",\"con\": "+ String(val) +"}}");
        http.end();
        Serial.println(code);
        if(code==-1)
        {
                Serial.println("UNABLE TO CONNECT TO THE SERVER");
                ledFlag=0;
                ledUpdate();
        }
        delay(300);
}
```

f. connecttoWiFi() - try to make the connection with given ssid using the given password

```
void connect_to_WIFI()
{
        int flag=1;
        WiFi.mode(WIFI_STA);// Set WiFi to station mode and disconnect from an AP if it was previously connected
        WiFi.disconnect();
        delay(100);
        WiFi.begin(wifi_ssid, wifi_pwd);
        Serial.println("Connecting to WiFi..");
        while (WiFi.status() != WL_CONNECTED || WiFi.status()==WL_CONNECT_FAILED)
        {
                delay(500);
                Serial.print(".");
        }
        if(WiFi.status()==WL_CONNECTED)
        {
                Serial.println("Connected to the WiFi network");
                ledFlag=1;
                ledUpdate();
        }
        Serial.println("Connected to the WiFi network");
}
```

2.6 Operational requirements :

1. 24x7 internet, 24x7 power and a working server.
2. **UI Design :**
   a. A dashboard is made in python to scrape the last data from the onem2m server.



   b. Tkinter library should be properly installed
   c. A computer to run the code.
3. **Analytical System :**
   i. Selenium, Firefox web driver, time, request libraries should be properly installed for data scraping.
   ii. Matplotlib, numpy should be properly installed for graph making of data scraped.
   b. Python3 environment to run the code.

Part 2: User Documentation

1. Introduction

The system helps us to monitor the water level in the Bakul overhead tank(East side) by sending data to a server and retrieving it back using a dashboard.

Our project is expected to upload 1 on the server when there is water at 25% mark in the tank and upload 0 if the water is less than 25% mark.

1.1. Objective:

To measure the level in the overhead tank and notify whether water is up to 25% mark or not.

As 25% mark is too low water level for such a big hostel-like Bakul, therefore, it will act as an alarming system for the students as well as the authority that soon the water is going to get over in Bakul and some necessary steps need to be taken.

Moreover collecting above data can help us know about water consumption rate in Bakul over the year.

1.2. Scope :

As water scarcity is a big issue nowadays our project aims at pre alarming the students and the authority that the water level has gone too down(as 25% is too low for such a big hostel). Hence our basic motive behind the project was to pre alarming system for bakul for future use.

2. Product Operational requirements

2.1. Hardware requirements :

a. **Power:** This includes a power source (any power outlet would work), an adapter and a USB type B wire to connect the ESP board to the power source.
b. **Male-Female Connectors:** They should be in proper working condition otherwise circuit will break.
c. **Waterproofing material over the wires:** To keep the wire underwater safe from any shorting.
d. **IP-65 box:** For protecting the hardware components from harsh weather condition

2.2. Software requirements :

1. **Arduino IDE**
2. **Drivers**
3. **The code**
4. **Python libraries**
5. **Web scraping driver**

3. Assembling the system

3.1. Hardware component :

1. Connect the esp32 board from the charger input inside the box.
2. Connect the VCC and GND wires from the sensor to ESP32 board.
3. Connect the input from a sensor to ESP32 board at the pin specified by the software.
4. Make sure that the sensor is at the right level and have proper waterproofing.
5. Make sure jio fi is powered.

3.2. Software component (Compiling and loading the code onto the board) :

1. Connect the laptop to the hardware setup (built above)
2. Open Arduino and download the drivers required for uploading the code on the board. Also, download the required header files. (refer the documentation specific to your operating system)
3. Copy-paste the code from **Table 1** (At Last of Documentation)into a new file:

4. Change the WiFi SSID and password to that of your network.
5. Compile and upload it. If in case, you face any issues while uploading it, press the reset button on the board during the loading time.
6. Open the serial monitor. You should be able to see WiFi connected, and the values read by the board and sent to the server.
7. Your setup is now ready for deployment.

NOTE: In an ideal case, the project should contain the hardware setup with the code uploaded in it. If it does not, or the circuit delivered is damaged, then get a new board and sensor and construct the hardware circuit from scratch and upload the code onto it.

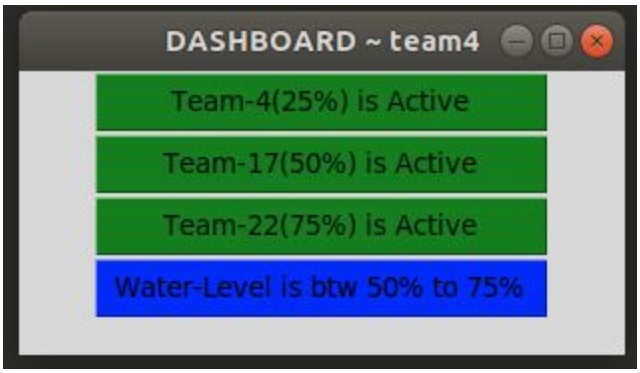
3.3. Deployment :

1. Insert the hardware component onto a box, that would keep the components secure from harsh weather.
2. Place it in a place where the WiFi signal is strong enough.
3. Connect it to the power supply.

Now the board shall send values to the servers as required. Leave the setup there for several days in order to get the data for a reasonable number of days.

3.4. Analytics :

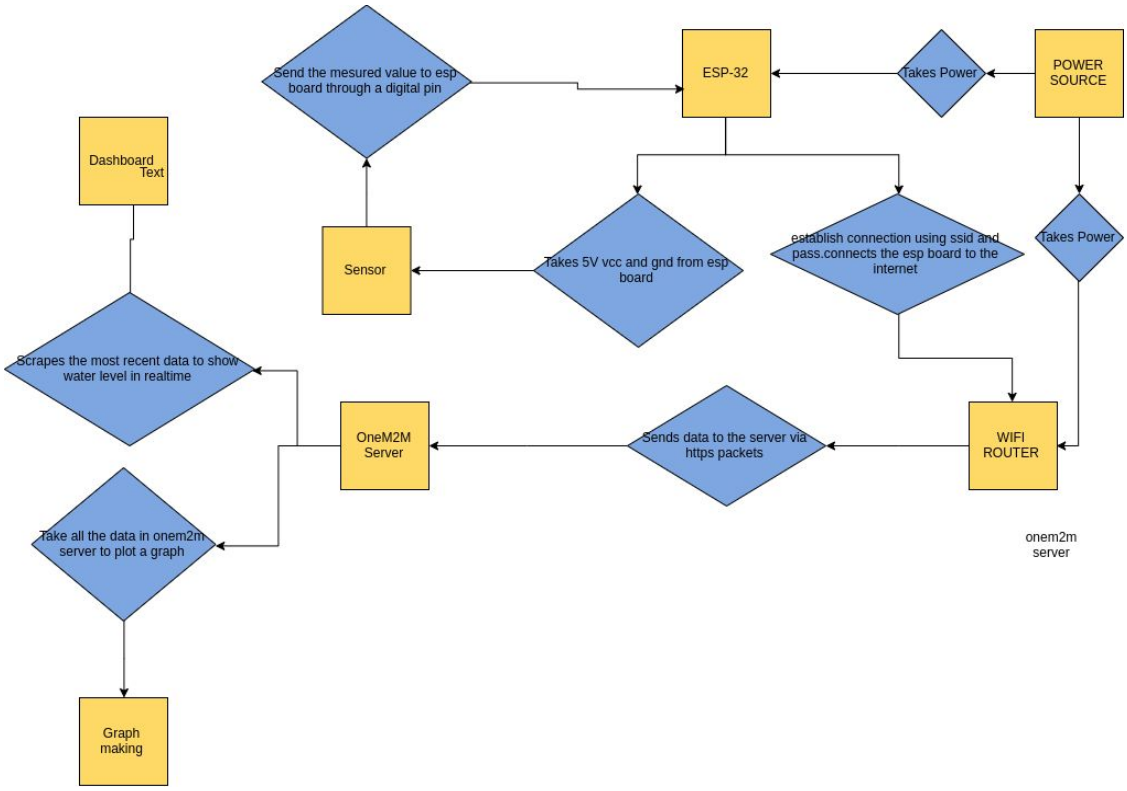
1. Run the data.py(**Table 2**) to get all the data of all 3 teams in file team17,team4 and team22.
2. Run the graph.py to get the matplotlib window for water level.
3. Observing the graph, we can find out the following things:
    i. Average water level
    ii. Consumption rate
4. Run board.py(**Table 3**) to get the dashboard.



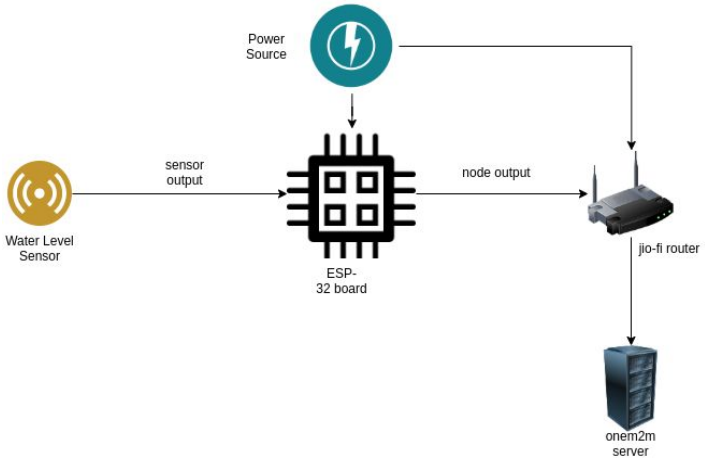
DASHBOARD

3.System Working Model

3.1. Working State :



3.2. Base State :



Part 4: IOT Project Components

1. Hardware Specifications

1.1. Box : IP65

1.2 ESP 32 (Hardware) :

a. Wi-Fi - 802.11 b/g/n
b. Bluetooth - v4.2 BR/EDR and BLE
c. 2 × I²C interfaces

1.3 ESP 32 (Software) :

a. HTTP
b. JSON time library
c. Thingspeak library

1.4 Sensor :

a. Operating Voltage: 5V
b. Output Current : 12 mA
c. Working Temperature: 25 ~ 105 ℃
d. Low level Output : < 0.1V
e. High level Output : > 4.6 V

1.5 USB Adapter: To provide power to the node MCU

1.6 AC/DC converter: One m2m MicroUSB to connect to node MCU

2. Communication

2.1 Sensor and MCU : Data communication happens with digital signal(data format) from a wire connection the sensor and a digital pin in ESP-32 board.

2.2 MCU and sever: Using https/onem2m protocol data packets(data format) are sent to the server.

2.3 server and dashboard: Using https/onem2m protocol get request is made to the server in reply of which server sends the asked container.

3. Software Specifications

3.1. Arduino IDE

3.2. Arduino libraries

a. Wifi.h
b. HTTPClient.h
c. ArduinoJSON.h (version 5)

3.3. Python libraries :

a. tkinter(for dashboard)
b. requests(for get request)
c. matplotlib,numpy (for graph formation)

3.4. Data Handling Model :

a. Data from the sensor is uploaded to the server every 5 minutes by sending a post request with JSON file as payload having sensor value.
b. It is stored in onem2m server kept in nilgiri building
c. From the server data is pulled by the dashboard for further analysis and record keeping.

3.5. Integration Framework :

i. Integration between sensor and ESP board - MCU reads the digital output provided by the sensor.No calibration or further processing of the data is required as data is in boolean form.
ii. Integration between ESP board and server -Data from the MCU is sent to the server via wifi router. The server is using onem2m, which have well defined mechanisms for data management.
iii. Integration between server and dashboard - Dashboard pulls data from the onem2m server through https request and plot a graph of water level.

3.6. Data Visualization / Analysis Framework :

a. Data visualization/analysis is done by the dashboard. It shows the last received values of all 3 water level sensors
b. From which we can calculate the level at which water is currently present.
c. It also shows whether a sensor is active or not in real time.
d. We can also plot a graph using Matplotlib and request libraries in python by scraping all data in onem2m server.

Table 1 (For 3.2):

```
#include "WiFi.h"
#include "HTTPClient.h"
#include "ArduinoJson.h"
#include "time.h"

char* wifi_ssid = "JioFi3_126C36";
char* wifi_pwd = "4p7ce3t8vv";
char* wifi_ssid2="";
char* wifi_pwd2="";

String cse_ip = "onem2m.iiit.ac.in";
String cse_port = "443";
String server = "https://"+cse_ip+":"+cse_port+"/~/in-cse/in-name/";

int ledFlag=0;
uint8_t SENSOR_PIN = 2;
int LED_G = 4;
int LED_R = 5;StaticJsonBuffer<200> jsonBuffer;
JsonObject& user_data = jsonBuffer.createObject();
JsonObject& temp_user_data = jsonBuffer.createObject();
JsonObject& sensor_data = jsonBuffer.createObject();

void ledUpdate(){
 if(ledFlag==1){
   digitalWrite(LED_R,0);
   digitalWrite(LED_G,1);
 }
 else{
   digitalWrite(LED_R,1);
   digitalWrite(LED_G,0);
 }
}

String createCI(String server, String ae, String cnt, String val)
{
 HTTPClient http;
 http.begin(server + ae + "/" + cnt + "/");
 http.addHeader("X-M2M-Origin", "admin:admin");
```

```
  http.addHeader("Content-Type", "application/json;ty=4");
  http.addHeader("Content-Length", "100");
  http.addHeader("Connection", "close");
  int code = http.POST("{\"m2m:cin\": {\"cnf\": \"text/plain:0\",\"con\": "+ String(val) +"}}");
  http.end();
  Serial.println(code);
  if(code==-1){
   Serial.println("UNABLE TO CONNECT TO THE SERVER");
   ledFlag=0;
   ledUpdate();
  }
  delay(300);
}

void connect_to_WIFI(){
  int flag=1;
  WiFi.mode(WIFI_STA);// Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.disconnect();
  delay(100);
  WiFi.begin(wifi_ssid, wifi_pwd);
  Serial.println("Connecting to WiFi..");
  while (WiFi.status() != WL_CONNECTED || WiFi.status()==WL_CONNECT_FAILED){
   delay(500);
   Serial.print(".");
  }
  if(WiFi.status()==WL_CONNECTED){
   Serial.println("Connected to the WiFi network");
   ledFlag=1;
   ledUpdate();
  }
  else{

  }
  Serial.println("Connected to the WiFi network");
}

void setup()
{
  Serial.begin(115200);
  ledUpdate();
  connect_to_WIFI();

  pinMode(SENSOR_PIN, INPUT);
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G,OUTPUT);
```

```
  Serial.println("Setup done");

  // DOUBT ----------------------------------------------------------------------------------------------------------------------------------??
  //createCI(server, "\"Team4_Water_Level_Monitoring_in_Overhead_Tanks", "project_description",
"measuring_water_level_in_overhead_tanks");
  //createCI(server, "Team4_Water_Level_Monitoring_in_Overhead_Tanks", "node_description",
"node_1:(photo_electric_fluid_level(=25%)");

}

void loop()
{
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Connection lost.. trying to reconnect");
    ledFlag=0;ledUpdate();
    connect_to_WIFI();
  }

  int OUT = digitalRead(SENSOR_PIN);
  Serial.println(OUT);
  // convert it to a string
  String val = (String)OUT;

  val = "\""  + val + "\"";

  // Send data to OneM2M server
  createCI(server, "Team4_Water_Level_Monitoring_in_Overhead_Tanks", "node_1", val);
  delay(1000*300); // DO NOT CHANGE THIS LINE 10 min delay


}
```

Table 2 (data.py For 3.4):

```
import requests
import json
import matplotlib.pyplot as plt
import numpy as np
from selenium import webdriver
from time import sleep
from selenium.webdriver.firefox.options import Options
#from selenuim.webdriver.chrome.options import Options

data4=[]
data17=[]
data22=[]

def get_data_group(group_name):
headers = {
'X-M2M-Origin': 'admin:admin',
'Content-type': 'application/json'
}

group_uri = server+group_name
print(group_uri)
response = requests.get(group_uri, headers=headers)
# print('Return code : {}'.format(response.status_code))
#print('Return Content : {}'.format(response.text))
try:_resp = json.loads(response.text)
except:print("lol:error")
val=[]
val.append(_resp['m2m:cin']['lt'])
val.append(_resp['m2m:cin']['con'])
val.append(group_name)
if(team==4):data4.append(val)
if(team==17):data17.append(val)
if(team==22):data22.append(val)
return response.status_code, _resp

if _name_ == "_main_":

opts = Options()
opts.set_headless()
assert opts.headless # Operating in headless mode

driver = webdriver.Firefox(options=opts)
# driver.get("http://139.59.42.21:8080/webpage/welcome/index.html?context=/~&cseId=in-cse")
driver.get("http://onem2m.iiit.ac.in:443/webpage/welcome/index.html?context=/~&cseId=in-cse")
sleep(3)
driver.find_element_by_tag_name("button").click()
sleep(5)
driver.find_element_by_xpath("//li/ul/li[6]").click()
sleep(4)
driver.find_element_by_xpath("//li/ul/li[19]").click()
sleep(7)
driver.find_element_by_xpath("//li/ul/li[24]").click()
sleep(6)
driver.find_element_by_xpath("//li/ul/li[6]/ul/li[1]").click()
sleep(6)
driver.find_element_by_xpath("//li/ul/li[19]/ul/li[3]").click()
sleep(7)
driver.find_element_by_xpath("//li/ul/li[24]/ul/li[1]").click()
```

```
sleep(7)

ele1 = driver.find_elements_by_xpath("//li/ul/li[6]/ul/li[1]/ul/*")
ele2 = driver.find_elements_by_xpath("//li/ul/li[19]/ul/li[3]/ul/*")
ele3 = driver.find_elements_by_xpath("//li/ul/li[24]/ul/li[1]/ul/*")
print(len(ele1))
print(len(ele2))
print(len(ele3))
lst1=[]
for i in ele1:
lst1.append(i.text)
lst2=[]
for i in ele2:
lst2.append(i.text)
lst3=[]
for i in ele3:
lst3.append(i.text)

server="http://onem2m.iiit.ac.in:443/~/in-cse/in-name/Team4_Water_Level_Monitoring_in_Overhead_Tanks/node_1/"
# team=4
team=4
for i in lst1:
get_data_group(i)
server="http://onem2m.iiit.ac.in:443/~/in-cse/in-name/Team17_Water_Level_Monitoring_in_Overhead_Tanks/node_1/"
team=17
for i in lst2:
get_data_group(i)
server="http://onem2m.iiit.ac.in:443/~/in-cse/in-name/Team22_Water_Level_Monitoring_in_Overhead_Tanks/node_1/"
team=22
for i in lst3:
get_data_group(i)
```

Table 3 (board.py For 3.4):

```
import requests
import json
from time import sleep
from datetime import datetime
import pytz
import tkinter
data4=[]
data17=[]
data22=[]

def convert_datetime_timezone(dt, tz1,tz2):
    tz1 = pytz.timezone(tz1)
    tz2 = pytz.timezone(tz2)
```

```
    dt = datetime.strptime(dt,"%Y-%m-%d %H:%M:%S")
    dt = tz1.localize(dt)
    dt = dt.astimezone(tz2)
    dt = dt.strftime("%Y-%m-%d %H:%M:%S")
    return dt

def get_data_group(group_name):
    headers = {
    'X-M2M-Origin': 'admin:admin',
    'Content-type': 'application/json'
    }

    group_uri = str(group_name)
    print(group_uri)
    response = requests.get(group_uri, headers=headers)
    # print('Return code : {}'.format(response.status_code))
    # print('Return Content : {}'.format(response.text))
    try:_resp = json.loads(response.text)
    except:print("lol:error")
    val=[]
    val.append(_resp['m2m:cin']['lt'])
    val.append(_resp['m2m:cin']['con'])
    val.append(_resp['m2m:cin']['rn'])
    if(team==4):data4.append(val)
    if(team==17):data17.append(val)
    if(team==22):data22.append(val)
    return response.status_code, _resp

if __name__ == "__main__":

    root = tkinter.Tk()
    team=4

get_data_group("http://onem2m.iiit.ac.in:443/~/in-cse/in-name/Team4_Water_Level_Monitor
ing_in_Overhead_Tanks/node_1/la")
    team=17

get_data_group("http://onem2m.iiit.ac.in:443/~/in-cse/in-name/Team17_Water_Level_Monit
oring_in_Overhead_Tanks/node_1/la")
    team=22

get_data_group("http://onem2m.iiit.ac.in:443/~/in-cse/in-name/Team22_Water_Level_Monit
oring_in_Overhead_Tanks/node_1/la")
    print(data4)
```

```
    time1=data4[-1][0]
    time2=data17[-1][0]
    time3=data22[-1][0]

    #
server="http://onem2m.iiit.ac.in:443/~/in-cse/in-name/Team4_Water_Level_Monitoring_in_O
verhead_Tanks/node_1/"
    time1=time1[0:4]+"-"+time1[4:6]+"-"+time1[6:8]+"
"+time1[9:11]+":"+time1[11:13]+":"+time1[13:15]
    time2=time2[0:4]+"-"+time2[4:6]+"-"+time2[6:8]+"
"+time2[9:11]+":"+time2[11:13]+":"+time2[13:15]
    time3=time3[0:4]+"-"+time3[4:6]+"-"+time3[6:8]+"
"+time3[9:11]+":"+time3[11:13]+":"+time3[13:15]
    print(time1)
    print(time2)
    print(time3)
    time1i=convert_datetime_timezone(time1,"UTC","Asia/Kolkata")
    time1i = datetime.strptime(time1i,"%Y-%m-%d %H:%M:%S")
    time2i=convert_datetime_timezone(time2,"UTC","Asia/Kolkata")
    time2i = datetime.strptime(time2i,"%Y-%m-%d %H:%M:%S")
    time3i=convert_datetime_timezone(time3,"UTC","Asia/Kolkata")
    time3i = datetime.strptime(time3i,"%Y-%m-%d %H:%M:%S")

    curtime=datetime.now()
    diff1=curtime-time1i
    diff2=curtime-time2i
    diff3=curtime-time3i
    a1=1;a2=1;a3=1;

    if(diff1.seconds/60 > 7):a1=0
    if(diff2.seconds/60 > 7):a2=0
    if(diff3.seconds/60 > 7):a3=0

    root.title("DASHBOARD ~ team4")
    if a1==1:b1=tkinter.Button(text="Team-4(25%) is Active",width=25,bg="green")
    else:b1=tkinter.Button(text="Team-4(25%) is Inactive",width=25,bg="red")
    if a2==1:b2=tkinter.Button(text="Team-17(50%) is Active",width=25,bg="green")
    else:b2=tkinter.Button(text="Team-17(50%) is Inactive",width=25,bg="red")
    if a3==1:b3=tkinter.Button(text="Team-22(75%) is Active",width=25,bg="green")
    else:b3=tkinter.Button(text="Team-22(75%) is Inactive",width=25,bg="red")
    b1.pack()
    b2.pack()
    b3.pack()
    v1=int(data4[-1][1])
    v2=int(data17[-1][1])
```

```
v3=int(data22[-1][1])
l=int()
if(v1==1 and v2==1 and v3==1):l=3
elif(v1==1 and v2==1 and v3==0):l=2
elif(v1==1 and v2==0 and v3==0):l=1
elif(v1==0 and v2==0 and v3==0):l=0
else: exit()
arr=["btw 0% to 25%","btw 25% to 50%","btw 50% to 75%","btw 75% to 100%"]
b=tkinter.Button(text="Water-Level is "+arr[l],width=25,bg="blue")
b.pack()
print(curtime)
print(l)
root.mainloop()
```