CS461 CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 14, 2017- FALL TERM

DEVELOP AN INTERNET OF THINGS IRRIGATION VALVE

PREPARED FOR

OSU |OPENLY PUBLISHED ENVIRONMENTAL SENSING LAB (OPENS)

CHET UDELL

PREPARED BY

PETER DORICH

CONTENTS

1 Part 1: Micro-Controllers

1.1 Overview

In order to complete this project, hardware and software tools must be chosen with precision. The goal of this section is to compare and contrast three different types of micro-controllers that could potentially be used in this project. The micro-controllers are required in order to physically control the flow of information and program our system. Without any sort of micro-controller, the soil moisture data won't get where it needs to go, and commands from the web application won't reach the irrigation valve. For this document, in order to continue to discuss relevant information, I will not be discussing any previous systems that the OPEnS lab may operate that may have certain technology requirements. For example, all of the technology was required by our client in order to be compatible with previous systems. For this document, I will assume that any of the three choices are viable options for this project.

1.2 Criteria

As noted, it is required that this micro-controller can fully interact with the SDI-12 library, which is the library used and required by the soil moisture sensor. In addition, the micro-controller must be able to support an Ethernet shield, in order to connect to the world wide web. The controller must also be compatible with some type of communications transmitter and receiver in order to reach long distances with no wires. There are no size or weight requirements for this device. It should be noted that the following three micro-controllers are offered with each of the three potential communications transmitter/receiver discussed in Part 2.

1.3 Potential Choices

1.3.1 Adafruit

The first option is the Adafruit control board. This board is shipped with all three communications options. The Adafruit micro-controller uses the Arduino open source programming language based on C/C++. This language allows conditionals to be programmed onto the controller, which we can use to design the flow of information in our system. The Adafruit board comes with a huge variety of pinouts for micro-controller accessories. There are 20 logic pins on this board, allowing for plenty of devices to be connected. Two of the pinouts are serial connections as well. What makes the Adafruit a good candidate is the versatility. There are a huge amount of accessories that can be attached, making future development very flexible.

1.3.2 Raspberry Pi

The Raspberry Pi is a very small computer. The Raspberry Pi is not a micro-controller, it has the ability to perform much more complex tasks. It can, however, be used as a simple micro-controller. The Raspberry Pi is shipped with a full operating system on it, which may or may not be useful for our project. What is useful, however, is that the Raspberry Pi board can be loaded with any software of our choosing. The Raspberry Pi supports Ethernet shields, antennas for communication, and our potential communications protocols. Even though the Raspberry Pi is a full computer, it is by no means large in size, and can fit in the palm of your hand.

1.3.3 SparkFun ESP 8266

The SparkFun ESP 8266 is a micro-controller that is similar to the Adafruit. This board has the options for different accessories such as Ethernet shields, radio receivers, antennas, and more. This board has 10 input output pins. This board comes with no headers attached, like the Adafruit board. The headers need to be soldered on in order to connect to an extra board, like a power supply. The SparkFun is very similar to other micro-controller boards.

1.4 Discussion

As far as our criteria goes, all three of these options are acceptable. They all can load the SDI-12 library, allowing connection to the soil moisture sensors. All three of them can connect to an Ethernet shield for world wide web access. In addition, all three of them can support transmitters and receivers for the three potential communications options. That being said, not all of them are ideal. The Raspberry Pi is the weakest link in this section. Even though it can work for our purposes, it would be much more difficult to design. Research says that it would be improbable to use the Arduino language to program a Raspberry Pi from scratch to work as our micro-controller, since the device itself is an actual computer with more moving parts than one isolated micro-controller. The Adafruit and SparkFun boards are designed to run custom Arduino language software for short, repetitive tasks. The Raspberry Pi would be a better option for a system with more complex tasks that the Adafruit cannot handle. The SparkFun and the Adafruit are very similar in terms of micro-controller boards, however, the Adafruit comes with more pinouts, and also comes with the LoRa radio built-in.

1.5 Conclusion

In conclusion, all three of these choices would potentially work. Though, the Raspberry Pi is too unnecessarily powerful for our purposes, which actually would make development more difficult. The Adafruit and the SparkFun are very similar, but the Adafruit simply has more pinouts and comes in options that have built-in communications devices in them, such as the Adafruit 32u4 with LoRa Radio Module. In addition, the online resources supplied by Adafruit and Arduino are much better than SparkFun. This doesn't relate to the technology directly, but it is an important factor that developers must consider when choosing a tool. A more powerful tool is useless unless you know how to use it.

2 PART 2: COMMUNICATIONS

2.1 Overview

Now that we have the micro-controllers chosen, we need to figure out what they will use to communicate. The wireless hub, with an Adafruit connected to it, must communicate with another Adafruit far away. Formal distance requirements have not been discussed, but we can assume a distance of at least a mile for this discussion. For this reason, the system can't utilize Bluetooth or Wi-Fi due to distance and location, respectively.

2.2 Criteria

As noted, the communications device must connect two micro-controllers from at least a mile away. These devices must be able to transfer data back and forth. The data will consist of very small exchanges, such as a few integer values at a time that will signal conditionals on the other end. No video, pictures, or large files will be transmitted. It is important that the communications is somewhat fast, as we do not want to be viewing old soil-moisture data. The communications device also must not be too much for the Adafruit in terms of overhead.

2.3 Potential Choices

2.3.1 LoRa

The first is Long Range Radio, known as LoRa. This communications receiver and transmitter is built into the Adafruit 32u4, and a full Arduino Library is included. The device would transmit at 915MHz, making it much more powerful that Bluetooth or Wi-Fi. LoRa can reach distances of up to about 13 miles, making it ideal for our system. LoRa is also unique in that it takes very little power to operate. The radio can run off of battery for a matter of years, and can stay in remote places. This project has no security requirements, but it is important to note that LoRa comes with multiple encryption schemes that will help ensure data integrity. Because LoRa uses radio frequency, it is not equipped to send large files back and forth.

2.3.2 SigFox

SigFox is much different than LoRa in a lot of ways. It still uses radio frequency, but transmits the data much differently. It also is primarily for uploading data, not downloading. A two-way connection is still possible, but the download link needs to be separated. SigFox works by exchanging small messages very slowly. It can be as slow as the average modem back in the 1990's, 300 baud, which is symbols/pulses per second. This slow message speed allows for longer distances. Like LoRa, this uses very little power, allowing batteries to operate it for years at a time.

2.3.3 LTE-M

LTE-M is a way to connect an IoT device to a 4g LTE network. This would potentially allow the soil-moisture sensors to directly upload the information to the world wide web before sending to the wireless hub first. This could change the project vastly, but LTE-M has some issues. First, it is designed for larger amounts of data. That being said, the overhead that comes with this service is too much for the Adafruit. Also, Adafruit/Arduino don't offer any built-in LTE-M boards. This is still a potential option, however, that could be more efficient than our current plan.

2.4 Discussion

The three potential communication types are all prime candidates for the project. LTE-M is the weakest option, since it isn't fully supported for our choice of micro-controller. SigFox is another great option, but with a large caveat. SigFox is designed to primarily upload small, infrequent bursts of data. This is apparent by examining the speed at which it transmits, as well as the uplink and downlink scheme. It was noted before that SigFox is primarily for data uplink, not downlink. It would be difficult to send conditional control statements from the web application to the irrigation valve.

2.5 Conclusion

With the very apparent issues with LTE-M and SigFox, it seems that LoRa is the best option for this project. Our chosen board luckily has a built-in LoRa as well. This option was best suited for our bidirectional flow of information, while maintaining both a low overhead and long battery.

3 Part 3: Communications Protocol

3.1 Overview

Now that we have chosen the control board, and the communication type, we need to chose a communication protocol, which describes exactly how the information will depart from a machine and arrive to another. Even though we have chosen the LoRa, we must still chose a protocol that will make sense of the radio transmissions.

3.2 Criteria

We want our communications protocol to be fast, efficient, and lightweight. The Adafruit boards do not have large amounts of room to store lengthy protocols. In addition, we want the protocol to be optimized to send small packages of data. We don't want to focus on protocols that do much more than what we want, do to overhead issues.

3.3 Potential Choices

3.3.1 MQTT

MQTT is a very small messaging protocol for machine to machine connectivity. MQTT stands for Message Queue Telemetry Transport. This protocol is designed for applications such as ours, with small control boards. MQTT is optimized for remote devices with a small code base, and for situations and systems with a very limited bandwidth.

3.3.2 AMQP

AMQP stands for Advanced Message Queuing Protocol. It provides the same end result that MQTT does, but with a lot more messaging options included. AMQP has allows a user to see and create the message structure, including headers, text bodies, and properties. With this protocol, a developer could customize everything about how the protocol deals with certain messages. For example, one could design a system with access control, and a complex structure describing where certain messages get sent based on certain parameters.

3.3.3 STOMP

The last potential protocol is STOMP, which stands for Simple/Streaming Text Oriented Message Protocol. Much like MQTT, this protocol is very lightweight and simple. Like AMQP, STOMP can differentiate separate parts of the message structure, potentially increasing the amount of control over the protocol. The options for STOMP are pretty vast, but not as vast as AMQP.

3.4 Discussion

The three potential messaging protocols are harder to compare and contrast, since they all work toward the same goal. MQTT is the simplest protocol out of the three. It has the lowest overhead, and also the lowest amount of functionality, with no control over message structure. AMQP is quite the opposite, with a huge list of features and ways to control any type of messaging situation. STOMP lies in the middle.

3.5 Conclusion

The purposes of the irrigation valve project is a proof-of-concept system that can exchange soil moisture data and valve conditional controls. The type of data that will be flowing through the system is very simple, and small. That being said, the system does not require such a convoluted messaging protocol. There is no use to create a complex message system with custom headers and bodies. AMQP is used in large enterprises, and controls complex structures of data flow, which are not present in our system. Even STOMP is too complex for our system, and an unnecessary addition of overhead. MQTT does everything we need it to do, and is far smaller than the alternatives. In addition, and much like LoRa, MQTT supports several different security implementations, such as TLS/SSL.