# IOT Water Level Project

IOT SEMESTER 2 REPORT

Aslam Patel
@00520874 | A.PATEL54@EDU.SALFORD.AC.UK

## Table of Contents

## Table of figures

## Introduction

In today's homes and constructions, water tanks are the principal source of water delivery. It's hardly surprising that these water tanks include built-in water level meters, considering our current technology. Others are also affected, notably those who built their own water tanks without built-in water level sensors years ago.

The report describes how to use an Arduino Microcontroller to measure the amount of water in a tank automatically and display a reading of millimeters on a LCD screen, along with a LED display.

A flowchart was created from well-written algorithms, from which Codes were generated and compiled on the Arduino IDE.

This project's prime goal was to build a water level detecting system for a water tanker that would be used in developing countries. The reasoning for this design and the problem it has come to solve will be discussed in the research section of this report.  It will allow us to detect the amount of water in our houses by utilizing three distinct colored LEDs to signal the level of water in the tank. It is a very easy and inexpensive method.

The components used will be discussed in the functional components section. The build of this device will be included including its schematic diagram, flow chart, code and physical pictures of the build will also be discussed.

## Research

Access to clean and safe water is one of the most pressing issues facing the world, including Africa. Water is a basic human requirement that must be met for humans to survive the next day, there for when clean water is available it needs to be stored and monitored. According to the World Health Organization (WHO), over 40% of Africa's Sub-Saharan population still lacks access to safe drinking water. *("The Challenge: Clean and Safe Water - Africa.com", 2022)*

However, some experts predict that by 2025, more than half of the world's population would be vulnerable to water-related hazards. The existence of a water level indicator in a reservoir can aid in the control of wastage and water scarcity. A water level indicator is used to display the level of water in an overhead tank. This keeps the user informed about the water level at all times, preventing water from running out when it is most required. Alarm functionalities are available on indicator circuits. It not only shows how much water is in the overhead tank, but it also sounds an alarm when the tank is full.

One of the outcomes of the contemporary era is the proliferation of automatic equipment. Operating systems are frequently wasted manually; also, the cost of employing operators can eat into firm finances. Apart from fixing the challenge, the design also goes into current Arduino design approach.

Automatic systems reduce the number of personnel required to attend to industrial processes; as a result, wages and losses are drastically reduced, resulting in increased profitability.

For this reason, this report looks to simply deal with water tanker for lower developed countries and how with minimal technology this can also be managed.

## Overview of functional components

### Arduino

A computer on a chip is what a microcontroller is called. Input and output pins are provided. It includes a processor with memory for storing programs written in the C programming language, unlike a microprocessor.

*Figure 1 Arduino Uno*

### Water level sensor

The Water Level Sensor operates on a very simple concept.
Essentially, the sensor's series of parallel conductors serves as a variable resistor (Similar to the potentiometer). The resistance changes as the water level rises or falls. As a result, the change in resistance is inversely proportional to the distance between the sensor's top and the water's surface. *("Water Level Sensor - Renke", 2022)*

*Figure 2 Water Level Sensor*

1. The S (Signal) pin in Arduino is an analogue output pin that is always connected to the analogue inputs pin.
2. The + (VCC) pin is the sensor's power supply. provides the sensor with power. The water level sensor should be powered between 3.3V and 5V. The analogue output of the sensor will vary based on the voltage applied to it.
3. The ground connection is made with the (GND) pin.

### LED's

Lights to show data when certain parameters have been reached. For example, blink red when water level low.

### LCD Screen

The term LCD stands for liquid crystal display. Any display, including alphanumeric character LCD displays, monochrome graphic LCD displays, color TFT LCD displays, and IPS LCD displays, may be utilized with Arduino. This will be used to show the device's readings in milliliters.

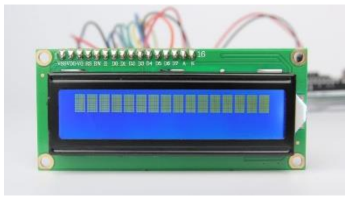


*Figure 3 LCD Screen*

## ESP8266 WIFI

The ESP8266 is a highly integrated chip that was created to meet the demands of a future connected world. It provides a comprehensive and self-contained Wi-Fi networking solution, allowing it to host or offload all Wi-Fi networking tasks from another application processor. ("ESP8266 ESP-12F Wifi Module System On A Chip - ESP8266 Shop", 2022)

## Breadboard

A breadboard is simply a circuit-building or prototype board. It permits you to construct circuits without the need of soldering by permitting you to set parts and links on the board. This will be used to connect the device with the LED’s.

## Wires

Connecter wires to interconnect all devices and devices within the design.

## Design of functional elements

### Block Diagram

This diagram shows the overall function of the sensors and how they function with the Arduino, LCD Display and LED display.

Once the level is measured by the water level sensor it is captured and shown on the LCD Display and the LED display with its respective light turning on. Shown in figure 4

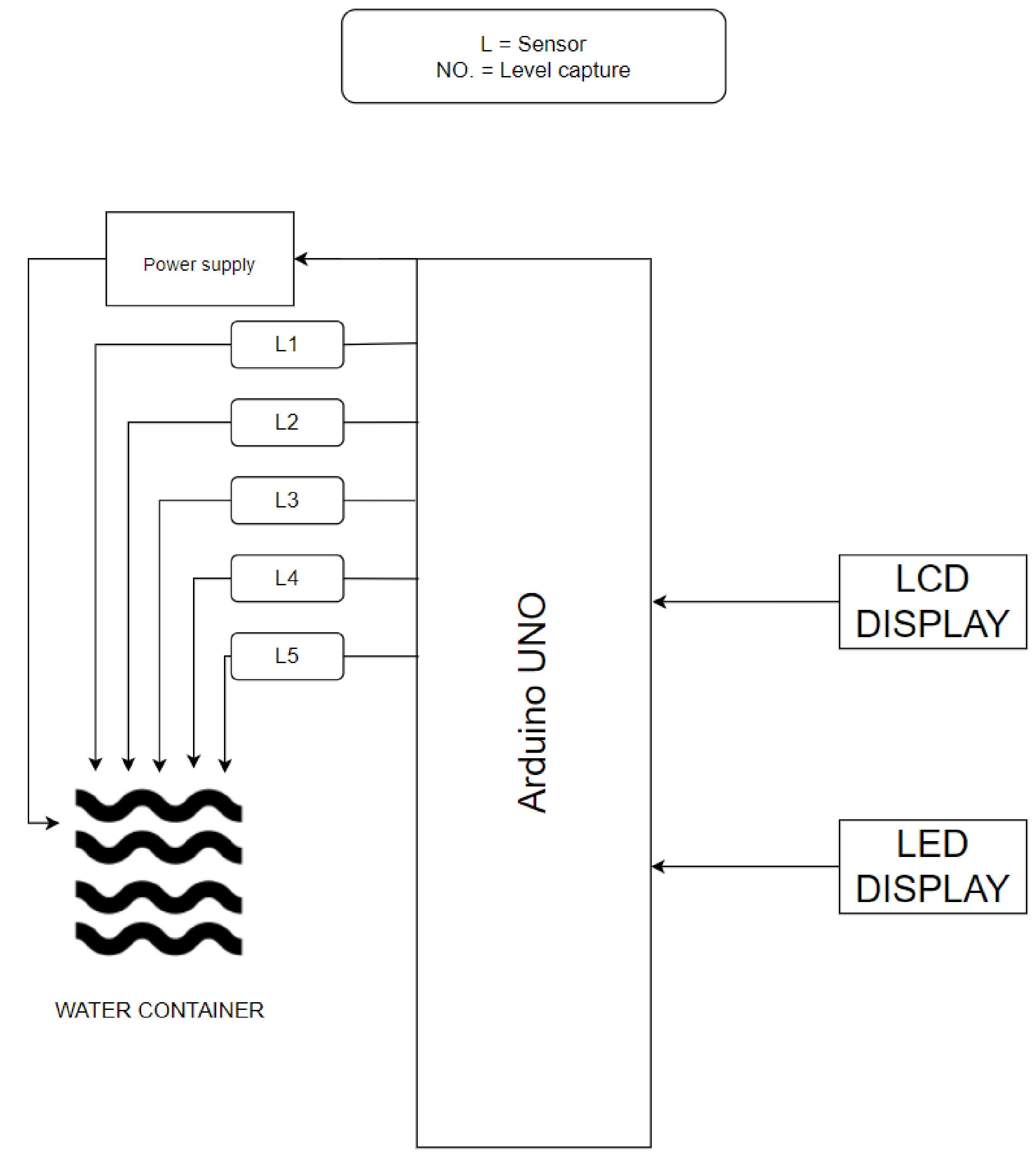


*Figure 4 Block Diagram*

## Flow chart

If the condition is satisfied, the microcontroller indicates the same on the display unit and also lights up the relevant LED. If the condition is not satisfied, the microcontroller checks if the tank is filled up to level 5, and the process repeats, with the appropriate level indicated in the display unit. Shown in figure 5

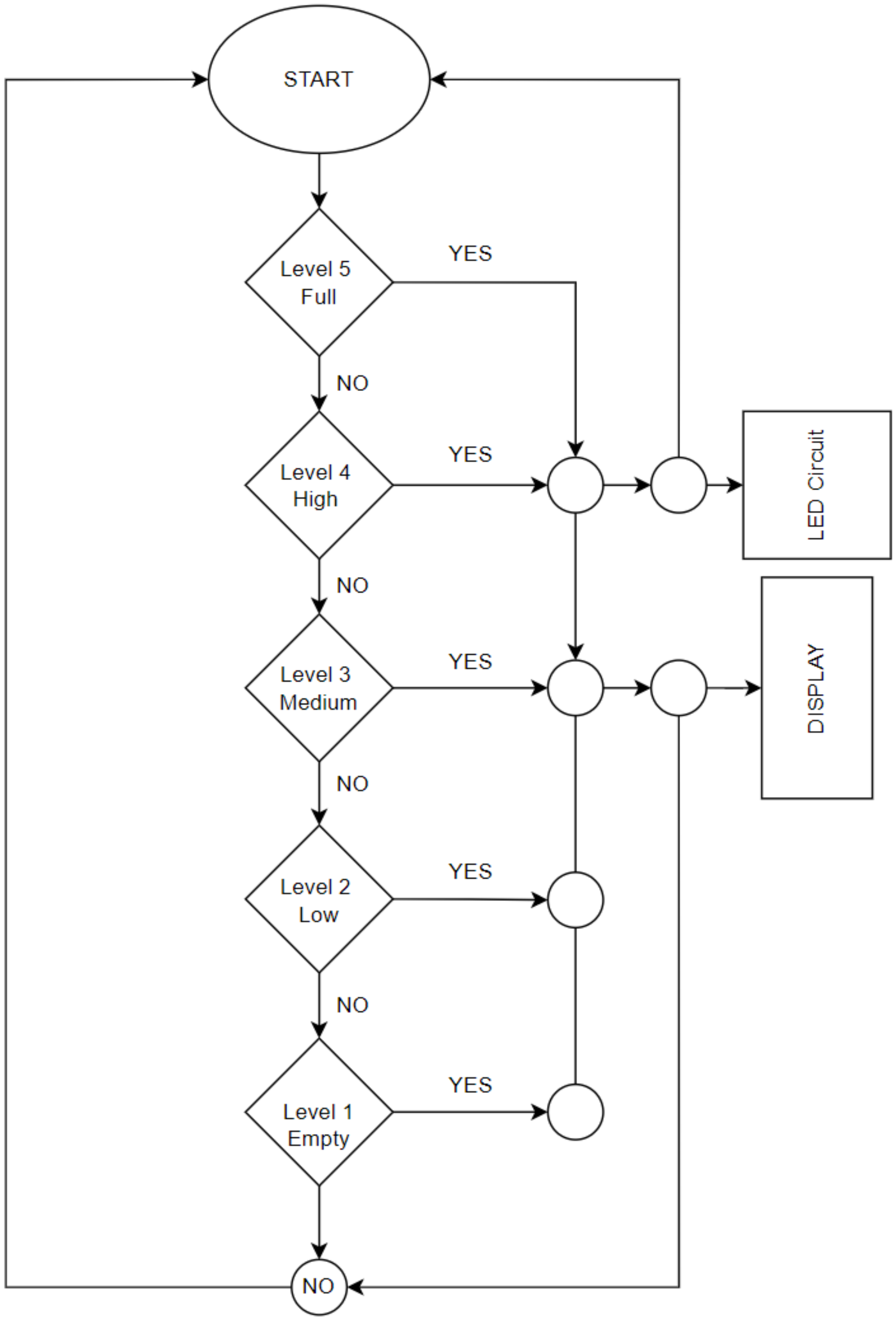
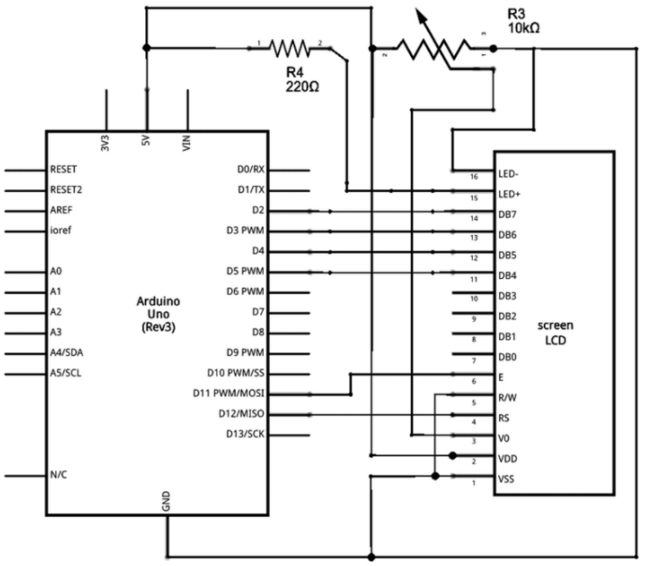


*Figure 5 Flow Chart*

## Project assembly

### LCD Screen Schematic and setup

The LCD screen has been set up in the universal way provided by Arduino shown in figure and figure 7 and figure 6. The schematic that was provided by Arduino was followed and set up.



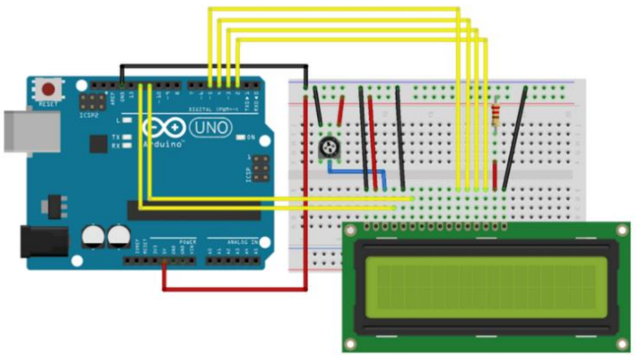
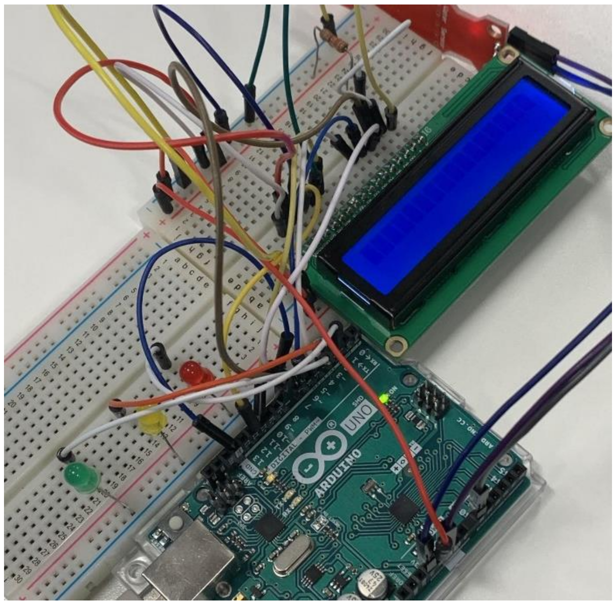


*Figure 6 Tinker CAD Schematic LCD*

*Figure 7 Circuit Schematic LCD*



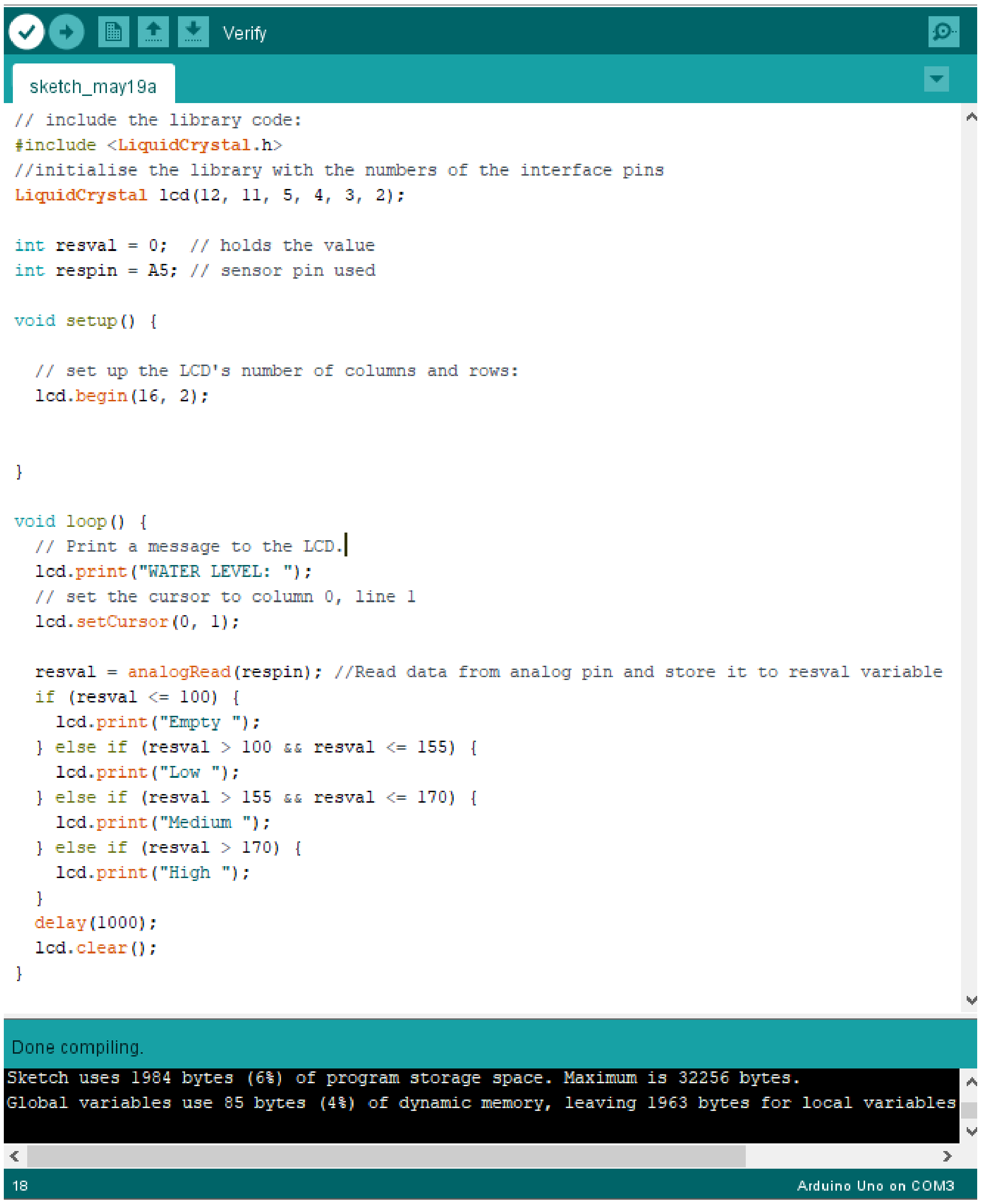
This was assembled in the following manner.

1. LCD RS pin → digital pin 12 (Arduino)
2. LCD Enable pin → digital pin 11 (Arduino)
3. LCD D4 pin → digital pin 5 (Arduino)
4. LCD D5 pin → digital pin 4 (Arduino)
5. LCD D6 pin → digital pin 3 (Arduino)
6. LCD D7 pin → digital pin 2 (Arduino)
7. LCD R/W pin → GND (Arduino)
8. LCD VSS pin → GND (Arduino)
9. LCD VCC pin → 5V (Arduino)
10. LCD LED+ → 5V through a resistor (Arduino)
11. LCD LED - → GND (Arduino)

*Figure 8 Physical Contruction LCD*

## Code for LCD

The LCD version of the programme code is shown below in figure. In essence it adds a loop to print to the LCD and return the cursor on each cycle shown in figure 9.



```
// include the library code:
#include <LiquidCrystal.h>
//initialise the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int resval = 0;  // holds the value
int respin = A5; // sensor pin used

void setup() {

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);


}

void loop() {
  // Print a message to the LCD.
  lcd.print("WATER LEVEL: ");
  // set the cursor to column 0, line 1
  lcd.setCursor(0, 1);

  resval = analogRead(respin); //Read data from analog pin and store it to resval variable
  if (resval <= 100) {
    lcd.print("Empty ");
  } else if (resval > 100 && resval <= 155) {
    lcd.print("Low ");
  } else if (resval > 155 && resval <= 170) {
    lcd.print("Medium ");
  } else if (resval > 170) {
    lcd.print("High ");
  }
  delay(1000);
  lcd.clear();
}
```

```
Done compiling.
Sketch uses 1984 bytes (6%) of program storage space. Maximum is 32256 bytes.
Global variables use 85 bytes (4%) of dynamic memory, leaving 1963 bytes for local variables
```

```
18                                                    Arduino Uno on COM3
```

*Figure 9 LCD code (Arduino.com)*

## Water Sensor Schematic and Set up

The full design has been attached in figure. The main steps of this set up have been detailed below (shown in figure 10)

12. Water sensor S pin → A0 (Arduino)
13. Water sensor + pin → 5V (Arduino)
14. Water sensor – pin → GND (Arduino)
15. Red LED short pin → Negative pin (Breadboard)
16. Yellow LED short pin → Negative pin (Breadboard)
17. Green LED short pin → Negative pin (Breadboard)

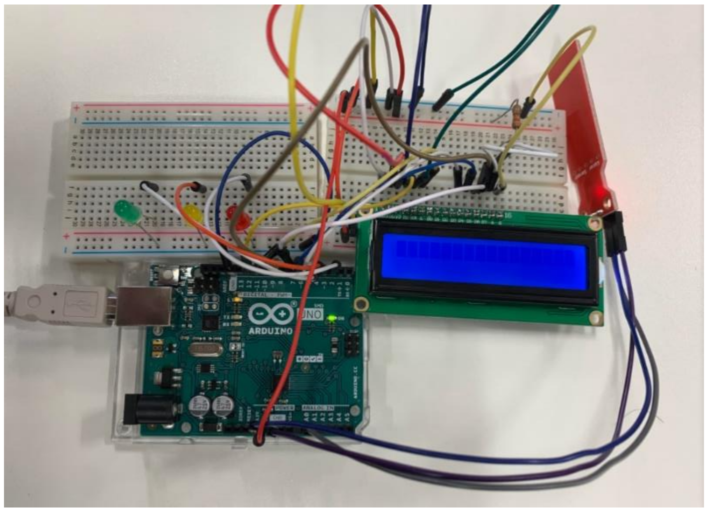


*Figure 10 Water level sensor integrated set up*

If LED 1 is illuminated, the water level in the tank is at its lowest, indicating that there is little or no water remaining. If LED 2 is illuminated, the tank is half-filled, and if LED 3 is illuminated, the alarm will sound, signaling that the tank's water level is full.

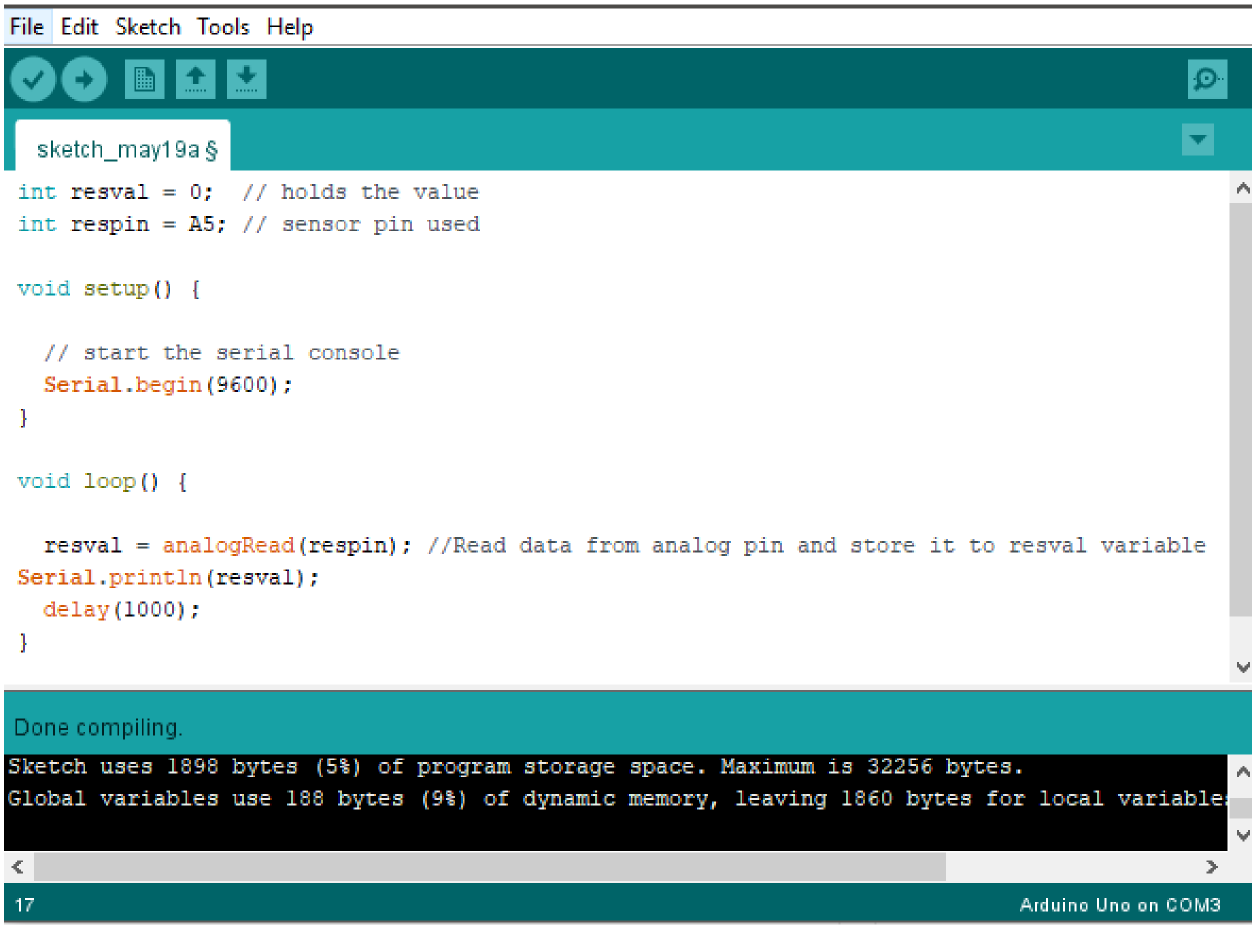
The readings in milliliters were shown on the LCD screen which was referenced back to the serial monitor on the Arduino software.

## Code for Water sensor calibration

As there can be varied readings on the resistance of the water sensor, according to the journal on Arduino it is highly recommended to calibrate the sensor in relation to different water types, e.g., tap water compared to bottled natural water.

As pure water is non-conductive the minerals inside it is what causes the sensor to pick up the readings.

To ensure that the sensor was calibrated the following code was used and results were gained from the serial monitor which showed a sensor value of 130 at first when the sensor was not yet submerged even a little in the water but after calibration the sensor showed a result of 0, which meant the sensor was now calibrated correctly shown in figure 11.



```
int resval = 0;  // holds the value
int respin = A5; // sensor pin used

void setup() {

  // start the serial console
  Serial.begin(9600);
}

void loop() {

  resval = analogRead(respin); //Read data from analog pin and store it to resval variable
Serial.println(resval);
  delay(1000);
}
```

```
Sketch uses 1898 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 188 bytes (9%) of dynamic memory, leaving 1860 bytes for local variable
```

*Figure 11 Code for Water sensor calibration*

## ESP8266 Setup

18. Pin One → RTX
19. Pin TWO → GPIO
20. Pin THREE → GPIO2
21. Pin FOUR → GND
22. Pin FIVE → TX
23. Pin SIX → 3.3v
24. Pin SEVEN → Reset
25. Pin EIGHT → 3.3v

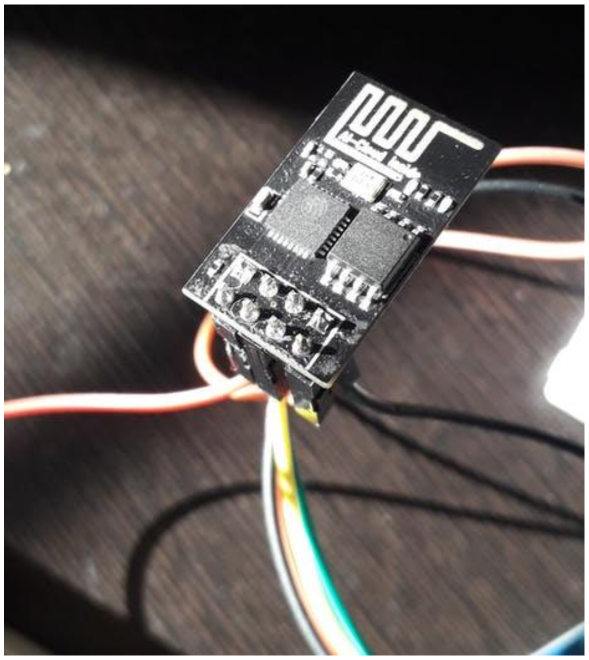


*Figure 12 ESP Device*

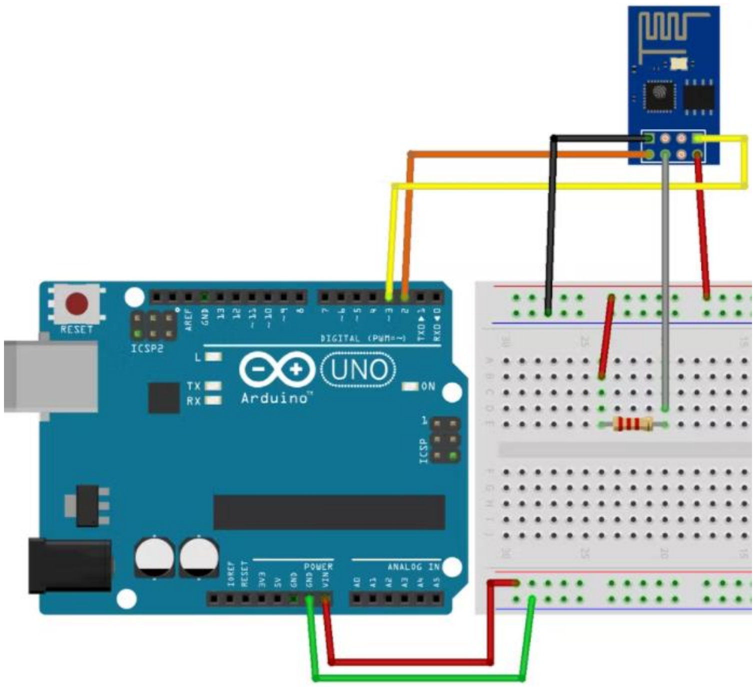


*Figure 13 ESP Schematic*

## Full schematic Setup

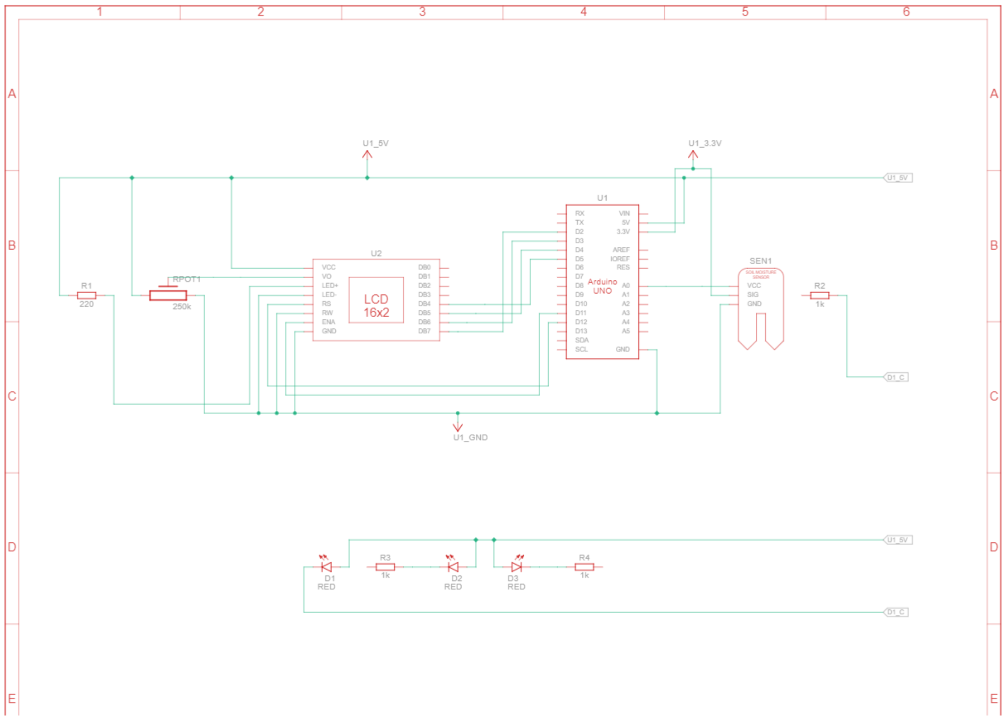


Figure 14 Full schematic

The overall schematic design of the full design is shown in figure compared to the physical design shown in figure 13.

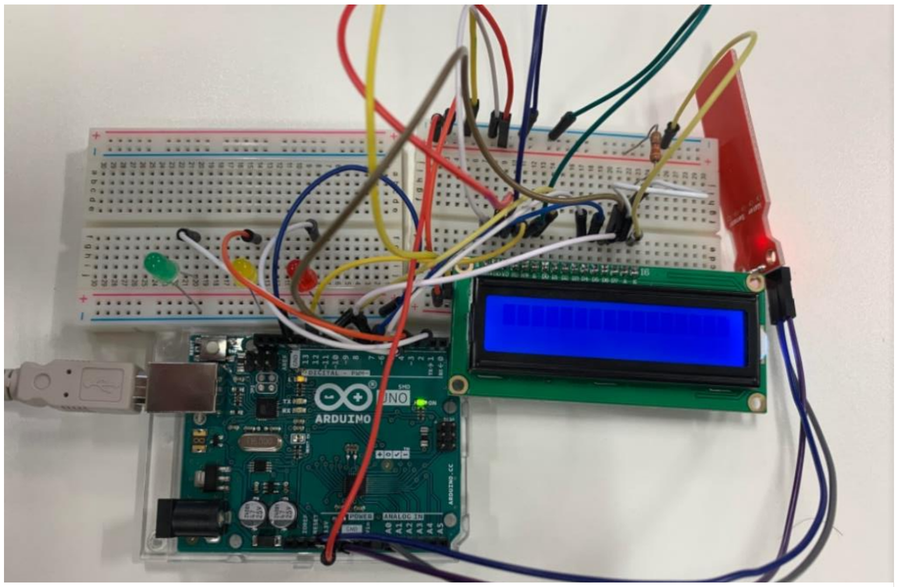
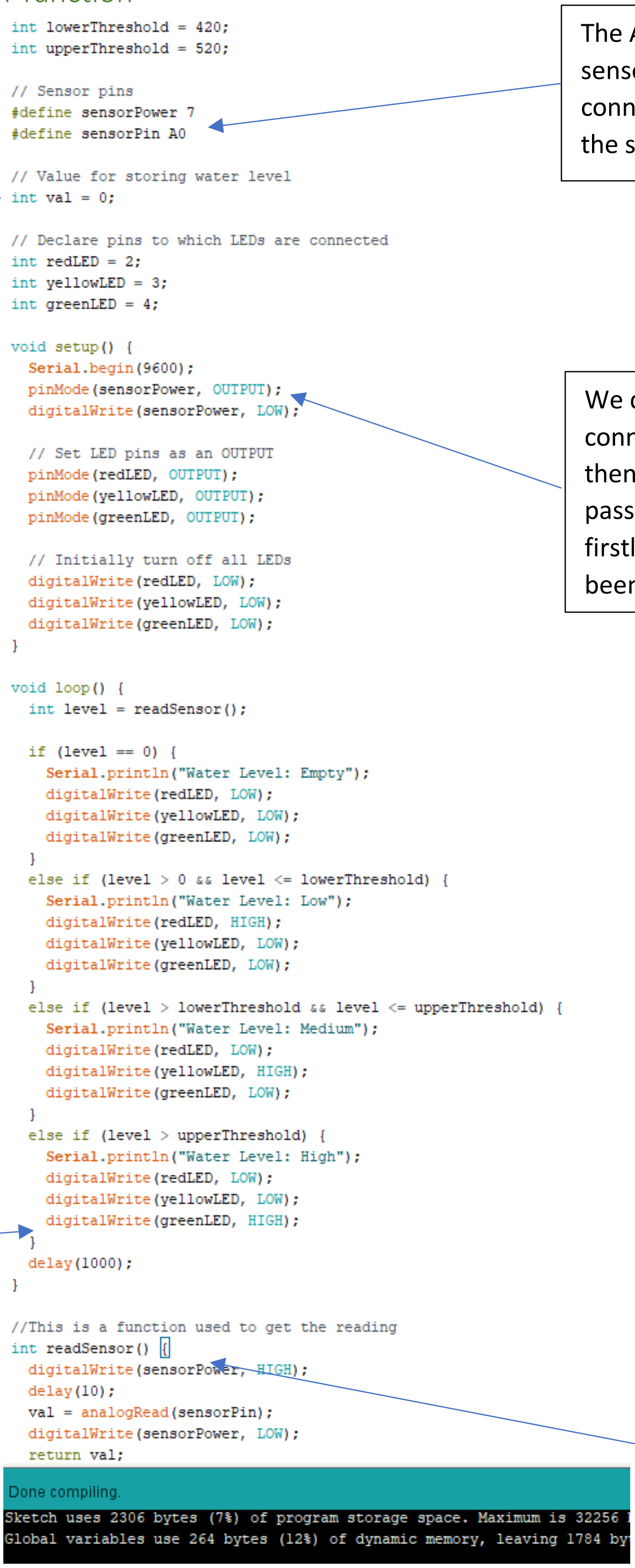


Figure 15 Full Physical set up

## Code for water sensor function



```
int lowerThreshold = 420;
int upperThreshold = 520;

// Sensor pins
#define sensorPower 7
#define sensorPin A0

// Value for storing water level
int val = 0;

// Declare pins to which LEDs are connected
int redLED = 2;
int yellowLED = 3;
int greenLED = 4;

void setup() {
  Serial.begin(9600);
  pinMode(sensorPower, OUTPUT);
  digitalWrite(sensorPower, LOW);

  // Set LED pins as an OUTPUT
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);

  // Initially turn off all LEDs
  digitalWrite(redLED, LOW);
  digitalWrite(yellowLED, LOW);
  digitalWrite(greenLED, LOW);
}

void loop() {
  int level = readSensor();

  if (level == 0) {
    Serial.println("Water Level: Empty");
    digitalWrite(redLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);
  }
  else if (level > 0 && level <= lowerThreshold) {
    Serial.println("Water Level: Low");
    digitalWrite(redLED, HIGH);
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);
  }
  else if (level > lowerThreshold && level <= upperThreshold) {
    Serial.println("Water Level: Medium");
    digitalWrite(redLED, LOW);
    digitalWrite(yellowLED, HIGH);
    digitalWrite(greenLED, LOW);
  }
  else if (level > upperThreshold) {
    Serial.println("Water Level: High");
    digitalWrite(redLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, HIGH);
  }
  delay(1000);
}

//This is a function used to get the reading
int readSensor() {
  digitalWrite(sensorPower, HIGH);
  delay(10);
  val = analogRead(sensorPin);
  digitalWrite(sensorPower, LOW);
  return val;
```

```
Done compiling.
Sketch uses 2306 bytes (7%) of program storage space. Maximum is 32256 
Global variables use 264 bytes (12%) of dynamic memory, leaving 1784 by
```

The Arduino pins to which the sensor's + (VCC) and S (signal) pins are connected are declared at the start of the sketch.

Create a variable value to keep track of the current water level.

We define the sensor's power connection as output initially, then set it low so that no power passes through to the sensor firstly. The serial monitor had also been set up.

The readSensor() function is called periodically at a one-second interval in the loop section, and the resulting value is printed.

The current water level is obtained using the readSensor () function. It turns the sensor on, waits 10 milliseconds, then reads the analogue value from the sensor before turning it off and returning the analogue value.

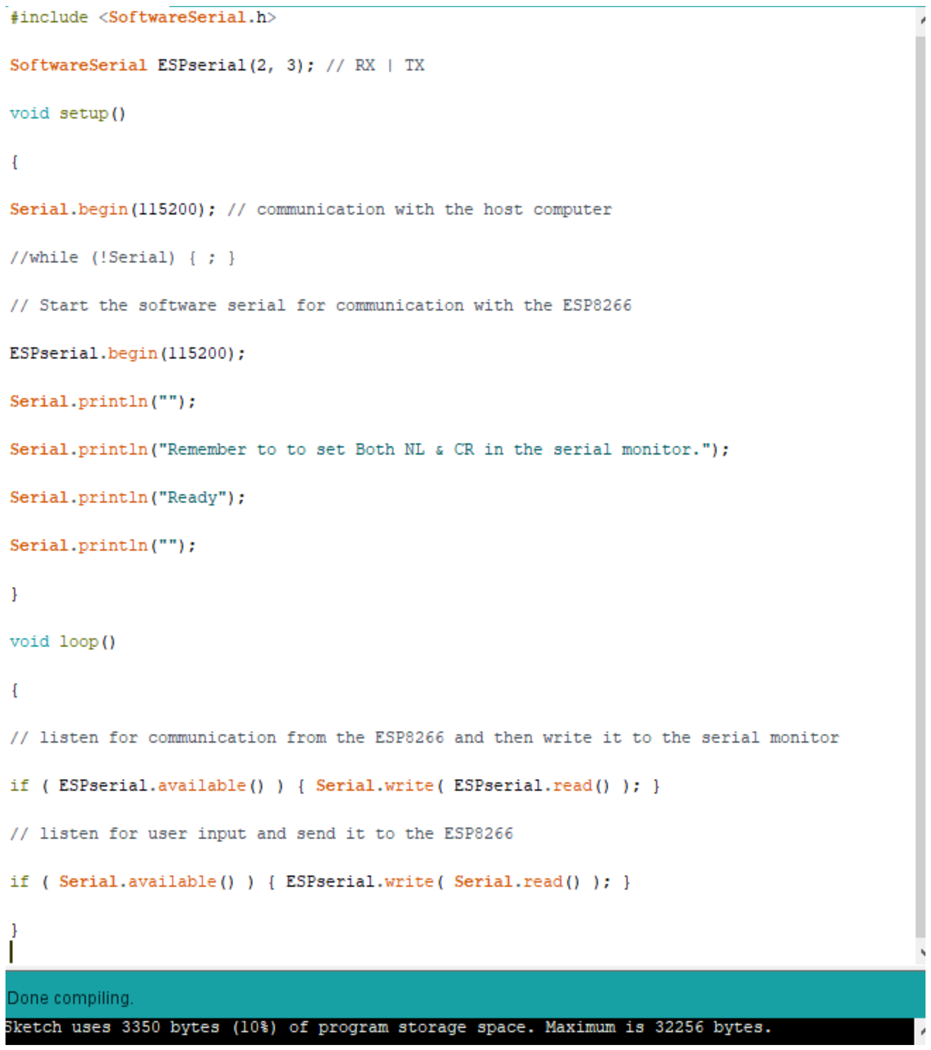
*Figure 16 Code for water sensor function*

## Testing

### Wi-Fi connectivity

In order for the device to be connected to a network the ESP8266 device was used. It was powered by the Arduino 3V pin as the 5v pin is too powerful for this device.

The code which was used for the setup of the Wi-Fi device was the universal base code retrieved from the Arduino website attached in figure 15.

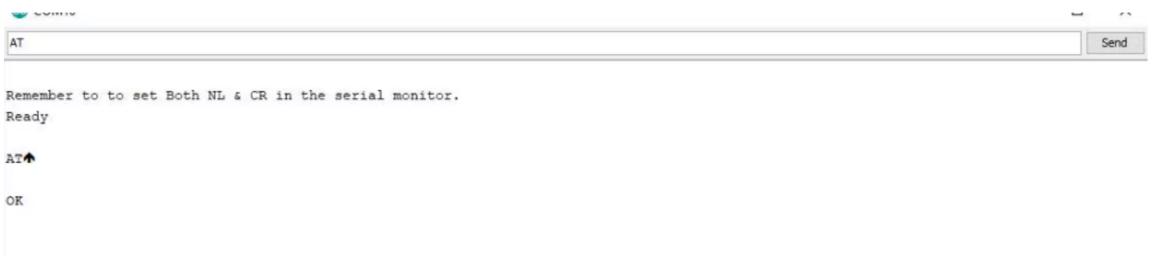


```
#include <SoftwareSerial.h>

SoftwareSerial ESPserial(2, 3); // RX | TX

void setup()

{

Serial.begin(115200); // communication with the host computer

//while (!Serial) { ; }

// Start the software serial for communication with the ESP8266

ESPserial.begin(115200);

Serial.println("");

Serial.println("Remember to to set Both NL & CR in the serial monitor.");

Serial.println("Ready");

Serial.println("");

}

void loop()

{

// listen for communication from the ESP8266 and then write it to the serial monitor

if ( ESPserial.available() ) { Serial.write( ESPserial.read() ); }

// listen for user input and send it to the ESP8266

if ( Serial.available() ) { ESPserial.write( Serial.read() ); }

}
```

```
Done compiling.
Sketch uses 3350 bytes (10%) of program storage space. Maximum is 32256 bytes.
```

*Figure 17 Wi-Fi Code | (UoS IoT Labs)*

The code was then entered, complied, and uploaded and the serial monitor was brought up. The initial command to send to the serial monitor was an AT command. This was to ensure that the ESP was set up correctly and functioning this is shown in figure 16.

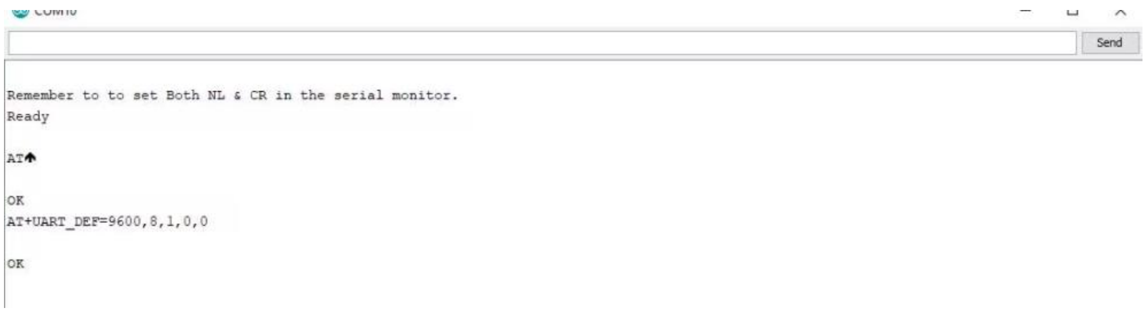


```
AT
```

```
Remember to to set Both NL & CR in the serial monitor.
Ready

AT

OK
```

*Figure 18 AT Command*

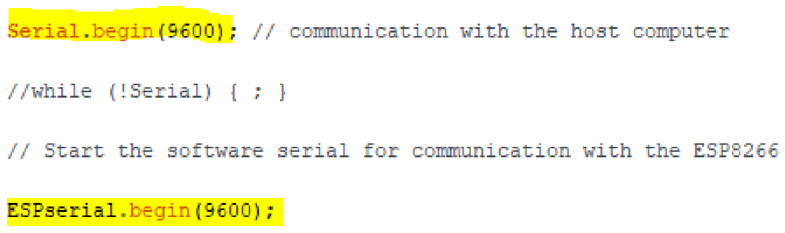
An OK response was received therefore it was acknowledged that we would be able to communicate to the ESP. we changed the BAUD rate from 115200 to 9600 as this was too fast shown in figure 17.



```
Remember to to set Both NL & CR in the serial monitor.
Ready

AT

OK
AT+UART_DEF=9600,8,1,0,0

OK
```

*Figure 19 Baud rate command*

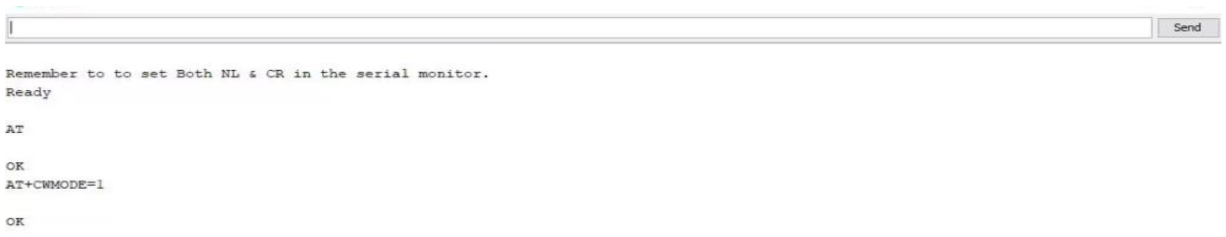
The following command was sent a AT+UART_DEF=9600, 8, 1, 0, 0 and an OK response was received. Once this was done it was also changed within the code shown in figure 18.



```
Serial.begin(9600); // communication with the host computer

//while (!Serial) { ; }

// Start the software serial for communication with the ESP8266

ESPserial.begin(9600);
```

*Figure 20 Alter Baud rate on code*

The module now needs to set to mode 1- and the command that was used for this was AT+CWMODE=1. An OK response was again received attached in figure 19



```
Remember to to set Both NL & CR in the serial monitor.
Ready

AT

OK
AT+CWMODE=1

OK
```

*Figure 21 Mode command*

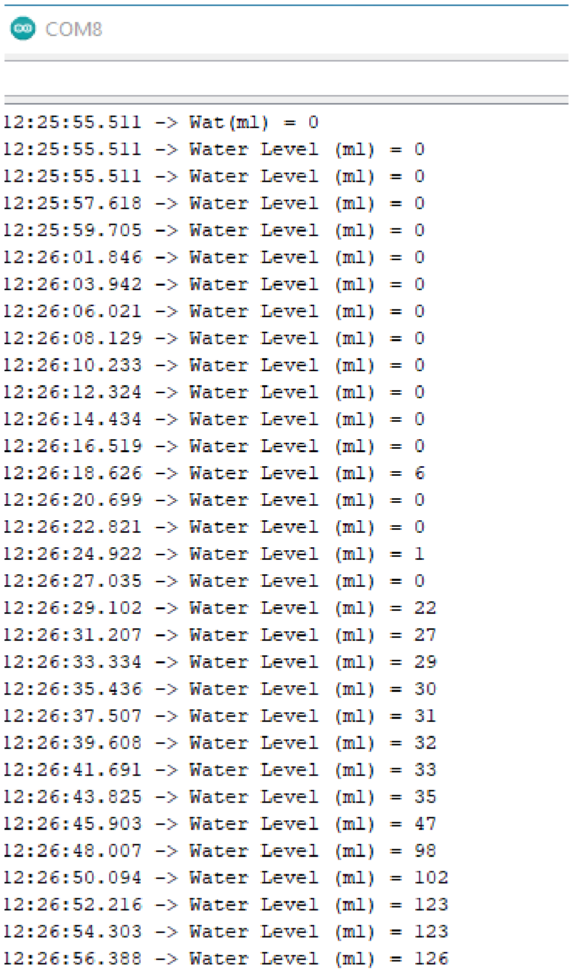
The next command that was used was to monitor the local Wi-Fi networks. The command ran was AT+CWLAP. And to connect to the specific network the command AT+CWJAP="Iotlab","743DE2A148 was inputted. This provided the terminal with the network which we want to connect to which is the IoT lab and its password which is 743DE2A148.

## Final results

The first test was to slightly submerge the sensor in water. The output that was expected was a sensor reading between 0-50. This was triggering the red LED switch as there is minimal or no water in the tank. The LCD screen also gave the same reading with the water level shown. The data can be seen in the serial monitor capture in figure

The second test was to submerge the sensor more in water. The output that was expected was a sensor reading between 51-100. This was triggering the yellow LED switch as there was a medium amount of water in the tank. The LCD screen also gave the same reading with the water level shown. The data can be seen in the serial monitor capture in figure

The final test was to submerge the sensor fully in water. The output that was expected was a sensor reading between 100-<. This was triggering the green LED switch as there is maximum amount of water in the tank. The LCD screen also gave the same reading with the water level shown. The data can be seen in the serial monitor capture in figure 20



```
12:25:55.511 -> Wat(ml) = 0
12:25:55.511 -> Water Level (ml) = 0
12:25:55.511 -> Water Level (ml) = 0
12:25:57.618 -> Water Level (ml) = 0
12:25:59.705 -> Water Level (ml) = 0
12:26:01.846 -> Water Level (ml) = 0
12:26:03.942 -> Water Level (ml) = 0
12:26:06.021 -> Water Level (ml) = 0
12:26:08.129 -> Water Level (ml) = 0
12:26:10.233 -> Water Level (ml) = 0
12:26:12.324 -> Water Level (ml) = 0
12:26:14.434 -> Water Level (ml) = 0
12:26:16.519 -> Water Level (ml) = 0
12:26:18.626 -> Water Level (ml) = 6
12:26:20.699 -> Water Level (ml) = 0
12:26:22.821 -> Water Level (ml) = 0
12:26:24.922 -> Water Level (ml) = 1
12:26:27.035 -> Water Level (ml) = 0
12:26:29.102 -> Water Level (ml) = 22
12:26:31.207 -> Water Level (ml) = 27
12:26:33.334 -> Water Level (ml) = 29
12:26:35.436 -> Water Level (ml) = 30
12:26:37.507 -> Water Level (ml) = 31
12:26:39.608 -> Water Level (ml) = 32
12:26:41.691 -> Water Level (ml) = 33
12:26:43.825 -> Water Level (ml) = 35
12:26:45.903 -> Water Level (ml) = 47
12:26:48.007 -> Water Level (ml) = 98
12:26:50.094 -> Water Level (ml) = 102
12:26:52.216 -> Water Level (ml) = 123
12:26:54.303 -> Water Level (ml) = 123
12:26:56.388 -> Water Level (ml) = 126
```

*Figure 22 Serial monitor results*

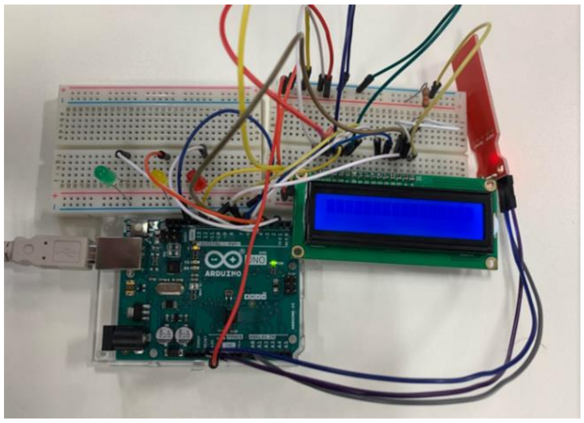


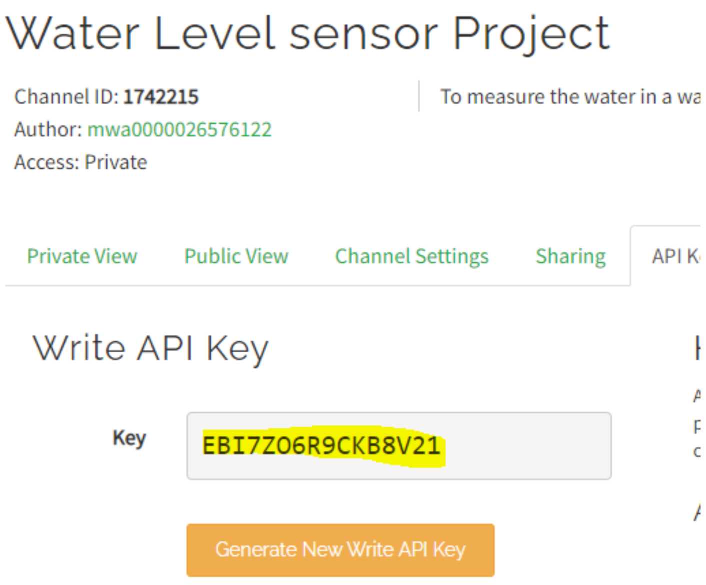
*Figure 23 LED Receivers*

the results from the serial monitor were shown and they were also shown on the LCD screen and to its relevant LED lights on the breadboards shown in figure 21.

## IoT cloud (Think Speak)

In order for us to store our readings in an IoT based cloud the think speak server will be used. Firstly, I created an account and opened up a new project.

An API key was retrieved EBI7ZO6R9CKB8V21 shown in figure 22


Water Level sensor Project

Channel ID: **1742215**
Author: mwa0000026576122
Access: Private

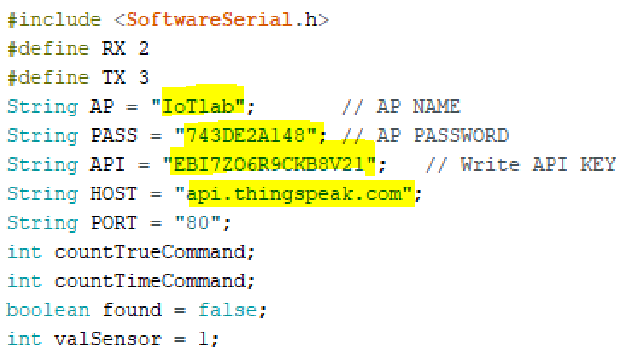
To measure the water in a wa

Write API Key

Key EBI7ZO6R9CKB8V21

*Figure 24 API*

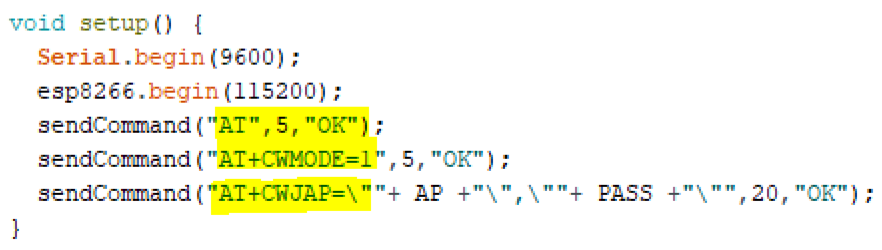
This was then inputted into the IoT code alongside the WIFI name and password. This is shown in figure 23



```
#include <SoftwareSerial.h>
#define RX 2
#define TX 3
String AP = "IoTlab";       // AP NAME
String PASS = "743DE2A148"; // AP PASSWORD
String API = "EBI7ZO6R9CKB8V21";   // Write API KEY
String HOST = "api.thingspeak.com";
String PORT = "80";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;
```

*Figure 25 Input received API*

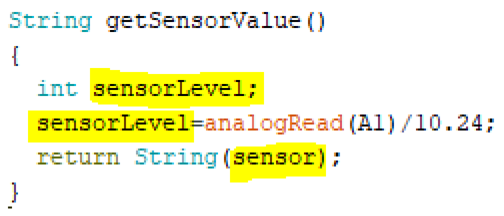
The code also included the Wi-Fi connectivity commands shown in figure 24



```
void setup() {
  Serial.begin(9600);
  esp8266.begin(115200);
  sendCommand("AT",5,"OK");
  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");
}
```

*Figure 26 Wi-Fi Connectivity to Think speak*

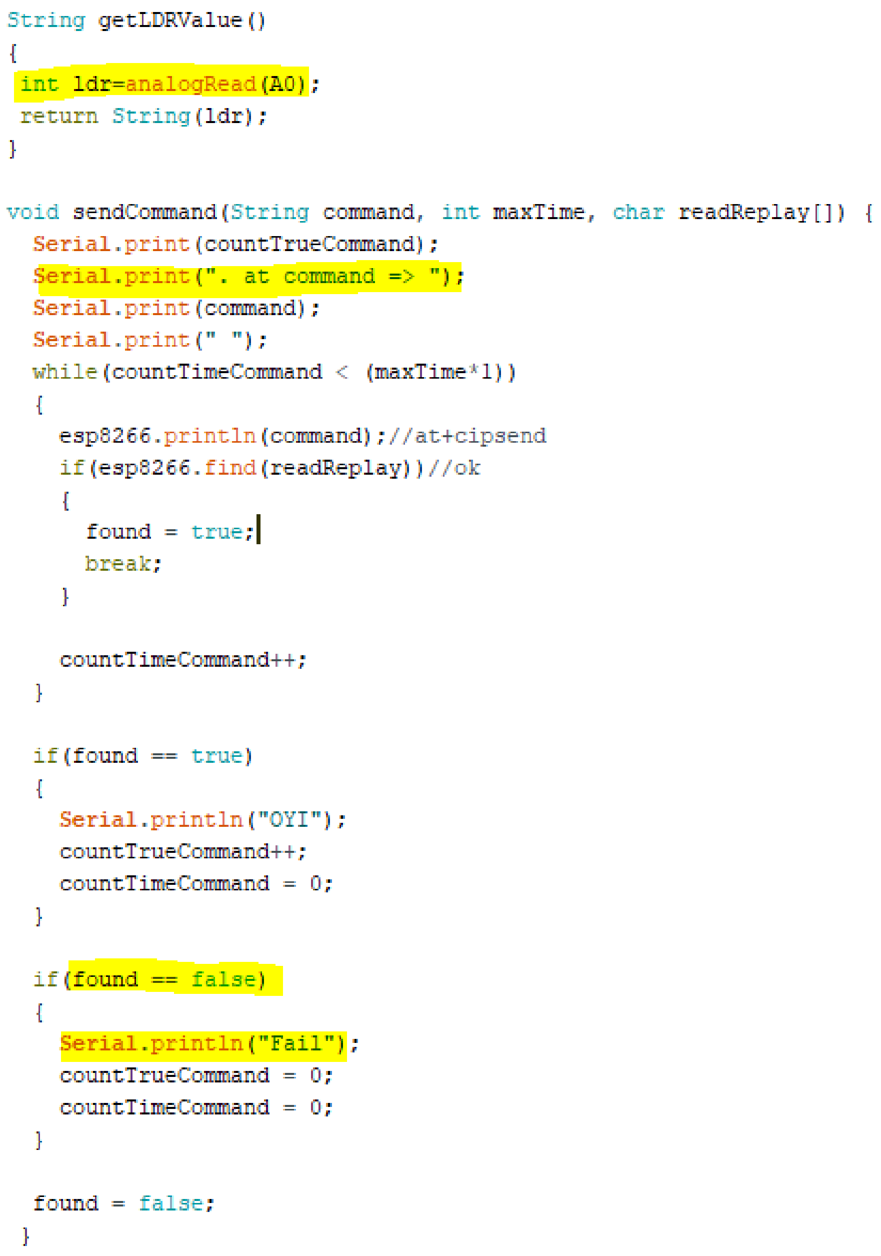
The code then retrieves the value of the water sensor shown in figure 25



```
String getSensorValue()
{
  int sensorLevel;
  sensorLevel=analogRead(A1)/10.24;
  return String(sensor);
}
```

*Figure 27 Value code*

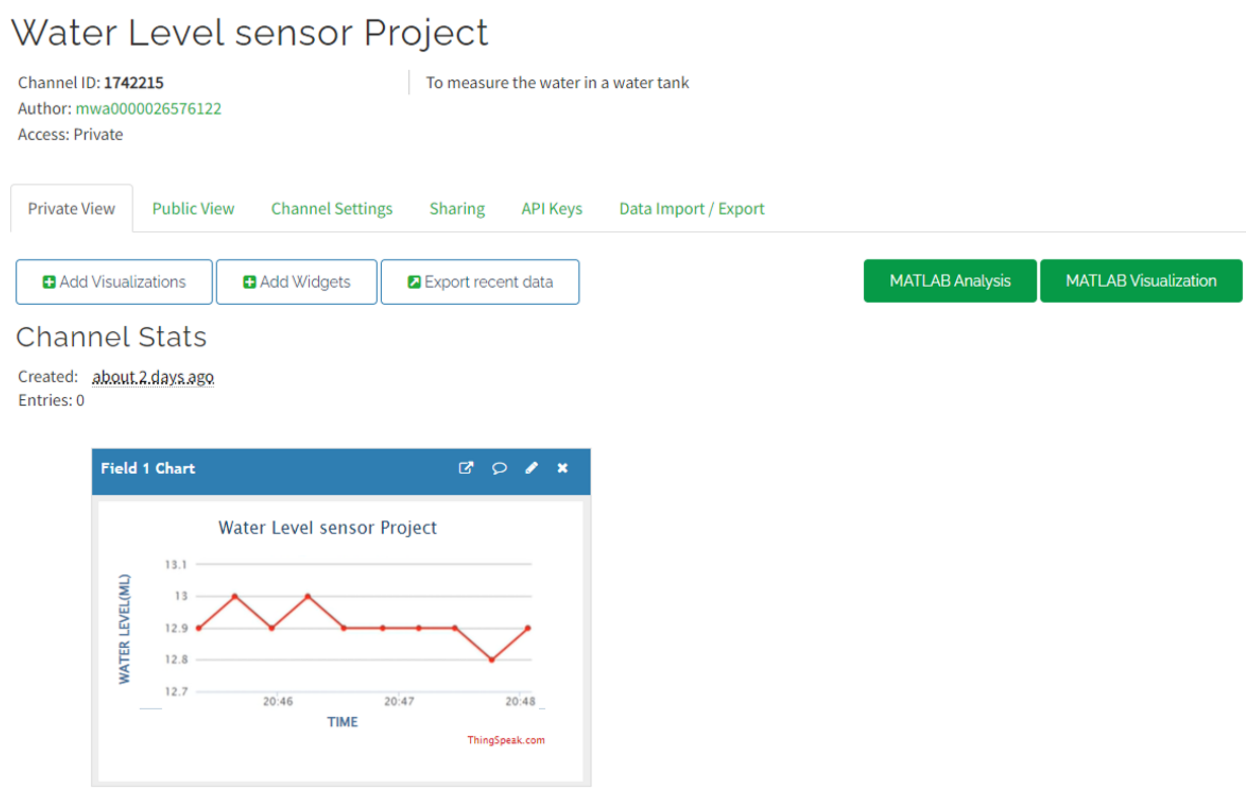
The remainder of the code prints the relevant information to the think speak server. The serial print includes the sensor value and stores it to the think speak server shown in figure 26.



```
String getLDRValue()
{
 int ldr=analogRead(A0);
 return String(ldr);
}

void sendCommand(String command, int maxTime, char readReplay[]) {
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
      found = true;
      break;
    }

    countTimeCommand++;
  }

  if(found == true)
  {
    Serial.println("OYI");
    countTrueCommand++;
    countTimeCommand = 0;
  }

  if(found == false)
  {
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }

  found = false;
 }
```

*Figure 28 Print code for Think speak*

Once the device was connected to the internet using the ESP and connected to the cloud, we were now able to retrieve a simple graph showing the data. This has been attached to the figure 27.


Water Level sensor Project

Channel ID: 1742215
Author: mwa0000026576122
Access: Private

To measure the water in a water tank

Channel Stats

Created: about 2 days ago
Entries: 0

*Figure 29 Cloud Data (Think speak)*

These readings can now be stored on the cloud database for future or even past reference which will assist in making key decision for the proposed design.

## Conclusion

We wanted to learn more about how to use an Arduino and sensors in this project. Not only that, but the collection of these devices must be connected to the internet in order to be connected to a cloud device. This is important not only in industry, but also in the project proposal I wrote to address the needs of a water tanker for developing countries.

Now, using the parts above, you can create a water monitoring tool that can be kept in the cloud and used to visualize data trends. For example, a village may only have one tanker and wants to observe which days and times the residents of that village use the most in order to predict future trends and behaviors in order to maximize their water capacity.

To expand on this work, we would incorporate additional sensors into the design, including a temperature sensor that would allow us to monitor the water temperature as well.

Many issues arose during the development of this design, but the majority of them were caused by malfunctioning sensors. If this is to be produced on a large scale, it is critical that the sensors and devices be maintained on a regular basis, as well as the calibration of the water sensor mentioned in the calibration section.

## References of code and Content

*Arduino - Ultrasonic Sensor. Tutorialspoint.com. (2022). Retrieved 2 April 2022, from https://www.tutorialspoint.com/arduino/arduino_ultrasonic_sensor.htm.*

*Arduino Cloud. Create.arduino.cc. (2022). Retrieved 7 May 2022, from https://create.arduino.cc/example/library/liquidcrystal_1_0_7/liquidcrystal_1_0_7%5Cexamples%5CHelloWorld/HelloWorld/preview.*

*ESP8266 ESP-12F Wifi Module System On A Chip - ESP8266 Shop. ESP8266 Shop. (2022). Retrieved 5 May 2022, from https://esp8266-shop.com/product/esp8266-esp-12f-wifi-module/#:~:text=Overview-,ESP8266%20is%20a%20highly%20integrated%20chip%20designed%20for%20the%20needs,functions%20from%20another%20application%20processor.*

*Parajuli, A. (2022). Water Level Sensor Arduino Tutorial. The IOT Projects. Retrieved 4 April 2022, from https://theiotprojects.com/water-level-sensor-arduino-tutorial/.*

*The Challenge: Clean and Safe Water - Africa.com. Africa.com. (2022). Retrieved 13 May 2022, from https://www.africa.com/clean-fresh-water/.*

*The University of Salford | IoT lab 1 to 7 | Steve Hill | retrieved from January 2022- May 2022.*

*Water Level Sensor | What, How, Where, Benefits, Types - Renke. Renkeer.com. (2022). Retrieved 5 May 2022, from https://www.renkeer.com/water-level-sensor-definition-applications-benefits-types/.*