Introduction

Objective

This document provides comprehensive guidelines and step-by-step instructions for working with the project: Pump performance monitoring system.

Scope

This project is meant to be used by people who want to monitor their pump's power, energy vs flow, temperature and humidity of the area and provide appropriate analytics based on the data collected. This will be useful in areas where we have limited power supply and have to critically manage energy. Also we will be able to detect power failures and compromises to the pump performance.

Product Operational requirements

Hardware requirements

Operating Environment

* Any Operating system that supports Arduino IDE.

Flow Sensor

We have used a 1.5 inch pulse count interrupt based flow sensor.

Energy Meter

We have used an RS485 Schneider Electric Energy Meter which is able to provide three phase data (like frequency, current, power factor, voltage etc).

Temperature and Humidity Sensor

The temperature and RH sensor used is based on I2C communication protocol and temperature varying resistance to measure the temperature.

OneM2M server hosted by the college

We push to the college server using the OneM2M API provided on the OneM2M website. Additionally we will host a flask app on a linux server on which we will fetch the data from OneM2M server.

This includes the PCB, on which the circuit is constructed and the connection wires.

Components required for deployment

Power:

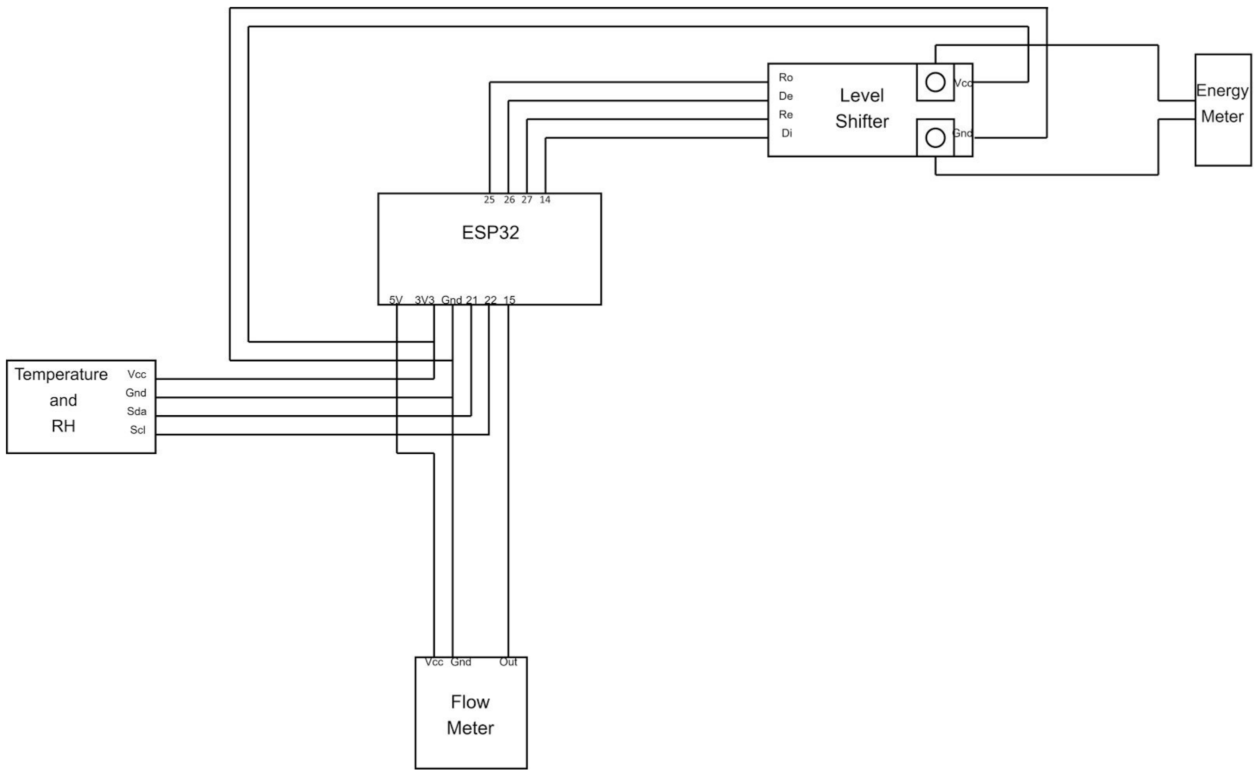
This includes a power source (any power outlet would work), an adapter and a micro USB type B wire to connect the ESP board to the power source

WiFi:

The setup requires WiFi connection (preferably a steady one that works 24*7) in order to send the values to the server for analytics.

Software requirements

>> Arduino software
>> ESP32 Libraries and Drivers
>> The code (shipped along with the project)
>> Hardware components
>> Assembling the system
>> The circuit consists of the ESP32 board, a level shifter (for energy meter), temperature and humidity sensor, and flow meter connected using wires on a PCB. (Refer the figure given below)



Setup

Software component

>> Connect the laptop to the hardware setup (built above)
>> 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)
>> Open the main.ino file provided.
>> Change the WiFi SSID and password to that of your network.
>> For intra-college network, use http and for other networks, use https in the server link.
>> 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.
>> Open the serial monitor. You should be able to see WiFi connected, and the values read by the board, and sent to the server.
>> Your setup is now ready for deployment.

Deployment

>> Insert the hardware component into a box, with 2 holes, one for temperature sensor, and other for connection wires from energy meter and flow meter to pass through.
>> Place it in a place where the WiFi signal is strong enough.
>> 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.

Analytics

Server:

In the serverCode folder, run “fetch.py”.
It would collect all the data from the oneM2M server and place them in a .db (database file type) file.
Now run “run.py”.
This would start the python web app that displays all the analytics.
Run

```
$ ipconfig
```

Or

```
$ ifconfig
```

On your terminal depending on your operating system and write the server’s IP down.
(For example, say the ip is XX.XX.XX.XX)

Client:

Run “XX.XX.XX.XX/home” on any web browser that supports HTML5 (eg. Chrome, Edge, Firefox, Safari, etc.)

Reading and analyzing the data

Observing the graphs, we can find out the following things:

1. Flow vs Time
    a. The status button just above this graph tells us whether the motor is in safe state to run or not.
    b. The graph shows different value of flows at different times.
2. Power vs Time
    a. This graph tells the user about the status of power the motor consumed at different times.
    b. This graph also has a related status button. If the user just want to know the status of power and do not want to see the whole data, then he can just see this buttons.
3. Temperature vs Time
    a. This graph tells us about the temperature at different times.
    b. The status button of this graph tells about whether it is too cold, moderate or too hot.
4. Humidity vs Time
    a. This graph tells us about the humidity at different times.
    B. The status button of this graph tells about whether humidity is too high, moderate or too low.
5. Litres vs Day
    a. This graph tells us about the different water pumped during the different days.
    b. We can get some idea about the consumption and which day is the most busy for the motor.
6. Efficiency vs Time
    a. This graph describes the efficiency at different times.
    b. The efficiency status tells us about the current condition of the pump.

$$efficiency = \frac{flow(l/min) * height(m) * 60}{367 * power}$$

Design Details

Components

>> ESP 32 board
>> Flow meter
>> Energy Meter
>> Temperature and RH sensor
>> OneM2M Server
>> Front end server
>> Power bank
>> Other components required for making the circuit like the PCB, wires, power source, WiFi source etc.

Constraints

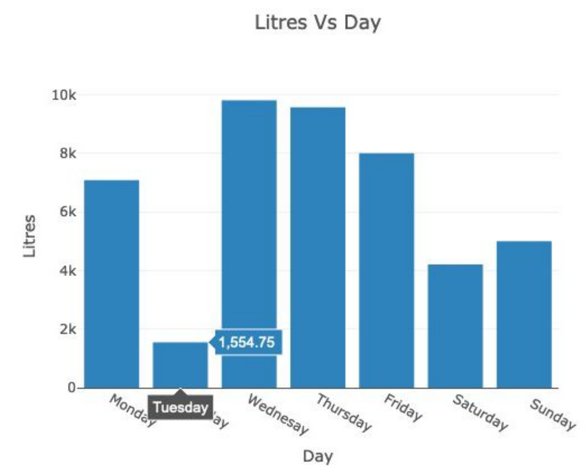
The performance of the device would depend on server capacity, WIFI speed and also on the amount of possible storage in the server.
The amount of accuracy that can be obtained from the flow sensor depends on how steady the flow is.
The accuracy of the temperature and humidity sensor depends on where the resistor part is placed, and how much of the external environment it is exposed to.
The energy meter might display error 0xE0 or 0xE3. These errors can be referred in the energyMeterFuntions.h file's comments.
Solutions: Make sure connections to and from the level shifter are firm. Make sure there is enough delay between successive calls of reading energy meter's register values.
Other solutions are error specific, look into the energy meter's documentation for rectification.

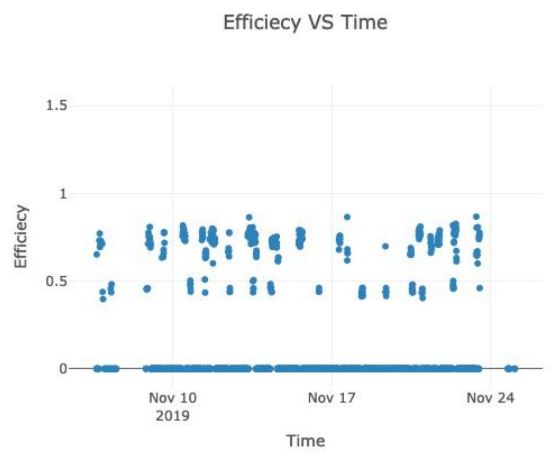
Workflow

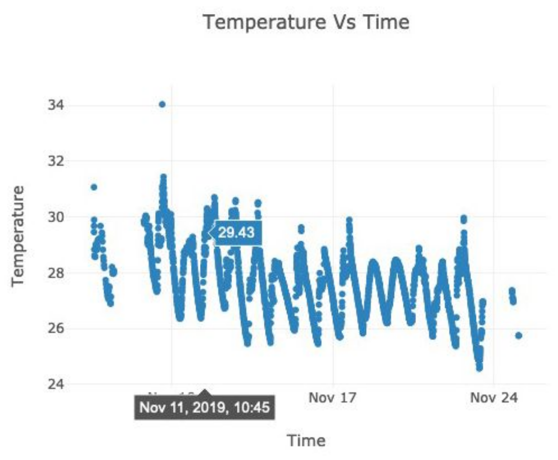
The energy meter essentially just measures the current and voltage passing through it, and based on these and the power factor, it calculates values of power, energy, frequency, etc. and stores them in it's registers to be read by modBus.
The flow meter measures flow by measuring pulse counts in an interval and sending these as interrupts to the board.
The temperature and RH sensor measures temperature using change in resistance wrt change in temperature of the surroundings.
The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measures, processes this changed resistance values and change them into digital form.
The data from the board is sent to the OneM2M server.

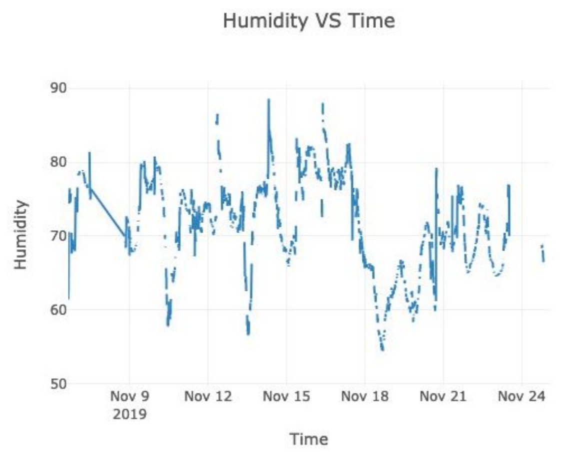
At the (front-end) server, further analytics would be done to monitor the performance of the pump.

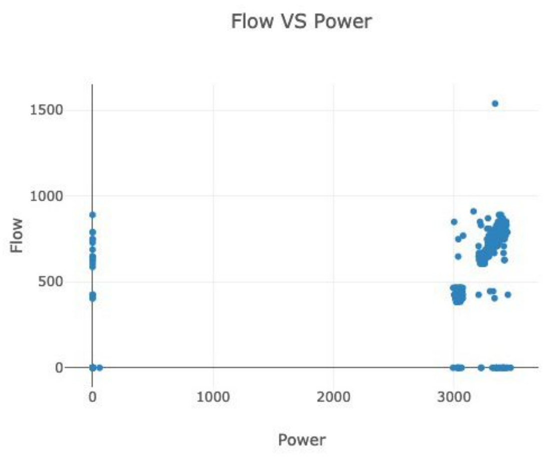
Experimental Results

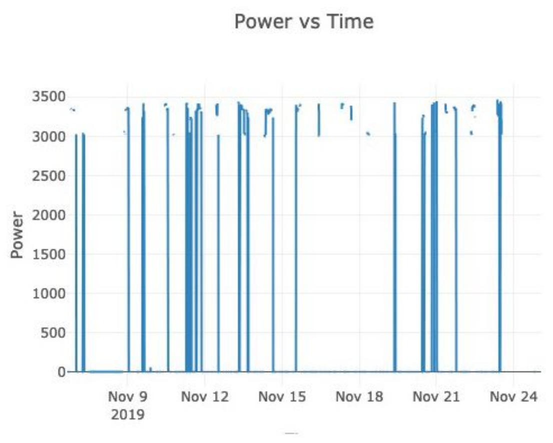

Litres Vs Day
Litres
10k
8k
6k
4k
2k
Monday
Tuesday
Wednesay
Thursday
Friday
Saturday
Sunday
Day


Efficiecy VS Time
Efficiecy
Nov 10
Nov 17
Nov 24
Time


Temperature Vs Time
Temperature
Nov 11, 2019, 10:45
Nov 17
Nov 24
Time


Humidity VS Time
Humidity
Nov 9
Nov 12
Nov 15
Nov 18
Nov 21
Nov 24
Time


Flow VS Power
Flow
Power


Power vs Time
Power
Nov 9
Nov 12
Nov 15
Nov 18
Nov 21
Nov 24