Using Everyday Physics to Monitor Water Levels

Janav Nagapatla (09), Trae Kuok (17), Reyes Lee (27) 2.10

AKA hACker people

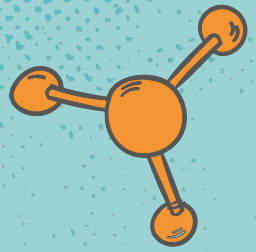
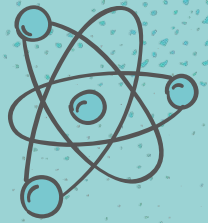
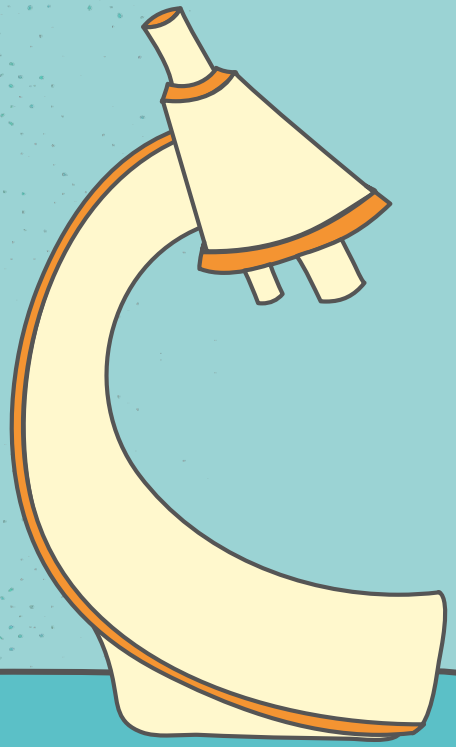


Table of Contents





01

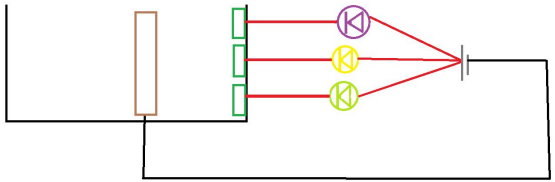
Our Product

Live Demo, Usage, Brief Explanation

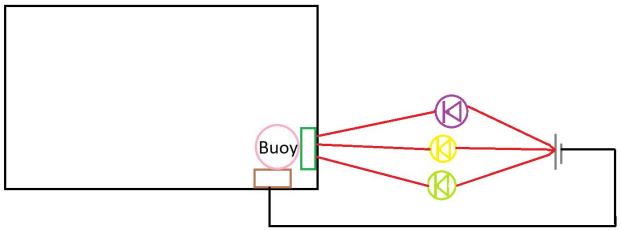
we actually made two products!

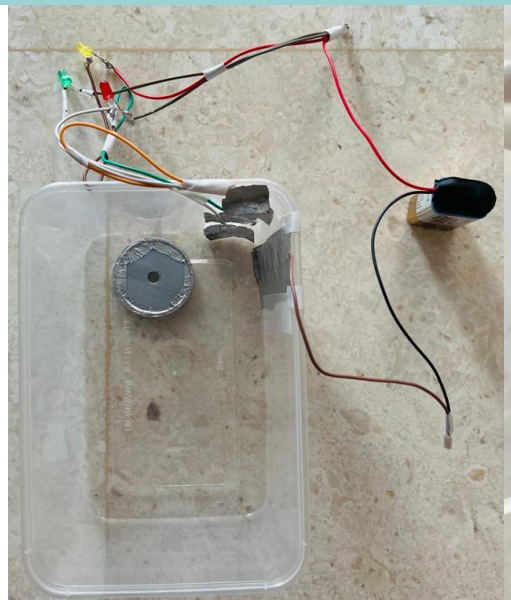
The Initial Plan

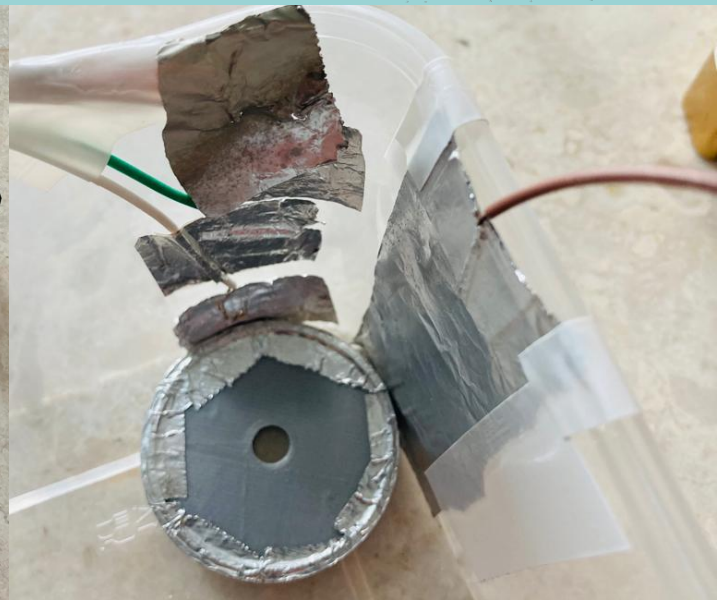
Side Diagram



Top Diagram







Why it sucked: Reliability

1. Soldering issues especially when wet
2. Connection issues
3. The HARD Paradox
4. Buoy sometimes gets stuck under contacts.
5. Precision Limitations: There were three contact “levels”, but it is impractical to add more for any given length.

So we scrapped it...

Our Final Product

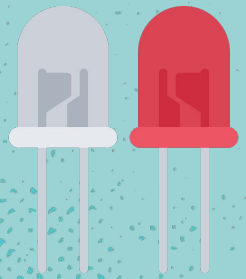
Usage Instructions

Connect to Power

Plug the Arduino into a USB power supply, preferably a computer.

Connect Sensor

Using four wires.



Insert LEDs

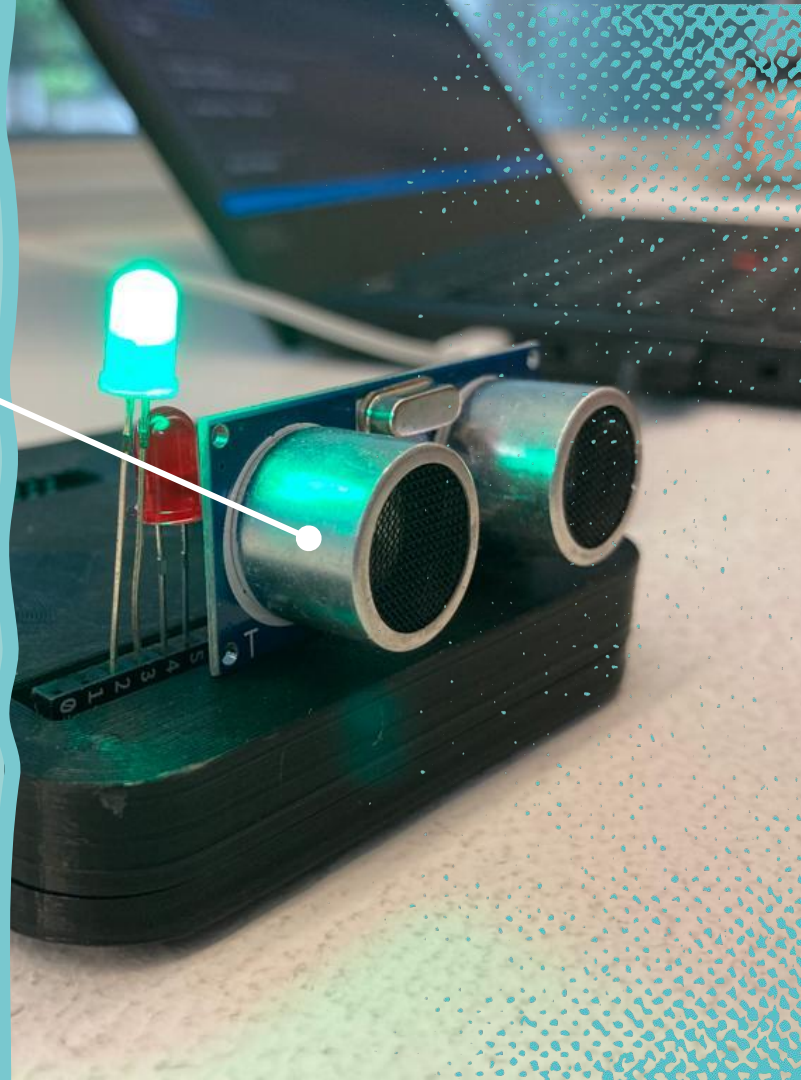
Push the LED pins into designated holes.

Attach Sensor

Just snap it onto the container!

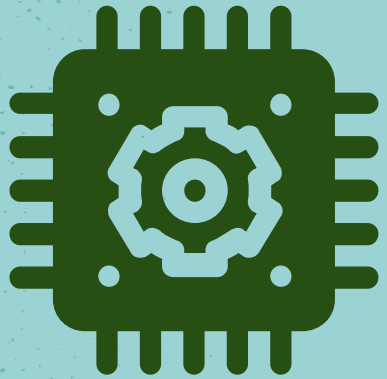
Demo!

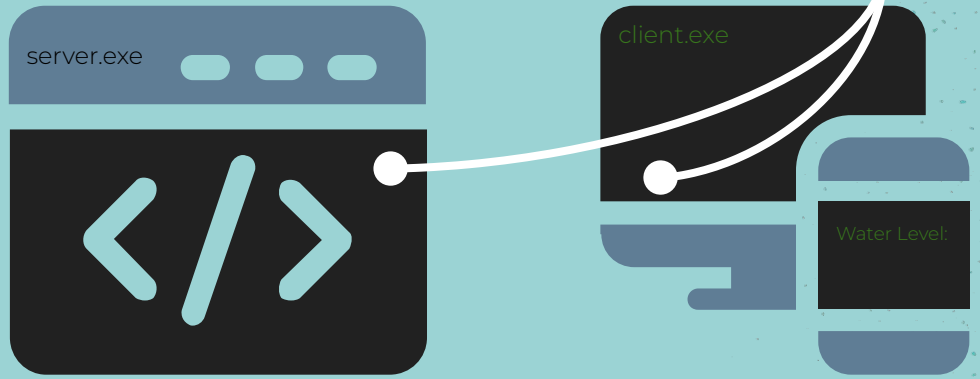
Do look closely at the demonstrators and look for features of the hACker People Water Sensor



How It Works

Totally Optional!
If you feel that these are too troublesome or complicated, simply stick with the on-site system!





Arduino

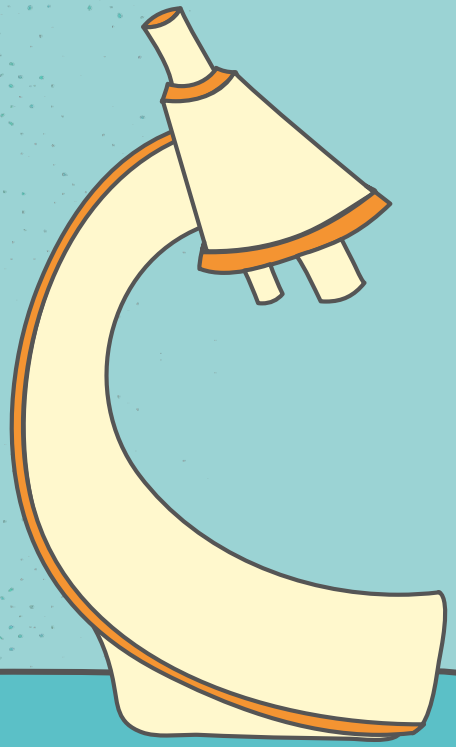
Takes in data from the ultrasonic sensor, computes the distance, and controls the LEDs and Buzzer

Server

Middleman between the Arduino and client

Client

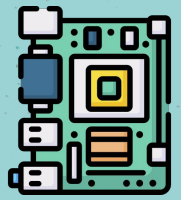
Able to see the data and control the sensor remotely



02 Building

Electronics, Casing, Programming

Electronics Used



Arduino

A microcontroller that accepts input from the HC-SR04 and actuates the buzzer and LEDs

LEDs

LEDs (Light-Emitting Diodes) allow the end-user to visually see data from the HC-SR04

Buzzer

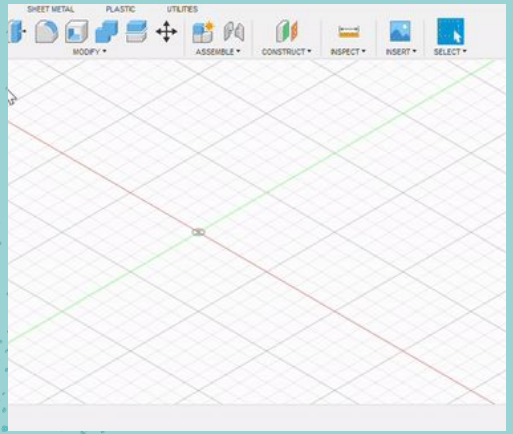
Uses an electromagnet to vibrate a diaphragm, producing sound waves when voltage is applied

HC-SR04

An ultrasonic sensor that when actuated, returns the time taken for a sound to return

Computer-Aided Design

We then made 3D models in software such as Fusion360 and printed them out as encouraged.



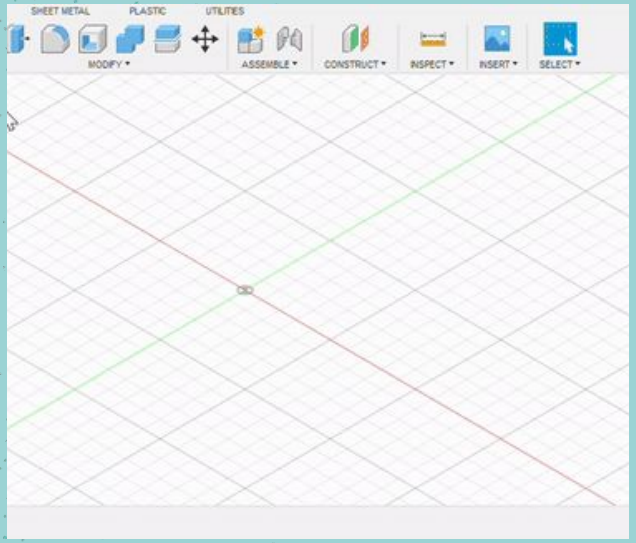
Much better reliability and standardisation

PETG provides better durability

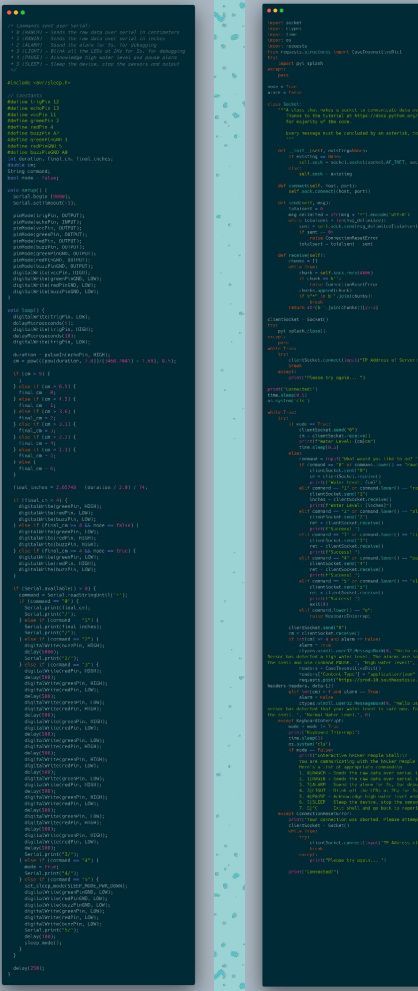
Custom-designed by Trae

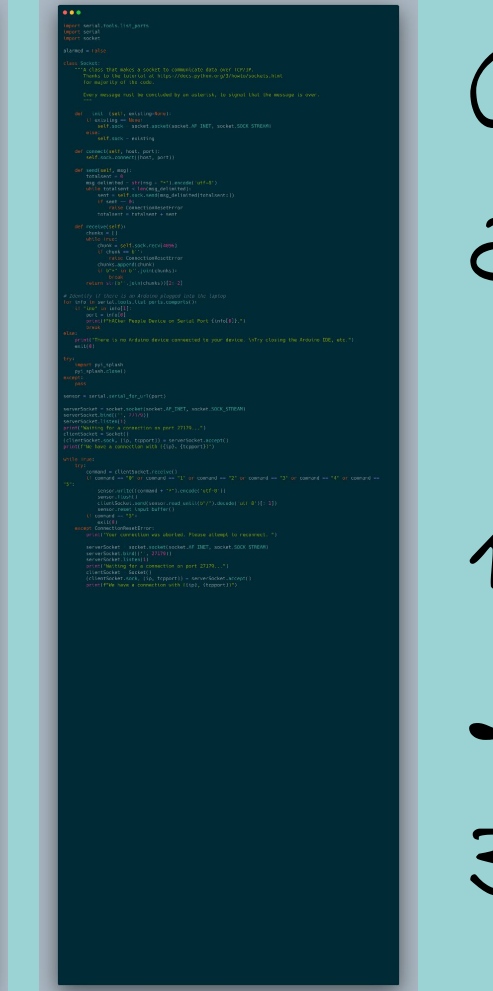
Electronics less exposed

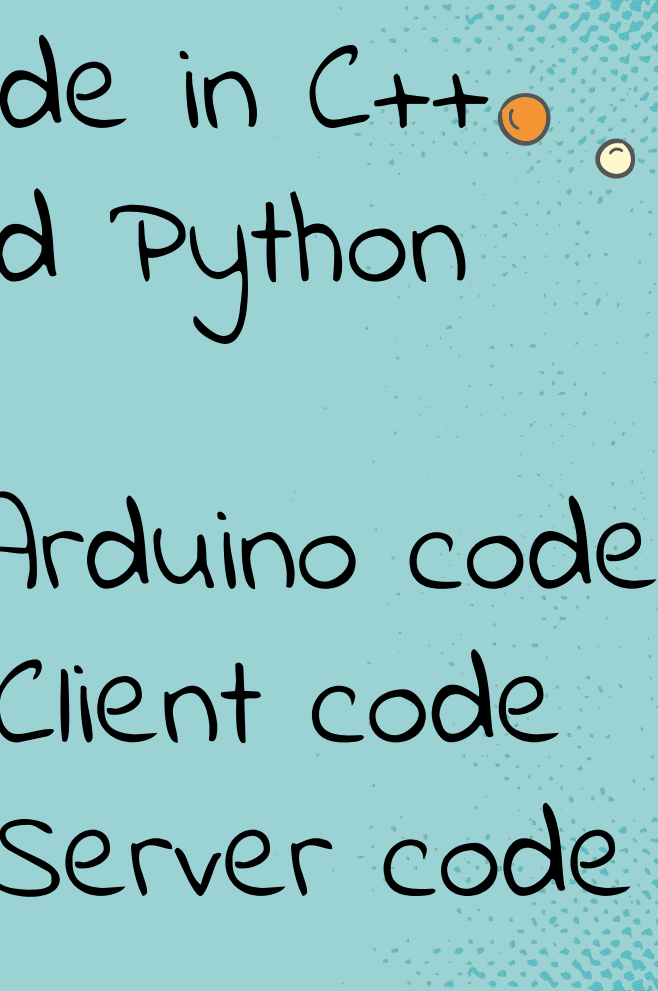
How CADding works



- 2D Drawing
- Extrude to a 3d object
- Vernier caliper → accurately and precisely measure dimensions of objects, such as electronics
- Print with 3d printer using PETG filament





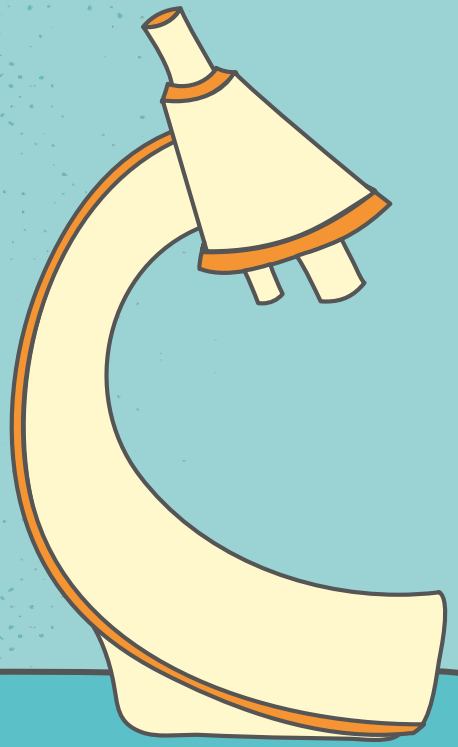


Code in C++ and Python

1: Arduino code

2: Client code

3: Server code



03

Science Concepts

Ultrasound & Electric Communications

Concept No. 1

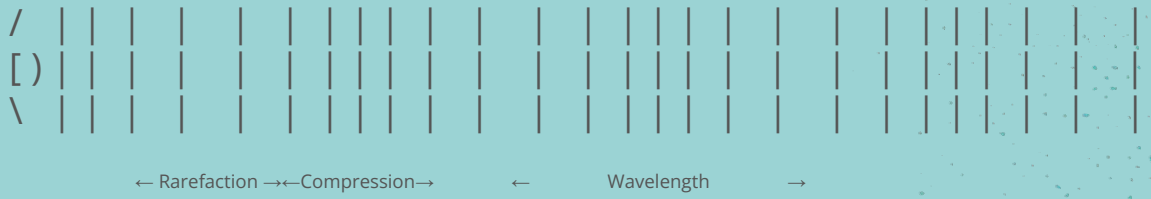
Ultrasound

Using ultrasonic sound waves to measure distance

How Sound works

Generated when an object vibrates and displaces particles surrounding it

Eg. Speaker's diaphragm vibrating when voltage is applied to it. This displaces the surrounding air molecules which creates a longitudinal wave



MORE rarefactions and compressions per unit time, the wavelength is shorter and the pitch of the audio goes up

More particles are displaced by the diaphragm, the amplitude is higher and the volume of the sound is higher

How Ultrasound Works

Humans generally hear between 20Hz and 20000Hz (i.e. 20 waves per second to 20000 waves per second)

Therefore, sounds above 20000Hz are **ultra**sonic and cannot be heard, such as the 40kHz (40000Hz) generated by our ultrasonic sensor.

Special microphone on the ultrasonic sensor can capture these sounds and time how long the reception took, starting from the time the pulses were generated.

The MATH

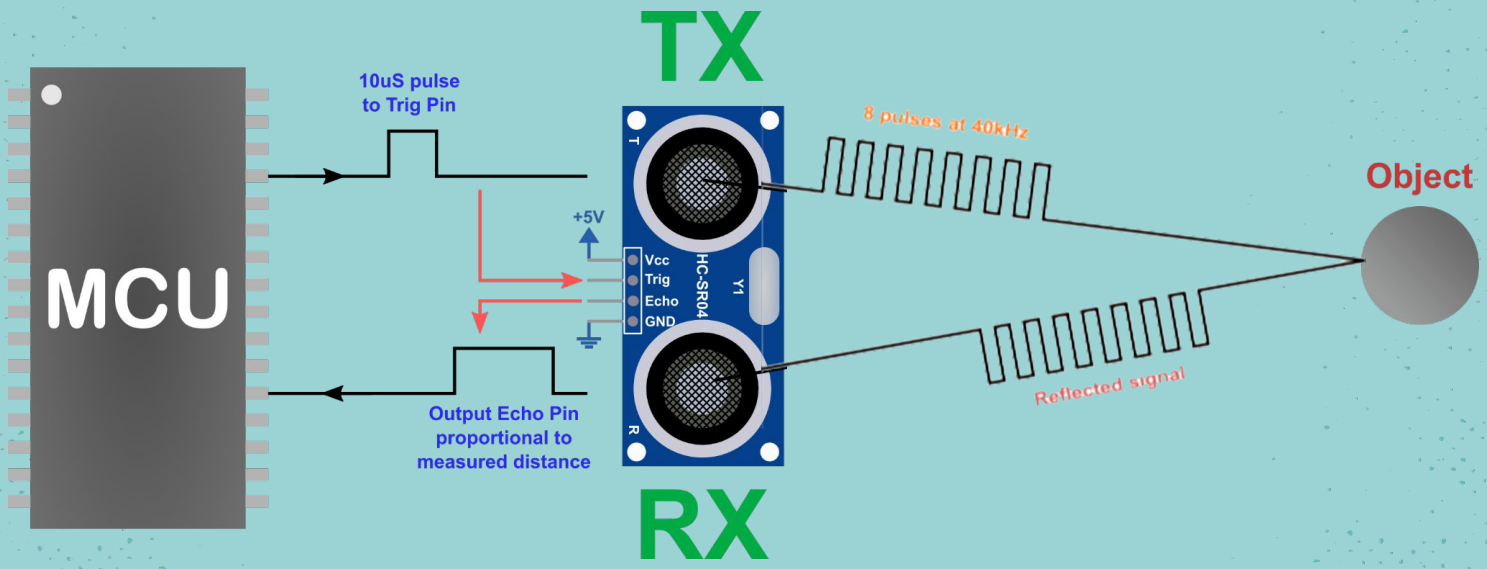
Speed of Sound in Air = **330 (±20) m/s**

Speed is calculated as the distance travelled per unit time,
Ergo, distance is speed multiplied by time taken

Distance travelled by sound wave = 330 m/s × time (seconds)

If a sound wave took **10 s** to reach a wall and reflect back to the source, we can do **330 m/s × 10 s ÷ 2**, as the sound wave has travelled the distance twice, to get **1650m**

Our Implementation



Our Implementation (Simplified)

```
#define trigPin 12
#define echoPin 13
#define greenPin 2
#define redPin 4

int duration, cm;

void setup() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(redPin, OUTPUT);
}

void loop() {
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    cm = 6 - (duration / 2 * 0.33);

    if (cm < 4) {
        digitalWrite(greenPin, HIGH);
        digitalWrite(redPin, LOW);
    } else if (cm >= 4) {
        digitalWrite(greenPin, LOW);
        digitalWrite(redPin, HIGH);
    }
}
```

1. Trigger the sensor
2. Sensor emits 8 40 kHz pulses, receives the reflected pulses
3. It sends a positive voltage pulse as long as the time taken for echo to come back
4. Measure this pulse
5. Calculate the distance in cm.
   We know sound travels in air at ~330m/s
   We have time in μs and want cm so we need speed in cm/μs
   330m/s is 0.033cm/μs, hence we do
   Distance = Duration ÷ 2 × 0.033
6. Having derived this, we use simple logic to colour the Light-Emitting Diodes, and activate the buzzer.

Concept No. 2

Comms

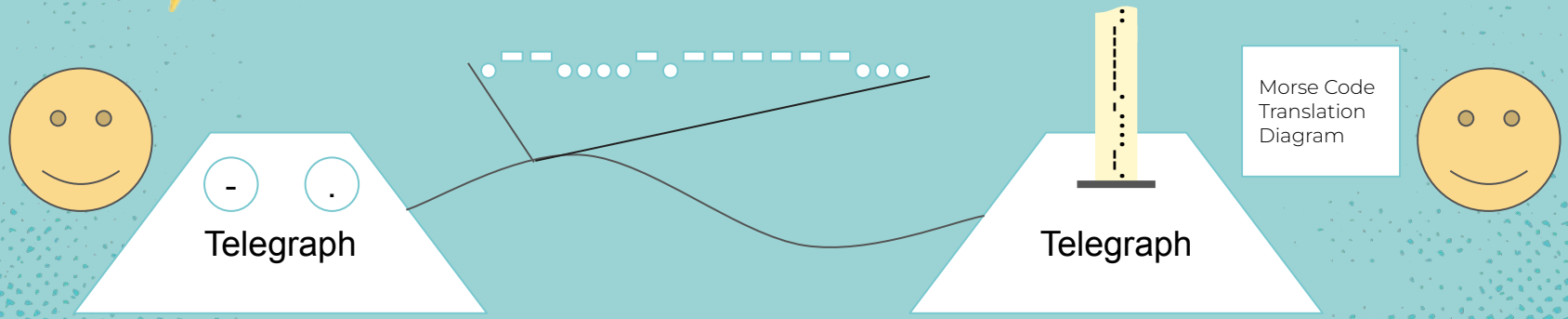
Using energy to communicate data over long distances

Electric Current at its Core

Electric current (usually) = movement of electrons through a wire

If we can read and write electrons at 2 different points, we can send data between these nodes

A good, simple visualisation is the telegraph, which used to be the fastest way to get data across



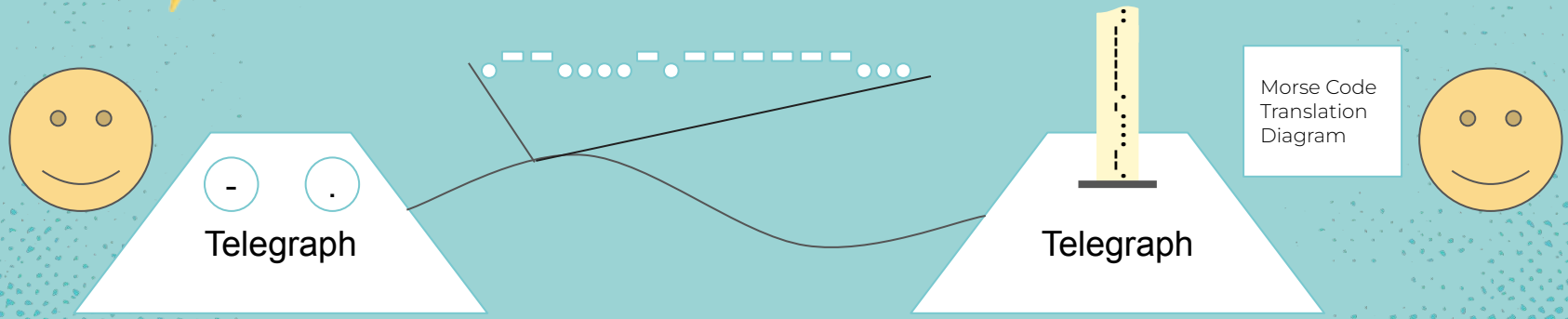
Transmitting data

Unlike ink, electricity has two easily-measurable states - on or off.

Therefore, we use base-two (binary)

A byte represents one number, which means a certain character based on the encoding.

Devices generally use Unicode



Transmitting data - 1

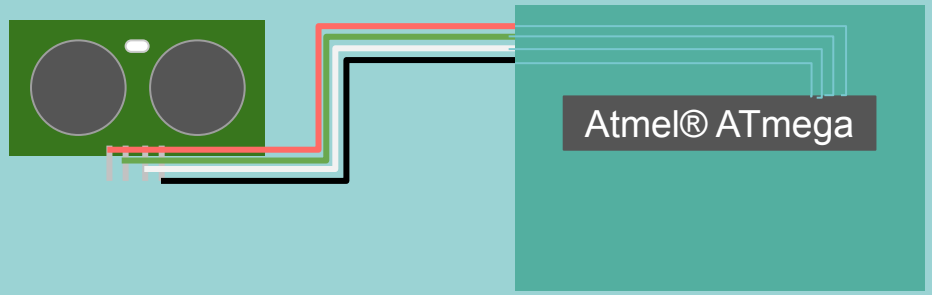
```
#define trigPin 12
#define echoPin 13
int duration;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
}
```

Transferring data from the sensor to the Arduino and vice versa

This is using jumper wires that go between components. These are regular wires. They send data through electric pulses

These pulses can be made by and measured by the board, so we have data.

Data is transferred almost instantaneously.



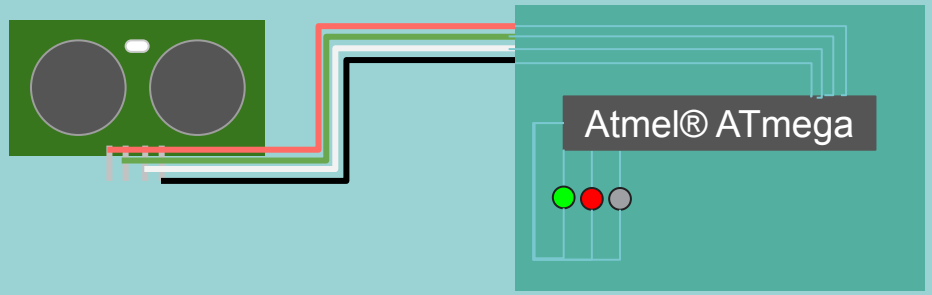
Transmitting data - 2

```
#define greenPin 2
#define redPin 4
#define buzzPin A2

void setup() {
    pinMode(greenPin, OUTPUT);
    pinMode(redPin, OUTPUT);
    pinMode(buzzPin, OUTPUT);
}

void loop() {
    digitalWrite(greenPin, HIGH);
    digitalWrite(redPin, LOW);
    digitalWrite(buzzPin, LOW);
}
```

Next, we must transfer data to the user, through audiovisual data

If the water level is acceptable, the chip sends current to the green LED and none to the buzzer and red LED, and vice versa

When current flows through the devices, sound or light is produced and users get audiovisual data



Transmitting data - 3

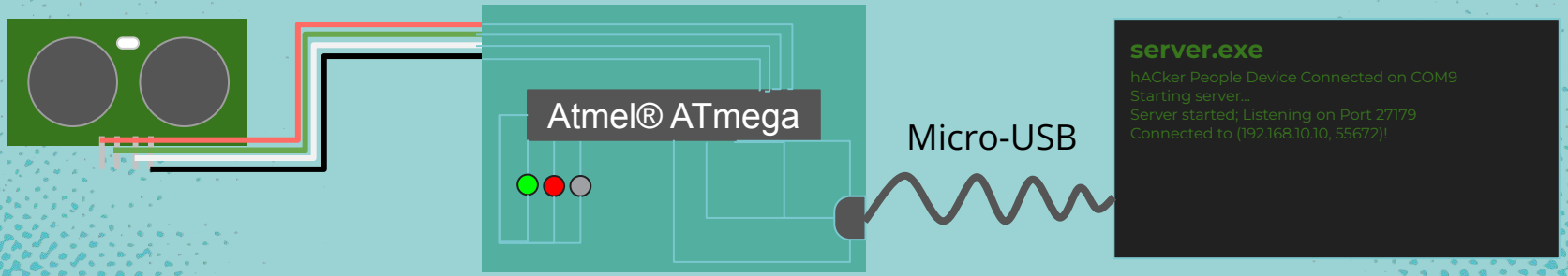
```
import serial

arduino = serial.serial_for_url("COM9")
arduino.write(b"Hello Arduino!")
output = arduino.read_until(b"\n").decode('utf-8')[:-1]
```

Enabling communication between the Arduino and outside world

This is done via the Serial protocol

Data is transferred in binary one bit at a time

A 0 is the lack of an amount of voltage and vice versa. The presences of voltage are decoded from UTF-8 into Unicode.



Transmitting data - 4

```
class Socket:
    """A class that makes a socket to communicate data over TCP/IP.
       Thanks to the tutorial at https://docs.python.org/3/howto/sockets.html
       for majority of the code.

       Every message must be concluded by an asterisk, to signal that the message is over.
    """

    def __init__(self, existing=None):
        if existing == None:
            self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        else:
            self.sock = existing

    def connect(self, host, port):
        self.sock.connect((host, port))

    def send(self, msg):
        totalsent = 0
        msg_delimited = str(msg + "*").encode('utf-8')
        while totalsent < len(msg_delimited):
            sent = self.sock.send(msg_delimited[totalsent:])
            if sent == 0:
                raise ConnectionResetError
            totalsent = totalsent + sent

    def receive(self):
        chunks = []
        while True:
            chunk = self.sock.recv(4096)
            if chunk == b'':
                raise ConnectionResetError
            chunks.append(chunk)
            if b"*" in b''.join(chunks):
                break
        return str(b''.join(chunks))[2:-2]
```

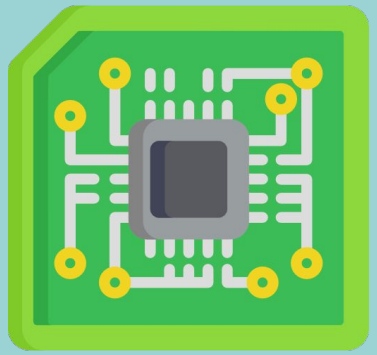
Finally, information needs to be transferred to the client

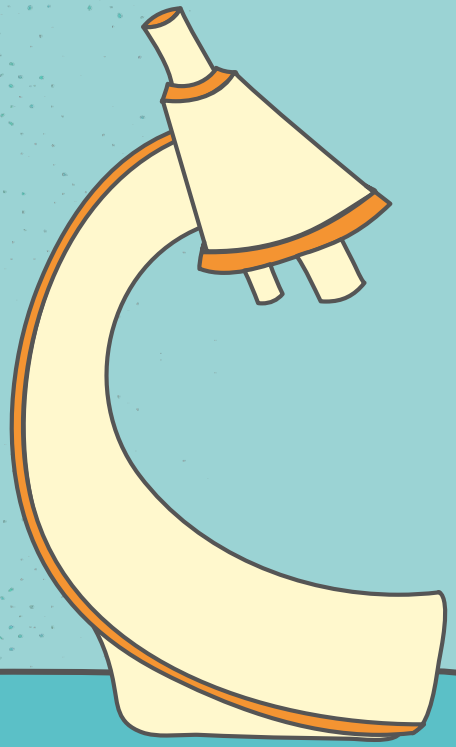
This is done via radio waves 2.4/5 GHz and fiber optic cables till the client is reached.

The antennae of computers convert the binary data into radio waves and vice versa, so data can be transferred without wires over long distances.

So what's the lesson?

Electricity doesn't just power things, it can also be used to communicate data!





04

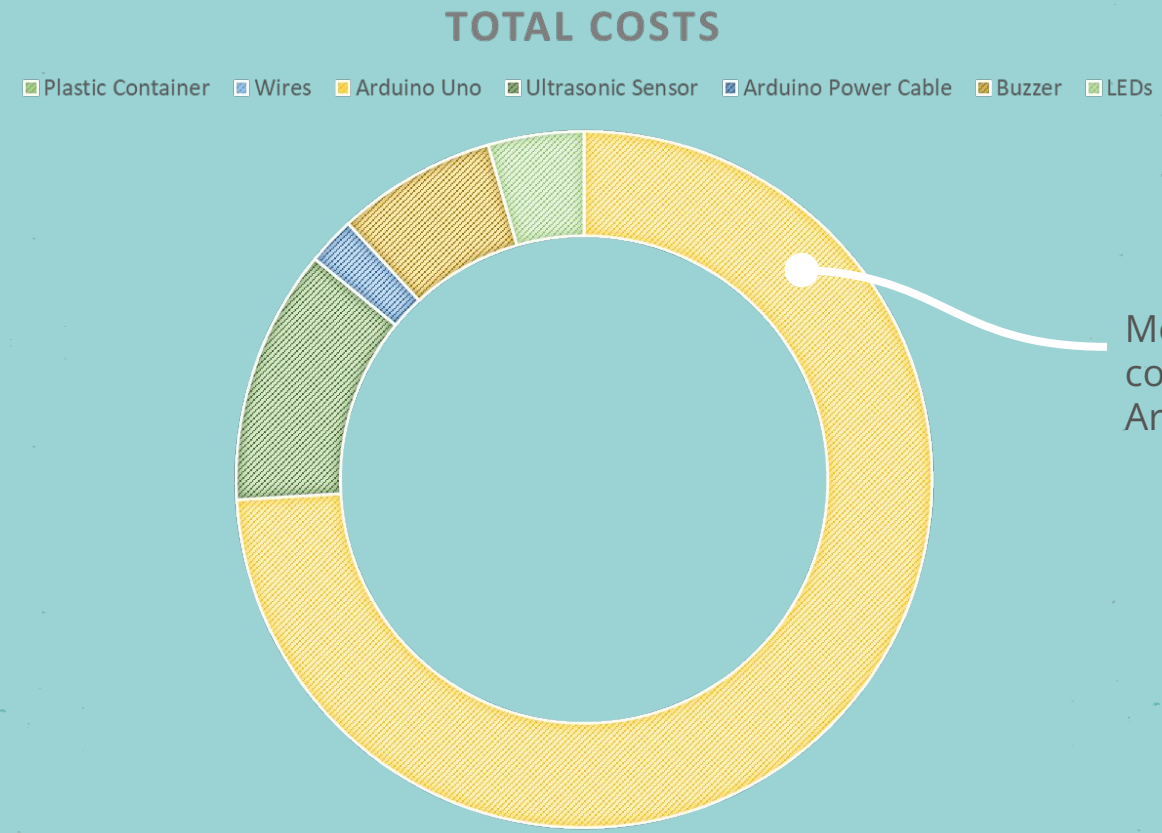
Viability Analysis

Feasibility of our Product

Cost Analysis

Item	Cost	Quantity	Remarks	Subtotal
Plastic Container	$ 0.00	1	Irrelevant cost (Pantry staple)	$ 0.00
Jumper Cables	$ 0.00	4	Reused (From other projects)	$ 0.00
Arduino Uno	$ 10.00	1	Second-Hand (Old Arduino)	$ 10.00
HC-SR04 Sensor	$ 1.60	1	New	$ 1.60
Micro-USB Cable	$ 0.30	1	New	$ 0.30
Buzzer	$ 1.00	1	New	$ 1.00
LEDs	$ 0.30	2	New	$ 0.60
Total	**$**			**13.50**

Cost Analysis



Most of the costs come from the Arduino!

Benefits of our Product

1. Extremely Reliable - Based on our experience and tests, works 100% of the time.
2. Extremely Precise - Distance can be measured to the millimetre.
3. Extremely Accurate - Distance calculated will only deviate by maximum of 3mm.
4. Results/Measurement communicated to user instantly.
5. Product is robust and tough,
6. Every component is **separate**.

- Ease of Maintenance: If the LEDs, Buzzer, Sensor, or even Arduino spoil, just unplug and replace it!
  (Compared to troublesome resoldering/glue/tape)
- Ease of (Mass) Production and Assembly
- Highly Portable and Small (Reassemble onsite)

Disadvantages of product

Although the sensor is meant to be used in wet conditions, the electronics are not able to be used when wet, due to microscopic ions in water causing a short circuit.

If the sensor ever gets wet, disconnect the circuit *IMMEDIATELY*, to prevent a short circuit.

Another con is that users may forget the ports being used in the setup, and a single wire connected wrongly could potentially cause the product to fail.

How we abated the cons

While the electronics may spoil when submerged in water, the 3D printed casing helps to reduce the amount of moisture entering the electronics.

We have composed a list of the correct ports to be used, which is on a paper pasted on the Arduino, should one accidentally connect the circuit wrongly.

Thanks!

Thanks to Trion Projects for many images and computational assistance!
Thank you for your kind attention!

You Ask

We Answer