IoT Based Water Flow Meter using ESP8266 & Water Flow Sensor

This project we are creating an IOT based water flow sensor using ESP8266 and water flow sensor. We will display the water flow rate, total volume and the current liquid flowing. The water flow rate, total volume and the current liquid flowing will be uploaded to firebase where it can be viewed/monitored from anywhere. The sensor uses the principles of electromagnetism, such that, when liquids flow through the sensor, the flow action impacts the fins of a turbine in the sensor, causing the wheel to spin. The shaft of the turbine wheel is connected to a hall effect sensor so that with every spin, a pulse is generated, and by monitoring that pulse with a microcontroller, the sensor can be used in determining the volume of fluid passing through it.

Components required

Following are the components required for making this project.

1.1 Hardware:

1. NodeMCU- ESP8266-12E Board
2. Water Flow Sensor
3. Connecting Wires- Jumpers
4. Breadboard
5. Micro USB cable

1.2 Software:

Arduino IDE - https://www.arduino.cc/en/Main/Software

1.3 Supporting Softwares

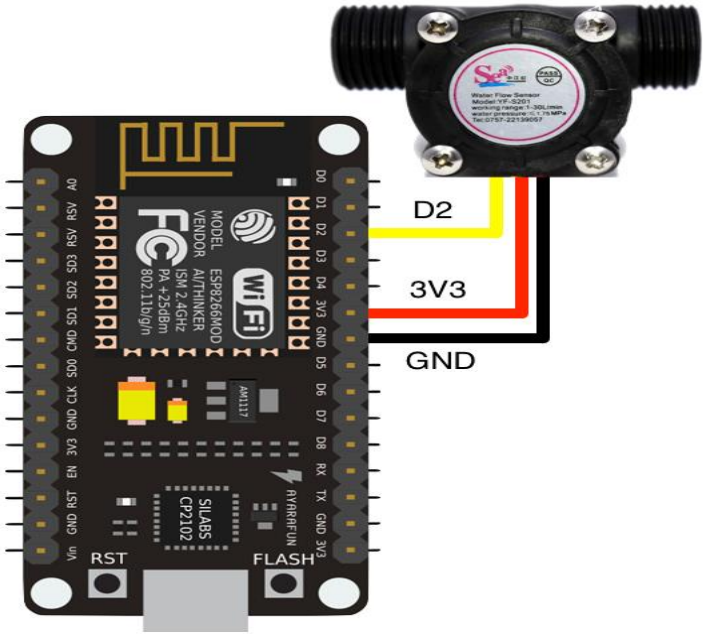
For data collection from sensors and storage will use a cloud service that is - https://firebase.google.com/

2.0 Circuit connection

The yellow wire(signal/pulse) connect to pin D2

The black wire(ground) connects to GND pin

The red wire connects to pin 3v



3.0 Code

The main objective is to monitor the signal pin of the **YF-S201** (Hall Effect Water Flow Meter / Sensor) to detect when the hall sensor is triggered (flow is detected) and increment a variable to show the increased inflow. However, to do that efficiently and accurately, we will use the interrupt feature of the ESP8266 such that whenever the hall sensor detects the rotating magnet, a rising edge interrupt is fired and registered by the ESP8266. The total number of interrupts fired over a particular time is then used in generating the flowrate and the total volume of liquid that has traveled through the flow meter.

```
#include "FirebaseESP8266.h” // Install Firebase ESP8266 library
#include <Arduino.h>
#include <ArduinoJson.h>
#include <EEPROM.h>
#define USE_SERIAL Serial
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

//set these to run
#define FIREBASE_HOST"flowrate-7d99b-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH"3LfKKfnsxFpTMoerPSV0n5im3yLjq8ZoV1OKNsjj"
#define WIFI_PASSWORD"Incubator#2013"
#define WIFI_SSID"Incubator"



// Variable init
const int buttonPin = D2; // variable for D2 pin
const int ledPin = D7;
char push_data[200]; //string used to send info to the server ThingSpeak
int addr = 0; //endereço eeprom
byte sensorInterrupt = 0; // 0 = digital pin 2
```

```
// The hall-effect flow sensor outputs approximately 4.5 pulses per second per
// litre/minute of flow.
float calibrationFactor = 4.5;

volatile byte pulseCount;

float flowRate;
unsigned int CurrentLiquidFlowing;
unsigned long totalMilliLitres;
unsigned long totalLitres;

unsigned long oldTime;

//SSID and PASSWORD for the AP (swap the XXXXX for real ssid and password )
const char * ssid = "Incubator";
const char * password = "Incubator#2013";

FirebaseData;
FirebaseData ledData;

FirebaseJson json;
```

```
//HTTP client init
HTTPClient http;
void ICACHE_RAM_ATTR pulseCounter() {
  pulseCount++;
  Serial.print("+");
}
void setup() {
  Serial.begin(115200); // Start the Serial communication to send messages to the computer
  delay(10);
  Serial.println('\n');

startWIFI();

  // Initialization of the variable “buttonPin” as INPUT (D2 pin)
  pinMode(buttonPin, INPUT);

  // Two types of blinking
  // 1: Connecting to Wifi
  // 2: Push data to the cloud
  pinMode(ledPin, OUTPUT);

  pulseCount = 0;
  flowRate = 0.0;
```

```
  CurrentLiquidFlowing = 0;
  totalMilliLitres = 0;

  oldTime = 0;

  digitalWrite(buttonPin, HIGH);
  attachInterrupt(digitalPinToInterrupt(buttonPin), pulseCounter, FALLING);

}

void loop() {
  if (WiFi.status() == WL_CONNECTED && (millis() - oldTime) > 1000) // Only process counters once per second
  {
    // Disable the interrupt while calculating flow rate and sending the value to
    // the host
    detachInterrupt(sensorInterrupt);

    // 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() - oldTime)) * pulseCount) / calibrationFactor;

// Note the time this processing pass was executed. Note that because we've
// disabled interrupts the millis() function won't actually be incrementing right
// at this point, but it will still return the value it was set to just before
// interrupts went away.
oldTime = 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.
CurrentLiquidFlowing = (flowRate / 60) * 1000;

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

unsigned int frac;

// Print the flow rate for this second in litres / minutezz
Serial.print("Flow rate: ");
Serial.print(int(flowRate)); // Print the integer part of the variable
Serial.print("."); // Print the decimal point
// Determine the fractional part. The 10 multiplier gives us 1 decimal place.
```

```
  frac = (flowRate - int(flowRate)) * 10;
  Serial.print(frac, DEC); // Print the fractional part of the variable
  Serial.print("L/min");
  // Print the number of litres flowed in this second
  Serial.print("  Current Liquid Flowing: "); // Output separator
  Serial.print(CurrentLiquidFlowing);
  Serial.print("mL/Sec");

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

  if (flowRate > 0) {
    digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(100);

   if (Firebase.setFloat(firebaseData, "/FirebaseIOT/flowRate", flowRate))
{
  Serial.println("PASSED");
  Serial.println("PATH: " + firebaseData.dataPath());
  Serial.println("TYPE: " + firebaseData.dataType());
  Serial.println("ETag: " + firebaseData.ETag());
```

```
  Serial.println("------------------------------------");
  Serial.println();
 }
 else
 {
  Serial.println("FAILED");
  Serial.println("REASON: " + firebaseData.errorReason());
  Serial.println("------------------------------------");
  Serial.println();
 }

 if (Firebase.setFloat(firebaseData, "/FirebaseIOT/CurrentLiquidFlowing",
CurrentLiquidFlowing))
 {
  Serial.println("PASSED");
  Serial.println("PATH: " + firebaseData.dataPath());
  Serial.println("TYPE: " + firebaseData.dataType());
  Serial.println("ETag: " + firebaseData.ETag());
  Serial.println("------------------------------------");
  Serial.println();
 }
 else
 {
```

```
    Serial.println("FAILED");
    Serial.println("REASON: " + firebaseData.errorReason());
    Serial.println("------------------------------------");
    Serial.println();
  }
  if (Firebase.setFloat(firebaseData, "/FirebaseIOT/OutputLiquidQuantity", totalMilliLitres))
  {
    Serial.println("PASSED");
    Serial.println("PATH: " + firebaseData.dataPath());
    Serial.println("TYPE: " + firebaseData.dataType());
    Serial.println("ETag: " + firebaseData.ETag());
    Serial.println("------------------------------------");
    Serial.println();
  }
  else
  {
    Serial.println("FAILED");
    Serial.println("REASON: " + firebaseData.errorReason());
    Serial.println("------------------------------------");
    Serial.println();
  }
}
    // Reset the pulse counter so we can start incrementing again
```

```
    pulseCount = 0;

    // Enable the interrupt again now that we've finished sending output
    attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
  } else if (WiFi.status() != WL_CONNECTED) {
    startWIFI();
  }
  Firebase.begin(FIREBASE_HOST,FIREBASE_AUTH);

}



void startWIFI(void) {
  digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(100);

  WiFi.begin(ssid, password); // Connect to the network
  Serial.print("Connecting to ");
  Serial.print(ssid);
  Serial.println(" ...");
  oldTime = 0;
  int i = 0;
```

```
  digitalWrite(ledPin, LOW);    // turn the LED off by making the voltage LOW
  delay(100);

  while (WiFi.status() != WL_CONNECTED) { // Wait for the Wi-Fi to connect
    digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(2000);
    Serial.print(++i);
    Serial.print('.');
    digitalWrite(ledPin, LOW);    // turn the LED off by making the voltage LOW
    delay(100);
  }
  delay(2000);
  Serial.print('\n');
  Serial.print("Connection established!");
  Serial.print("IP address:\t");
  Serial.print(WiFi.localIP()); // Send the IP address of the ESP8266 to the computer


}
```

Code running in the serial monitor arduino

```
Flow rate: 0.0L/min  Current Liquid Flowing: 0mL/Sec  Output Liquid Quantity: 953mL
++++++++++++++++++++++++++Flow rate: 5.7L/min  Current Liquid Flowing: 96mL/Sec  Output Liquid Quantity: 1049mL
PASSED
PATH: /FirebaseIOT/flowRate
TYPE: float
ETag: YVt3OJbLZAxcdNql8AevFhDX+lE=
------------------------------------

PASSED
PATH: /FirebaseIOT/CurrentLiquidFlowing
TYPE: int
ETag: LeRnFSK/V16ZiAU+NUhOlc/M81c=
------------------------------------

PASSED
PATH: /FirebaseIOT/OutputLiquidQuantity
TYPE: int
ETag: 3oHMpAnryedYUv2lATiOX3Vaj34=
------------------------------------
```

Firebase

Realtime data is recorded in the firebase i.e. the flowrate, current liquid flowing and output liquid quantity are shown in the firebase.