International Institute of Information Technology, Hyderabad

Water flow monitoring - Team 27

Shourja Mukherjee Harshika Jain Sriharshitha Bondugula

DEVELOPER DOCUMENT

November 25, 2019

1. Introduction

1.1 Problem Statement as mentioned in the question

To find the water flow rates in the water pump in the faculty quarters in order to get an idea about the water usage and its implications.

1.2 Project Description

This work aims to build an Internet based system that enables the remote monitoring of a water/flow meters.It enables us to detect the flooding or the sudden cut off of the water supply at the location where it is deployed.

Our project is deployed at the pump room in the faculty quarters and measures values whenever the automatic pump system is turned on and sends the values to a onem2m sever. This data can be used to understand the water usage patterns of the faculty quarters.

1.3 Project Overview

-Get the flow rate in the pipe in Faculty quarters pump room through ESP32 board

-Send the data to the Onem2m server to store it

-Analyse data to monitor and understand water usage by the faculty quarters.

2. Design Documentation

2.1 System Requirements

A location where the sensor is to be deployed. It is better to choose a place where there is a constant flow of water so that we can read the values and the data recorded comes out to be useful. The location chosen should also be convenient for the sensor to be deployed, i.e the size of the sensor and the water flow that it can handle should be taken into consideration.

2.2 System Specifications(technical terminology)

-ESP-32 Arduino

- 2.4 GHz band Wi-Fi
- Bluetooth 4.2
- Dual Core

-BT-DFS-700 Water flow sensor

- Max Rated Current: 8 mA

  Withstand Current: 15 mA
- Working Voltage: 4.5 to 24 VDC
- 2” inlet and outlet pipe

2.3 Stakeholders

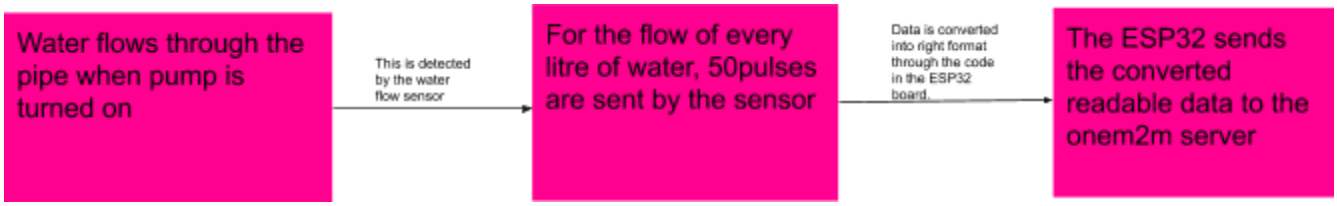
The different stakeholders in this equation include the residents of the faculty quarters whose water usage we are measuring and the water management team of IIIT-H who will get to have clearer statistics for the water usage in this area.

2.4 Design Entities / Main Components

- ESP-32 Arduino microcontroller
- BT-DFS-700 water flow sensor
- OneM2M server

2.5 Design Details

The conceptual flow and entity interaction in the project is clearly illustrated in the flowchart below:



(It's 50 pulses per minute or 8 pulses per second in case of our sensor when the rate is 1L/min.)

2.6 Operational Requirements

- The ESP-32 board requires a constant 5V power supply through its micro-USB adapter. The board provides power to the LDR sensor as well.
- The data is sent to a OneM2M server, where the user can login and monitor the data through a web browser. To avoid post errors, the OneM2M server must be operational throughout.
- The wires, connections, and board must be kept in a protected storage unit with appropriate warnings installed, to avoid risk of electrical shock and damage to

components.

3. User or Operational Document

3.1 Introduction

3.1.1 Objective

This document provides comprehensive guidelines and step-by-step instructions for working with the project: Water flow detection in the college water pump.

3.1.2 Scope

This document is supposed to be used by those people who wish to monitor water flow of their pipeline to detect the flooding or sudden cut off of the water supply at the location it is deployed. This data can be used further for various other applications, like to measure the volume of water used in commercial and residential buildings to take a look at wastage of water.

3.2. Product Operational requirements

3.2.1 Hardware requirements

1) **ESP 32 NodeMCU Development Board**

   ESP 32 NodeMCU is use to deploy our code which calculates the value of flow in the given pipe in L/min.

2) **Water flow sensor**

   This sensor gives the digital value in the form of pulses which are further calculated to find out the flow.

3) **Operating environment**

   A PC is required which support arduino software to dump your code to ESP 32 for building the project from scratch.

4) Server

To store our readings and data of the server we use OneM2M server of the college.

5)Components required for deployment

a. Power supply

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. WiFi

To send our data to the server we required an active WiFi setup which works 24x7.

3.2.2 Software requirements

1. Arduino software
2. Drivers
3. The code

3.3. Assembling the system

3.3.1 Hardware component

The circuit consists of the ESP32 board and a water flow sensor connected using wires and a jiofi for sending containers to server.

3.3.2 Software component

1. Connect the laptop to the hardware setup.
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 following code into a new file:

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

char* wifi_ssid = "JioFi3_127A1C";
char* wifi_pwd = "0rn36u30sm";
String cse_ip = "onem2m.iiit.ac.in";

#define LED_BUILTIN 2
#define SENSOR  27

long currentMillis = 0;
long previousMillis = 0;
int interval = 1000;
boolean ledState = LOW;
float calibrationFactor = 8;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;

String cse_port = "443";
String server = "http://"+cse_ip+":"+cse_port+"/~/in-cse/in-name/";

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.print(code);
  delay(60000);
}

void IRAM_ATTR pulseCounter()
```

```
{
  pulseCount++;
}
void check(){
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
}
void setup()
{
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(SENSOR, INPUT_PULLUP);

  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;
  Serial.println("Setup done");

  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
  sei();

  // Set WiFi to station mode and disconnect from an AP if it was previously connected
  WiFi.mode(WIFI_STA);

  WiFi.begin(wifi_ssid, wifi_pwd);
  check();
}

void loop()
{
  check();
  currentMillis = millis();
  if (currentMillis - previousMillis > interval && WiFi.status() == WL_CONNECTED) {

    pulse1Sec = pulseCount;
    pulseCount = 0;

    // Because this loop may not complete in exactly 1 second intervals we calculate
    // the number of milliseconds that have passed since the last execution and use
    // that to scale the output. We also apply the calibrationFactor to scale the output
```

```
    // based on the number of pulses per second per units of measure (litres/minute in
    // this case) coming from the sensor.
    flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
    previousMillis = millis();

    // Divide the flow rate in litres/minute by 60 to determine how many litres have
    // passed through the sensor in this 1 second interval, then multiply by 1000 to
    // convert to millilitres.
    flowMilliLitres = (flowRate / 60) * 1000;

    // Add the millilitres passed in this second to the cumulative total
    totalMilliLitres += flowMilliLitres;

    // Print the flow rate for this second in litres / minute
    Serial.print("Flow rate: ");
    Serial.print(int(flowRate));  // Print the integer part of the variable
    Serial.print("L/min");
    Serial.print("\t");       // Print tab space

    // Print the cumulative total of litres flowed since starting
    Serial.print("Output Liquid Quantity: ");
    Serial.print(totalMilliLitres);
    Serial.print("mL / ");
    Serial.print(totalMilliLitres / 1000);
    Serial.println("L");
  //user_data.printTo(Serial);

  // Send data to OneM2M server
  String sensor_string = String(int(flowRate)) + "L/min"; // + " Output Liquid Quantity:" +
String(totalMilliLitres)+ "mL/" + String(totalMilliLitres/1000) + "L";
  sensor_string = "\"" + sensor_string + "\""; // DO NOT CHANGE THIS LINE
  createCI(server, "Team27_Water_flow_monitoring", "node_1" , sensor_string);
  delay(15000); // DO NOT CHANGE THIS LINE
  }
}
```

4. Change the WiFi SSID and password accordingly.

6. 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.

7. Open the serial monitor. You should be able to see WiFi connected, and the values read by the board, and sent to the server.

8. Your setup is now ready for deployment.

3.3.3 Deployment

1. Insert all the hardware components into the box to protect it from outer environment and connecting wires are to be connected to the water flow sensor which comes out through a hole in the box.
2. Place the box in a place with good Wifi signals for the Jiofi.
3. Connect it with the power supply.

Now it will send the values of water flow calculated in the form of containers to the server and you can retrieve the data from the server.

3.3.4 Analytics

Data from the server can be scraped out using a simple python script and we can store that in an excel sheet through which we can make day or date-wise graph for analysis.

3.3.5 System working model

1. Base state

   It will show 0 L/min whenever the pump is off (there is no flow inside the pipe). In our project we got very few non-zero values as the pump is closed most of the time and it is only switched on when it is required(when the tank is empty).

2. Working state

   It will measure the flow inside the pipe and send the flow value to server in L/min.

4. Experimental Results

This section explains the problems we faced during deployment and how we solved them.

1. **PROBLEMS FACED**

1) The sensor we have used to work on(YF-S201) and the sensor that was deployed are different(BT-DFS700).
2) Water flow is not continuous in the pipe to which the sensor was attached. It is only switched on when it is required(when the tank is empty).So,our data consisted of very few non-zero values.
3) The network problem is solved by putting a jio-fi and there was no problem with the power. There would be a problem only if there are power-cuts.
4) We have faced problems due to the One-M2M server. We were afraid that the server would crash as we were sending the data every minute that would create a lot of containers. We couldn't scrape off the whole data as the server was stopped.
5) We lost data due to frequent server crashes and our data was not consistent.
6) We also couldn't get continuous data due to power cuts. Each time there was a power cut or heating up of the board or whenever Jio-fi stops working the creation of containers stopped and we had to switch the power off and on again to make it work again.

2. **SOLUTIONS OR COMPROMISES MADE**

1) We have found out the calibration factor of the deployed sensor from the internet and verified.
2) Network problem is solved using a Jiofi for network connections.
3) There is a proper power supply at the place of deployment to which our ESP board is connected.
4) We changed our server many a times due to clash of the server and now the final result is stored on http://onem2m.iiit.ac.in:443/webpage .
5) Whenever our Jiofi stops working we have to restart it to continue.

3. **FINAL RESULT**

1) We could get the data for about a week.
2) Data was continuous for 2-3 days.

3) We have analysed the data and calculated the volume of water flowing through the pipe each day.

DAY	VOLUME(in L)
2019-11-06	5761
2019-11-07	6705
2019-11-15	350
2019-11-19	4097
2019-11-20	4786
2019-11-21	3228
2019-11-23	0

4. CONCLUSION

We were able to calculate the flow rate through the pump. We have done this in a very restricted domain. This can further be extended for multiple purposes. It can be used along with water-level sensor to analyse the flow in the pump along with the height of the water in the tank. It helps us find out when there is damage of pipe or overflow or water-cut.

It can also be extended for the following applications :

Water Management - pump monitoring, system-flow balancing, monitor waste water recycling, sewage.

This project gave us a better idea about how a water flow sensor works , oneM2M , and arduino coding.

———————————————————— THE END ————————————————————