Dawson Road Maintenance Weather Station Proof-of-Concept Final Report

Authors:

Doug Walch

Alex Frison

Problem

Winter road maintenance is a large industry in BC and making sure crews have the correct information so that they can apply the appropriate material to road surfaces at the correct temperature is critical for not only public safety but also so the business limits potential wastefulness of applying the incorrect material to road surfaces. Currently crews self-evaluate by looking at weather forecasts and judgements based on personal experience with the current and future weather forecasts. This can be a problem when the incorrect material is applied to the surface leading to not only a waste of material but also a potential safety hazard. The current critical problem lies around –8C when the de-icing solution begins to lose effectiveness and starts to freeze. The crews need a set of devices to better equip themselves to make informed decisions regarding road material application.

Microcontrollers

When it comes to IoT devices, there are 2 types of devices that hold much of the market, The raspberry pi and Arduino. Raspberry pi's are more like a lightweight computer, while Arduino has a wide variety of microcontrollers for many different purposes.

Raspberry Pi

Our initial design we chose to use raspberry Pis for both the data gathering and data handling. After doing some more research into power consumption and analyzing the use of a Pi when deployed, we decided to shift towards using the Pi as the reception point for data handling and to use an Arduino Uno for the data gathering.

Arduino Uno

We decided on the Uno for our proof of concept for multiple reasons. First was ease of access, we had ready access to a multitude of Uno microcontrollers and that allowed us to work simultaneously on different parts of the project. Second was the Uno's wide adaptability as we are looking at building a proof of concept and picking a more general-purpose microcontroller allows us to be very flexible as the project evolves through our research and analysis. Finally, the power consumption of an Arduino is far lower compared to that of a Pi, requiring about one fifth of the power. This will allow the device to be deployed with little need to replace the power source compared to the Pi which will need either a far larger battery pack or more consistent maintenance to ensure data collection is not interrupted from loss of power.

Temperature Sensors

When looking at the different types of temperature sensors, we quickly decided on the use of infrared temperature sensors as they are the only way to read the temperature of something the sensor is pointed at. We did a lot of research into finding an appropriate sensor that would be able to both operate and collect the data in freezing cold temperatures, but it also had to fit the use with a microcontroller. The MLX90614 infrared temperature sensor was ultimately decided as we found it fit our requirements the best; it has the lowest potential object temperature detection possible, which allows the system to be deployed across a potentially wide distance of environments.

Name	MLX90614	MLX90632	BMH06203-32
Object Temperature Range	-40C to +125C (Ambient) -70C to +300C (Object)	-20C to +100C	-40 to +300C (Industrial)
Operational Temperature Range	Unlisted (Tested in controlled environment down to -15C)	-20C to +85C	-40C to +85C
Accuracy	± 0.5C (In wide range)	± 0.2C (Between +35C to +42C)	-3% to +3% (Depending on set frequency)

Data Communication

After collecting the data, the next problem for the device is how we are going to retrieve the data. With IoT devices, there are several ways to transmit data; the internet, Bluetooth, radio frequency or direct connection. When it comes to an internet connection, there are a few ways to connect an IoT device to the internet, whether through Wi-Fi or a direct connection, or for more remote where neither option is available, a cellular connection can be used to connect to the 3G or 4G network. Due to the potential remoteness some of the devices may be deployed in, this is a concern and we decided to look at the other options. Bluetooth was immediately crossed out due to its short range, this was infeasible for the project. Direct connection was similarly removed as a viable option as it would require a staff member to physically visit each data collection device and plug in a computer of some type to retrieve the data. This left us with using a radio frequency transmitter and receiver to handle the data.

Radio

We then began researching the different types of radio frequency transmission and quickly stumbled upon the LoRa, long range, radio communication technique and several LoRa radio modules that operate on the license-free ISM band in North America, 915MHz. They also seemingly have a good transmission range of up to 2km, depending on factors such as obstructions, strength of signal, line of sight, etc... The specific modules we first attempted to use were the RFM95W LoRa modules, and this began the first struggles of our project in learning how to work with hardware.

RFM95W

After acquiring the RFM95W modules, we acquired a soldering tool and some solder, we soldered pins to the module to allow us to begin connecting them to the raspberry Pi and Arduino Uno. Approximately two weeks were spent on intense troubleshooting to get the radio modules working. We reached out to the vendor to seek help and they were unable to assist us with our problem. We searched for and used many different libraries in case the issue of the radios not working was related to a library that is no longer supported, and after attempting many different libraries with no luck, we contemplated briefly writing our own library before deciding that it would take far too long to learn the specifics of the module needed to write a library for it. We ordered additional modules and soldered them in case the modules we first received had some kind of malfunction before we acquired them. After we spent another week and a half on these new modules with the same problems, we ordered a different pair of radio transceivers, the RFM69HCW, that have a slightly lower range as we were focused on trying to create our proof of concept and judged that the loss of range for a working proof of concept was an acceptable tradeoff.

RF69HCW

We then spent roughly 3 weeks working with these new modules and having learned some more about how sensitive radio transceivers could be, we contacted someone in TRU's physics lab to gain access to their temperature-controlled soldering pens to ensure we didn't potentially accidentally overheat and fry the sensitive electronics in the module. During these weeks we also acquired a third backup plan from another Professor at TRU who loaned us their xbee S2 radio transceiver modules that they used for a previous project.

Xbee S2

The Xbee S2 modules presented their own set of problems, namely their range and the requirement of special software to configure them for use. While the range can't be changed, the easiest way to connect to this software was with the company's Explorer Boards. After ordering and waiting for them to arrive, time was spent in reading and learning how we needed to configure the modules so they would work as we needed them to. Eventually we were successful in getting them to communicate with each other through the explorer boards and then also when both modules were connected to the respective Raspberry Pi and Arduino Uno boards. With this step complete, we were finally able to assemble the other working parts we had finished and begin working towards testing the device in a controlled environment to prepare to present our working proof of concept to our client.

Proof of Concept

Our first proof of concept that we showed off to our client consisted of the Arduino Uno, the temperature sensor, and an LCD display to more easily display the temperature readings, all connected to a battery pack. Our final proof of concept consists of two devices, a Raspberry Pi4 connected to one of the xbee S2 modules, this will be our master data handler, it receives any data sent out from the Arduino data collector, saves it to a csv file and then after some time has passed, emails that file for crews to use the data to help them make informed decisions. The Arduino data collector is connected to the MLX90614 temperature sensor, the xbee S2 module, and currently for the proof of concept, a small battery pack. For testing, the Arduino works by pulling object and ambient air temperature from the sensor every 10 seconds and then transmitting over serial through the xbee.

Controlled Tests

In our controlled tests so far, the Arduino device can work consistently in –15C temperatures while stored in a freezer for long extended periods of time, 24+ hours. The first test had both devices sitting next to each other, but the Raspberry Pi has problems operating for extended periods of time in freezing temperatures and occasionally shuts itself off before turning back on after 8-10 hours of running. The next tests were run by keeping the Pi itself outside of the freezer, and then using longer wires to leave the xbee module inside the freezer to ensure it would still be able to receive the data from the Arduino.

Appendices