TITLE:

REMOTE MONITORING AND CONTROL OF WATER LEVEL IN A TANK

File created by:

Sanjay Bisu (20MI10047)

Experiment performed by:

Rakesh Bisu (19CH10034)

Sanjay Bisu (20MI10047)

## 

# PROJECT BRIEF AND PURPOSE:

PROBLEM STATEMENT, MOTIVATION AND OBJECTIVES:

We can see in our surroundings that many times when we or our neighbor tries to fill the water tank from regular motor, the only way to check if tank is full or not is by waiting till the water starts falling from the tank. And due to this a lot of water is wasted. And even sometimes we turn on the motor and get engaged in other work and totally forget to notice when will water fall from the tank, and that results in loss of a huge amount of precious water and electricity. The problem statement is to devise methods to get rid of such loss of water and electricity.

Our aims is to provide a method by which you can not only prevent loss of water and electricity but it will also save your precious time, because you just have to turn a switch on and forget about it completely! The product will automatically control the water level in the tank in such a way that it never overflows and also the water level never goes below a minimum level so that we always have sufficient water in store.

We also intend to enable the user to get real-time information like water level in centimeters and whether the pump is currently in ON or OFF condition through his/her smartphone.

COMPONENTS

All the components required in the project:

|  |  |
| --- | --- |
| * A submersible water pump | * Male to female and male to male jumper wires |
| * A pipe | * Arduino UNO R3 |
| * Long wires | * A relay module |
| * 2 buckets | * USB cable (to connect Arduino UNO to laptop) |
| * An ultrasonic sensor |  |

# SUBMERSIBLE WATER PUMP

We took an AC 180-240 V, 50 HZ, 20 W, rust proof submersible single phase cooler water pump of output pipe size 1/2 inch and 3/4-inch, water output of 1100 liters per hour (approx.) and it can push water up to a height of 1.85 m.



# ULTRASONIC SENSOR

We took a 5 V DC, static current< 2mA Digital ultrasonic sensor of range 2cm to 4m of high precision up to 0.3cm which can be used for measuring distance, object sensors, motion sensors, etc. It sends eight 40Khz square wave pulses and automatically detects whether it receives the returning signal. If there is a signal returning, a high- level pulse is sent on the echo pin. The length of this pulse is the time it took the signal from the first triggering to the return echo.

# ARDUINO UNO R3

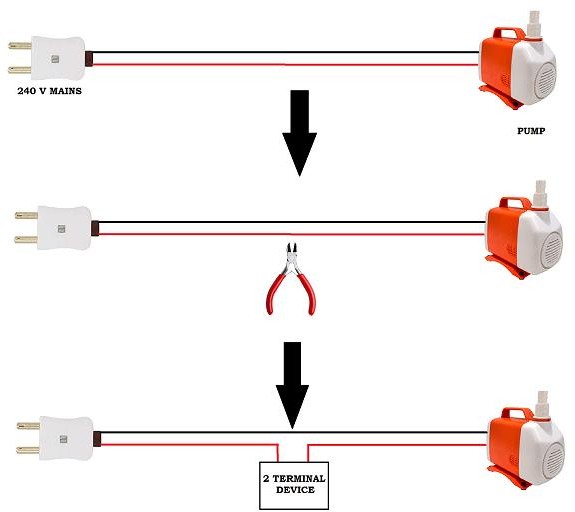
We took an Arduino UNO R3 board of operating voltage of 5V, input voltage 7-12V, DC Current 40mA, flash memory of 32 KB, 6 analog input pins, 14 digital I/O pins (of which 6 provide PWM output), SRAM of 2 KB, EEPROM of 1 KB, clock speed of 16 MHz and ATmega328P microcontroller which can get power by connecting it to a computer with a USB cable or powering it with a DC adapter or battery.

# RELAY MODULE

We took a 5V 1-Channel Relay Module of dimensions 75 mm x 55 mm x 19.3 mm, control signal type of TTL level, input voltage of 5V, maximum output of AC250V 10A and Dc30V 10A, of 1 relay (which mean it can only control 1 device) and is compatible with Arduino, AVR, PIC, ARM, Raspberry Pi etc. It controls larger loads and devices like DC motors, AC motors, and other AC and DC devices with the digital outputs from controllers and processors.

# PRODUCT DESCRIPTION, WORKING PRINCIPLE AND NATURE OF INPUT AND OUTPUT:

Ours is a two terminal device (the COM and NC terminals of relay module are the two terminals) to be lodged over water tank and connected to the appliance (here the pump) in the way as shown:



The device mainly consists of an Arduino Uno board, HCSR04 ultrasonic sensor (for distance measurement) and a module. The relay basically acts as a digital switch which controls the flow of current to the pump from the source depending on digital input signal fed to its In pin from the Arduino board. The Arduino is

programmed in such a way that it can measure the water level in the tank with an analogue input from the sensor and when water level is below a certain minimum height it turns its output pin high (output pin of Arduino is input pin of relay) and the relay allows the flow of current through the external circuit so that

pump turns ON. The exact opposite thing happens when water level goes beyond a maximum water level so that the pump is turned OFF automatically.

To sum up,

OUTPUT OF SENSOR = INPUT OF ARDUINO: ANALOGUE OUTPUT OF ARDUINO = INPUT OF RELAY: DIGITAL

THE OUTPUT OF RELAY CAN BE CONSIDERED DIGITAL CONSIDERING IT CAN EITHER ALLOW OR NOT ALLOW CURRENT (HIGH AND LOW STATES) IN THE EXTERNAL CIRCUIT.

## CIRCUIT LAYOUT OF THE PROJECT:

HC-SR04 ULTRASONIC SENSOR

USB

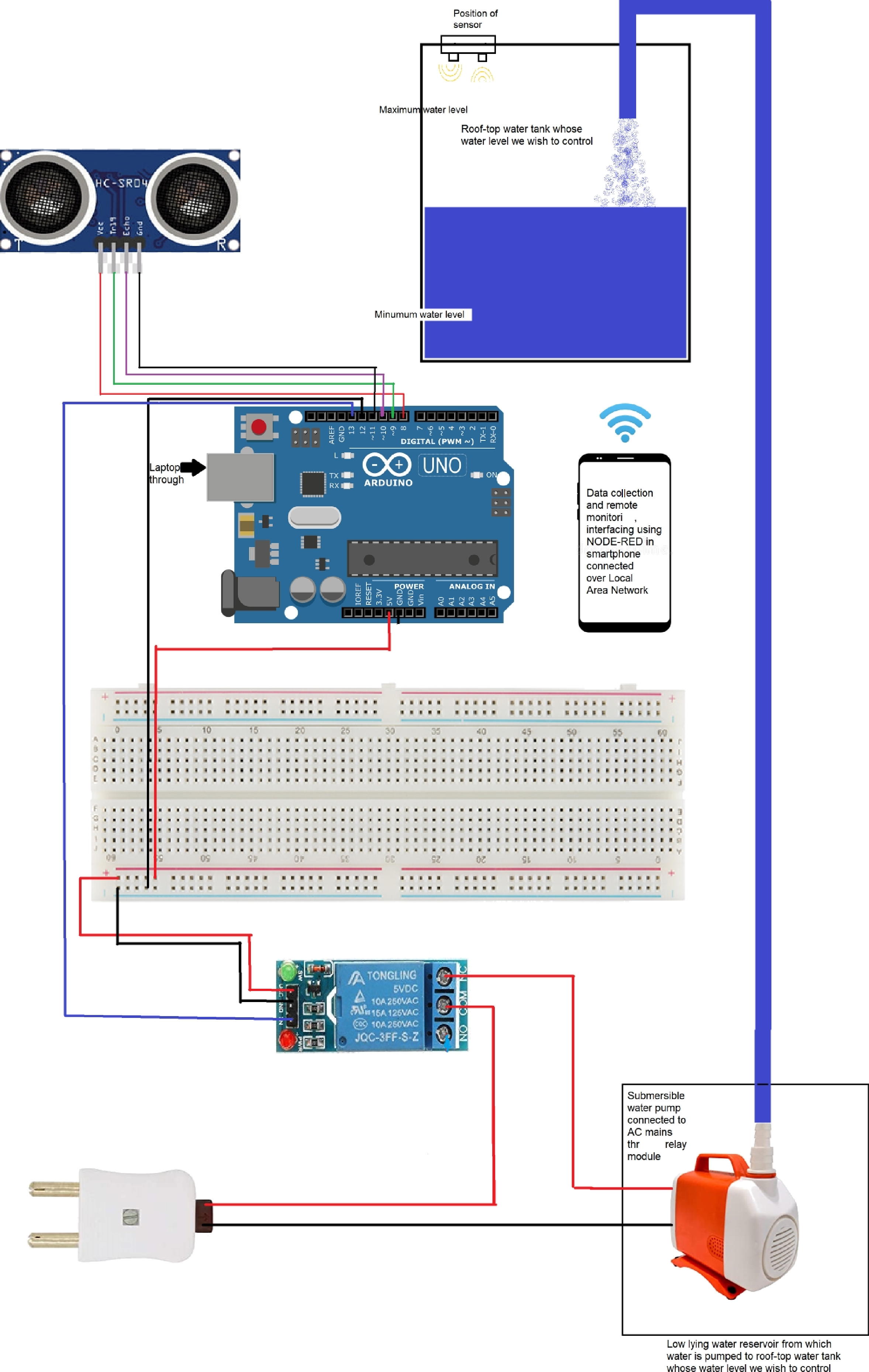
ng

ARDUNO UNO BOARD

BREADBOARD

5V1CHANNELRELAY MODULECONNECTED IN NORMALLYCLOSED MODE

ough



240V AC MAINS

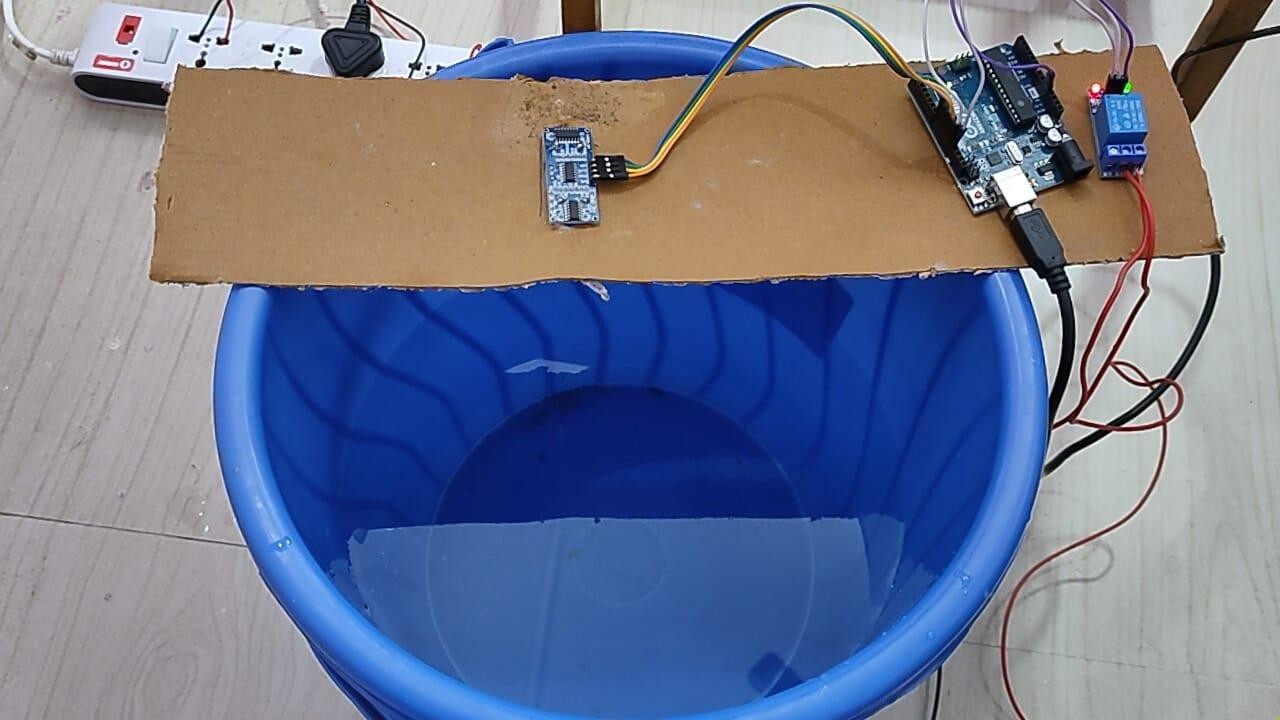
Basically, the connections can be summarized as:

|  |  |
| --- | --- |
| Arduino pin 8 | Sensor vcc |
| Arduino pin 9 | Sensor Trig |
| Arduino pin10 | Sensor Echo |
| Arduino pin 11 | Sensor gnd |
| Arduino pin 12 | Relay gnd |
| Arduino pin 13 | Relay In |
| Arduino 5V | Relay vcc |
| Relay COM | AC Source live |
| Relay NC | Pump live |
| AC Source neutral | Pump neutral |

Photographs:



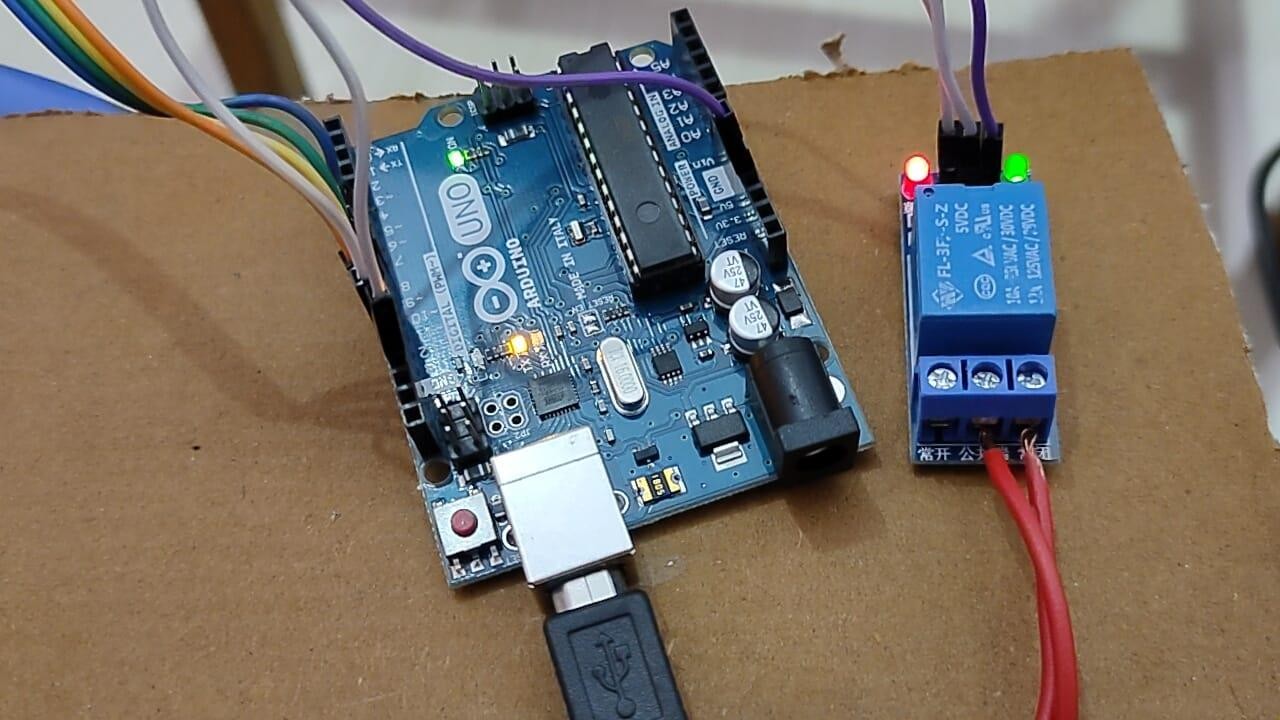
We know that having the tank and the reservoir (and hence the pump) at the same level does not make sense! But this was for demonstration purpose only.



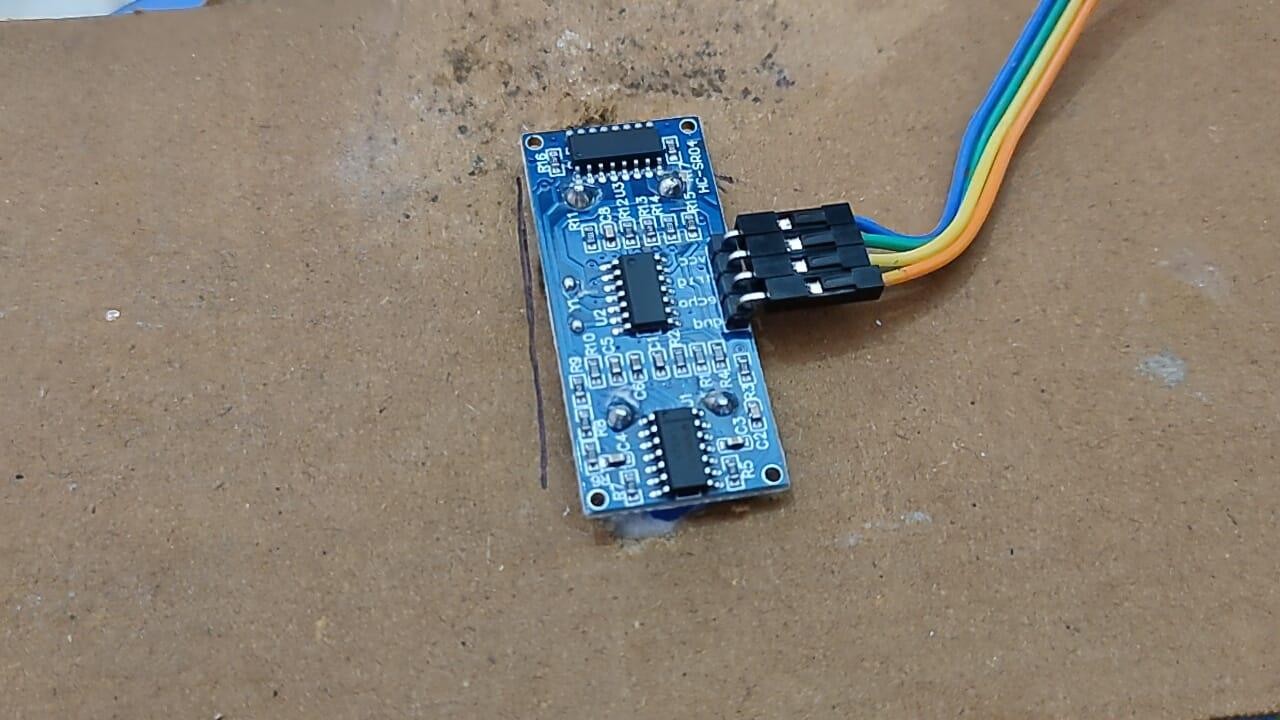
The role of the tank was played by a bucket. On its top sensor, arduino and relay module are lodged with the help of a cardboard plank.



The water-reservoir (supposed to be at a lower height) along with the submersible water pump



The Arduino board and relay module.



The circular windows (emitter and receiver) pass through holes cut on cardboard.



240V AC source

# Water level calculation using reading of sensor:

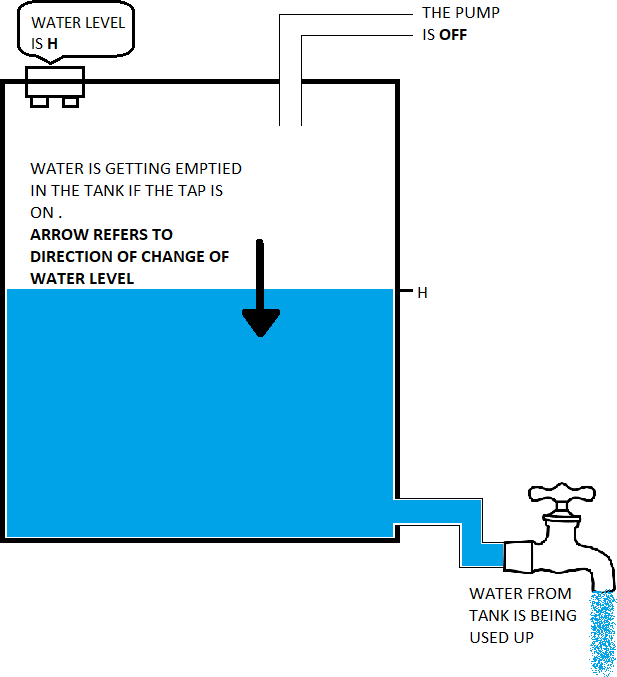
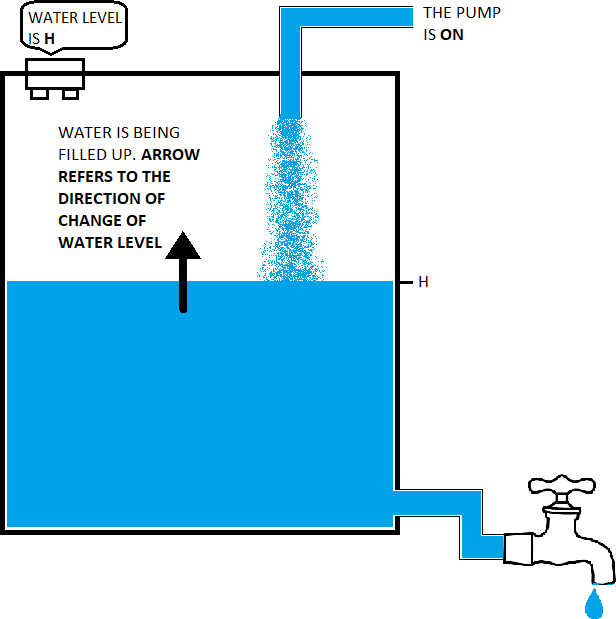
### h = (0.034\*duration)/2 cm x = (H-h) cm

(H is a constant known to us)

So, x = (H - (0.034\*duration)/2) cm

THE TWO-WAY PROBLEM / SEPARATE FILLING AND EMPTYING CYCLES:

Consider the following two situations:



In the left-hand side picture, the pump is ON and tank is being filled up, in the right-hand side picture water is being emptied and the pump is OFF. But in both the cases water level is same (say H). Based on the reading taken by sensor both the situations would look exactly same to the Arduino. Then how can the Arduino decide at this point that whether it should feed a high input signal or a low input signal to the relay?

This observation leads to the conclusion that we have to program the Arduino in such a way that it cannot only measure the water level but it should also be able to identify whether the filling cycle or the emptying cycle is running.

We have proposed two different solutions to overcome this challenge in the later sections of this project report (see “THE ARDUINO CODE” section).

# THE ARDUINO CODE:

*Solution 1*

Following is the final Arduino source code copy-pasted from Arduino IDE *(Note that the bucket that we used as tank was of 31 cm height, max and min levels were set by us as 25 and 8cm)*

#### //THIS IS THE FINAL ARDUINO SOURCE CODE OF DIY PROJECT SUBMITTED BY GROUP 13, SEC. 3

**//"REMOTE MONITORING AND CONTROL OF WATER LEVEL IN A TANK"**

**#define vccPin 8 //The pins of sensor are connected to consecutive pins on arduino for efficient space management & integration**

**#define trigPin 9**

**#define echoPin 10**

**#define gndPin 11**

**#define relay\_gnd 12**

**const int H=31; //here the height of the water tank is entered const int h\_max=25; //here the maximum water level beyond which pump**

**should be switched off is entered**

**const int h\_min=8; //here the minimum water level below which pump**

**should be switched on is entered**

**long duration; // variable for the duration of sound wave travel int distance; // distance of water level from sensor**

**int water\_level; // height of water level from bottom of tank**

**void setup() {**

**// put your setup code here, to run once: pinMode(trigPin, OUTPUT);**

**pinMode(vccPin, OUTPUT); pinMode(gndPin, OUTPUT); pinMode(echoPin, INPUT); pinMode(12,OUTPUT); pinMode(13,OUTPUT);**

**Serial.begin(9600); // Serial Communication is starting with 9600**

**of baudrate speed**

**Serial.println("THIS IS THE MESSAGE TO BE DISPLAYED WHEN SERIAL MONITOR OPENS");**

**//HOWEVER IN THIS PARTICULAR PROJECT WE INTEND TO USE NODE-RED AS USER INTERFACE NOT THE ARDUINO IDE SERIAL MONITOR**

**}**

**void loop() {**

**digitalWrite(relay\_gnd,LOW); digitalWrite(vccPin, HIGH); digitalWrite(gndPin, LOW); digitalWrite(trigPin, LOW);**

**delayMicroseconds(2); digitalWrite(trigPin, HIGH);**

**delayMicroseconds(10); // Sets trigPin HIGH (ACTIVE) for 10**

**microseconds**

**digitalWrite(trigPin, LOW);**

**duration = pulseIn(echoPin, HIGH); // Reads the echoPin. Returns the**

**sound wave travel time in microseconds**

**// Calculating the distance**

**distance = ((duration \* 0.034 / 2)+2); // Speed of sound wave divided by 2 (go and back).**

**//Here speed of sound is taken as 340 m/s.**

**//Please note that our HCSR04 sensor was having some error in measurement, we had to add 2 to correct it**

**// Calculating water level water\_level=(H-distance);**

**//Instructions as to when the pump should remain ON/OFF if(water\_level>=h\_max){**

**digitalWrite(13,LOW);**

**}**

**else if(water\_level<h\_min){ digitalWrite(13,HIGH);**

**}**

**//Data to be transferred to the computer via serial port. It will be displayed in Node-Red dashboard**

**Serial.print(water\_level); Serial.print(", ");**

**Serial.println(digitalRead(13)); //digital state of pin 13 indicat es whether the pump is on or off**

**delay(2000);**

**}**

Extensive use of comments throughout the code makes it to a large extent self-explanatory. Let’s try to

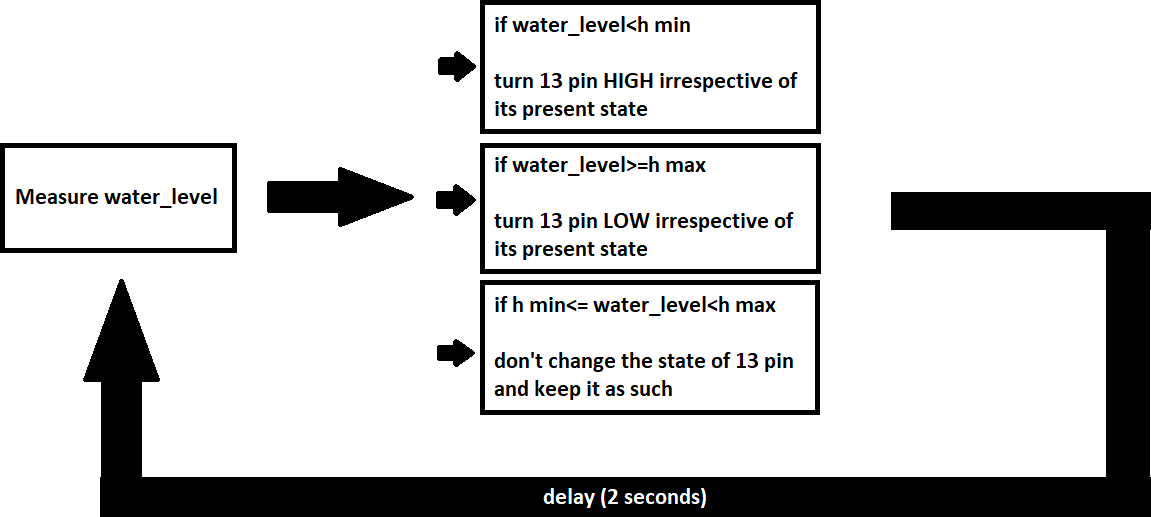
understand how the logical flow happens in the void loop.

As the loop starts the 12 pin (relay\_gnd) is turned low and throughout the course of the loop it remains low. Same is the case for 8 (vcc) and 11 (gnd) pins which are turned high and low respectively and kept so throughout the time. The pin 9 (trigPin) is initially turned low for 2 microseconds following which it is turned high for 10 microseconds and it is during this time that it emits ultrasonic sound pulses. Then the trigPin is again turned low and based on the time of flight of the ultrasonic sound waves (emission from sensor-->reflection-->return to sensor) as detected by the sensor a voltage is generated in echoPin which is connected to the pin 10 of Arduino. Arduino reads this analogue signal and converts it into distance in centimetres according to the conversion formula specified in the code.

So, the current value of distance and hence the water level has been measured. Reception of analogue signal from sensor and converting it into relevant data is thus complete.

Now comes the decision-making part.

The instructions are set such that the following steps are executed:



The decision of keeping the 13 pin as such when water\_level lies within the maximum and minimum levels is specifically helpful in dealing with the two way-problem. The usefulness of this procedure can be acknowledged after presenting an alternate source code.

*Solution 2:*

Alternate source code:

(please note that the code that will be presented in the subsequent sections was the one that we developed during the earlier weeks of the project. It was meant to give same results as the above code and after carrying out certain tests we also were convinced that there is nothing wrong in the logic but unfortunately it failed sometimes when the water level dropped below minimum as the sensor started giving erroneous results and the pump would not turn ON!)

**#define vccPin 8**

**#define trigPin 9**

**#define echoPin 10**

**#define gndPin 11**

**const int H=31; const int h\_max=25; const int h\_min=8;**

**long duration; int distance; int count;**

**void setup() {**

**pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT pinMode(vccPin, OUTPUT);**

**pinMode(gndPin, OUTPUT); pinMode(echoPin, INPUT); pinMode(13,OUTPUT); pinMode(12,OUTPUT);**

**Serial.begin(9600); //Serial Communication is starting with 9600 of baudrate speed**

#### Serial.println("THIS MESSAGE IS DISPLAYED WHEN THE SERIAL MONITOR STARTS");

**}**

**void loop() {**

**digitalWrite(12,LOW); digitalWrite(13,LOW); digitalWrite(vccPin, HIGH); digitalWrite(gndPin, LOW); digitalWrite(trigPin, LOW);**

**delayMicroseconds(2); digitalWrite(trigPin, HIGH); delayMicroseconds(10); digitalWrite(trigPin, LOW);**

**duration = pulseIn(echoPin, HIGH); // Reads the echoPin. Returns the sound wave travel time in microseconds**

**// Calculating the distance distance = duration \* 0.034 / 2;**

**if((H-distance)>=h\_min){ count=0;**

**}**

**else if((H-distance)<h\_min){ count=1;**

**}**

**while(count==1){**

**delayMicroseconds(2); digitalWrite(trigPin, HIGH);**

**delayMicroseconds(10); // Sets trigPin HIGH (ACTIVE) for 10 microseconds**

**digitalWrite(trigPin, LOW); duration = pulseIn(echoPin, HIGH);**

**// Calculating the distance distance = duration \* 0.034 / 2; Serial.print((H-distance)); Serial.print(", "); Serial.println(count);**

**if((H-distance)<h\_max){ digitalWrite(13,HIGH);**

**}**

**if((H-distance)>=h\_max){ digitalWrite(13,LOW); count=0;**

**}**

**delay(800);**

**}**

**Serial.print((H-distance));**

**Serial.print(", "); Serial.println(count); delay(2000);**

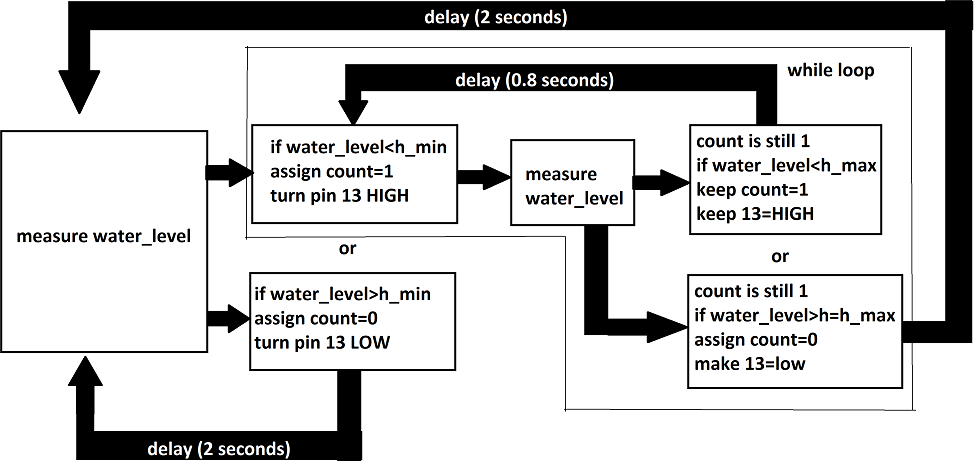
**}**

In fact, in this code the emptying and filling up cycles are more distinctly identifiable. The block enclosed

in the while loop represents the filling up cycle.

First the water\_level is measured. If water\_level<h\_min then the variable called ‘count’ is assigned the value 1 and the pump turns ON. Until and unless the tank is fully filled the ‘count’ variable value is not changed and the pump is kept ON. When water\_level goes beyond maximum the ‘count’ is assigned 0 and the pump (i.e., the pin 13) is turned OFF. It thus comes out of the while loop and the void loop again starts from the beginning but simply bypasses the while loop as the ‘count’ variable is no longer 1. Thus, the purpose of the ‘count’ variable is to define the condition which when true the while loop will run.

The steps for it can be schematically represented as:



It is evident that this flow of logic is much more complicated and in-fact unnecessary considering simplicity of the first one.

I would reemphasize that we tried out both these solutions but the one that worked more efficiently for us is the first one. Thus, the final source code submission on our behalf is the Solution 1 and not solution 2.

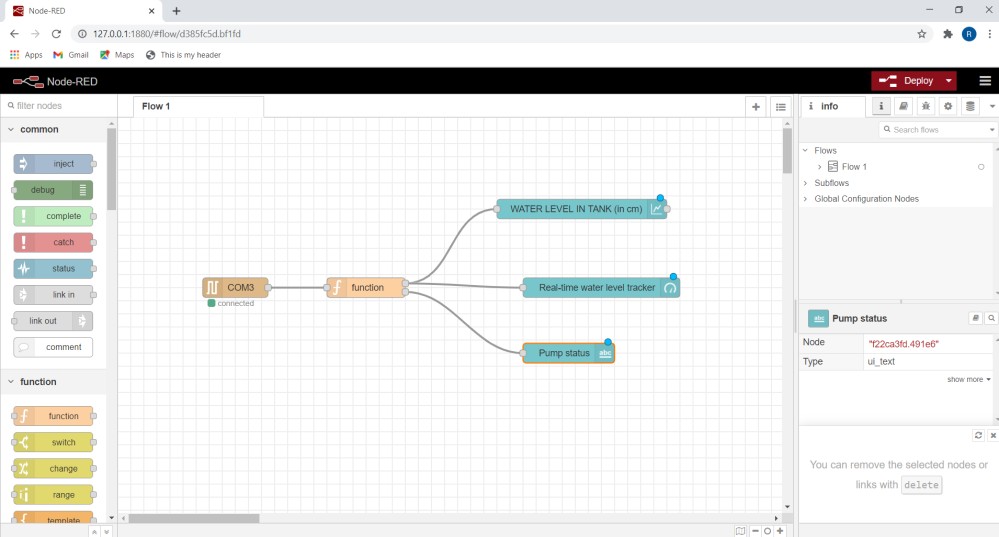
# INTERFACING WITH Node-Red:

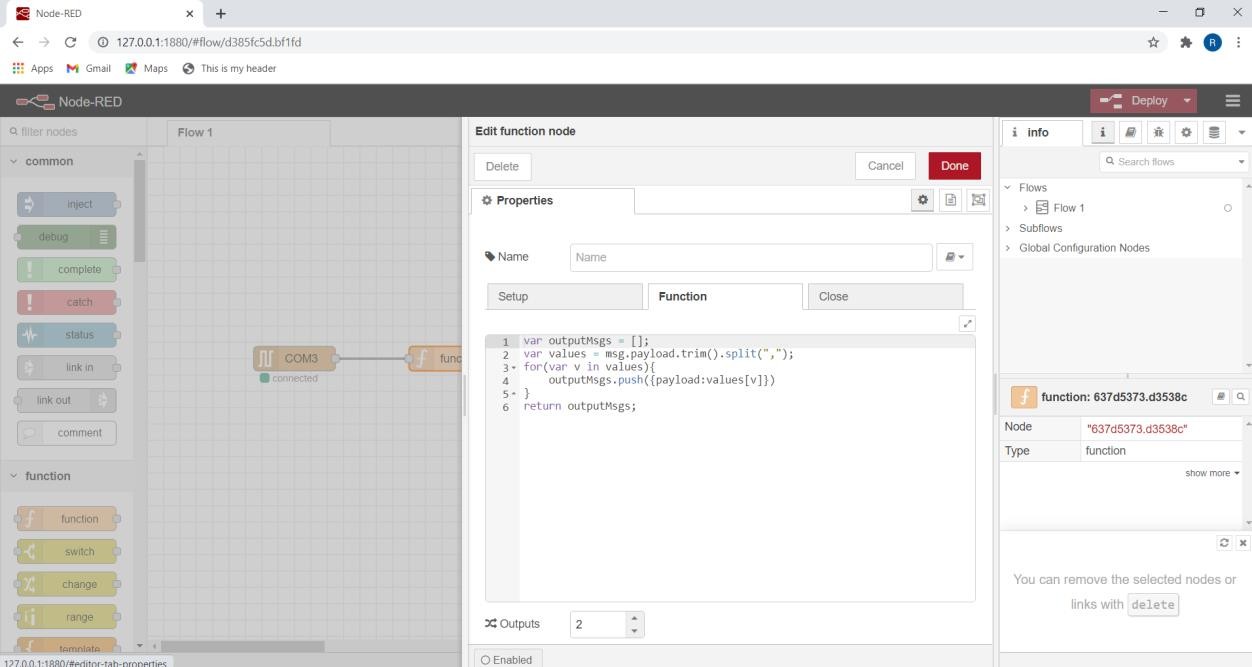
The Node-Red interface facilitates the user to get real time data about the water level in water tank and whether the pump is ON/OFF with the help of attractive dashboards in a smartphone connected to the same LAN as the Node-Red server (here the laptop).

The following steps were followed to build the Node-Red interface:

1. Making Arduino connections and uploading the code.
2. Starting the Node-Red server from command prompt.
3. Downloading the ‘dashboard’ and ‘serial’ palette. The dashboard palette as the name suggests presents lots of useful interfacing tools/templates like Gauge, Bar chart, text etc. The ‘serial in’ node in serial palette connects the Arduino to the server and receives data from it through serial port (COM3 in our case).

Following are some relevant snapshots:





The following piece of code was written into function editor:

**var outputMsgs = [];**

**var values =msg.payload.trim().split(“,”);**

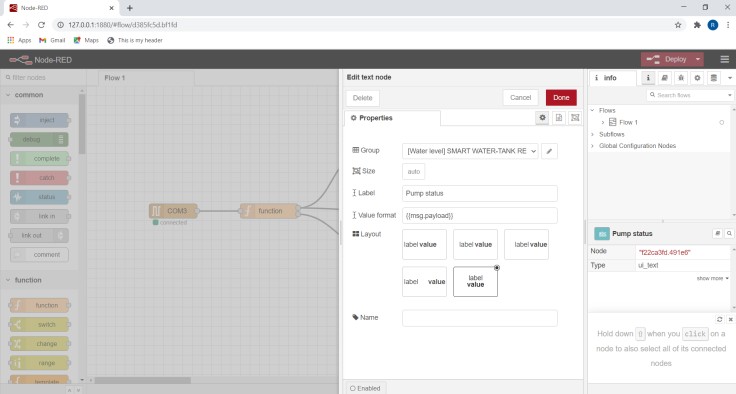
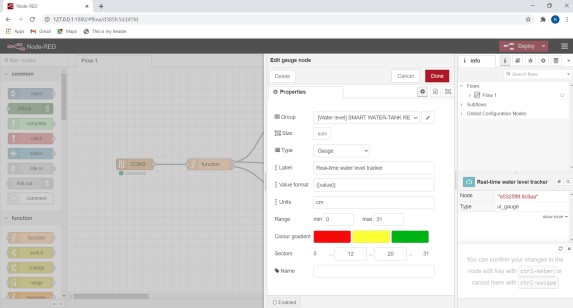
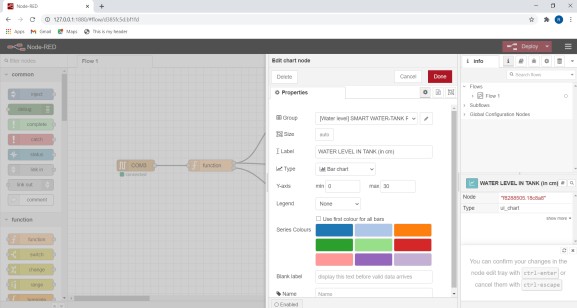
**for(var v in values){ outputMsgs.push({payload:values[v]})**

**}**

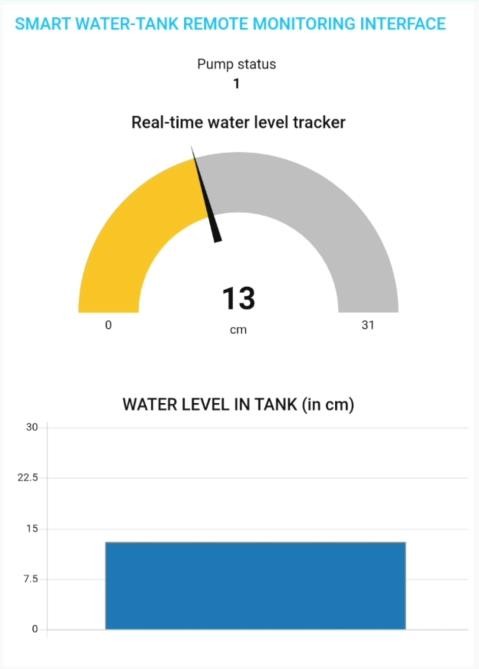
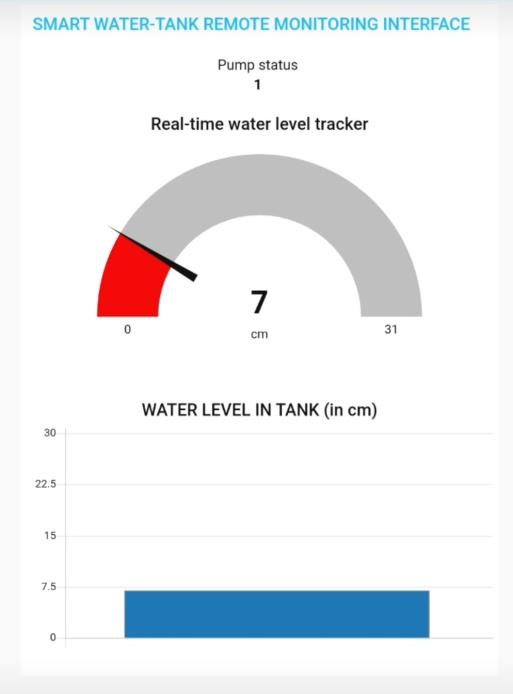
**return outputMsgs;**

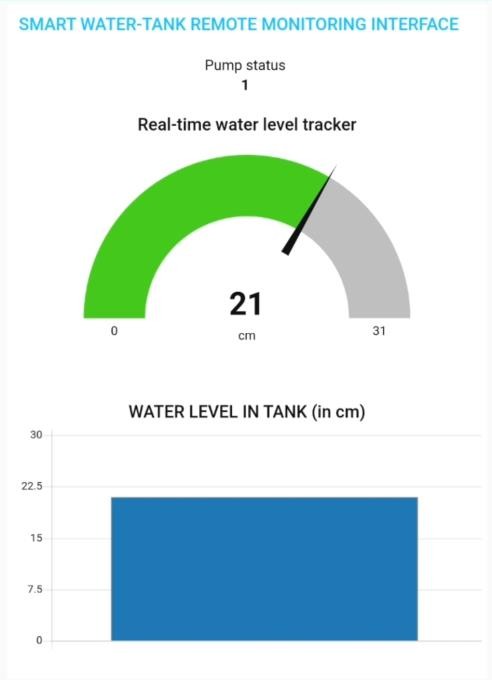
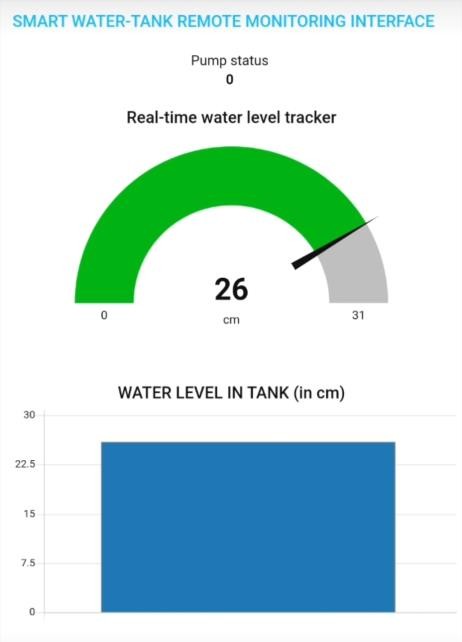
And the number of outputs was set as 2.

Please note that this particular piece of code is borrowed from ‘W5V6: Creating Node-Red Code for Interfacing with Arduino’ video in ‘DIY Lab IIT Kharagpur’ You-Tube channel. It converts the single line input from the serial port (which is in the form “water\_level, digitalRead(13)”) in the form of a string and assigns it to the 2 output values.

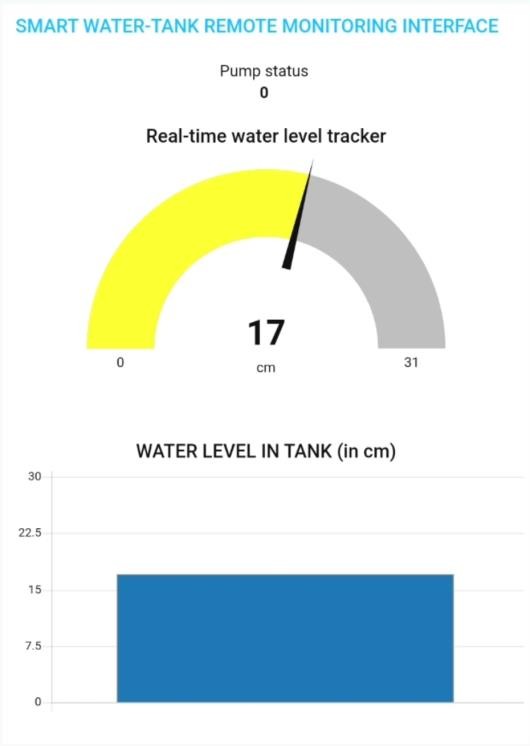
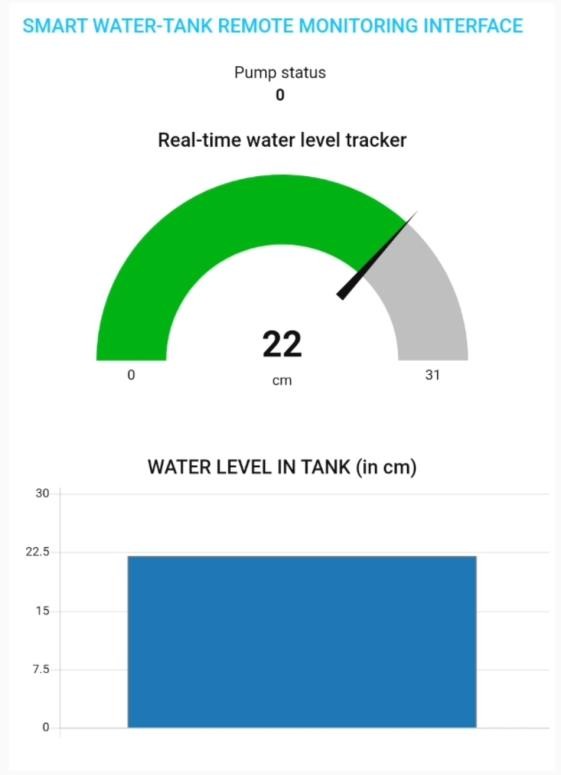


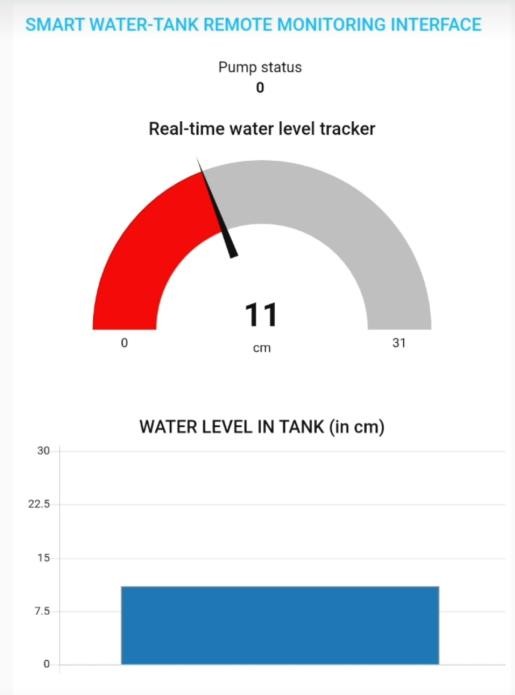
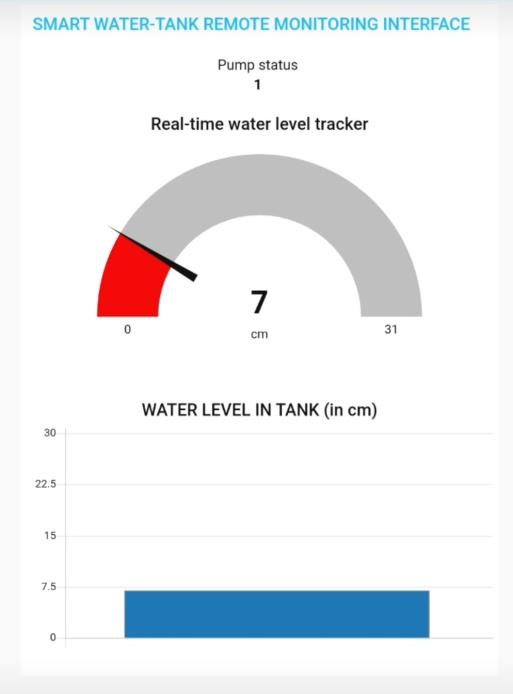
These were the steps involved in making the building the interface. Lets have a look at how it looks like when opened from smartphone



These are the dashboards when the tank is filling up, see that pump status is displayed 1 till 25 cm.



These were dashboards at the time of emptying, note that pump status is 0 till 8 cm and becomes 1 when water level goes below it.

### POWER REQUIREMENTS OF THE TWO-TERMINAL DEVICE:

* Let's consider the controller circuit as the device that we are trying to build. It’s a two-terminal device which needs to be connected to the main circuit to control flow of current through it.
* Well then, the energy requirement of the device is quite small and in fact once the code is loaded into the Arduino a 9V DC source (battery) is enough to power it up. We only need to take care that the relay does not get damaged by the amount of current passing through the external (main circuit).
* The relay can tolerate a maximum AC current of 10A (that’s quite high). For demonstrating purpose, we will probably be using a low power aquarium/cooler pump (maybe 20 -30 W) and considering a 240V supply we can show that the current does not exceed 0.08-0.13 A. So, it should not be a problem.

Even when high power pumps (like 1500W) are used then also current does not exceed 10 A so ideally, we can use the same relay with them too (but we are not quite sure whether anything else will go wrong or not in that case).

One of the key perks of our product is that it is not affected by what is going on in the external circuit, suppose the main AC source becomes OFF due to power cut, the Arduino will still be receiving power

from the battery and as soon as power comes back pump will start from exactly where it left off (it would turn ON if it was ON when the power went away else it will remain in OFF condition).

# SUMMARY AND CONCLUSIONS:

1. It is possible to control the water level in a simple overhead tank using HC-SR04 sensor, Arduino Uno (programmed accordingly) and relay module so as to minimize wastage of water and electricity.
2. The water-tank if too narrow may lead to erroneous readings of the sensor due to unwanted echoes from its walls. Such errors can be avoided if we use tanks of larger diameter/dimensions.
3. The Arduino board and the sensor do not like water so sufficient precautions must be taken to make the setup as much moisture proof as possible in order to use it on a long-term basis.
4. If the user needs to go out for some days, he/she might want to stop the operations of the pump for that period so that the water level remains low. It can be done by simply turning OFF the switch of the socket to which the pump is connected. Although the Arduino remains powered up (because it receives power from an internal battery) the pump does not receive power. When the user is back home, he/she will just turn the main switch on and the automatic controlling of water level again starts off from where it was left.

*˜˜END˜˜*