Team-7

Embedded Systems Workshop - Monsoon 2019

Water Turbidity Measurements

November 25, 2019

Project Overview

The idea behind this project was to measure the turbidity levels in the Overhead tank of the Boys' Hostel, Bakul Nivas. The project employs an ESP-32 board and a turbidity sensor whose output lies in the range of 0-5 V

Goals

1. Measure the **turbidity levels** in the overhead tank of the Boys' Hostel and store the values in a remote **OneM2M** server.

Specifications

Components used:

1. ESP-32: ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions.
2. Working of the Sensor:
    - ➔ The sensor has 2 pins, with a transmitter at one pin and a receiver at the other.
    - ➔ An electromagnetic wave is emitted from one which is received at the other.
    - ➔ When there is turbidity in the water sample, it would cause the wave to disperse and the intensity of the wave received is less.

Developer Project Documentation

Introduction

The project aims at measuring the turbidity in water at the Bakul overhead tank. The hardware consists only of the ESP-32 board, water turbidity measuring sensor and a connector to connect the two components.

The code can be edited and compiled on Arduino IDE, available for both Windows and Linux.

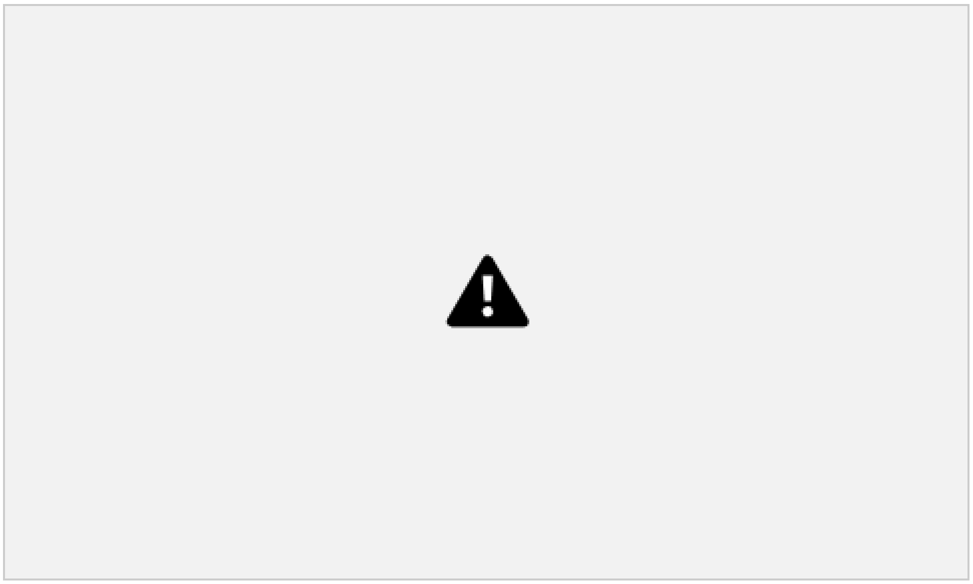
Stakeholders

- ➔ Users of water from overhead tank
- ➔ People in charge of maintenance of tanks

Design Entities

1. OneM2M server to which data is uploaded
2. User Interface which helps monitor the data collected

Conceptual Flow



System Requirements

Hardware:

➔ A 5V power supply for the functioning of the ESP-32 board.
➔ Ensure proper connection of sensor to ESP-32.

Software:

➔ Arduino IDE can be used to edit, compile and flash the code to ESP-32 board being used in this project.

Working of the System

Working of the Sensor:

➔ The sensor has 2 pins, with a transmitter at one pin and a receiver at the other.
➔ An electromagnetic wave is emitted from one which is received at the other.
➔ When there is turbidity in the water sample, it would cause the wave to disperse and the intensity of the wave received is less.

The sensor gives voltage output ranging from 0-5V, 5V showing 0 turbidity value and 0V showing maximum turbidity.

Calibration:
The sensor outputs values in Volts ranging from **0-5V**. We need to convert this into **NTU (Nephlometric Turbidity Units)**, the standard unit for turbidity measurements. For this purpose, we use a formula as described below:

NTU***^** = $-1120.4 * V^2 + 5742.3 * V - 4353.8$

* NTU (Nephlometric Turbidity Unit) from Voltage values; Here, V = Voltage
^This formula is obtained experimentally and we have tried to replicate the same calibration in our sensor as well

The above formula was experimentally obtained. Hence, to replicate the same calibration, we have calibrated our sensor to give minimum turbidity at **4.2V output** of the sensor.

Data Collection and Analytics:

Data is collected from the sensor at an interval of 5 minutes and sent to the OneM2M server. A flask application has been designed by us that scraps the incoming data values at the remote server and stores it in a local database.

On running the application, a graph is generated that gets updated in real-time. The values in the database are also separately displayed in the UI.

User Project Documentation

Introduction

The project aims at measuring the **water turbidity** levels at the Bakul overhead tank.

What is Turbidity?

Water turbidity is the opaqueness associated with the water. It is a first level indication of pollution as the opaqueness is usually caused by the presence of undesired solid particles.

System Requirements:

- Code can be edited, compiled and dumped through **Arduino IDE** software program, available across Windows and Linux platforms.
- Python 3 to run the **flask application** to view the data and the real-time updation of the graph.
- Any browser to view the data analytics on localhost.
- 5V power supply for the ESP-32 board.

Warning: Make sure that the sensor is completely sealed and water-proofed before deployment

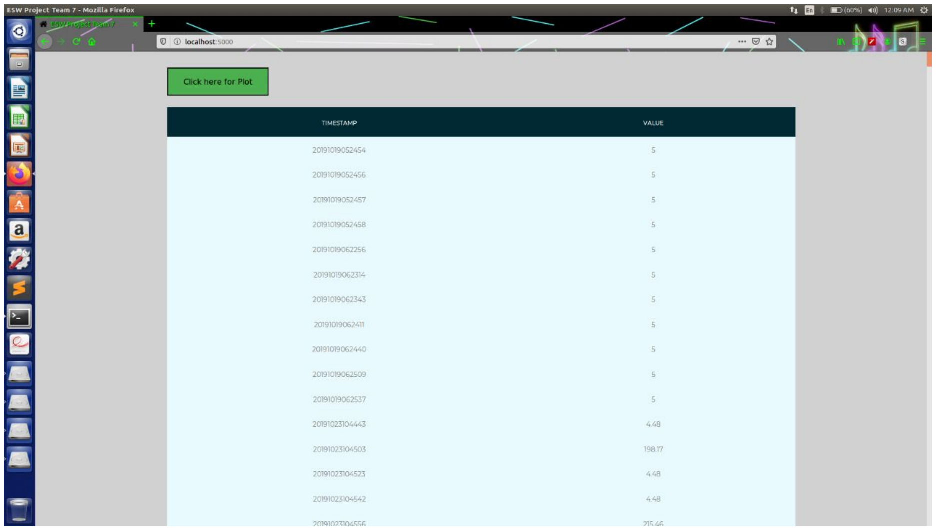
Deployment

- Connect the sensor to the ESP-32 board with the help of a connector.
- Waterproof the sensor, preferably by sealing it using MSeal.
- Connect the ESP-32 board to a 5V power supply using appropriate adapters.
- Ensure that the ESP-32 board is sealed in a waterproof container since it is easily fails when it comes in direct contact with enough water.
- There is an LED light connected to the ESP-32 board which shows if the board is connected to the internet or not.

Working

The sensor readings are available in the remote OneM2M server. To check out these values, run the Flask application and open the results on a browser.

The interface looks like this:



TIMESTAMP	VALUE
20191019052454	5
20191019052456	5
20191019052457	5
20191019052458	5
20191019062256	5
20191019062314	5
20191019062343	5
20191019062411	5
20191019062440	5
20191019062509	5
20191019062537	5
20191023104443	4.48
20191023104503	198.17
20191023104523	4.48
20191023104542	4.48
20191023104556	215.46

The sensor output is in Voltage ranging from 0-5V which has been converted to NTU.

Turbidity values may range from 0 to 3000 NTU. 0 NTU signifies absolutely clear
water with no turbidity in it while 3000 NTU suggests that the sample of water is highly turbid.

IoT Project Components

Hardware Specifications

ESP-32:

- ❖ Code compilation and flashing through Arduino IDE
- ❖ Operating Temperature: 40 - 125 degree Celsius
- ❖ Bluetooth support present
- ❖ WiFi connectivity available
- ❖ 5V power supply

Sensor:

- ❖ Output in Volts: 0-5V
- ❖ Not waterproof by itself. Requires waterproofing before deployment

Communication

The sensor outputs values in Voltages ranging from 0-5 V. The corresponding NTU values are obtained using the following formula:

NTU*^ = $-1120.4 * V^2 + 5742.3 * V - 4353.8$

* NTU (Nephlometric Turbidity Unit) from Voltage values; Here, V = Voltage
^This formula is obtained experimentally and we have tried to replicate the same calibration in our sensor as well

The sensor has been calibrated to give 0 NTU at 4.2V and hence the voltage values will range between 0-4.2 V.

Software and Code

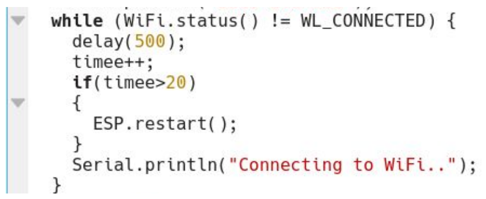
Introduction

The coding was done on an Arduino IDE where it was compiled and flashed onto the ESP-32 board. The code was carefully written keeping in mind the possible causes for failures including power failures and absence of WiFi connectivity.

Key Features

Error handlings:

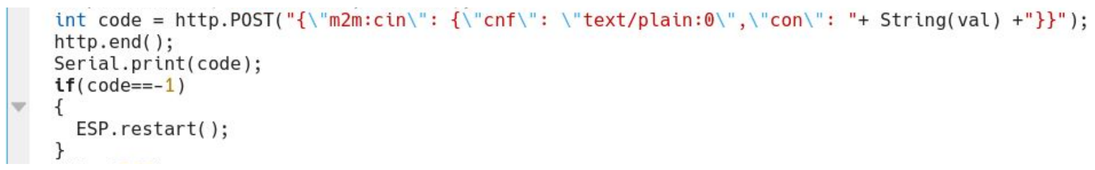
1. **Fail to connect to WiFi**



```
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  timee++;
  if(timee>20)
  {
    ESP.restart();
  }
  Serial.println("Connecting to WiFi..");
}
```

The ESP restarts when the device fails to connect to WiFi within a certain cutoff time.

2. **Reset WiFi when not connected to Internet**



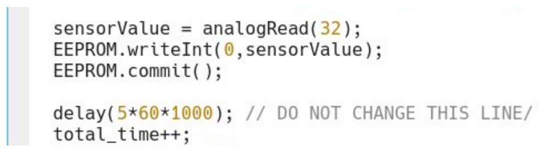
```
int code = http.POST("{\"m2m:cin\": {\"cnf\": \"text/plain:0\",\"con\": "+ String(val) +"}}");
http.end();
Serial.print(code);
if(code==-1)
{
  ESP.restart();
}
```

Restart and search for new connection if WiFi is connected but Internet is unavailable.

3. **Power Failure**

   The ESP board restarts when power fails and returns.

Time Interval:

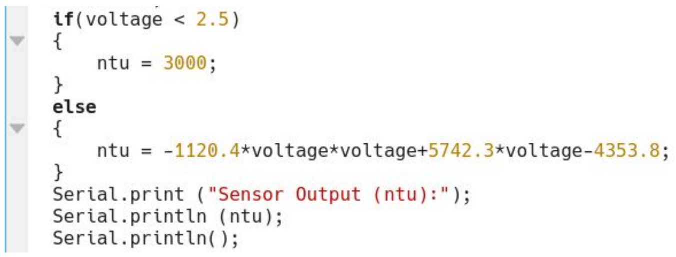


```
sensorValue = analogRead(32);
EEPROM.writeInt(0,sensorValue);
EEPROM.commit();

delay(5*60*1000); // DO NOT CHANGE THIS LINE/
total_time++;
```

The code uploads data into the OneM2M server at a regular interval of 5 minutes.

Voltage to NTU conversion:



```
if(voltage < 2.5)
{
    ntu = 3000;
}
else
{
    ntu = -1120.4*voltage*voltage+5742.3*voltage-4353.8;
}
Serial.print ("Sensor Output (ntu):");
Serial.println (ntu);
Serial.println();
```

The sensor after calibration gives values between 0 to 3000 NTU based on the sensor readings.

User Interface

A flask application was developed to display the data collected.

Working

The application powered by Selenium scrapes data off the remote OneM2M server and stores the data on a MySQL database. This data is then retrieved and displayed on a webpage.

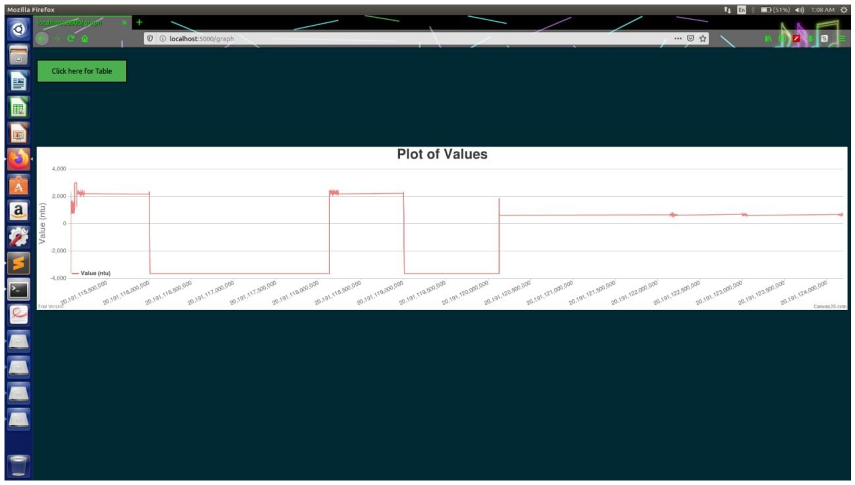
A graph of the data collected so far is also generated alongside the tables.

Using

- ❖ Run the python script
- ❖ Open the link generated on a web browser to view the data

Requirements

- ❖ Python 3
- ❖ Active internet connection
- ❖ Web browser to view the results


Click here for Table
Plot of Values
Value (ntu)