LiDAR-based Water Level Detector for Smart City Applications

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Daniel Huynh

Spring, 2024

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

N. Rich Nguyen, Department of Computer Science

LiDAR-based Water Level Detector for Smart City Applications

Daniel Huynh
tap7ke@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

As part of the FloodWatch project's mission to develop an early warning detection system for floods, I developed a new water level detection sensor that leverages LiDAR technology, greatly expanding the number of environments and configurations that our sensors can be deployed within, and adding more sources of data for our data collection pipeline. This data is then used by our project to train machine learning models and predict flood risk in regions all across Vietnam, and later, the world. To build this sensor, I used an Arduino Uno board, SX1276 LoRa module, and VL53L0X LiDAR sensor, detecting water level height based on the distance between the surface of the water and the LiDAR sensor, and transmitting the data through the LoRaWAN network to our online servers on The Things Network as well as our own API. After connecting all modules together and writing the software to power the integrated device, I tested the accuracy of LiDAR distance readings and water level detections, and verified the transmission of data payloads from the device to The Things Network server. The tests found promising results on the accuracy of the LiDAR sensor, though to a certain extent, finding that detected water surfaces that are 6 cm or closer to the sensor will lead to greater errors in detection accuracy. Next steps in this project will be in the integration of the device to our API using a webhook, the development of an outer casing for the device for the final prototype, and testing in the field after deployment.

1 INTRODUCTION

One of the most vulnerable regions of the world affected by flooding and climate change, Vietnam has a complex relationship with water, especially with its great traditional dependence on water to support rice farming, fishing, shrimp fishing, and other livelihoods. With the annual wet season comes an increase in flooding, as a result of large amounts of rainfall. This increase in rainfall will cause flooding through two major sources: when rivers overflow, through what is called *fluvial flooding*, or when sewers overflow, flooding inner-city streets. Rainfall will cause a rise in water level in both of these sources, and lead to widespread flooding across the country.

In the context of fluvial flooding, most of Vietnam's population (around half of the population of Vietnam) resides in two major deltas, or wetlands created when a major river deposits sediment into another slower moving body of water. These deltas, as they are formed by rivers, are concentrated spatially around those rivers as well. The Mekong Delta and the Red River Delta, the two deltas in this discussion, are both named after the two major rivers in Vietnam of the same names, of which run through the regions of wetlands that over half the population resides. When either of these two rivers overflow, a massive percentage of the population of Vietnam is at risk for harm due to extensive flooding.

As researchers, we want to leverage new technologies to introduce more and more effective solutions to protect these communities at risk. As part of the FloodWatch team at the University of Virginia, we want to develop an early warning system tool to help detect flooding early and accurately, providing critical time for local communities to prepare and plan how to protect themselves and their community from flood risk. We also hope to expand our platform into an overarching smart city application, utilizing sensors collecting a diverse range of different types of data and machine learning models to predict across multiple domains such as soil quality to support agriculture, air quality, and water quality. However, our main focus is addressing the most pressing environmental issue faced by Vietnam, flooding, which will continue to worsen as climate change gets worse.

2 BACKGROUND

The FloodWatch team is made up of three subteams – the Fullstack team, developing our full stack application software at floodwatch.io, the machine learning team, developing machine learning models that predict flood risk in localized areas based on collected data as well as to detect anomalies from data that is sent in from sensors to remove outliers and possibly incorrect data points, and finally, the Internet of Things (IoT) team, in charge of managing our current network of sensors across Charlottesville, monitoring the network in Vietnam, France, and Germany, and handling the data transmission pipeline. When it comes to the IoT team, we already have a deployed design that was developed by our partners. This design is made from a modified Arduino board called the UCA board, and has a hole at the top to collect rainwater through. It leverages a balance and magnet situated immediately underneath the hole inside of the casing which seesaws back and forth each time 0.03 milliliters of water pass through the sensor. The magnet counts the number of flips that the balance performs, and the board multiplies it by 0.03 milliliters of water. Then, the board prepares the packet for transmission to our server, leveraging a LoRa module, a chip that is able to handle transmission of data with an attached coil antenna.

The main mode of transmission for our devices is the use of the LoRaWAN network, a networking protocol that allows for *long range and wide area* communication, and uses much less power than connection through other methods like WiFi. I will also use this mode of transmission for my LiDAR-based device. Any LoRa node simply has to transmit its readings in all directions, and has to be picked up by a LoRaWAN gateway. A gateway is simply a router for the LoRaWAN network, and may be managed by organizations or users all across the world. After a join request is accepted by the gateway, in which keys are exchanged for encryption of further communication, the LoRa node is connected to the rest of the

network, and can continue to send encrypted messages from the node to the gateway, which reroutes the data to a specified location, in which we want to route our messages to The Things Network server. This is a server that is widely used by thousands of Internet of Things companies worldwide to manage LoRaWAN data transmission. Each user may register a device with The Things Network with a LoRaWAN device's set of keys uniquely generated either by the device manufacturer or manually by The Things Network. These sets of keys will uniquely identify the device. When data is received by a gateway, its keys will identify the user who registered the device, and the data will be routed to the user's The Things Network dashboard. From there, the user can reformat the payloads received by the server, and write code to manage the data collected.

For our team, we reformat the data on The Things Network into a JSON format, and set up an API call to our own platform's exposed external API endpoints, submitting a POST request with the data we receive on The Things Network to our own server, which stores it in our internal database, managed on Amazon's AWS. From there, the Machine Learning team can use the data to train models and predict flood risks from corresponding locations. The Fullstack team is also able to display the data on the application for users to see and interact with.

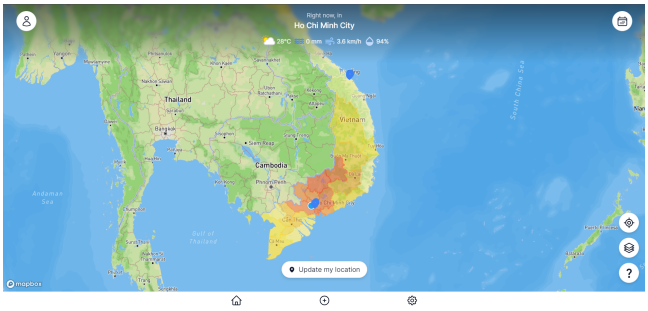

Figure 1: The user interface of our application's website, allowing users to interact with the map to view flood risk and weather data for different regions across the world, monitor sensors, and view live sensor data.

2.1 LiDAR

LiDAR is a technology that uses light to measure distances and map out its surroundings. Taking advantage of the constant value for the speed of light, it uses the *time of flight principle*, as well as basic physics of kinematics to calculate distances of objects that it wants to detect. [2]

The sensor simply shoots out a beam of light from the sensor, and records the amount of time that it takes for the light to have been reflected back into the sensor. How does this allow the device to calculate for distance? Take the formula for average velocity:

$$v = \frac{\Delta x}{\Delta t}$$

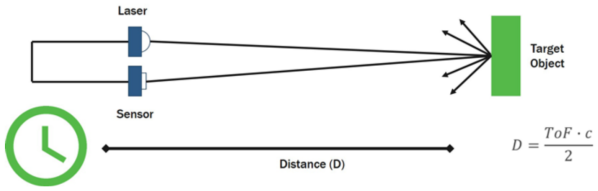

Figure 2: An illustration of how LIDAR sends and receives lasers to calculate distance, of which ToF stands for time of flight, via LIDAR News (https://blog.lidarnews.com/lidar-technology-explained-at-the-electronics-level/).

Since the speed of light is constant, and the sensor is able to measure the amount of time for the light to travel from the sensor to the detected object (and back, but we divide the time by 2 in order to only obtain the amount of time it takes to travel to the object), this means that one can solve for the displacement, which represents the distance to the object. [6]

How may distance measurements therefore be utilized for water level measurements for our project? By periodically measuring a grounding distance for the sensor to the ground, we obtain a base measurement for the height of the sensor to the ground. Using this calibration, for any measurement we obtain thereafter, if this measurement is less than the grounding distance we originally obtained, we know that there must be water on the ground, and can take the water level depth to be the difference between the original grounding distance and the new distance measured. There may also be other objects that may pass through the sensor's line of sight to the ground at times. However, we can perform anomaly detection on sensor readings on our server side to determine the feasibility of detected levels.

3 RELATED WORK

LiDAR, though a fairly older piece of technology, has not seen many applications to smart city applications until recently, when the rise of cyber physical system research empowered by major technological developments in the 21st century. This includes usage of LiDAR for flood detection. In a paper published in *Water Resources Research*, J.D. Paul and others expressed that LiDAR had rarely been used to measure water levels, but can be very viable for doing so. The paper experiments with the accuracy of using LiDAR for determining water levels, and tests its effectiveness based on certain factors, including sensor temperature, measuring distance, incidence angle, and surface roughness. Through experiments, they assert that the best deployment location for LiDAR-based sensors for flood detection will be under bridges and or inclined along river banks, minimizing the effects of all the previous factors. It also identifies the usage of LiDAR flood detection to be cost efficient, with a high energy efficiency. [4]

In addition, a group of researchers led by P. J. Basford have leveraged the use of LoRaWAN transmission-based devices to monitor air quality in Southampton, England, running experiments upon

the efficiency and reliability of LoRaWAN for use in smart city contexts. It was found that it was the most reliable mode of data transmission when balanced with power usage. [1]

Finally, a group of researchers led by Indra Riyanto proposed a flood water monitoring system that integrates the use of a web camera to take images of water levels, processing them to detect flooding, and integrate LiDAR to ensure accuracy and confirm correctness of data, through a combined processing of both sources of information. They are moving in a similar direction to our project, but with their main focus on the usage of web cameras over other detection devices. LiDAR is simply used for correction and corroboration purposes. [5]

4 DESIGN

4.1 Hardware

For our LiDAR-based water level detection sensor, there are four main components that make up the device. They are as follows:

- **Arduino Uno board** - a programmable microcontroller board with a large number of pins with which one can attach multiple different modules onto the board, each with different functionalities, and integrate all of the modules' individual functionalities into an overall device with a single program, using C++ code.
- **SX1276 LoRa Module** - a module that supports LoRaWAN transmission. I had to solder headers and wires onto the module, as well as a coil antenna. This coil antenna is specifically designed for transmission at 915 MHz, which is the standard LoRaWAN frequency in the United States. [3]
- **VL53L0X LiDAR sensor** - a small LiDAR module that receives one-dimensional distance measurements (in a straight line from the sensor). It utilizes a Median Filter on the received times to calculate for distance.
- **Battery**

I connected each module and the battery to the Arduino Uno board with the configuration found in Tables 1 to 3.

4.2 Software/Firmware

The Arduino Uno board can be programmed using the Arduino IDE, an environment that allows you to write, compile, and upload code to a board, written in the language C++. This allows you to define the functionality of the integrated board as a whole, and write code that utilizes the modules in a way that allows interactions between all of the modules, creating a cohesive device in its defined behavior. In an Arduino file, there are two main functions that must be defined – a setup() function that is called exactly once when the device first starts up to set all desired settings, as well as a loop() function, which is called continuously as long as the board is connected to power, and is where the behavior of the board is written.

In order to interact with each of the two specialized modules in the code for the Arduino Uno board, a corresponding library that can interact with each module must be used, with defined functions that can interface your code with the hardware. For the VL53L0X

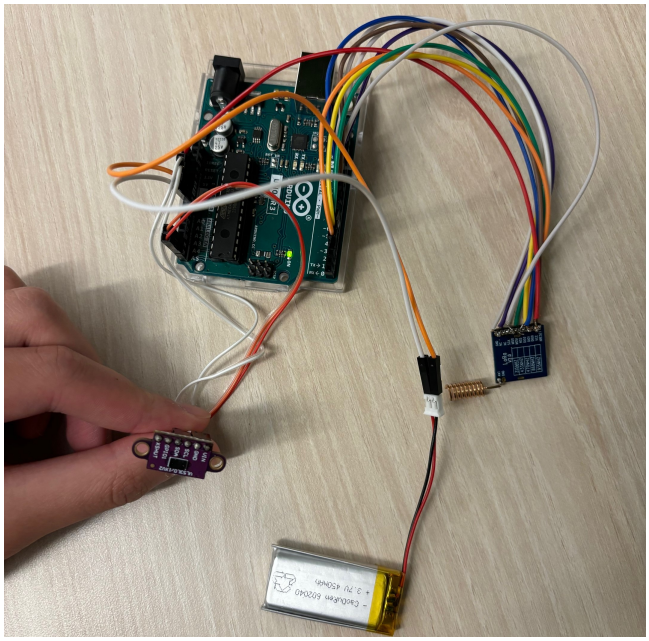

Figure 3: The final setup of the hardware for this sensor device. The LiDAR sensor is the module that is being held, with the battery at the bottom of the screen, LoRaWAN module to the right, and board to the top-middle, in which all modules are connected into.

LiDAR sensor, this is the *VL53L0X* library. For the SX1276 LoRa module, this is the *MCCI LoRaWAN LMIC* library, maintained by IBM.

Using the *VL53L0X* library, we simply set a timeout of 500 milliseconds in the setup() function, meaning that it will abort sensor start up if it is not responsive within 500 milliseconds, and the library's startContinuous() function is called to start the sensor and to keep detecting distances continuously without stopping.

The *MCCI LMIC* library instead supports an event-driven implementation such that it only wants the board to act when it receives a certain event, from a list of events. Events include a joining event, a joined event (when the device has successfully joined the LoRaWAN network), error events including when the packet being transmitted fails to reach the specified destination, as well as a completed transmission event. This was accomplished by writing a function called do_send(), in which I call once right after the setup() is complete, and only after every successful transmission event such that the next packet of detection data should be prepared and sent. Within do_send(), I first check if there is a current transmission or reception job in progress (so that I do not take up the bandwidth in the LoRaWAN module and block currently incoming or outgoing messages). Then, if there is no current job running, I start to prepare the next packet of data, using my reference to the VL53L0X sensor to call readRangeContinuousMillimeters(). This requests the LiDAR sensor to detect a distance reading. Then, if

Table 1: Connection between Arduino Uno and VL53L0X

Arduino Uno	VL53L0X
A4	SDA
A5	SCL
5V	VIN/VCC
GND	GND

Table 2: Connection between Arduino Uno and LoRa SX1276

Arduino Uno	SX1276
GND	GND
D12	MISO
D11	MOSI
D13	SCK
D10	NSS
D9	RST
3.3V	VIN/VCC
D3	DIO0
D4	DIO1
D5	DIO2

Table 3: Connection between Arduino Uno and Battery

Arduino Uno	Battery
VIN	+
GND	-

the sensor's reading is successful, I set the packet payload to be the reading I received from the sensor, and queue the packet to be sent. The packet will then be sent on completion of the previous transmission event (plus an additional delay to prevent sending too many measurements too quickly and draining the battery). This is because the do_send() function is set as the callback function for each sendjob() call performed by the library.

```
void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possible time.
        distance = sensor.readRangeContinuousMillimeters(); // This collects the LiDAR sensor reading
        Serial.println(distance);
        if (sensor.timeoutOccurred()) { Serial.print(" TIMEOUT"); }
        LMIC_setTxData2(1, distance, sizeof(distance), 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}
```

Figure 4: The do_send() function implementation in my Arduino code for the device.

4.3 Overall Behavior of Device

With all of these integrated together, this means that every single time a packet is ready to be sent, a request is sent to the LiDAR sensor to obtain a reading for distance. Once obtained, the reading is set as the payload for a packet, and transmitted by the LoRaWAN SX1276 module into the air to be picked up by nearby gateways. Once picked up and routed to The Things Network server, we can reformat the received packet into JSON to send to our server endpoints. However, before the packet is translated into JSON, we must calculate the water depth using the distance measurement. Each device will have a saved value for the grounded distance that came from calibration of the height of the sensor from the ground. By taking the difference between the original distance and this new reading's distance, we obtain water level depth in millimeters. We can then translate this value to JSON, and then make an API call to our server endpoint with a POST request, storing our data into our AWS database.

5 TESTING

To ensure the functionality of each module is working, there were two separate demonstrations that had to be performed. First, the LiDAR sensor must be able to detect changes in water height. Second, transmissions by the LoRaWAN module must be detected and received by a LoRaWAN gateway.

By simply wiring the LiDAR sensor to an Arduino Uno board, and prompting the sensor to receive continuous readings by placing a call to the readRangeContinuousMillimeters() function, we can get the sensor to record readings without any worry of other

functionality (being placed in the loop() function, it is called immediately as the last reading was received and printed, so I was able to obtain a practically continuous stream of data with a new reading every approximately hundredth of a second). However, to see outputs from the sensor, I connected a USB to USB-B adapter between my laptop and the Arduino Uno board, and by utilizing the Serial Monitor within the Arduino IDE, I can see the messages that are printed out to the serial port from the Arduino Uno board. Therefore, I needed to add statements in the code to also print the received measurements to the serial port each time a new one was received.

After setting up this configuration, I attached the LiDAR sensor to the top of a cup, pointing face down to the bottom of the cup. I gradually poured water into the cup, and watched as the stream of distance measurements declined each time I poured more water into the cup. To test for accuracy, I attached a ruler to the inside of the cup, with ruler precision up to 1 millimeter. Then, I attached the LiDAR sensor connected to the board, which was connected to the laptop through the configuration noted earlier. I then slowly poured water into the cup, noting the height of the water with the ruler, versus the detected water depth (which I calculated by subtracting the original detected distance of the sensor to the bottom of the cup by the new detected distances). See the setup in Figure 5:

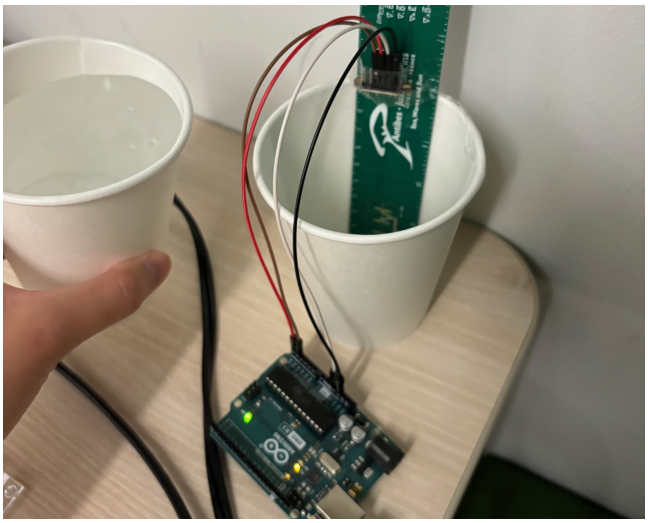

Figure 5: The setup for the LiDAR accuracy experiment. As shown, there is a ruler attached to the inside of the cup to verify ground-truth water height. The LiDAR sensor is attached above the water looking down towards the bottom of the cup.

In addition, in setting up our own LoRaWAN gateway in the presence of my final prototype with the configuration noted in the Design section, I was able to test for the functionality of the LoRaWAN module by checking the Live Data feed on The Things Network, and seeing the physical payloads of the packet being sent.

6 RESULTS

From the first experiment testing for the accuracy of the LiDAR module, we find many interesting discoveries. The first is the accuracy range of the module (which creates limitations to its deployment). In our first experiment, we found the initial height of our sensor being 132 mm, which states that the sensor is 132 mm from the bottom of the cup. As we begin our experiment and obtain measured distances, we will subtract measured distances from this original distance to find the detected depth of the water. The results of this experiment can be found in Table 4.

From these findings, it is revealed that the accuracy of the sensor diminishes as the water level approaches the sensor. At around a water height of 60 mm, or when the water level is around 6 cm from the sensor, the accuracy of the sensor drops, going from an absolute error of around +/- 1-3 to an absolute error of 6 mm. This is a drastic change, relative to the scale at which we are measuring the level of water, in millimeters. This means that in deployment, sensors should not be placed so close to the surface in which they are to be measuring and detecting water heights for, with 6 centimeters recommended as being the absolute minimum distance from the sensor to any surface's typical height, due to the clear increase in error at that distance away. However, the best minimum distance between the sensor and its base surface of measurement should be much greater for best results. In addition, deployment should also take into account other factors that were not tested in this experiment as well, such as external effects as a result from other objects interfering with the beam of light's path, which are further investigated in the study by J.D Paul [4].

Next, I investigated the transmission of payloads from a LoRa SX1276 to The Things Network. By simply connecting the battery to the board, and uploading the software code I wrote to the Arduino Uno board, it started to transmit by itself to the LoRaWAN network, first sending a join request, and on having the join request accepted, transmitting data to the network. I am able to see a live feed of the transmitted data through the Live Data window on The Things Network as seen here in Figure 6:

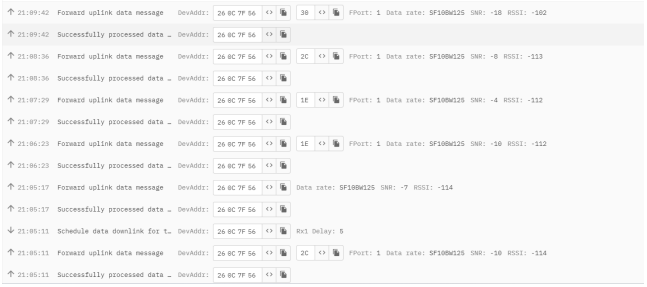

Figure 6: Live payload data received from my registered device on The Things Network's Live Data window.

As you can see, the payloads sent by the device are in bytes, found in the "Forward uplink data message" events after the DevAddr field. These received bytes can be reformatted through a payload

Table 4: Experiment Results: Testing Accuracy of LiDAR sensor

True Water Depth (mm)	Average Detected Water Depth (mm) (Initial Distance Detected - Distance Detected)	Error (mm)
0	132 - 132 = 0	0
12	132 - 119 = 13	1
20	132 - 110 = 22	2
30	132 - 99 = 33	3
41	132 - 89 = 43	2
50	132 - 84 = 48	-2
60	132 - 78 = 54	-6
70	132 - 74 = 58	-12
80	132 - 72 = 60	-20

formatter, such that it can now be displayed and read in a more human-readable form. This can be seen in Figure 7.

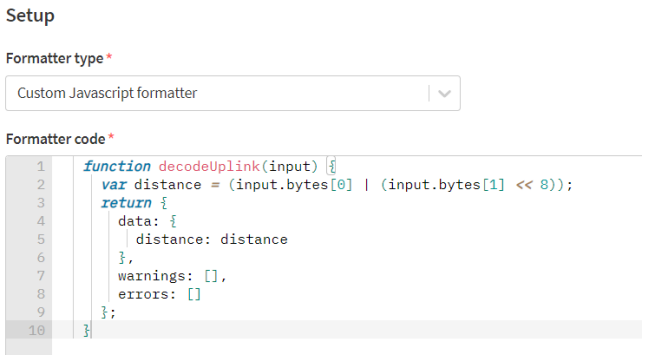

Figure 7: The custom JavaScript payload formatter that I employ in The Things Network server to decode the uplink message payload received from my device.

Figure 8: Updated payload data in the Live Data window after reformatting using the payload formatter tool.

Through using this interface, we are able to ensure that the data that is collected from the LiDAR sensor and transmitted by the LoRaWAN module in our device is successfully transmitted to The Things Network server. This means it can now be submitted to our FloodWatch API through a POST request, and be stored to our database on AWS.

7 CONCLUSION

In conclusion, my development of a LiDAR-based water level detection sensor enables a much more low-cost and energy efficient solution for water level detection in the context of early warning detection systems for floods across the world as part of FloodWatch's overall smart city platform and mission. With such flood monitoring capabilities, communities in vulnerable regions such as Vietnam are able to know, plan, protect, and evacuate their communities and livelihoods ahead of time when a flood is likely to occur, greatly increasing the amount of lives and infrastructure that can be saved when flooding does occur. My device demonstrated very promising results in terms of accuracy, though there were some limitations that can be found when in close proximity to water surface. However, testing showed and confirmed the functionality and reliability of the sensor to detecting changes in water levels, and the ability to transmit this data to a cloud server on The Things Network, as well as the ability to therefore forward data to our internal databases through our FloodWatch API.

8 FUTURE WORK

Moving forward, I hope to integrate the sensor with the rest of the data collection pipeline that our project has in place, joining the ranks of the rain gauge devices and cameras we currently use to collect data for our platform. This includes updating the handling of uplinks to send to our FloodWatch API, through the creation of a webhook. By doing this, we will further our efforts towards a multi-modal data collection pipeline and system. It will take more testing for accuracy in a wider range of conditions and environments (outside of a simple ideal controlled environment experiment), and test deployment in the field. I will also have to design and create an outer casing to protect and enclose the hardware of the sensor when deployed into the public. Finally, with our project's aspirations to turn our sensors into DIY kits for students and others interested in citizen science opportunities, allowing them to explore building their own IoT devices for use in a real-world smart city network, I must develop a streamlined and user-friendly development process and tutorial that will ease the development of these sensors, and allow for as many people without a background in computer science or engineering to participate in building new sensors. This will likely involve the designing of my own PCB file that interfaces with the SX1276 module to remove the need for soldering, a written and recorded tutorial, and outer casing to enclose the hardware.

REFERENCES

[1] Bulot F. M. J. Apetroaie-Cristea M. Cox S. J. Ossont S. J. Basford, P. J. 2020. LoRaWAN for Smart City IoT Deployments: A Long Term Evaluation. (2020).

[2] Lidar Technology Explained at the Electronics Level. (2021). LIDAR News. [n. d.]. Retrieved from https://blog.lidarnews.com/lidar-technology-explained-at-the-electronics-level/.

[3] David Meaney. 2024. LORA AND LORAWAN TIMING. *ECS Inc International* (2024).

[4] Buytaert W. Sah N. Paul, J. D. 2020. A technical evaluation of lidar-based measurement of river water levels. *Water Resources Research* (2020).

[5] Margatama L. Ariawan A. Bayuaji L.-Rizkinia M. Sudiana D. Sudibyo H. Sumantyo J. Riyanto, I. 2019. Web Camera Sensor Coupled with Lidar Data Flood Map for Flood Warning System. *IEEE International Geoscience and Remote Sensing Symposium* (2019).

[6] U. Wandinger. 2005. *Lidar: range-resolved optical remote sensing of the atmosphere.* Springer, Chapter 1: Introduction to lidar.