Project Report

Instructor: Dr. Avinash Eranki

Embedded Systems

Samar Singhai, Vidya Ajay, Ramanathan Annamalai

April 25, 2022

BM2003

Part 1

Measuring flow rate using YFS401 and Arduino

The YF-S401 sensor

The YF-S401 water flow sensor consists of a PVC body, a water rotor and a hall effect sensor. When water flows through the rotor, it rolls, and its speed changes with the rate of flow. The hall effect sensor outputs the corresponding pulse signal, and this can be read using software. The sensor consists of a PVC body, and is suitable to detect flow in water dispensers, water heaters, gardening or coffee machines.

Specifications:

- Max Supply Voltage: 5-12V DC
- Water Flow Range: 0.3-6 L/min
- The Output Pulse High Level: $>$ 4.5 VDC (DC input voltage 5 V)
- The Output Pulse Low Level: $<$ 0.5 VDC (DC input voltage 5 V)
- Operating Current: 15 mA (DC 5V)

Figure 1: The YFS401

The Hall-effect principle

The effect of generating a measurable voltage in a conducting material using a magnetic field is called the Hall Effect. The basic physical principle underlying this effect is the Lorentz force Hall Effect Sensors consist basically of a thin piece of rectangular p-type semiconductor material passing a continuous current through itself.

When the device is placed within a magnetic field, the magnetic flux lines exert a force on the semiconductor material which deflects the charge carriers, electrons and holes, to either side of the semiconductor slab. This movement of charge carriers is a result of the magnetic force they experience passing through the semiconductor material.

As these electrons and holes move sideways, a potential difference is produced between the two sides of the semiconductor material by the build-up of these charge carriers.

Calculating the water flow

With each revolution of the wheel, the volume of fluid flowing through it is certain. At the same time, the number of pulses generated per revolution of the wheel is also a certain amount. Thus, we can establish an equation between the number of pulses and the water flow.

In fact, the YF series uses a ring magnet with six alternating poles, so that for every revolution, three low levels and three high levels, i.e., three pulses, are generated.

In the YF-S401, for every liter of water that flows, the Hall Sensor outputs a fixed number of pulses, say "m".

Also, the total volume of fluid flowing through the water flow sensor is equal to the water flow rate(Q - unit L/s) multiplied by time t(sec).

Number of pulses in 1L = m
So for each pulse, $\frac{1}{m}$ liters of water flows.
Number of pulses in t sec = N
So N pulses corresponds to $\frac{N}{m}$ liters.

$$\frac{N}{m} = V_t$$
$$flowrate, Q(L/sec) = \frac{V_t}{time}$$
$$Q(L/s) = \frac{N}{m * t}$$

Calibrating the sensor

On using the 5880 factor (taken from the datasheet as the number of pulses in 1L), we noticed substantial errors in our water flow measurement, so we tweaked the factor around till we got an accurate reading. The calibration factor that got us the correct reading was $\frac{5880}{4330} = 1.357$, meaning our sensor gave 4330 pulses per litre.

The Arduino code

```
1
2  int inputPin=2;//input from flowmeter sensor
3  float flowRate;
4
5  // factor = pulses generated for 1 liter of water flow;
6  float factor=4330;
7  float volume;
8  float currentTime;
9  volatile unsigned long pulseCounter=0; //varible for counting pulses generated
       due to water flow
10
11 void setup() {
12   Serial.begin(9600);
13
14   //configuring pin for INPUT
15   pinMode(inputPin,INPUT);
16   //Initially set the inputPin to HIGH
17   digitalWrite(inputPin,HIGH);
18
19   //for continuously checking if any Rising edge came or not, if there is
       rising edge call IRS_Counter() function
20   attachInterrupt(digitalPinToInterrupt(inputPin),IRS_Counter,RISING);
21
```

```
22   //millis() gives the time since arduino started, setting current time to
       millis()
23   currentTime=millis();
24   volume=0; // initial volume is zero
25
26 }
27
28 void loop() {
29
30   //do all the calculations every 1 second
31   if(millis()-currentTime>=1000){
32     //stop checking for rising edge through interrupt
33     detachInterrupt(digitalPinToInterrupt(inputPin));
34     currentTime=millis();
35
36
37     //changing the current time to millis() so millis()-currentTime=0
38     // flowrate= (pulse count)/(pulses per liter) (in liter/s)
39     // for milliliter/min flowrate=(pulse count)*60*1000/(pulse per liter)
40
41
42     flowRate=(pulseCounter*60*1000)/factor;
43
44     //volume=(prev volume)+(flowrate*time)
45     volume+=(flowRate/60);
46
47     // Serial.print("Pulse Count:");
48     // Serial.println(pulseCounter);
49
50     // printing the flowrate and volume
51     Serial.print("Flow rate of water(in mL/min):");
52     Serial.println(flowRate,DEC);
53     Serial.print("Volume of water(mL):");
54     Serial.println(volume,DEC);
55
56     //setting pulse count to zero for another loop
57       pulseCounter=0;
58
59    // again set up the interrupt to check for rising edge
60    attachInterrupt(digitalPinToInterrupt(inputPin),IRS_Counter,RISING);
61   }
62 }
63
```

```
void IRS_Counter(){
  // if we hit a rising edge increase the pulse count by 1
  pulseCounter++;
}
```

The circuit setup

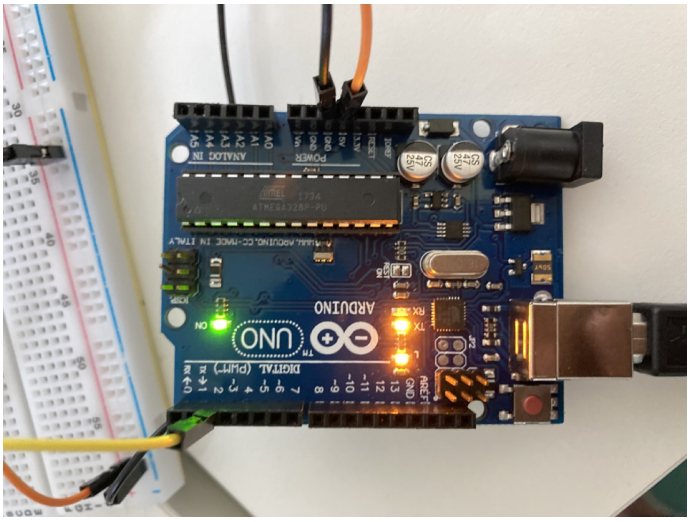

Figure 2: Arduino board connections

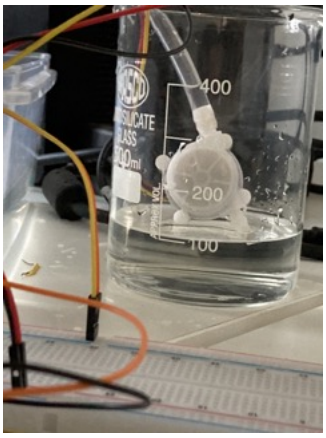

Figure 3: Water level measurement

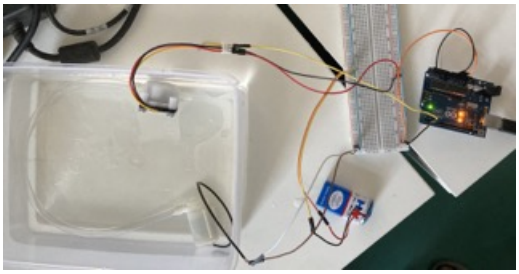

Figure 4: The measurement setup

Part 2

Measuring water level using HC-SR04 and Arduino

The HC-SR04 sensor

The HC-SR04 is an ultrasonic sensor which uses sonar to calculate the distance between the sensor and the nearest object in front of it. It is a setup of a piezoelectric sheet. Triggering the device will activate the piezoelectric sheet by creating an electric field. This generates an ultrasound.

It has a transmitter module and a receiver module. When power is supplied to it, the transmitter module emits a sound wave which is reflected at the nearest surface. This wave travels back to the receiver, and the time elapsed is recorded. This sensor can detect surfaces up to 4 meters away from it.

On working the sensor with Arduino, we observed a high pulse signal as the output from the device every time a surface was introduced close to it.

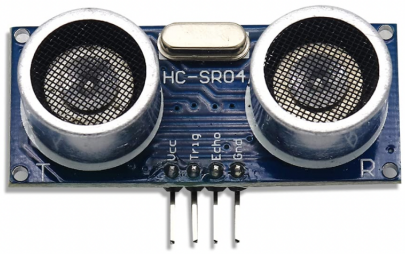

Figure 5: The HC-SR04

The sensor has four pins:

- Pin1 (V_{cc}): This pin provides a +5V power supply to the sensor.
- Pin2 (Trigger): This is an input pin, and it initializes measurement by transmitting ultrasonic waves, on keeping this pin high for 10µs.
- Pin3 (Echo): This is an output pin, which gives a high signal until the wave returns back to the sensor.

- Pin4 (GND): This is a ground pin.

Specifications:

- Operating Voltage: 5V DC
- Operating Current: 15mA
- Max Range: 4m
- Min Range: 2cm
- Ranging Accuracy: 3mm
- Operating Frequency: 40KHz

Calculating distance

When a pulse of at least 10 µS (10 microseconds) in duration is applied to the Trigger pin, the sensor transmits a sonic burst of eight pulses at 40 KHz. This 8-pulse pattern makes the "ultrasonic signature" from the device unique, allowing the receiver to differentiate the transmitted pattern from the ambient ultrasonic noise.

The Echo pin goes HIGH as the pulses travel away from the transmitter, and it goes LOW as soon as the reflected pulse is received.

If the pulses are not reflected back within 38 ms then the Echo will return LOW. So a 38 ms pulse indicates no obstruction within the range of the sensor (around 4 meters).

The distance can be calculated using the following formula.

$$S = \frac{(V * t)}{2}$$

Where the 'S' is the required distance, 'V' is the speed of sound, and 't' is the time elapsed since the echo pin HIGH.

This produces a pulse whose width varies between 150 µS to 25 ms, depending upon the time it took for the signal to be received.

The width of the received pulse is then used to calculate the distance to the reflected object.

The Arduino Program

```
1
2  int trigger =4; //giving output for transmitting the ultrasound waves
3  int echo=3; //taking input from the pin that how later it received the waves
       back
4  long duration; // duration from transmitting till receiving
5  float distance; // distance between sensor and surface
6  void setup() {
7    //start serial connection
8    Serial.begin(9600);
9    //configure pins for input and output
10   pinMode(trigger, OUTPUT);
11   pinMode(echo,INPUT);
12
13   // setting trigger to LOW, for not transmitting any waves
14   digitalWrite(trigger,LOW);
15
16 }
17
18 void loop() {
19   delay(1000);
20
21   //setting trigger to HIGH for transmitting waves for 10 microsecond by using
       10 microsecond delay
22   digitalWrite(trigger, HIGH);
23   delayMicroseconds(10);
24
25   //stopping transmission
26   digitalWrite(trigger,LOW);
27
28   //taking duration in microseconds for which the echo was HIGH
29   duration = pulseIn(echo,HIGH);
30
31   // distance= velocity of sound*time(or duration)/2
32   distance=1.0*333*duration*(1e-6)/2;
33
34   //printing Duration and Distance
35   Serial.print("Duration:");
36   Serial.println(duration);
```

```
37   Serial.print("Distance:");
38   Serial.println(distance);
39 }
```

The circuit setup and calibration

Temperature and humidity affect the speed of sound, and hence the distance measured, so we ran the program with different values for the speed of sound till we got an accurate reading.

A speed of 333 meters per second gave the correct distance measurements, so we finalised the experiment with that.

Part 3

The final closed-loop feedback system

Controlling the pump through relay

Once we had set up both the sub-circuits, one measuring the water flow rate and the other measuring the water level, we went on to make a closed loop feedback system where we could regulate the water in the beaker at any point of time.

For this, we used a relay that switched the water pump on and off depending on the signals it received.

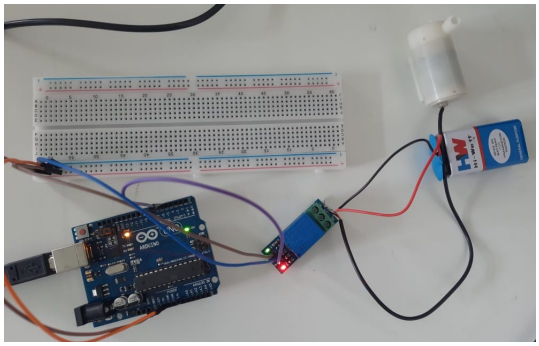

Figure 6: The relay-pump circuit

We set a lower limit (a minimum water level, 5cm below the ultrasound sensor) below which the pump would start functioning and filling water in the tub. How this worked was that the ultrasound sensor would give two consecutive readings of 5cm to the program, and that would give the relay a HIGH signal. The relay HIGH would switch the water pump on.

We also set an upper limit (a maximum water level, 4.25cm below the ultrasound sensor) above which the water pump would switch off and stop filling water.

If an external signal was given to switch off the pump when the water was still below 4.25cm (we introduced a card between the water and the sensor between 4.25cm and 2cm from it), there are two cases:

- Case 1: Water level is below 5cm: The moment you remove the external signal, the water pump would switch ON.
- Case 2: Water level is between 4.25cm and 5cm: The water pump would remain in the OFF state, because this water level is permissible according to our program.

The Arduino program for relay

```
1
2
3  int controlInput=5;// output pin for sending signal from arduino to turn ON or
       OFF the pump
4
5  // here we used relay with NO- normally open mode, so when the controlInput is
       LOW, pump is ON and when the controlInput is HIGH, pump is OFF
6  void setup() {
7    Serial.begin(9600);
8
9    //configuring OUTPUT pin
10   pinMode(controlInput,OUTPUT);
11
12   //switch OFF the pump initially
13   digitalWrite(controlInput,HIGH);
14 }
15
16 void loop() {
17   // wait for 1 second
18   delay(1000);
```

```
  //switch ON the pump for 3 second
  digitalWrite(controlInput,LOW);
  delay(3000);
  //switch OFF the pump
  digitalWrite(controlInput,HIGH);
}
```

The final program

```
int controlInput=5;
int trigger =4;
int echo=3;
long duration;
float distance;
int inputPin=2;
float flowRate;
// factor = pulses in 1 liter of water flow;
float factor=4330;
float volume;
float currentTime;
int counterStart=0,counterStop=0;
volatile unsigned long pulseCounter=0;
void setup() {
  Serial.begin(9600);
  pinMode(controlInput,OUTPUT);
  pinMode(inputPin,INPUT);
  digitalWrite(inputPin,HIGH);
  pinMode(trigger, OUTPUT);
  pinMode(echo,INPUT);
  digitalWrite(trigger,LOW);
  digitalWrite(controlInput,HIGH);
  attachInterrupt(digitalPinToInterrupt(inputPin),IRS_Counter,RISING);
  currentTime=millis();
  volume=0;
  counterStart=0;
  counterStop=0;
}

void loop() {
//  Serial.print("milis:");
//  Serial.println(millis());

```

```
34   if(millis()-currentTime>=1000){
35     detachInterrupt(digitalPinToInterrupt(inputPin));
36     currentTime=millis();
37     flowRate=(pulseCounter*60*1000)/factor;
38     volume+=(flowRate/60);
39       Serial.print("Pulse Count:");
40       Serial.println(pulseCounter);
41     Serial.print("Flow rate of water(in mL/min):");
42     Serial.println(flowRate,DEC);
43     Serial.print("Volume of water(mL):");
44     Serial.println(volume,DEC);
45       pulseCounter=0;
46    attachInterrupt(digitalPinToInterrupt(inputPin),IRS_Counter,RISING);
47     dist();
48   }
49
50 }
51
52 void IRS_Counter(){
53   pulseCounter++;
54 //  Serial.print("Pulse Count:");
55 //  Serial.println(pulseCounter);
56 }
57 void dist() {
58   //delayMicroseconds(2);
59   digitalWrite(trigger, HIGH);
60   delayMicroseconds(10);
61   digitalWrite(trigger,LOW);
62   duration = pulseIn(echo,HIGH);
63   distance=1.0*333*duration*100*(1e-6)/2;
64   //print out the value of the pushbutton
65   Serial.print("Duration:");
66   Serial.println(duration);
67   Serial.print("Distance(in cm):");
68   Serial.println(distance);
69   if(distance>5){
70     counterStart++;
71     if(counterStart>=3)
72       digitalWrite(controlInput,LOW);
73   }
74   else{
75     counterStart=0;
76   }
```

```
77    if(distance<=4.25)
78    {
79      counterStop++;
80      if(counterStop>=2)
81        digitalWrite(controlInput,HIGH);
82    }
83    else
84    {
85      counterStop=0;
86    }
87  }
```

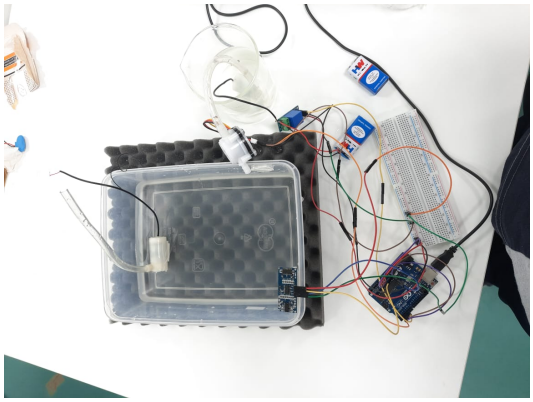

Figure 7: The final circuit

Maintaining a fixed water level

Some applications of this system require maintaining a fixed water level, so we made a few changes to the code to implement that. The only changes made here is removing the lower limit for the water level, and instead only keeping a counter for an upper limit (4.25cm here). Once the program gets three consecutive readings greater than 4.25, the pump will switch off, and any reading lower that than will switch the water pump on.

The counter was introduced to reduce faulty readings due to physical disturbances in the system.

```
1  void dist() {
2    //delayMicroseconds(2);
3    digitalWrite(trigger, HIGH);
4    delayMicroseconds(10);
5    digitalWrite(trigger,LOW);
```

```
  duration = pulseIn(echo,HIGH);
  distance=1.0*333*duration*100*(1e-6)/2;
  //print out the value of the pushbutton
  Serial.print("Duration:");
  Serial.println(duration);
  Serial.print("Distance(in cm):");
  Serial.println(distance);

  //maintain the level of water 4.25cm from sensor
  if(distance>4.25){
    counterStop=0;
    counterStart++;
    if(counterStart>=3)
      digitalWrite(controlInput,LOW);
  }
  else{
    counterStart=0;
    counterStop++;
    if(counterStop>=2)
      digitalWrite(controlInput,HIGH);

  }
}
```

Here's a link to a drive folder with more images and soft copies of our codes: BM2003 Project

References

YF-S401 datasheet
HC-SR04 datasheet

The end.

Project done by:

Samar Singhai BM20BTECH11012
Vidya Ajay BM20BTECH11017
Ramanathan Annamalai BM20BTECH11011