Project 17:

Control Flow using Flow sensor, and Dc Pump

1. Introduction

This Project is use to Control Flow Of using Water Flow Sensor and Dc pump, its help to control flow of water in Agriculture Field and its is very useful in Drip Irrigation system.

COMPONENTS: -

1. **WEMOS**
2. **DC PUMP**
3. **WATER FLOE SENSOR**

APPLICATIONS: -

Water flow sensors can measure the rate of flow of water either by measuring velocity or displacement. These sensors can also measure the flow of water like fluids such as measuring milk in a dairy industry etc.…

There are various types of water flow sensors available based on their diameter and method of measuring.

A cost-effective and most commonly used water flow sensor is

Paddlewheel sensor. It can be used with water-like fluids.

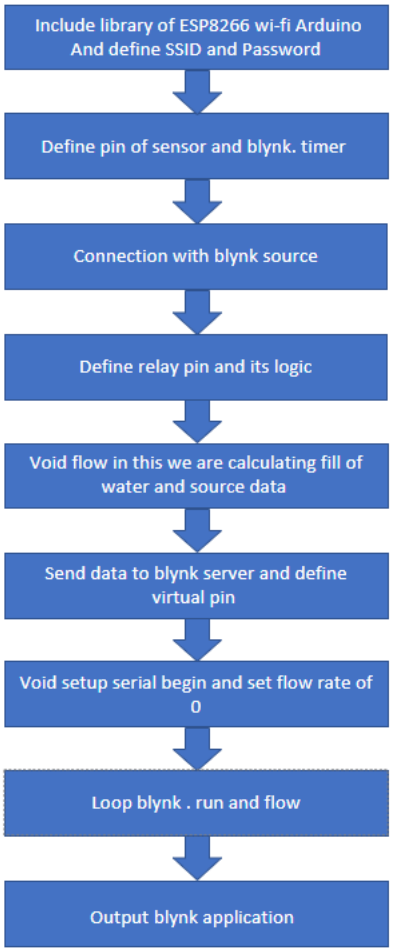
For the type of applications where a straight pipe is not available for inlet, Positive displacement flow meter is used. This type of water flow sensor can be used for viscous liquids also.

For working with dirty water and wastewater which may be conductive, Magnetic flow meter is used. For applications such as sewage water, slurries, and other dirty liquids Ultrasonic flow meters are used.

OBJECTIVES: -

Flow sensors are used to measure the flow rate of blood or oxygen through a vessel. Implantable flow sensors are commonly incorporated into a flexible cuff (Fig. 20.10) that is fitted around the vessel whose flow rate is to be measured.

FLOW CHART: -



PROGRAMMING: -

```
#define BLYNK_PRINT Serial          // Uncomment for debugging


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>


#include <ESP8266mDNS.h>  // For OTA with ESP8266
```

```
#include <WiFiUdp.h>  // For OTA
#include <ArduinoOTA.h>  // For OTA


// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "07DQbnsxODMhV3t0HC4QFZ5ZeHKIoFF6";


// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Tshimologong-General";
char pass[] = "letsinnovate";


#define PULSE_PIN D2  //gpio4
#define FLOW_CALIBRATION 8.2
#define VPIN_TOTAL_LITERS       V1
#define VPIN_FLOW_RATE          V2
#define VPIN_FLOW_MILLI_LITERS  V3
#define VPIN_RESET              V4


#define OTA_HOSTNAME "Test Flow Sensor Water Meter"
```

```
BlynkTimer timer;


volatile long pulseCount = 0;
float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;
float totalLitres;
float totalLitresold;
unsigned long oldTime;




BLYNK_CONNECTED() { // runs once at device startup, once connected to server.


  Blynk.syncVirtual(VPIN_TOTAL_LITERS); //gets last know value of V1 virtual pin


}
```

```
// Restores last know value of V1 virtual pin which we got it from blynk server
BLYNK_WRITE(VPIN_TOTAL_LITERS)
{
  totalLitresold = param.asFloat();


}



BLYNK_WRITE(VPIN_RESET) {  // reset all data with button in PUSH mode on virtual pin V4
  int resetdata = param.asInt();
  if (resetdata == 0) {
    Serial.println("Clearing Data");
    Blynk.virtualWrite(VPIN_TOTAL_LITERS, 0);
    Blynk.virtualWrite(VPIN_FLOW_RATE, 0);
    flowRate = 0;
    flowMilliLitres = 0;
    totalMilliLitres = 0;
```

```
    totalLitres = 0;
    totalLitresold = 0;
  }
}




void pulseCounter()
{
  pulseCount++;
}


void flow()
{


  if ((millis() - oldTime) > 1000)   // Only process counters once per second
  {
    detachInterrupt(PULSE_PIN);
    flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) / FLOW_CALIBRATION;
```

```
    oldTime = millis();
    flowMilliLitres = (flowRate / 60) * 1000;
    totalMilliLitres += flowMilliLitres;
    totalLitres = totalLitresold + totalMilliLitres * 0.001;
    unsigned int frac;


    // Print the flow rate for this second in liters / minute
    Serial.print("flowrate: ");
    Serial.print(int(flowRate));  // Print the integer part of the
variable


    Serial.print(".");             // Print the decimal point
    frac = (flowRate - int(flowRate)) * 10; // Determine the fractional
part. The 10 multiplier gives us 1 decimal place.
    Serial.print(frac, DEC) ;      // Print the fractional part of the
variable
    Serial.print("L/min");


    Serial.print("  Current Liquid Flowing: ");  // Print the number of
liters flowed in this second
    Serial.print(flowMilliLitres);
    Serial.print("mL/Sec");
```

```
    Serial.print("  Output Liquid Quantity: ");  // Print the cumulative total of liters flowed since starting
    Serial.print(totalLitres);
    Serial.println("L");


    pulseCount = 0;  // Reset the pulse counter so we can start incrementing again


    attachInterrupt(PULSE_PIN, pulseCounter, FALLING);    // Enable the interrupt again now that we've finished sending output
  }


}




void sendtoBlynk()  // In this function we are sending values to blynk server
{
  Blynk.virtualWrite(VPIN_TOTAL_LITERS, totalLitres);          // Total water consumption in liters (L)
```

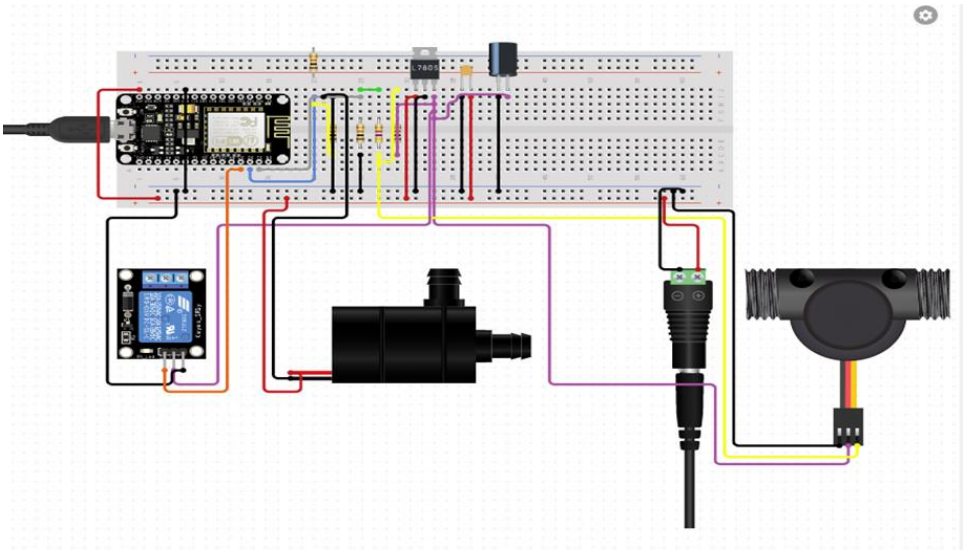
```
  Blynk.virtualWrite(VPIN_FLOW_RATE, flowRate);           // Displays the flow rate for this second in liters / minute (L/min)
  //  Blynk.virtualWrite(VPIN_FLOW_RATE, flowMilliLitres);  // Displays the number of liters flowed in second (mL/Sec)



}



void setup()
{
  Serial.begin(57600);
  Blynk.begin(auth, ssid, pass);
  ArduinoOTA.setHostname(OTA_HOSTNAME);  // For OTA - Use your own device identifying name
  ArduinoOTA.begin();  // For OTA



  pulseCount        = 0;
  flowRate          = 0.0;
  flowMilliLitres   = 0;
  totalMilliLitres  = 0;
  oldTime           = 0;
  totalLitresold = 0;
```

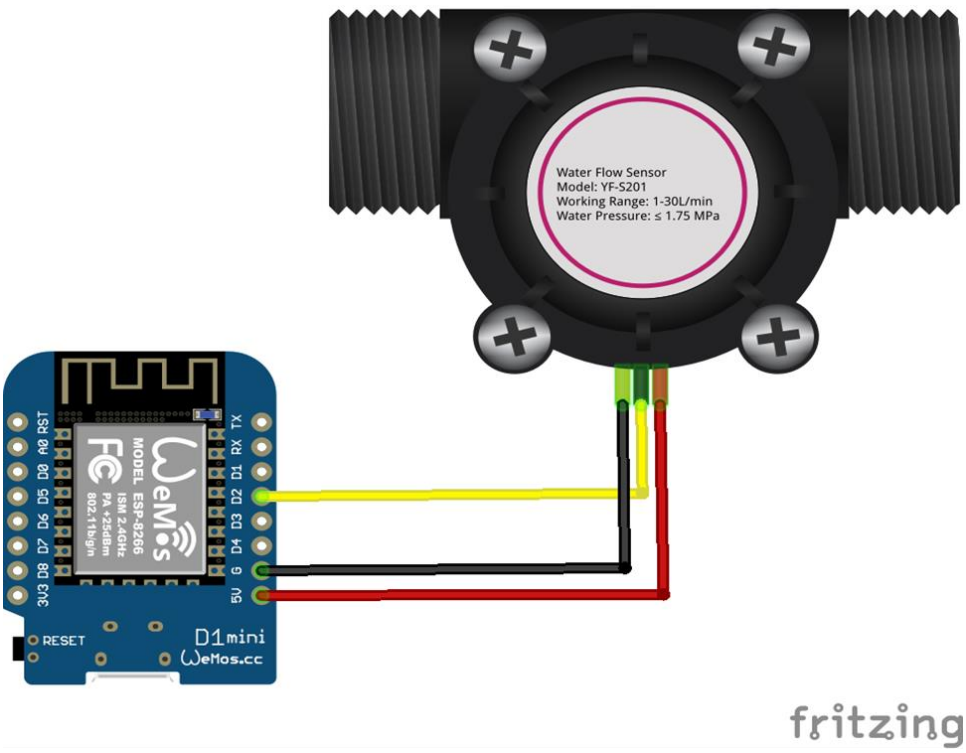
```
  pinMode(PULSE_PIN, INPUT);  // Initialization of the variable "PULSE_PIN" as INPUT (D2 pin)


  attachInterrupt(PULSE_PIN, pulseCounter, FALLING);


  timer.setInterval(10000L, sendtoBlynk); // send values blynk server every 10 sec


}



void loop()
{


  Blynk.run();
  ArduinoOTA.handle();  // For OTA
  timer.run();
  flow();


}
```

HARDWARE CONNECTION: -

1. Connect relay to flow sensor and wemos
2. connect pin D0 to D2
3. Connect pin GND to GND
4. Connect pin signal to D4
5. Connect pin GND to VCC

CIRCUIT DIAGRAM: -




Water Flow Sensor
Model: YF-S201
Working Range: 1-30L/min
Water Pressure: ≤ 1.75 MPa
RST A0 D0 D5 D6 D7 D8 3V3
RX TX D1 D2 D3 D4 G 5V
MODEL ESP-8266
ISM 2.4GHz
PA +25dBm
802.11b/g/n
RESET
D1mini
WeMos.cc
fritzing