Project 21:

Water Flow Controller using blynk

Introduction

In this activity, we will be reading and displaying the water flow meter on blynk using NodeMCU and also control the tap with the solenoid

COMPONENTS: -

1. **SOIL MOISTURE SENSOR**
2. **WEMOS**
3. **RELAY**

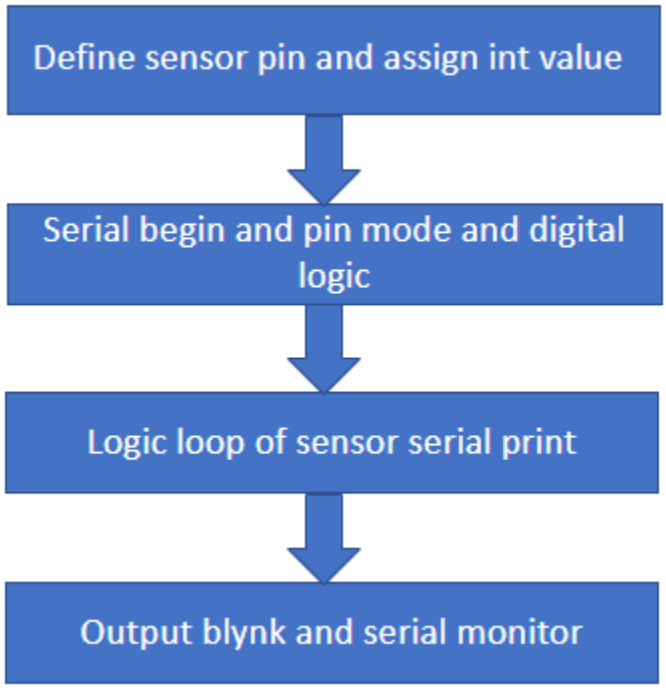
APPLICATION: -

One of the most common uses of a flow control valve is to regulate the speed of motors or cylinders within the system. ... They are also used in many consumer applications such as showers, faucets, and lawn watering systems to easily reduce the amount of water consumed without impacting the overall system performance.

OBJECTIVE: -

A water flow regulator is a common device that is used to maintain a specified flow rate no matter how the pressure varies throughout the supply line. If ever water is passing through the line at a high pressure, the regulator will close just enough to ensure that the water continues at a steady pace.

FLOW CHART: -



PROGRAMMING: -

```
int sensorInterrupt = 0;  // interrupt 0
int sensorPin       = 2; //Digital Pin 2
int solenoidValve = 5; // Digital pin 5
unsigned int SetPoint = 400; //400 milileter



/*The hall-effect flow sensor outputs pulses per second per litre/minute of flow.*/
float calibrationFactor = 90; //You can change according to your datasheet

volatile byte pulseCount =0;

float flowRate = 0.0;
unsigned int flowMilliLitres =0;
unsigned long totalMilliLitres = 0;

unsigned long oldTime = 0;

void setup()
{
```

```
  // Initialize a serial connection for reporting values to the
host
  Serial.begin(9600);
  pinMode(solenoidValve , OUTPUT);
  digitalWrite(solenoidValve, HIGH);
  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);


  /*The Hall-effect sensor is connected to pin 2 which uses
interrupt 0. Configured to trigger on a FALLING state
change (transition from HIGH
  (state to LOW state)*/
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
//you can use Rising or Falling
}


void loop()
{


   if((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.
  flowMilliLitres = (flowRate / 60) * 1000;
```

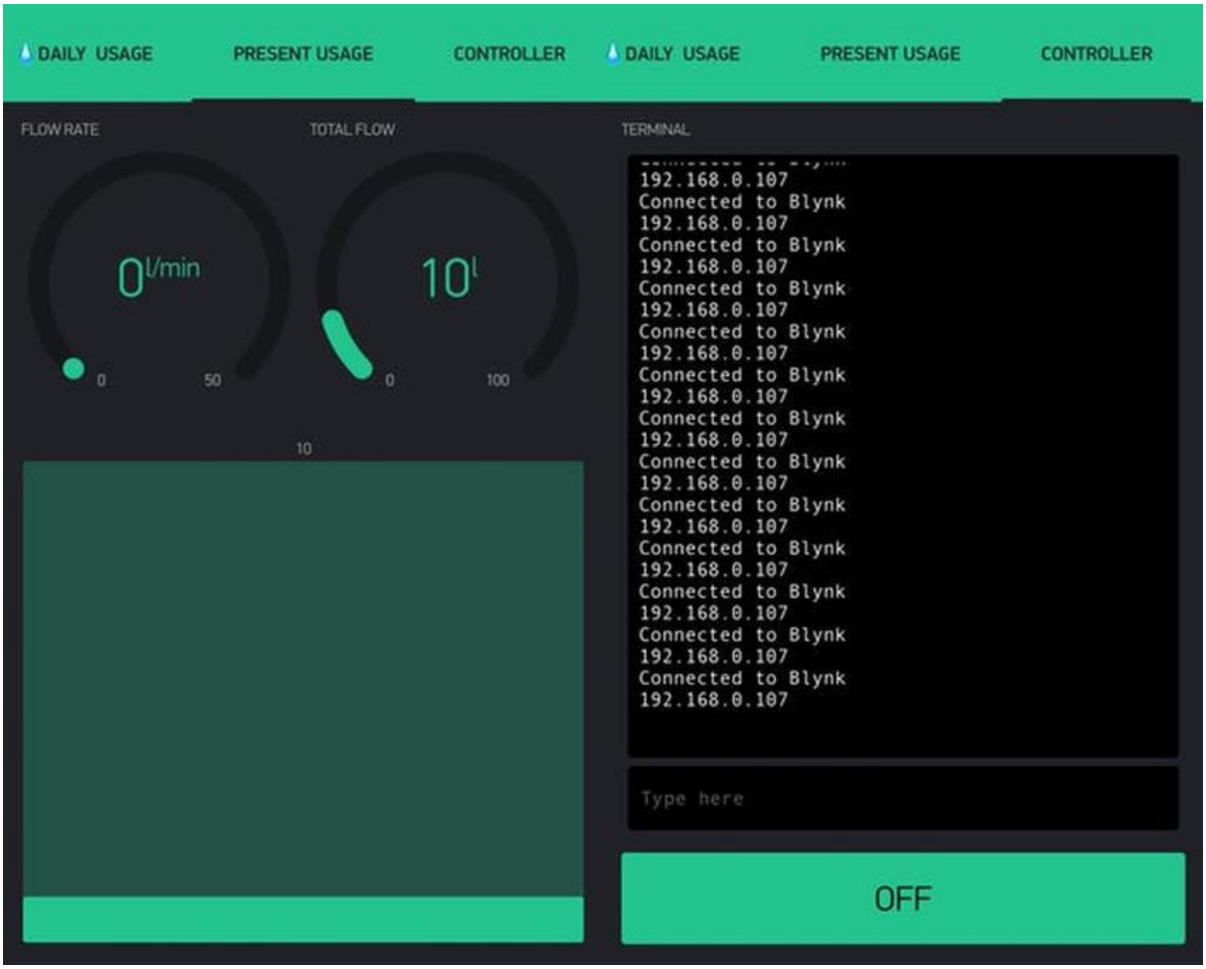
```
   // Add the millilitres passed in this second to the
cumulative total
   totalMilliLitres += flowMilliLitres;


   unsigned int frac;


   // Print the flow rate for this second in litres / minute
   Serial.print("Flow rate: ");


 Serial.print(flowMilliLitres, DEC);  // Print the integer part of
the variable
   Serial.print("mL/Second");
   Serial.print("\t");


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


   if (totalMilliLitres > 40)
   {
     SetSolinoidValve();
   }
```

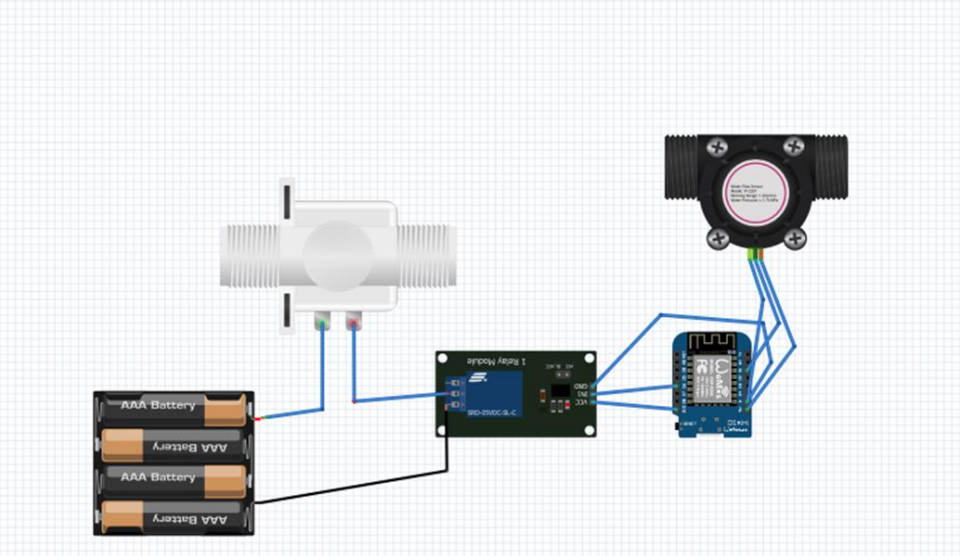
```
// Reset the pulse counter so we can start incrementing
again
    pulseCount = 0;


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

//Insterrupt Service Routine


void pulseCounter()
{
  // Increment the pulse counter
  pulseCount++;
}

void SetSolinoidValve()
{
  digitalWrite(solenoidValve, LOW);
}
```

HARDWARE CONNECTION: -

1. Connect sensor pin D1 to wemos D4 and relay to signal pin
2. Connect pin vcc to vcc
3. Connect pin GND to GND

CIRCUIT DAIGRAM: -




AAA Battery
AAA Battery
AAA Battery
AAA Battery
1 Relay Module
SRD-05VDC-SL-C
GND
VIN
VCC