University of Bahrain
College of Engineering
Electrical & Electronic Engineering Department

REPORT ABOUT

SMART WATER CONSUMPTION MEASUREMENT SYSTEM BASED ON IOT

Prepared By
Farooq Abdulaziz Jan Mohammed
Student ID# 19992198

Supervised By
Prof. Mohab Mangoud
Professor, Telecommunications and Networks Engineering

January 2022

Abstract

Several factors contribute to water waste, including user behavior and water use. Further, a lack of information or monitoring of water pipelines makes it challenging to know the system's current state. By managing a water pipeline properly, these problems can be alleviated. This project proposes a hardware and software implementation based on IoT technologies in collecting data and monitoring water consumption in residential. A low-cost water sensor attached to a low-power microcontroller with high-end capabilities provides the platform to understand the water volumes consumed daily through real-time data collection and analysis.

Contents

1. Introduction

The project's main objective is to develop a water management system where real-estate owners can monitor water consumption on a daily basis using the power of IoT (Internet of Things). The lack of a system that can monitor and alert the homeowners about any water faulty or leakage made it difficult to track these problems when they occurred. Thus, we must have a remotely accessible system to monitor and alert the family members or homeowners about an unusual water usage reading. The proposed project is built on an Arduino Node-MCU board equipped with external hardware to sense a tap's water flow rate. The water flow sensor will push the reading of water consumption to the cloud through the Wi-Fi connection, and then the records can be accessed through the dashboard or mobile application.

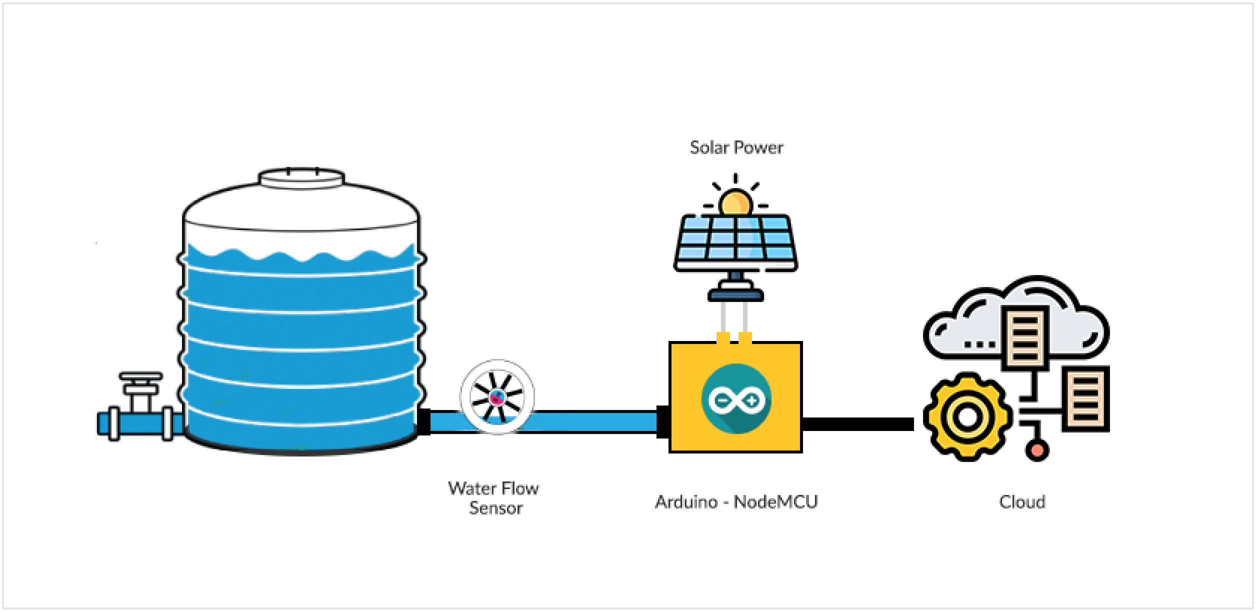


Figure 01

2. Methodology

This proposed method calculates water usage and records data in real-time. Sensors can be installed on the main water tank or reservoir for general water consumption readings, or we can install these sensors on different house inlets for more detailed information on water consumption. The water usage can be seen in the LCD monitor, and these values are constantly updated to the Thinger server with the support of the ESP8266 Wi-Fi module, which has a constant power supply. And these values can also be seen in the Thinger mobile application.

2.1. Block Diagram

The block diagram consists of the components that are utilized in this project. The Arduino Node-MCU microcontroller collects the sensor data and sends it to the cloud via wireless communication and hotspot Wi-Fi. The OLED display connected to the microcontroller will display a real-time reading of the water flow rate and the total volume. The collected data will be analyzed on Thinger cloud and then pushed to display on a web app. The cloud is integrated with IFTTT services for instant alerts and actions.

IFTTT is a third-party service that can be integrated through Thinger platform and can be triggered by predefined conditions.

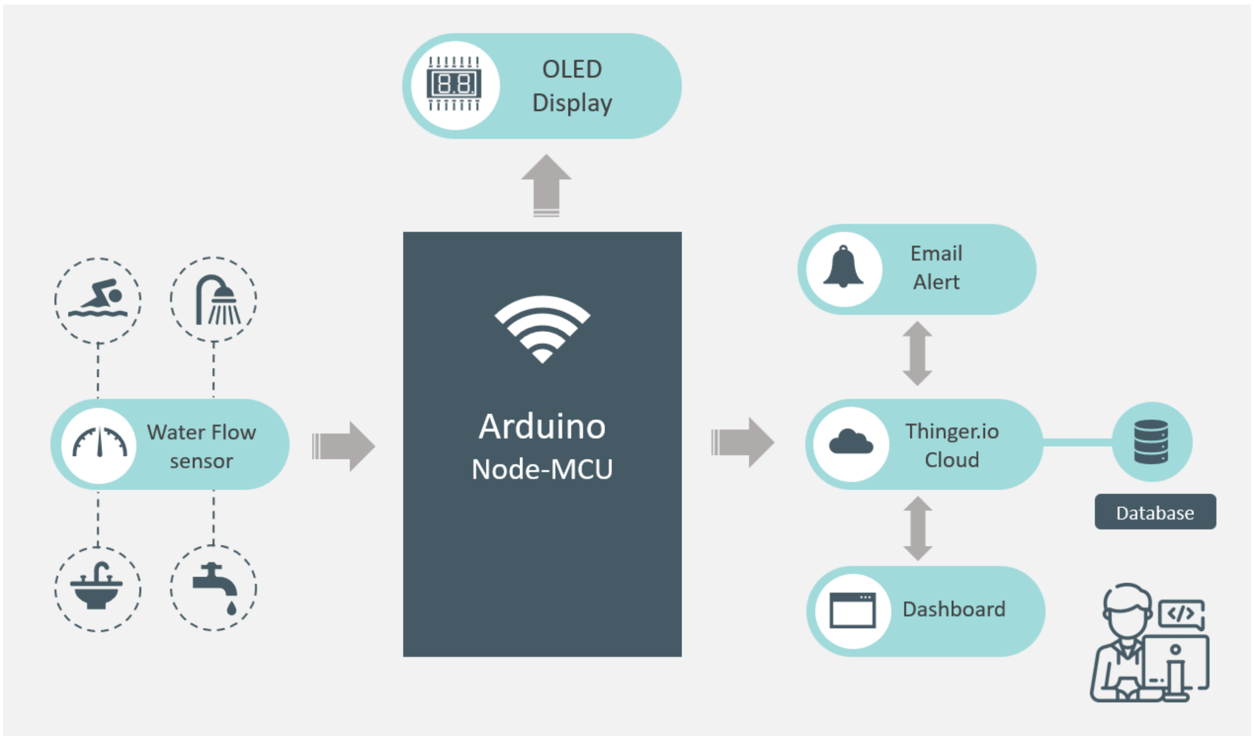


Figure 02

2.2. Flow Chart

The process of the system goes into different phases, starting from microcontroller Node-MCU initialization and then configuring the water flow sensor to read the data whenever water starts to flow. Next, the data is sent to Thinger cloud via wireless communication. This system automatically displays the water flow rate, volume, billing information, and historical data on Thinger portal.

Whenever the microcontroller failed to connect to the existing defined Wi-Fi, it will keep searching for the hotspot until it get connected.

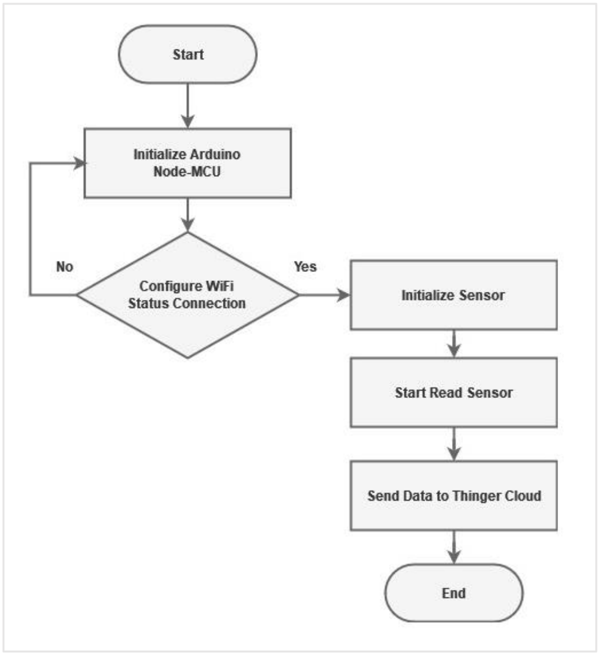

Start
Initialize Arduino Node-MCU
No
Configure WiFi Status Connection
Yes
Initialize Sensor
Start Read Sensor
Send Data to Thinger Cloud
End

Figure 03

2.3. Hardware Development

2.3.1. Components

Based on gathered requirements, Hardware components were selected upon their features to attain our goal efficiently. Below is the list of components used in this project

NodeMCU – Arduino
The NodeMCU is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip called the ESP8266, it can be operated with low voltage, and has built-in Wi-Fi along with other features

Water Flow Sensor
The sensor is responsible of reading the volume of water flow based on the rotator rolls and the speed of it changes with a different rate of flow.

OLED Display
A small screen size 0.96-inch OLED display with 128x64 resolution to display water flow rate and total consumed liters directly from the sensor.

Solar Panel
To operate the Arduino board with a reliable source power, low maintenance cost, and to avoid electricity power consumption, we should utilize the power of solar power to run the application board. For this project we have used 2 pcs. of solar plates that output 3.3w with total number of 15 watts.

Rechargeable Lithium Battery
To make sure that the power supply is available 24/7, we need to connect a rechargeable battery along with the Arduino board. The proposed power solution requires 1 lithium battery connected to a charging board.

TP4056 Micro USB 5V 1A 18650 Lithium Battery Charger Board
Micro USB lithium ion 18650 battery charging board, built-in TP4056 chip for lithium battery charging

2.3.2 Circuit Construction

For testing purposed, a bread board were used to build the circuit to make sure all the components are located correctly. Below a circuit board that shows the wiring were done by using an online software to build the circuit scheme.

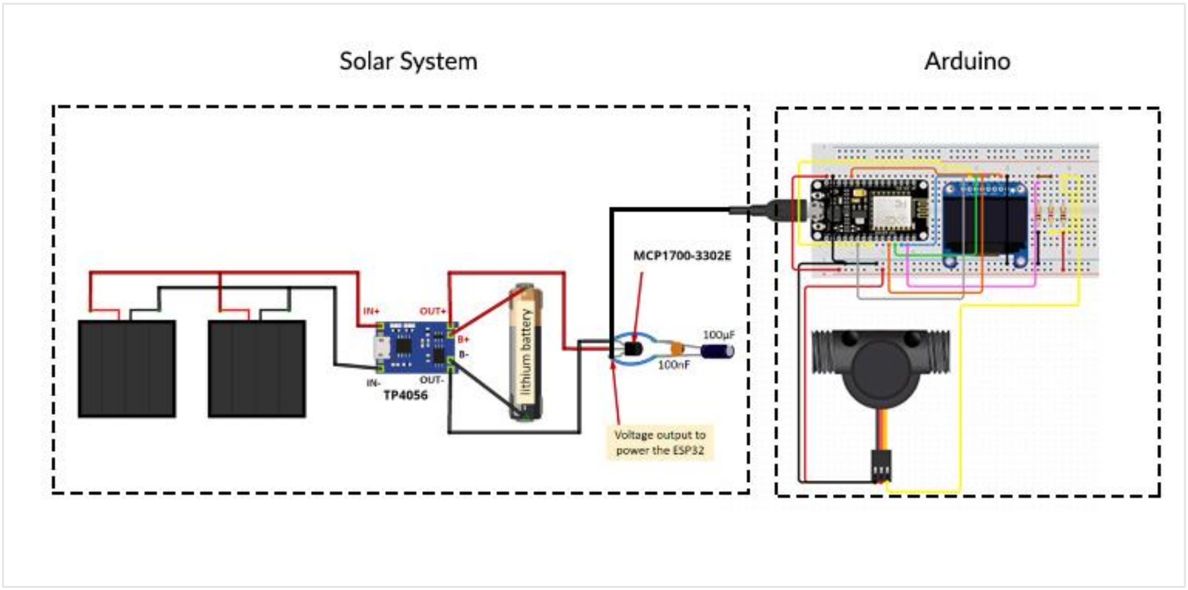


Figure 04

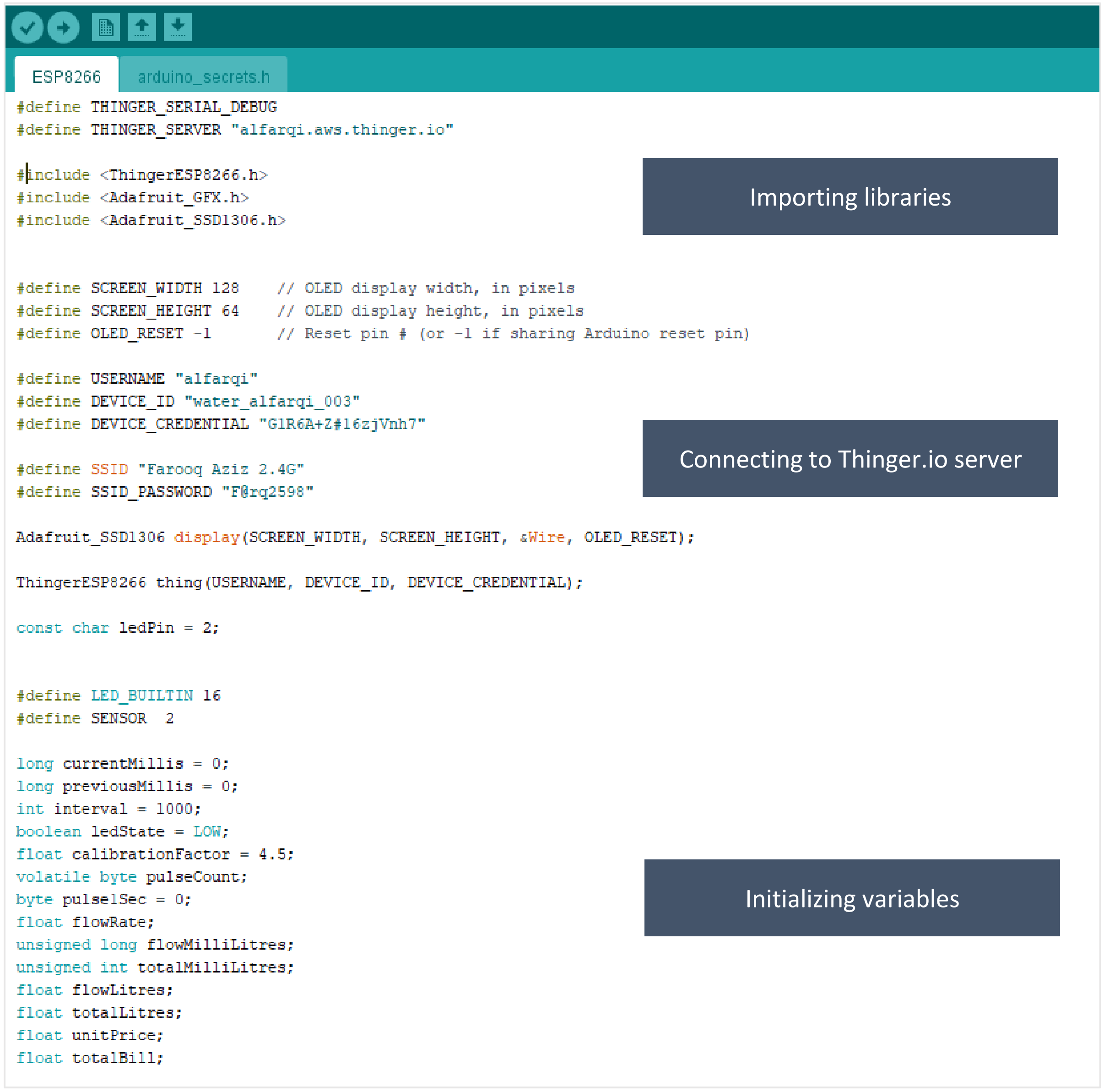
2.4. Software Development

2.4.1 Platform Used

Arduino IDE
This software is being used to program the microcontroller with the required features and formulas for calculating the water flow rate and volumes

Thinger.io
Thinger.io is a cloud IoT Platform that provides every needed tool to prototype, scale and manage connected products integrated with many plugins to extend the power of this platform, such as Node-Red, Vs Code….etc.

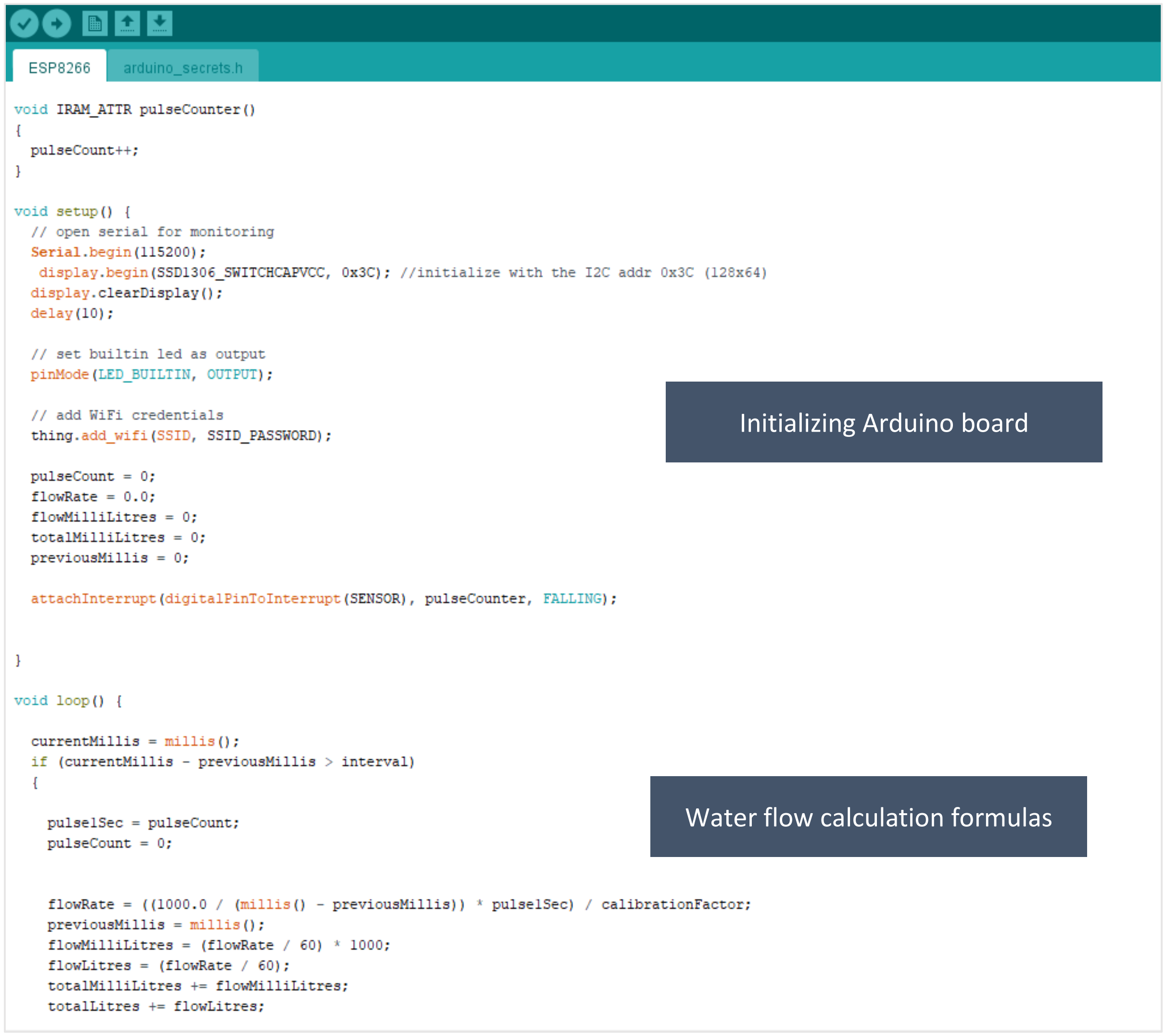
2.4.2 Source Code



```
#define THINGER_SERIAL_DEBUG
#define THINGER_SERVER "alfarqi.aws.thinger.io"

#include <ThingerESP8266.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>


#define SCREEN_WIDTH 128    // OLED display width, in pixels
#define SCREEN_HEIGHT 64    // OLED display height, in pixels
#define OLED_RESET -1       // Reset pin # (or -1 if sharing Arduino reset pin)

#define USERNAME "alfarqi"
#define DEVICE_ID "water_alfarqi_003"
#define DEVICE_CREDENTIAL "G1R6A+Z#16zjVnh7"

#define SSID "Farooq Aziz 2.4G"
#define SSID_PASSWORD "F@rq2598"

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

const char ledPin = 2;


#define LED_BUILTIN 16
#define SENSOR  2

long currentMillis = 0;
long previousMillis = 0;
int interval = 1000;
boolean ledState = LOW;
float calibrationFactor = 4.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned long flowMilliLitres;
unsigned int totalMilliLitres;
float flowLitres;
float totalLitres;
float unitPrice;
float totalBill;
```

Figure 05

In figure 05, shows the written code for initializing the Arduino board with libraries and global variables. For this project we have used OLED display and for that we need to initialize it by calling *Adafruit_SSD1306* method. Then we have initialized the connected Thinger.io cloud platform by calling *TingerESP8266* method and passing the login and device information. Next, we have set the variables with constants and defined the others to be used on setup and loop method.

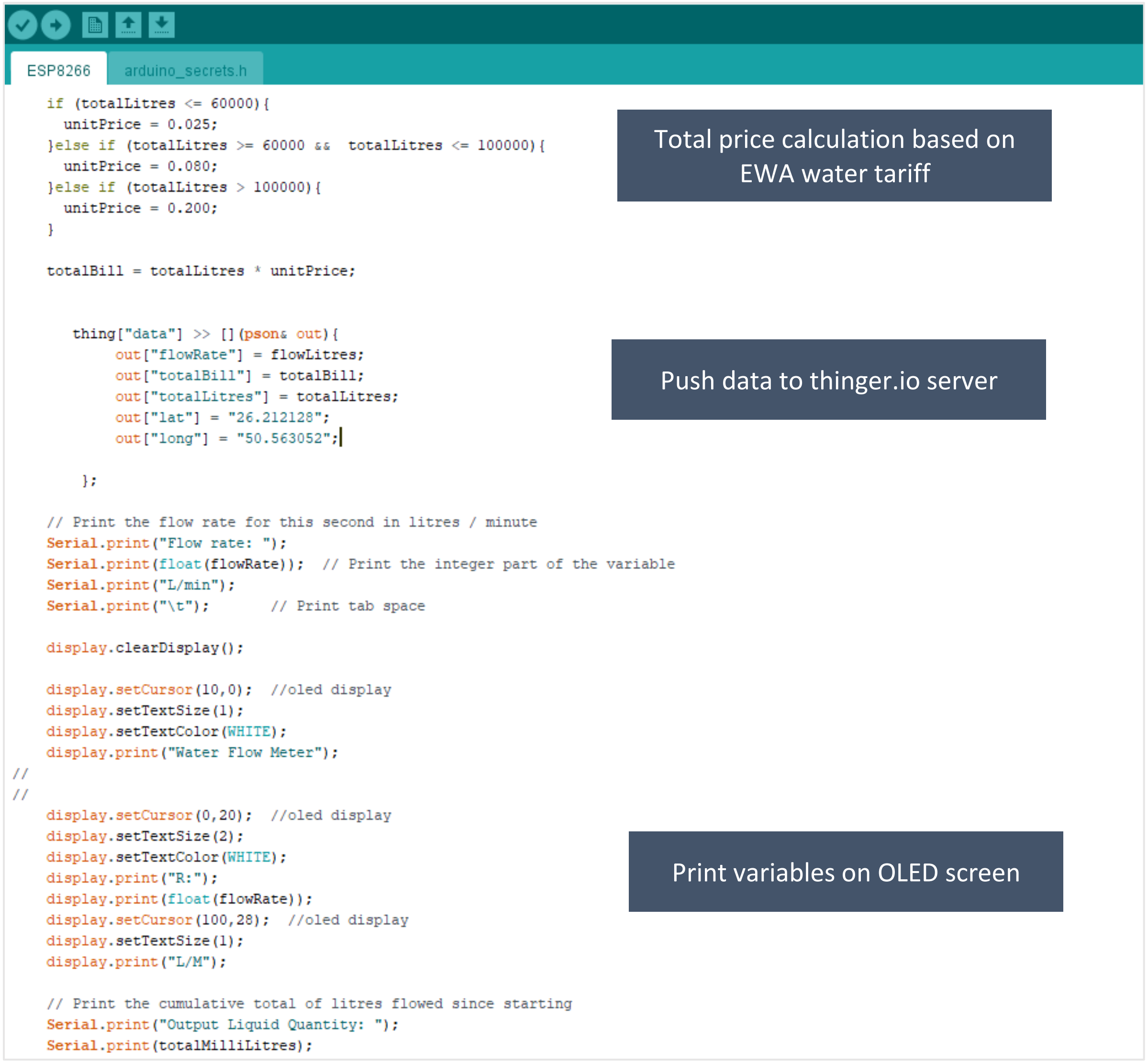


```
void IRAM_ATTR pulseCounter()
{
  pulseCount++;
}

void setup() {
  // open serial for monitoring
  Serial.begin(115200);
   display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //initialize with the I2C addr 0x3C (128x64)
  display.clearDisplay();
  delay(10);

  // set builtin led as output
  pinMode(LED_BUILTIN, OUTPUT);

  // add WiFi credentials
  thing.add_wifi(SSID, SSID_PASSWORD);

  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;

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


}

void loop() {

  currentMillis = millis();
  if (currentMillis - previousMillis > interval)
  {

    pulse1Sec = pulseCount;
    pulseCount = 0;


    flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
    previousMillis = millis();
    flowMilliLitres = (flowRate / 60) * 1000;
    flowLitres = (flowRate / 60);
    totalMilliLitres += flowMilliLitres;
    totalLitres += flowLitres;
```

Figure 06

In figure 06, inside the setup method we have initiated the microcontroller port number, and then cleared the OLED display. Next, the microcontroller was connected to the Wi-Fi by calling *thing.add_wifi* method. A constant value for the predefined variables were set to be used in *loop* method.

Inside the loop method, we have used some formulas to calculate the water flow rate and the volume of total liters consumed by looping through these formulas we can get the total number of liters consumed


ESP8266
arduino_secrets.h
Total price calculation based on EWA water tariff
Push data to thinger.io server
Print variables on OLED screen

```
    if (totalLitres <= 60000){
      unitPrice = 0.025;
    }else if (totalLitres >= 60000 &&  totalLitres <= 100000){
      unitPrice = 0.080;
    }else if (totalLitres > 100000){
      unitPrice = 0.200;
    }

    totalBill = totalLitres * unitPrice;


       thing["data"] >> [](pson& out){
            out["flowRate"] = flowLitres;
            out["totalBill"] = totalBill;
            out["totalLitres"] = totalLitres;
            out["lat"] = "26.212128";
            out["long"] = "50.563052";

        };

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

    display.clearDisplay();

    display.setCursor(10,0);  //oled display
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.print("Water Flow Meter");
//
//
    display.setCursor(0,20);  //oled display
    display.setTextSize(2);
    display.setTextColor(WHITE);
    display.print("R:");
    display.print(float(flowRate));
    display.setCursor(100,28);  //oled display
    display.setTextSize(1);
    display.print("L/M");

    // Print the cumulative total of litres flowed since starting
    Serial.print("Output Liquid Quantity: ");
    Serial.print(totalMilliLitres);
```

Figure 07

In figure 07, we are still doing the calculations inside the loop method to prepare the values to be sent to the cloud. Finally, the water flow rate and the total number of liters were sent to the screen by calling display.print method

3. Results and Data Analysis

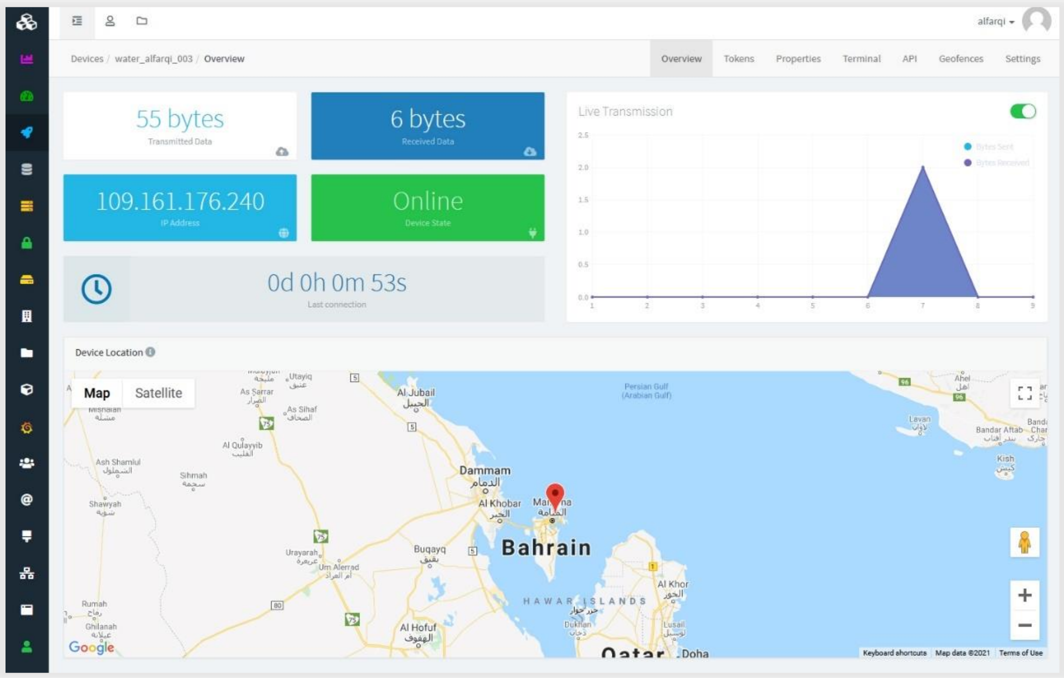
All the data were pushed to the cloud are saved in a database for advance analysis and calculations. Here are a few screenshots of the platform screen that displays different charts based on the given data

3.1. Thinger Platform

This platform provides a ready-to-use scalable cloud infrastructure for having things talk to each other, built on an Open Source platform. Device manufacturers and companies can start controlling their devices from the internet in minutes, without worrying about the additional infrastructure required

- **Connect Devices:** No matter what device, what network or what manufacturer you have, Connect is fully compatible with it. With Thinger.io, you can create bidirectional communications with Linux, Arduino, Raspberry Pi, or MQTT devices, as well as with edge technologies like Sigfox or LoRaWAN or any other API data sources that are connected to the internet.
- **Data Bucket:** Data Buckets are an efficient, scalable, and affordable way for IoT developers and developers of a variety of IoT applications to store and aggregate IoT data in real-time, allowing the data to be aggregated in real-time.
- **Display Real-time** or **Stored Data**: The Dashboard creates awesome dashboards on the fly by displaying real-time or stored data in multiple representations such as time series, donut charts, gauges, or even custom made representations for you to customize.
- **Trigger events and data values:** A Node-RED rule engine embedded in the Node-RED database is used for tracking events and values
- **Extend with custom features:** Enhance your platform with multiple plugins that can be used to run IoT projects as well as integrate with your company's software or engage with any other Internet service as needed.
- **Customize the appearance** and branding with our fully drag and drop frontend, that allows introducing your own branding colors, logotypes, and domain name

a. Connection Screen



55 bytes
Transmitted Data

6 bytes
Received Data

109.161.176.240
IP Address

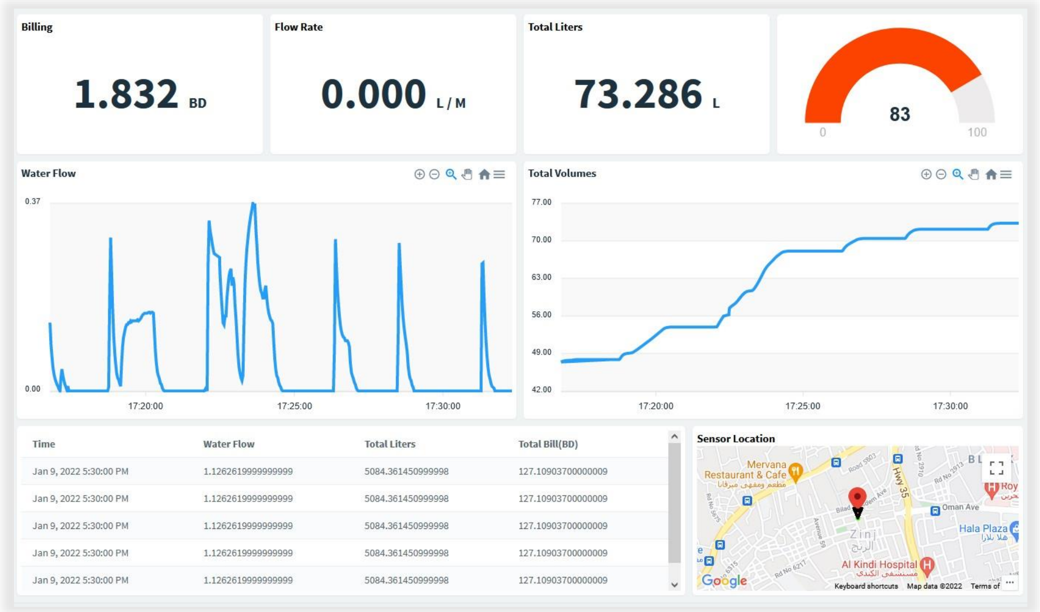
Online
Device State

0d 0h 0m 53s
Last connection

Figure 08

The platform offers a dashboard screen to display the connection status with the Arduino board and it shows how its interacting with the data and how many total bytes were transferred and from which IP address its connected. Additional to that, it shows the location of your device on a google. Also you have the option to enable or disable the connection to this device and stop receiving data from the connected microcontroller.

b. **Main Dashboard**



Billing: 1.832 BD

Flow Rate: 0.000 L / M

Total Liters: 73.286 L

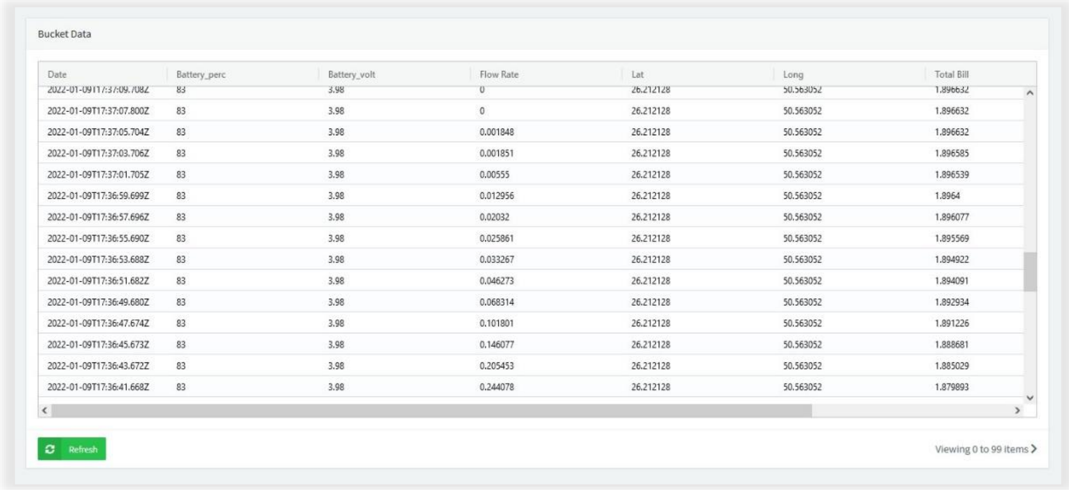
83 (0 – 100)

Time	Water Flow	Total Liters	Total Bill(BD)
Jan 9, 2022 5:30:00 PM	1.1262619999999999	5084.361450999998	127.10903700000009
Jan 9, 2022 5:30:00 PM	1.1262619999999999	5084.361450999998	127.10903700000009
Jan 9, 2022 5:30:00 PM	1.1262619999999999	5084.361450999998	127.10903700000009
Jan 9, 2022 5:30:00 PM	1.1262619999999999	5084.361450999998	127.10903700000009
Jan 9, 2022 5:30:00 PM	1.1262619999999999	5084.361450999998	127.10903700000009

Figure 09

To analyze and present the data as charts and numbers you can create your own dashboard and then you choose the chart style from the list and set the connection to the data and build the entire screen by drag and drop the boxes.

c. Database



Bucket Data

Date	Battery_perc	Battery_volt	Flow Rate	Lat	Long	Total Bill
2022-01-09T17:37:09.708Z	83	3.98	0	26.212128	50.563052	1.896632
2022-01-09T17:37:07.800Z	83	3.98	0	26.212128	50.563052	1.896632
2022-01-09T17:37:05.704Z	83	3.98	0.001848	26.212128	50.563052	1.896632
2022-01-09T17:37:03.706Z	83	3.98	0.001851	26.212128	50.563052	1.896585
2022-01-09T17:37:01.705Z	83	3.98	0.00555	26.212128	50.563052	1.896539
2022-01-09T17:36:59.699Z	83	3.98	0.012956	26.212128	50.563052	1.8964
2022-01-09T17:36:57.696Z	83	3.98	0.02032	26.212128	50.563052	1.896077
2022-01-09T17:36:55.690Z	83	3.98	0.025861	26.212128	50.563052	1.895569
2022-01-09T17:36:53.688Z	83	3.98	0.033267	26.212128	50.563052	1.894922
2022-01-09T17:36:51.682Z	83	3.98	0.046273	26.212128	50.563052	1.894091
2022-01-09T17:36:49.680Z	83	3.98	0.068314	26.212128	50.563052	1.892934
2022-01-09T17:36:47.674Z	83	3.98	0.101801	26.212128	50.563052	1.891226
2022-01-09T17:36:45.673Z	83	3.98	0.146077	26.212128	50.563052	1.888681
2022-01-09T17:36:43.672Z	83	3.98	0.205453	26.212128	50.563052	1.885029
2022-01-09T17:36:41.668Z	83	3.98	0.244078	26.212128	50.563052	1.879893

Figure 10

The data are pushed every 3 seconds to the server and are stored in a database for further analysis and computation. The database stores passed data date, flow rate, latitude, longitude, total bill, and total liters.

3.2. OLED display sensor data

The microcontroller (Arduino-NodeMCU) is connected to the OLED display and soldered on a printed PCB to accommodate the plastic box,

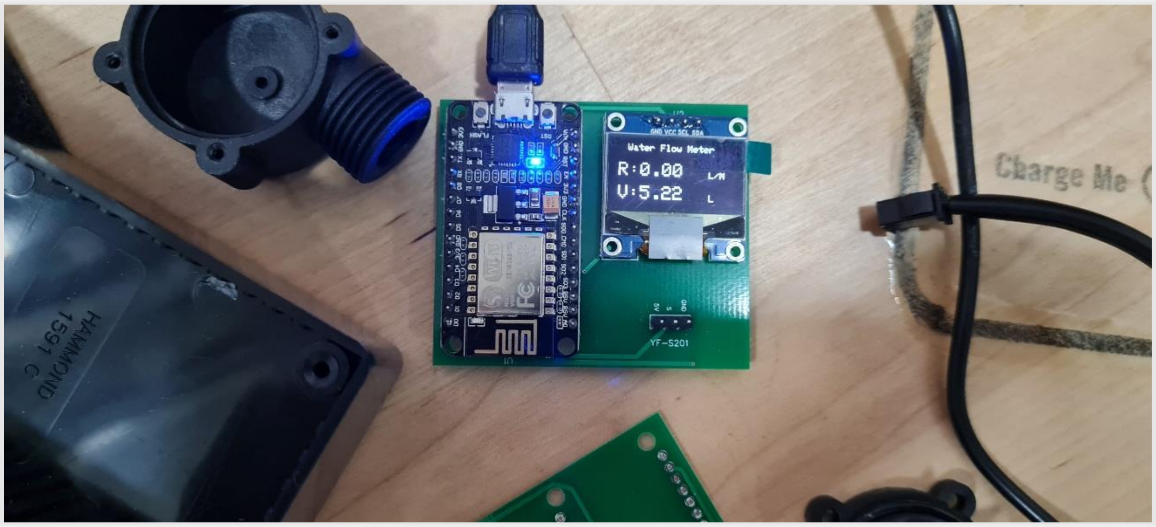
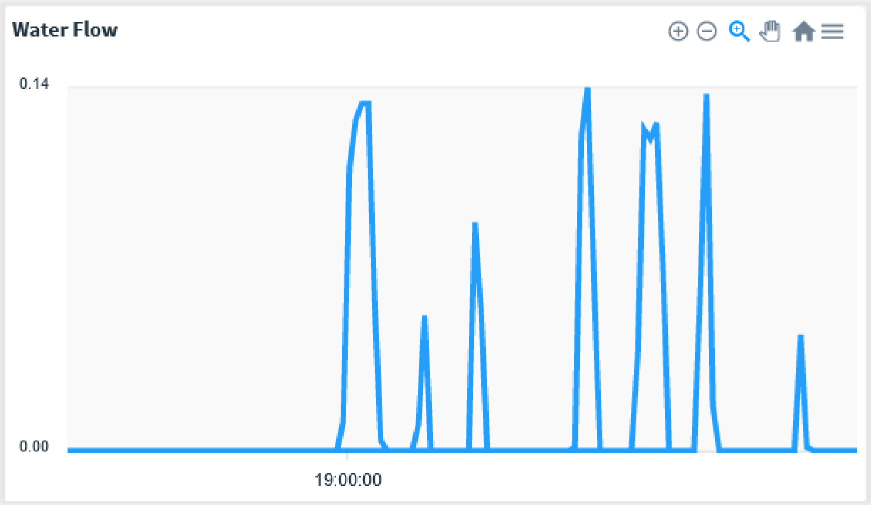


Figure 11

3.3. Data Analysis



Water Flow
Figure 12 display a chart that shows the water flow rate per minute on a real-time

Figure 12

Total Volumes
Figure 13 shows the total number of liters consumed

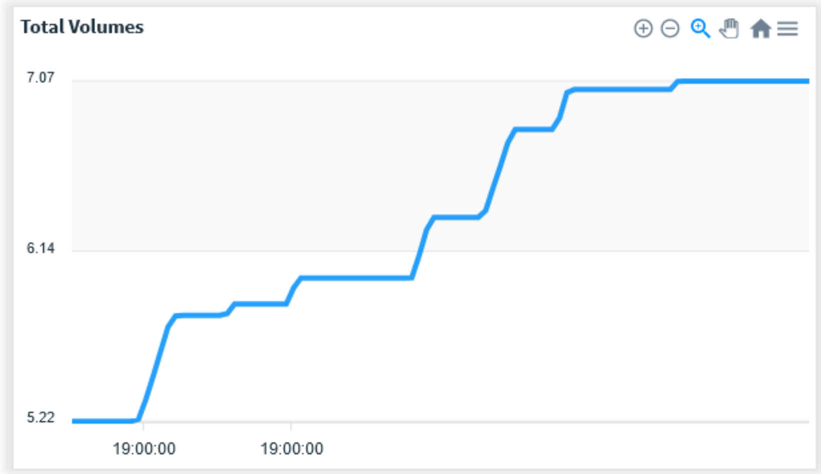
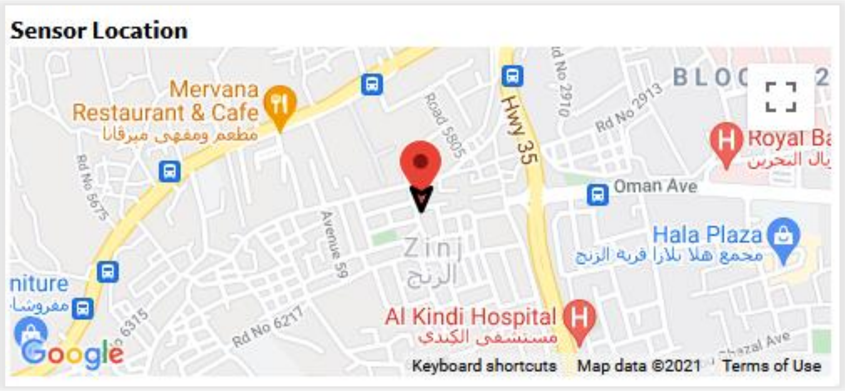


Figure 13



Sensor Location
Figure 14 shows the location of the water flow sensor. This feature is handy if we have different sensors that are located in other places

Figure 14

4. Challenges

During the development of this project, I went through two main challenges:

- Providing a continuous, low-cost power supply
- Wireless signals coverage

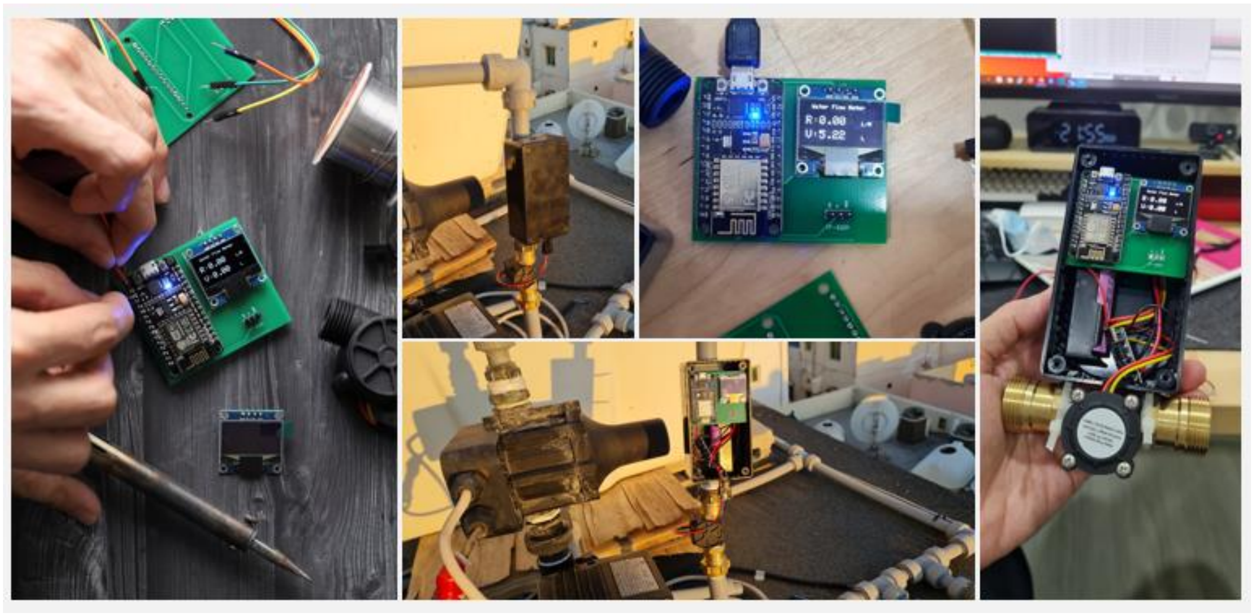
And after investigating more into a possible solution, I came up with a proper solution for these two challenges, and they are:

- Integrating a mini solar panel to Arduino board to charge the lithium battery during the morning time
- Installing wireless mesh solution near to the device to boost the internet signals

5. Cost and Time Calculations

Description	Cost	Time
Water Flow Sensor + Arduino board	10 BD	4 weeks / 3 hours per day to complete building the circuit and integrating with thinger.io cloud
Solar System	8 BD	
Total Cost	18 BD	

6. Clips



7. Conclusion

This report has clearly described how we can build a water consumption smart meter using Arduino Node-MCU microcontroller connected to a water flow sensor and a cloud-based platform to present real-time data that can help the household owners understand their daily water consumption. Although, integrating solar energy as source power to operate the microcontroller enabled us to achieve our goals more efficiently with low power consumption. Finally, future plans will focus on improving the proposed smart meter by integrating the power of machine learning algorithms into the existing system to help predict future water needs and costs based on the level of consumption.

8. References

1. IoT Water Flow Meter using ESP8266 & Water Flow Sensor, Alex Newton, accessed Oct 01, 2021, <https://how2electronics.com/iot-water-flow-meter-using-esp8266-water-flow-sensor/>
2. Fuentes, H., Mauricio, D. Smart water consumption measurement system for houses using IoT and cloud computing. *Environ Monit Assess* 192, 602 (2020). https://doi.org/10.1007/s10661-020-08535-4
3. Power ESP32/ESP8266 with Solar Panels, Nick, accessed Dec 12, 2021, <https://randomnerdtutorials.com/power-esp32-esp8266-solar-panels-battery-level-monitoring/>