Architecture Design

In this step of our design process, we will outline the specific components we have chosen based on our System Design, and explain why.

Processor: ATmega328P

Function:

- Used to control the sensors, storing and sending data, calculating water levels based on sensor input.

Why?

- A low power processor would maximise time between battery replacement/ maintenance.

Communication: LoRaWan

Function:

- Connect to a WAN to send data from the sensor/device.

Why?

- When comparing our three main choices - Wi-Fi, 3/4G and LoRaWan - we concluded with LoRaWan due to the following reasons:
    - Using Wi-Fi would require communication to get access to Wi-Fi hosted by organisations, which could prove to be difficult. This would also mean no connection the further out towards rivers we went to.
    - Using 3/4G would have been electronically expensive and signal would vary widely depending on the area and time of day. This would have made it hard to make the data transfers reliable and map.
    - Another reason we chose against 3/4g is due to the cost required in sending data for every device.

    - LoRaWan proved to be a power efficient technology that whilst require some setup, provides great range and signal strength in a mappable area.
    - With LoRaWan, we are using the things network which gives us enough transfer time to send the readings we want each day from our device.

API: Laravel

Function:

- Display data coming from the sensors in a meaningful way to users/ ourselves in the form of graphs.

Why?

- Laravel is an established web framework for PHP projects which two of our project members had used previously in their work placements. This prior experience along with the ease of setting it up are the primary reasons behind our choosing it as our API.

- It also has an easy way to setup support for the JSON API standard which is widely adopted as a standard for building APIs using JSON as a data format.

- We chose a JSON format due to how easy it is to parse and use within a JavaScript Frontend, whereas XML would require more setup.

Database: InfluxDB

Function:

- Used to store data — namely water level readings.

Why?

- When analysing the types of data we would be storing — timestamps and numbers — we agreed that we should select a database that can handle time series well. When comparing different databases including a MySQL database, we saw that InfluxDB is known for this therefore selecting it as our database.

- We also wanted a database that could handle high volumes of data as in theory many flood sensors work together at once providing data that could be condensed down into useful information. InfluxDB could guarantee this too.

Server-Side JavaScript execution: Node.js

Function:

-

Why?

- Firstly, Node.js is quick and easy to run once installed.
- Secondly, it had a supported InfluxDB library that was easy to setup.
- Finally, one of our group members had experience with it which allowed for it to be quickly setup and finished.

Storage: SD Card

Function:

- Will be used to store a log of any errors that might occur.
- Periodic measurements taken will also be stored here firstly as a permanent backup (until cleared by operator), before being transferred to a database.

Why?

- When comparing against a USB flash stick and EEPROM, we found that a USB flash stick isn't commonly used on an arduino as it needs a master device (e.g. a computer), and that EEPROM's memory would limit us in terms of size.
- An SD Card proved to be cheaper to purchase and attach to a breadboard and need be, and there are plenty of libraries to support it.
- This storage proved to be extremely optimal later on when using our development board the Adalogger as it included an SD Card slot ready to use.

Measurement Device: MB7040 I2CXL-MaxSonar-WR Ultrasonic Sensor

Function:

- Provides contactless installation to calculate water levels in specified time periods.

Why?

- When researching similar water level measuring project and after contacting the Environmental Agency, we acknowledged that ultrasonic sensors were widely used for this type of project.
- When comparing ultrasonic sensors to other measurement devices such as pulleys, we realised that whilst it was not the cheapest option, it was certainly the most efficient one with less room for errors. This was a big decider for us as we aimed for a reliable device which would provide us with the most accurate of results. Thus we chose an ultrasonic sensor.
- We chose this particular sensor because it filtered out small targets and only returns the distance to the largest object — which we thought would be convenient in reducing any anomalous values for irrelevant small objects.

Case: Showerproof 3D printed case.

Function:

- Used to store our device including our ultrasonic sensor.
-

Why?

-

Battery:

Function:

-
-

Why?

-

leave for module design:

- Used to attach to a secure base above a river (with the ultrasonic sensor faced down)

- Fairly easily disassembled with a hand clip/screw to remove the SD card or replace batteries leave for module design - sensor:

- Initial depth value is input by the installation operator with first setup to accurately calculate the water levels based on the total distance from the sensor to the bottom of the river (Total distance = initial distance from sensor to water + initial water level depth).

- Sensor sends pulse, and the processor calculates distance traveled based on the time taken to receive the reflected pulse back from the water top.

- Periodic measurements taken, stored locally on the SD card, then sent to the server to be stored in a secure database.

Power supply:

- 9V Lithium battery (10 year smoke detector)

- Performs well in colder weather

- Single use - low power usage

}

Data usage/analysis:

Pseudo code

Calculate water level (Ultrasonic sensor):

```
Const float initialWaterDepthInput;
//initialWaterDepthInput is populated by installation operator with first setup

Const float sensorToRiverBedDistance = sensorToRiverTopDistance() +
initialWaterDepthInput;

float sensorToRiverTopDistance()
{
        sendUltrasonicPulse;
        long timePassed = receivePulse; //miliseconds
        return microsecondsToCentimeters(time);
}


float riverLevelCM()
{
        return sensorToRiverBedDistance - sensorToRiverTopDistance();
}

float microsecondsToCentimeters(long microseconds)
{
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}
```