CSE 360: Computer Architecture

Sec-2

Project Report

Submitted to-

Dr. Ahmed Wasif Reza

Associate Professor

Department of Computer Science & Engineering

East West University

Submitted by:

(2018-1-60-063) Md. Habibur Rahman

(2018-1-60-042) Md Mania Ahmed Joy

(2018-1-60-076) Abdur Rahman Tumzied

Submission Date: 08/01/2021

Title:
Microprocessor based water level controller in domestic water storage tank (when water goes above a height, it switches off the pump motor, and when it goes below some level, it switches ON the pump)

Objective:
The main aim of our project was to create an autonomous water storage system. There is a water storage. The water level within the storage facility is monitored by a sensor and the system tries to maintain an optimal water level. If the water level falls below a critical point, the pump will start and if the water level goes above the overflow point the pump will turn off.

Theory:
In our system we have used the following components.

1) Water storage
2) Arduino
3) Bread board
4) Indicator LED
5) Buzzer
6) Ultra sonic sensor

1)Water Storage:
This is the water storage of our system. The entire project revolves around maintaining an optimal water level in it.

2) Arduino:
Arduino UNO is a programmable microcontroller board which is built with atmega328 microcontroller. We have used the digital pins of this board to take sensory data as input and after making correct decisions send it to the actuators. In this prototype model we have used LEDs and a Buzzer alarm as actuator. To implement this proto type in real life , we will have to debug the code and add a module.

3)Bread board:
A breadboard is a construction base for prototyping of electronics. We have used a bread board to connect our pseudo actuators i.e. the indicator LEDs and the buzzer to our system.

4) Indicator LED:

Here in our project we used 3 types of indicator LEDs. They are:

a) When the water is at critical level the RGB light is on.
b) When the water is at optimal level the blue LED is on.
c) When the water is overflowing the white LED is on.

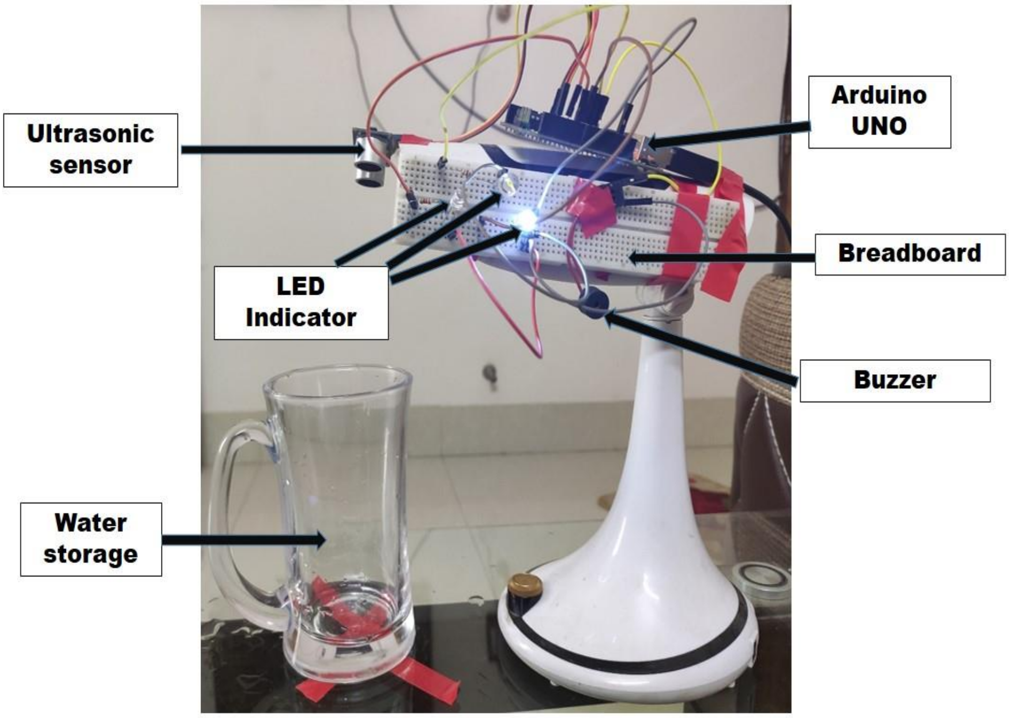


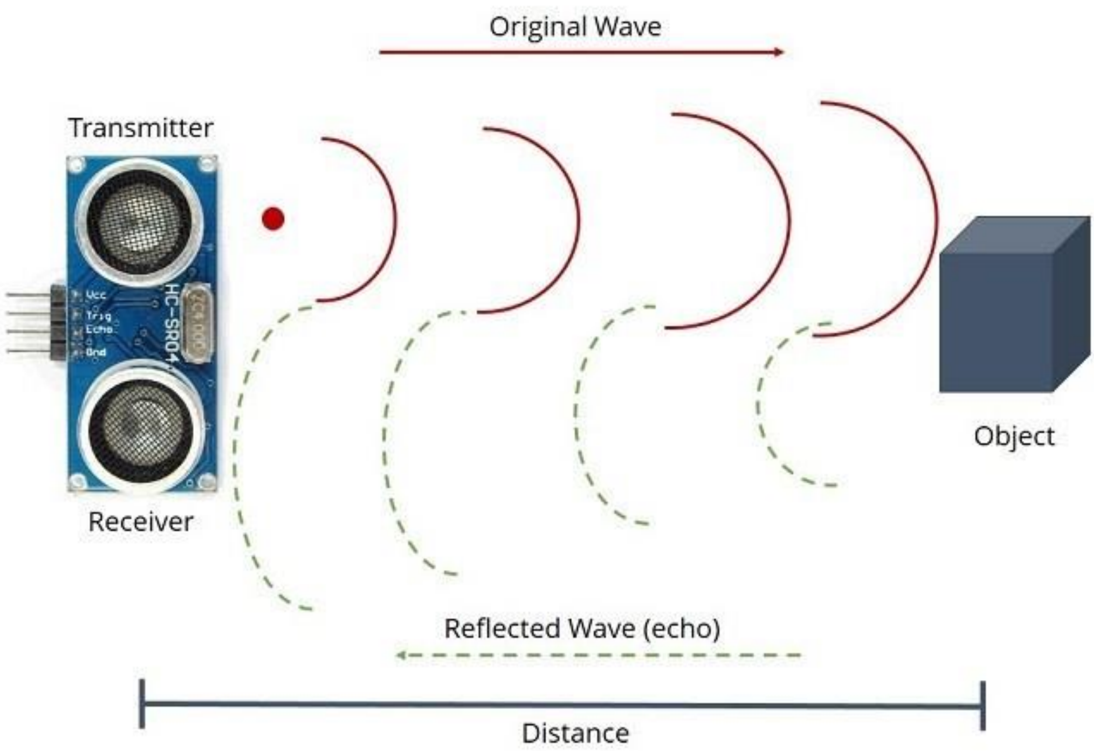
Fig: Project Diagram

5)Buzzer:

The Buzzer has been used to indicate alarming sound when water is overflowing.

6) Ultrasonic Sensor:

Ultrasonic sensor uses sound waves to measure distance. In our project we have used HC-SR04 Hardware. At its core, the HC-SR04 Ultrasonic distance sensor consists of two ultrasonic transducers. The one acts as a transmitter which converts electrical signal into 40 KHz ultrasonic sound pulses and measuring angle covered less than 15 degree. The receiver listens for the transmitted pulses. If it receives them, it produces an output pulse whose width can be used to determine the distance the pulse travelled.



For our project the most important mathematical formula is the depth calculation.
The depth is half the distance travelled by the ultra sound. It takes 1-unit time for the sound to go from the transducer to the reflector and another 1 unit to return to the receiver. The depth here is therefore 1 unit which is half of the distance travelled by our sound pulse.
So,

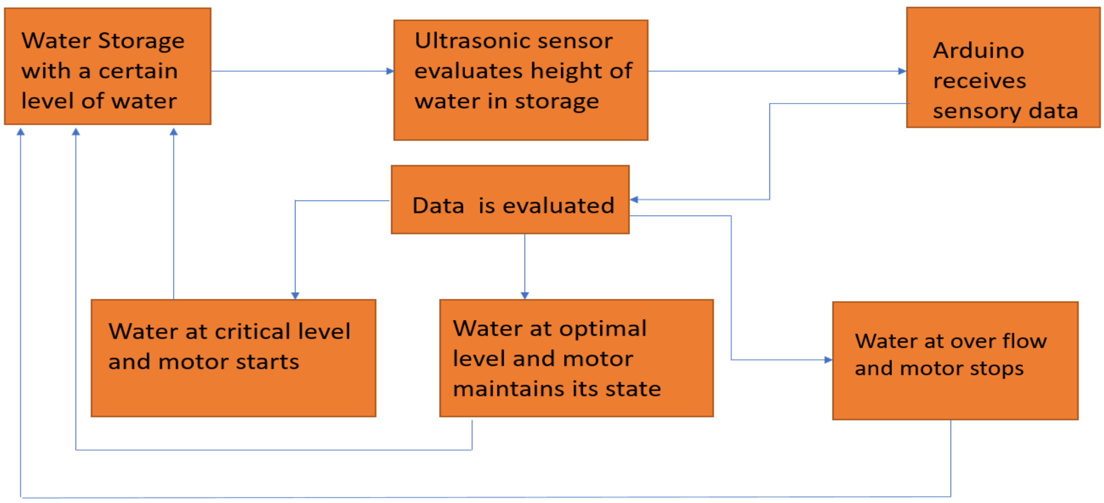
Distance= Speed * Time
Depth= Distance/2.

Here it is important to know that ultra sound travels 1 inches in 74 micro second and 1 centimeter in 29 microseconds.

Project Design:

Flowchart:

For a simple understanding of our project let us demonstrate a simple flow chart.



The different states of the motor are displayed through our pseudo actuators i.e LEDs and buzzer.

System Over-view:

Step-1) The depth of water in storage is sensed by the ultrasonic sensor.

Step-2) The Sensor sends this data to Arduino and it is evaluated.

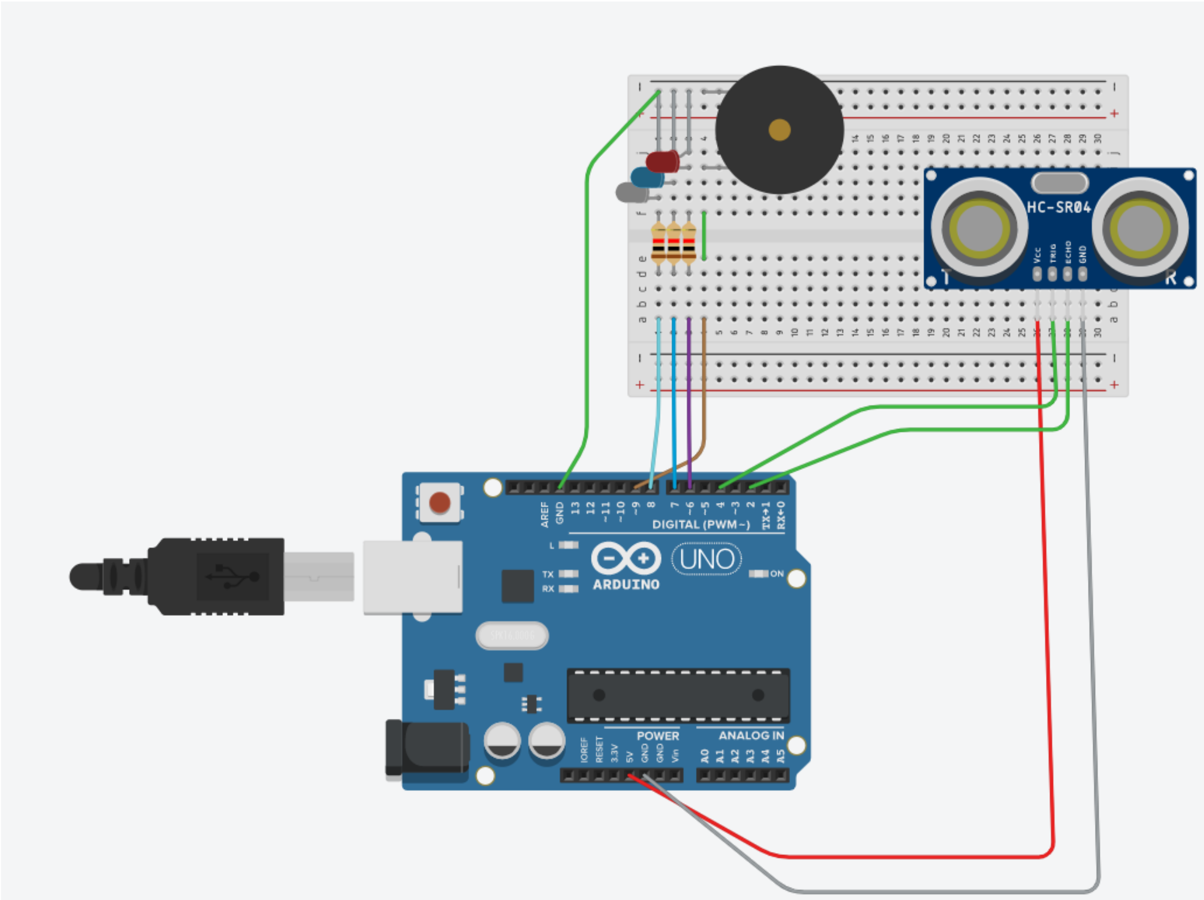
Step-3) The system takes action depending on sensory data:

a) If the water is below critical point, flickering LEDs (RGB) are activated indicating that the motor will start.

b) If the water is at optimal height, blue light is activated showing the motor will continue to maintain its state.

c) If the water is at over flow, white light and buzzer is activated indicating the motor will stop.

Schematics:



Video demonstration:
https://drive.google.com/file/d/1E-z1-VD0_UM62DbGs3azNKqY_PD0nkOa/view

Implementation

Because Arduino is programmable microcontroller, so we can control its behavior by code.

Code Description:

At first to grab the ultrasonic sensor data by Arduino we define two constant pin, which ultrasonic sensor is connected to, this will help us to determine when we want to transmit and receive ultrasonic wave

```
const int trigerPIN = 4; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 2; // Echo Pin of Ultrasonic Sensor
```

Define the over flow point of the water level and lowers point of water level

```
const long overFlow_POINT=10;//cm
const long lowest_water_level_POINT=23;//cm
```

And then define few extra pin for signal and debugging

```
const int blue_light=7;
const int white_light=8;
const int flickering_light=6;
const int buzzer=9;
```

- `blue_light` pin (*7*) connected to and blue light, for stable water signal
- `white_light` pin (*8*) and `buzzer` pin (*9*) connected to and white light and buzzer to determine overflow signal
- `flickering_light` pin (*6*) connected to a light that changes its color in certain time period, this is use for when the water level is lower than certain amount

Here, we store the last 8 value water level in circular array later this array will help us to reduce the error in reading the data for the sensor

```
const int MAX_SIZE = 8;
long arr[MAX_SIZE] = { 0 };//to store last few sensor
```

Because this array is circular we need special push function to insert an item to the array.

```
void push (long x)
{

  if (back == -1)
    {
      font++;
      back++;
      arr[back] = x;

    }
  else
    {
      if (((back + 1) % MAX_SIZE) == font)
  {
    font = (font + 1) % MAX_SIZE;
    back = (back + 1) % MAX_SIZE;
    arr[back] = x;
  }else{

    back = (back + 1) % MAX_SIZE;
    arr[back] = x;
  }

    }
  //printf ("push -> : %d\n ", arr[back]);
}
```

Now creating a `setup()` function, which initializes and sets the initial values, pin modes, start using libraries etc. this function will run once after each power up or reset of the Arduino board.

```
void setup(){
    Serial.begin(9600); // Starting Serial Terminal
    // opens serial port, sets data rate to 9600 bps

    pinMode(blue_light, OUTPUT);
    pinMode(white_light, OUTPUT);
    pinMode(flickering_light, OUTPUT);
    pinMode(buzzer, OUTPUT);

}
```

`Serial.begin(9600);` passes the value 9600 to the ***speed*** parameter. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second. That's 9600 binary ones or zeros per second, and is commonly called a *baud rate*.

How creating and `loop()` function, **`loop`** function and starts doing what the statements in its curly braces tell it to do. Any statements in **`loop`** will be repeated over and over again,

```
void loop() {
   long duration, current_Water_level_in_INCHES, current_Water_level_in_CM;

   pinMode(trigerPIN, OUTPUT);
   digitalWrite(trigerPIN, LOW);
   delayMicroseconds(2);
   digitalWrite(trigerPIN, HIGH);//turn on the trinsmitter
   delayMicroseconds(10);
   digitalWrite(trigerPIN, LOW);//turn off the trinsmitter
   pinMode(echoPin, INPUT);//turn on the receiver
   duration = pulseIn(echoPin, HIGH);

   current_Water_level_in_INCHES = microsecondsToInches(duration);
   current Water level in CM = microsecondsToCentimeters(duration);
```

Here set pin mood of `triger_pin` as `OUTPUT`, and start transmitting ultrasonic wave for 10ms then stop transmitting the wave, after that receiver receive the ultrasonic wave and get the duration of the receiving signal using **`pulseIn()`**function.

Now its time to evaluate error in the current water level distance, for that we call `check_for_error` user define function and pass `current_water_level` variable as argument

```
current_Water_level_in_CM=Check_for_error(current_Water_level_in_CM);
```

Here, we define some condition for control the water level, and turn on / off some light base of the condition, note that is condition in the **loop** function

```
if(current_Water_level_in_CM==0){
 //error
 digitalWrite(flickering_light, HIGH);
 digitalWrite(blue_light, HIGH);
 digitalWrite(white_light, HIGH);
}else if(current_Water_level_in_CM<=overFlow_POINT){
 //over flow point
 tone(buzzer, 10000, 1000);
 digitalWrite(blue_light, LOW);
 digitalWrite(flickering_light, LOW);
digitalWrite(white_light, HIGH);

}else if(current_Water_level_in_CM>=lowest_water_level_POINT){
 // lowest water level point
  digitalWrite(flickering_light, HIGH);
  digitalWrite(blue_light, LOW);
  digitalWrite(white_light, LOW);

}else{
 //stable point
  digitalWrite(white_light, LOW);
  digitalWrite(flickering_light, LOW);
 digitalWrite(blue_light, HIGH);
}
```

For turning on and off the buzzer at certain time, the tone() function is used, its works with two arguments, but can take up to three arguments, fist argument is pin number , second is frequency and third one is duration of the tone

`tone(buzzer, 10000, 1000);//`pin , frequency , duration in ms

Here we print the current water level in serial monitor. Every loop iteration there is a delay for 100 millisecond

```
Serial.print(current_Water_level_in_INCHES);
Serial.print("in, ");
Serial.print(current_Water_level_in_CM);
Serial.print("cm");
Serial.println();
delay(100);
```

This is end of **loop** function

user define function, **microsecondsToInches** this function takes microsecond and return and distance in inches

and **microsecondsToCentimeters** this function takes microsecond and return and distance in centimeters

```
long microsecondsToInches(long microseconds) {
   return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
   return microseconds / 29 / 2;
}
```

check_for_error function,

first check rate of changes in water level and if the rate of changes is in certain amount then we add difference of last and current water level to the last encounter value and return it, or otherwise its check for how many time its occur in last 8 iteration and take the percentage of current water level this is done by **get_probability** function, if the percentage is higher then 40% then we return the current water level value

```
long Check_for_error(long curr_water_lv_in_cm){
  static long last_counter = -1;

      push (curr_water_lv_in_cm);
      if (last_counter == -1)
  {
    last_counter =curr_water_lv_in_cm;

  }
      long rate_of_Change = (long) (abs (last_counter -curr_water_lv_in_cm));
      //printf("rate_of_Change %d \n",rate_of_Change);

      if (rate_of_Change >= 1 && rate_of_Change <= 3)
  {

    last_counter =
      last_counter +
      (long) ((sinh (curr_water_lv_in_cm- last_counter)) * rate_of_Change);

  }
      else
  {

    float prob = get_probability (curr_water_lv_in_cm);

    if (prob > 30)
      {
        last_counter = curr_water_lv_in_cm;

      }
  }
  return last_counter;

}
```

```
,
float get_probability (long item)
{

  //printf("item -> r : %d\n , ",item);
  int tfont = font;
  int tback = back;
  int counter = 0;
  int n = 1;

    for(int i=font;i!=back;i=(i+1)%MAX_SIZE){


      if (arr[i] == item)
  {
    counter += 1;

  }

      n += 1;


    }


     // printf("counter : %d ,n: %d \n",counter,n);

  return ((float)counter /(float) n) * 100.0;

}
```

Full code:

https://github.com/tz01x/Algorithm/blob/master/water_level_sensor.ino

Debug and Test run:

Fist we adjusting the value of the overFlow_POINT and lowest_water_level_POINT , in which this case those value are,

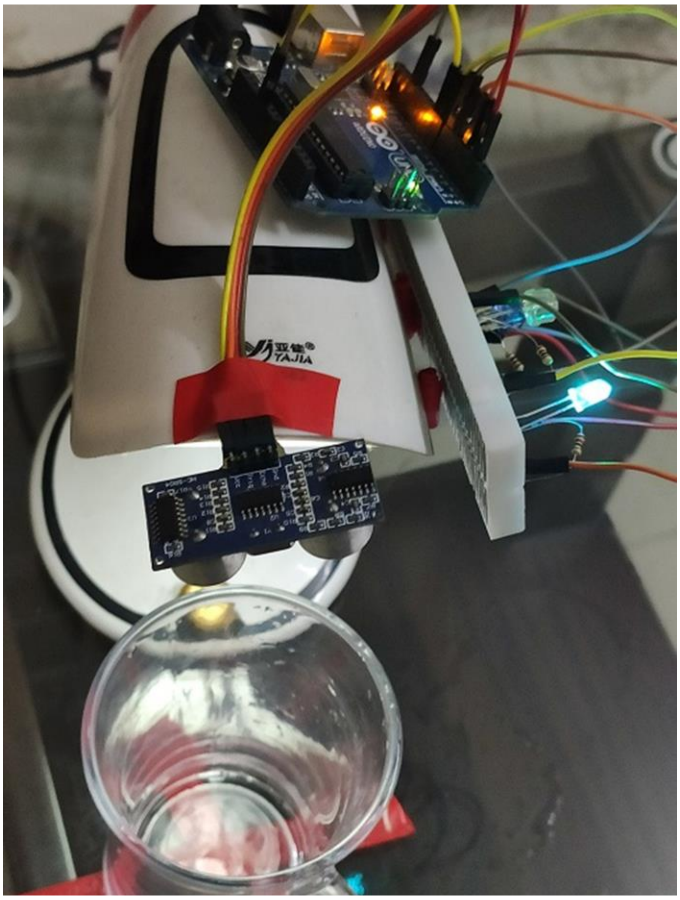
```
overFlow_POINT=10;//cm
```

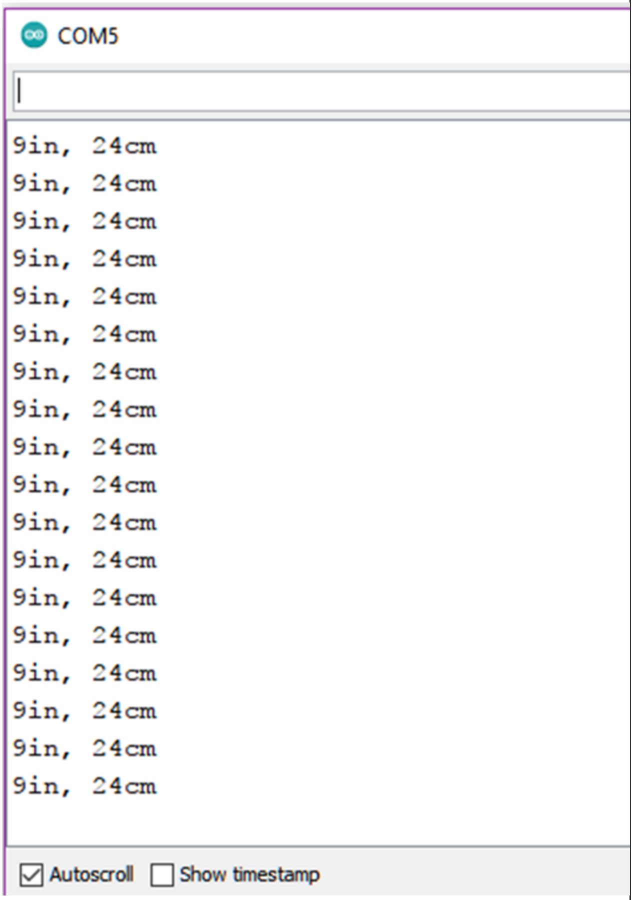
```
lowest_water_level_POINT=23;//cm
```

after setting up and upload the to Arduino,

Stage1: (Lowest Water level point)

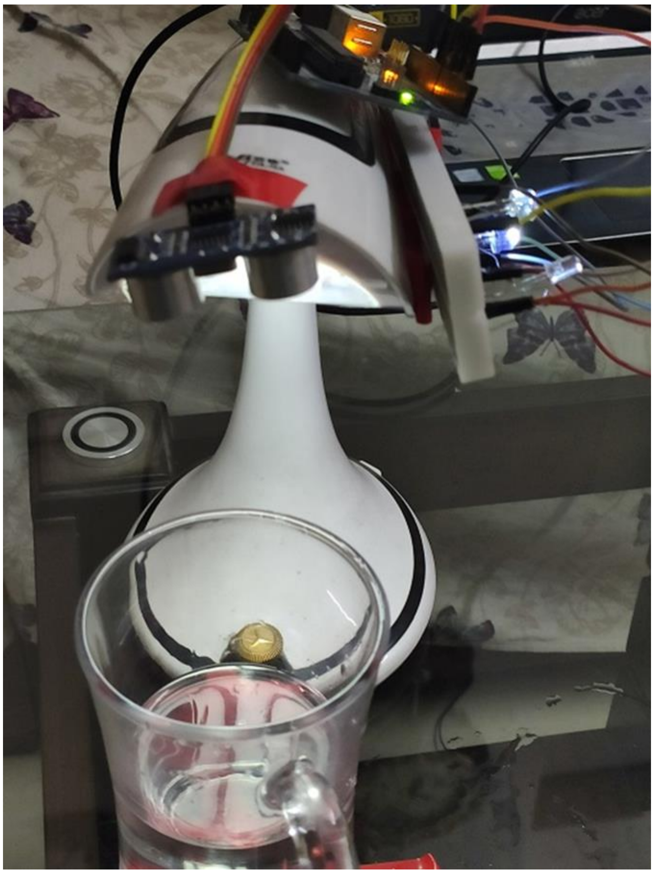
Here ultrasonic sensor measuring distance how much farther away the water is for the sensor point, in this stage, currently the jar is empty. we can also see the measured distance is 24cm which is lower than the declared water level in the code, so the RGB light is turn on.

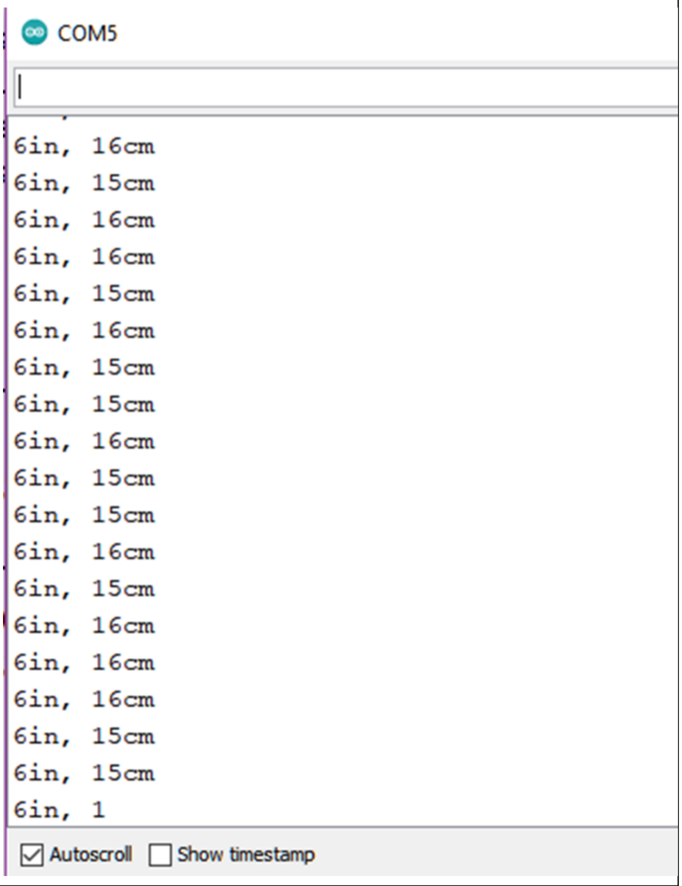




Stage 2: (Stable point)

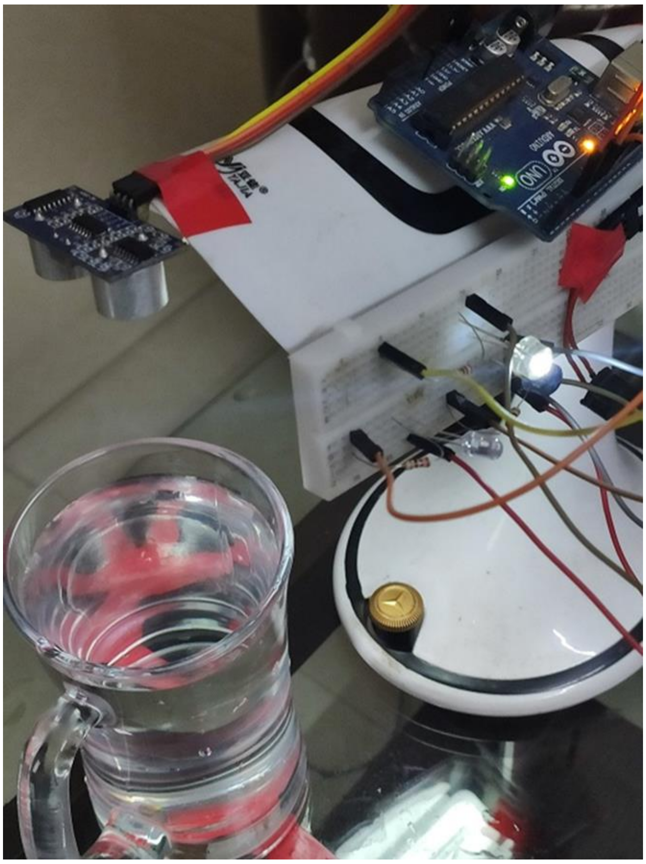
In stage 2, we see that our sensor mastering the how much farther way the water level is and output showing in the serial monitor is 16 cm or 15 cm, this indicate by Blue light is turn on and RGB light is off in the circuit

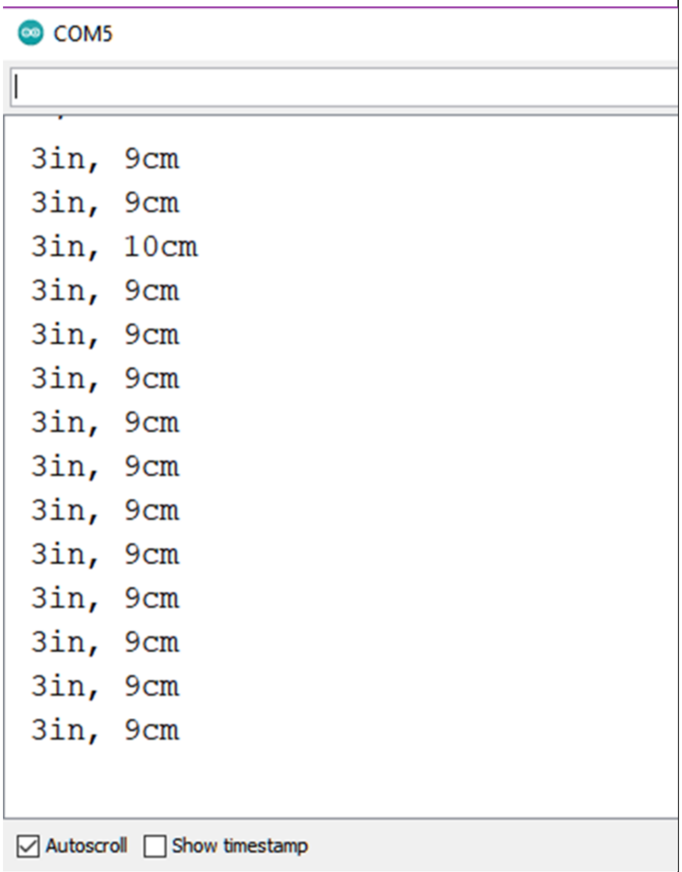




<u>Stage3: (Overflow point)</u>

In this final stage we trigger a buzzer to indicate that overflow point of the jar, we define the over flow point in the code with was 10 cm and the serial monitor showing the lass the 10 cm which is 9cm so trigger happened and buzzer turn on and with it the white light turn on.

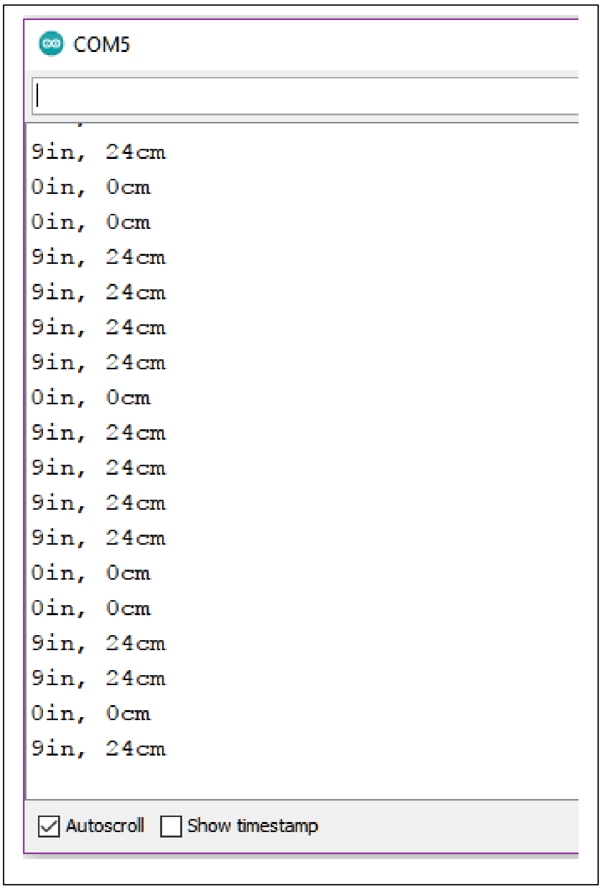


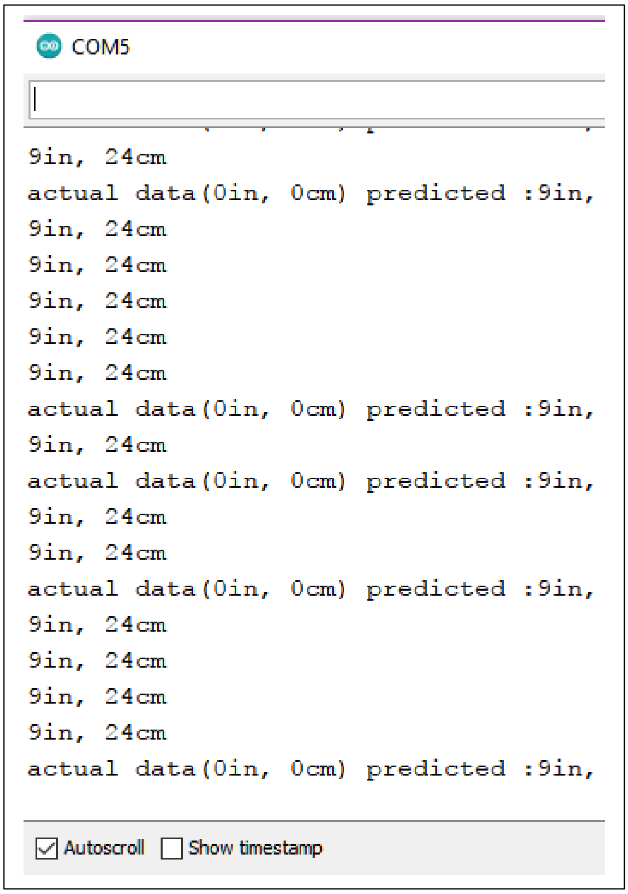


Detecting error:

Sometime hardware fail might happen and the measured distance will be jump to one to another random value in (fig:E2), to encounter this we apply and algorithm that might solve this problem,

Below the left serial monitor show that sensor is providing the wrong data because of the hardware fail. The right serial monitor shows that after the algorithm applied what was the actual data and predicted data by the algorithm.





Limitations:

Our only limitation is that we could not afford to procure a pump motor for our project. We displayed the motor activties using indicator LEDs and burger.

Possible improvements and future work:

When we will implement our prototype in real life scenario, we will use a real water tank then we will have to adjust our code according to the demand of the time. The pump motor that consumes higher voltage can be integrated with our system by using relay modules that actually work as the switch of the system.

Possible implementation of our project:

This project can be used in smart homes, offices, industries etc. Verily, it can be used to turn any water storage facility into an autonomous water storage facility.

Conclusion:

The project seemed to be a fun learning experience at first but as we started to work on it things started to get more and more difficult. It was extremely difficult for us to procure the resources that we used in our project. The assembly of our system was difficult and we had success after much trial and error. We learned a lot while developing this project.

Reference:

https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

https://www.tinkercad.com/

https://components101.com/ultrasonic-sensor-working-pinout-datasheet