Low Cost Environmental Data Collection with Arduinos

Omar White
School of Informatics and Computing
Indiana University
Bloomington, Indiana 47405
Email: omawhite@indiana.edu

Dr. Monica Anderson and Trey Harrison
University of Alabama
Tuscaloosa, Alabama 35487

***Abstract*—Abstract - Data collection is central to any kind of scientific study, but the costs and hazards associated with scientific data collection often serve as a significant barrier, particularly when attempting to collect environmental data. Without data, it is impossible for scientists to test their hypotheses, conduct experiments or verify results. Wireless Sensor Networks are emerging as an effective tool for reducing the costs and hazards associated with environmental data collection. In this paper, our research team aims to continue these cost reduction efforts, by testing and researching Arduinos as our data collection tool. With the proper coding, configuration, power source, and network architecture, Arduinos would be able to collect live data and transmit it through Wireless Sensor Networks to a computer database or server.**

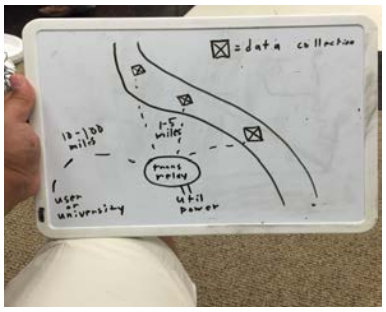


Fig. 1. An early concept drawing of our project

I. Introduction

Data is of crucial importance to scientists for their research efforts, but can sometimes prove difficult to obtain. Without data it is impossible to test hypothesizes, run valuable experiments, or verify results. Environmental data collection often requires the purchase of industrial grade equipment. This can be very expensive, especially when dealing with underwater sensors [1]. Wireless Sensor Networks(WSNs) have proven to be useful for collecting data for a variety of purposes [1]. Using previous work done with WSNs as a guide along with the capabilities of the Arduino, cost effective environmental sensors could be developed.

Arduino is an open source micro-controller that is designed to provide the functionality and power of a micro-controller while remaining flexible and easy to use. Because Arduinos are open source, there are a large number of code libraries that can be used to customize their capabilities. Each library also has its own extensive documentation and sample code. There are also a number of different models to choose from, making it even easier to use them for a custom project. Due to this customization we believe that Arduinos have great potential as data collection tools.

II. Related Work

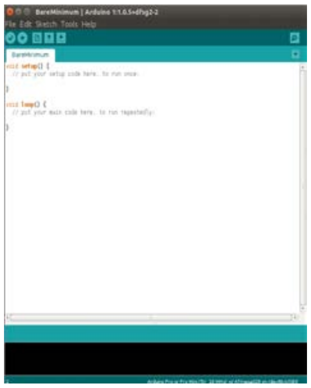
The use of WSNs for data collection is not new. The areas of application vary greatly, from academia, to military, to industrial and even the medical field. The Wireless Sensor Networks consist of a number of sensor nodes that can be deployed to collect data from an area [1]. Sensor nodes on their own do not have the capability to transmit the data they collect over long distances. As a result, past researchers have used existing wireless, or other communication networks to transmit their data to a database or server [1]. Previous work with WSNs has also demonstrated that environmental data transmitted to a database can be organized and sorted by location, data type, timestamp, and other beneficial factors [3] [2] [1].

Researchers have used WSNs for collecting data in difficult to reach or hazardous locations [2]. In order to monitor municipal river levels for example. Data collected from WSNs has also been used to make predictions on natural disasters [4]. In another project, researchers were able to predict flood occurrences and minimize property damage and loss of human life by combining collected data with a prediction algorithm [4].

A group of academic researchers at the University of Nevada in Las Vegas set out to see if Arduino based sensors could serve as a comparable alternative to the sensing equipment they were currently using [3]. They used Arduinos for data collection along with a web service they developed to keep track of their data [3]. Their tests of the Arduino based sensors compared to their old data collection methods showed that the Arduino could indeed serve as a comparable, less expensive alternative to their previous solution [3]. In addition, the researchers found that they were able to improve the efficiency of their data collection efforts, while making it easier to share their data with colleagues [3]. This work by previous researchers, further supports the idea that the Arduino/WSN model is a viable solution for scientific data collection.

III. Research and Experimentation

The goal of our initial research was to explore the Arduinos capabilities as a device for environmental data collection. Alabamas Sipsey river served as an initial deployment site. Tests were run using the Arduino as a sensor node. Networking solutions were explored and several candidate designs were proposed before a prototype was decided upon. In addition to barriers associated with keeping costs under control, the team was also forced to deal with obstacles like power consumption, stable network connectivity, and how to best protect equipment from the elements.


BareMinimum | Arduino 1:1.0.5+dfsg2-2
File Edit Sketch Tools Help
BareMinimum

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig. 2. The Arduino IDE with an example of the basic structure of an Arduino sketch.

A. First Steps

Before anything else could be accomplished the Arduino had to be developed into a data collection device. The Arduino Uno was selected for this stage of the project. The goals for the Uno included collecting water temperature data and recording water depth. Using a submersible temperature sensor with some of the Arduinos built in code libraries, the Arduino was able to capture temperature data and display it via serial communication.

In order to record that data that was being collected an SD card reader was integrated into the design. The SD card allowed data to be recorded without the team needing to devote extra time to research networking solutions or creating a server. This allowed for data collection to stay the main focus of this stage of the project. The SD card would serve as a failsafe or backup for future iterations of the project in the event that a connection to a server was lost, or data wasnt completely transmitted; a copy of it would be stored locally with the sensor node.

The Arduino Uno was further tested to see if accurate data could be collected and stored. Code libraries for both the SD card reader and the temperature sensor were readily available and easily adaptable to the design. Using these libraries, the team was able to code the Arduino Uno to collect and record data at regular intervals.

B. Water Depth Measurement

The Arduino also needed to be able to monitor and record changes in water depth overtime. Collecting this type of data wasn't as straightforward as equipping it with a thermometer.

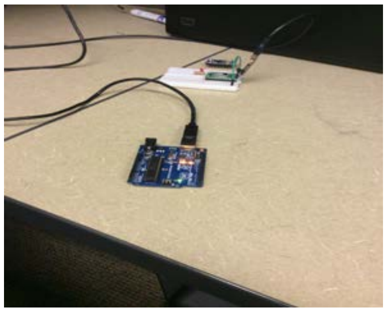


Fig. 3. An Arduino Uno and Arduino Mini Wireless, the two Arduino models used in the project, being programed.

The easiest solution would have been to equip the Arduino with a pressure transducer and then convert the pressure data into a depth measurement, but this method presented a major cost barrier. An affordable underwater pressure transducer could not be found. The majority of transducers that were found, were meant for industrial applications, and not suited for academic research projects that must operate under a tight budget. With pressure transducers too expensive to provide a viable solution, other less costly alternatives had to be explored.

Previous researchers ran into the same problem. In a similar project, researchers used ultrasonic sensors to create an underwater depth measurement system [4]. Their system used a PVC pipe in the river, with openings to allow for water to flow freely through the pipe without air pressure buildup. The inside of the pipe contained an ultrasonic transmitter and receiver. This allowed for depth measurement to be calculated by determining the amount of time it took for a signal to travel up and down the pipe. As the water levels changed, a hollow box at the bottom of the pipe would float on the surface of the water, and reflect the ultrasonic signal, allowing a scientist to measure the water level variation. The overall height of the PVC pipe, the thickness of the box, and the speed of the ultrasonic sensor signal, were all factored into the final depth measurement [4]. This solution seemed viable, but would drive up project costs due to the price of PVC pipe. Due to the potential cost of this second option more time had to be devoted to finding cost effective solutions before settling for this more expensive option.

The issue of depth sensing is one that has yet to be resolved. Late into the project, a more cost effective sensor was discovered, but as of this writing, this more cost effective sensor has yet to be ordered and tested. For more information, see the future works section below.

C. Power Consumption

After identifying the types of data to be collected, as well as various methods of obtaining the data, the next hurdle was power consumption by the sensor nodes. Sensor nodes in WSNs operate on a limited energy budget [5]. It is important to be able to optimize power consumption in the event that batteries used to power sensor nodes are unable to be charged or replaced [5]. During tests, the Arduino Uno was powered through the USB port of the computer the team used to develop and program it. This option would obviously not be possible

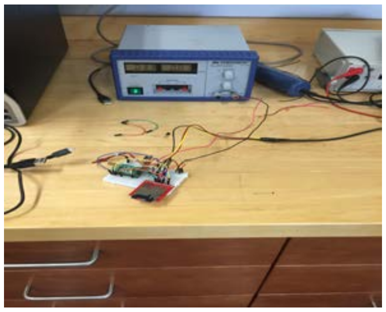


Fig. 4. Running the Arduino Mini Wireless from a power supply

during a live deployment, meaning that the Arduino would need to be battery powered.

1) Power Optimization: If long term data collection and deployments were the overall goal, optimization of the Arduinos power consumption had to be taken into account. Powering the Arduino Uno from a battery was simple. The Arduino Uno operated at 5 volts. As a small test, we hooked up a 9 volt battery to it, and then connected it to our temperature sensor and SD card reader. We discovered that we were able to successfully capture and record data with our device. Unfortunately a 9volt battery wouldn't keep the device powered for the extended periods of time.

Concerns over power consumption and long term deployment led to exploration of the capabilities of a different Arduino model that could replace the Arduino Uno, the Arduino Mini Wireless. The benefits of the Mini Wireless included: built-in wireless capability, the ability to operate at 3.3 volts versus the 5.0 volts of the Arduino Uno, a smaller form factor, meaning it required less material when using an environmentally sealed enclosure, and most important of all; the ability to use sleep mode. Sleep mode allows for unused components of the Arduino to be shut off for a period of time, allowing the device to consume less power. Since the device only needed to collect and transmit data at timed intervals; this technique was ideal for the project. The usage of sleep mode to conserve power is a common tactic when developing with WSNs [5].

Duty cycling - the interval between the time a device is active, to its total deployment time, is crucial when utilizing sleep mode [5]. Understanding which parts of a design are most energy intensive, helps with deciding when and where to activate sleep mode. Typically wireless communication is one of the most energy intensive aspects of a sensor node [5]. By employing a sleep/wake schedule, sending data at predetermined intervals while placing the device in sleep mode the rest of the time, a device uses less energy [5].

We used a third party library (Jeelib), to implement sleep mode into the Mini Wireless. This library only allowed it to sleep in intervals of a few seconds at a time. These intervals were further extended to conserve power using loops to chain together several instances of sleep mode together one after another.

In order to save the maximum amount of power a real-time clock could be added to the design. Using this clock to trigger interrupts the Arduino could be woken from an even deeper sleep, one that shuts off nearly all of its components.The clock also has the added benefit of being able to time stamp our data. This is a method that we didn't have time to test, but in theory it would reap the maximum benefits in terms of power conservation.

Fig. 5. We set up a makeshift workstation outside to monitor the communications sent by the Arduinos

D. Networking Solutions

Figuring out feasible networking solutions that could transmit data from the Arduino to a server or database, was one of the most important yet difficult aspects of the research project. We explored the possibility of using ham radio networks, the built-in networking capability of the Arduino Mini Wireless itself, and cellular radio towers.

1) HAM Radio: We began looking for networking solutions by exploring Ham Radio networks as our first option. The Arduino Mini Wireless could be equipped with a radio transmitter that would send a signal to another transmitter. A series of short range signals would be sent until the data reached a point where it could be transmitted long range via cell towers or Wi-Fi. Though this option was quickly abandoned, due to expense, it did make us take better notice of the data communication abilities that already existed in the Arduino Mini Wireless.

2) Arduino Wireless Communication: Similar to the Ham Radio option, the Mini Wireless required data to be relayed through a series of short range messages until it reached a stronger signaling point capable of transmitting data over a longer distance. The difference: unlike the Ham Radio option, the signals could be sent using the Mini Wireless own communication capabilities. This option also had the potential to be cheaper than the Ham Radio option.

We conducted two tests with the Arduino Mini Wireless network solution, one in the lab; the other in the area outside of our building. We connected two Arduino Minis to two separate laptops so we could monitor their communication. One Arduino sent a series of number values to the other, and when received - an acknowledgement message would be sent back. After every message transmission, the sensors were moved farther apart, allowing us to test their range. The tests were promising, but the range was limited to a little outside of our line of sight. Further testing is necessary to determine the maximum range the Arduino Mini Wireless can successfully transmit data. 5

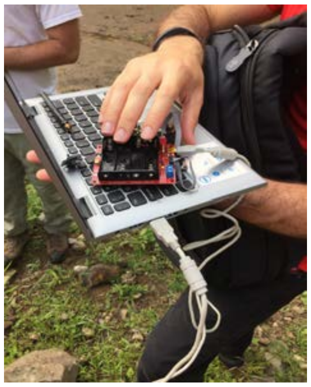


Fig. 6. A field test of a Cellular Radio

3) Cellular Radio: Cellular radio has proven to be a viable data transmission option for many sensor networks. Deployment sites that have strong enough signals to send text messages, are areas where cellular communication could be used to send data. If a deployment site has enough signal power, the sensor node could send the data as a text message to wherever the collected data was being stored. All the data collected by such a system could be saved as text which could be converted upon reaching a server.

Our team spent time exploring cellular radio as an option for data transmission. A small cellular radio was assembled and taken out to the field to be tested. We were able to send messages to our cell phones by hooking up the cellular radio to a laptop computer. This led us to believe that the same could be done with the Mini Wireless set up as a sensor node.

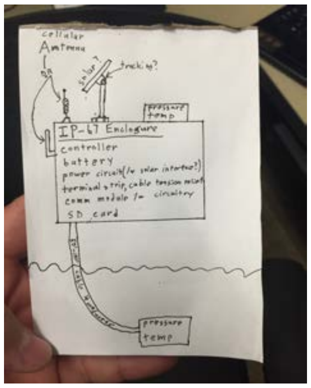


Fig. 7. A sketch of a proposed prototype design.

IV. Future Work

Most of the work done so far has been planning, scouting out components to use, testing two separate Arduinos with different features, and visiting possible deployment sites. One of the most important next steps is building a prototype that can be tested and eventually deployed. The Sipsey river is the first area that is planned for deployment of a prototype.

A promising candidate design has been decided upon 7. This design utilizes a IP-67 enclosure to house the Arduino, circuitry, and an SD card. Attached to the enclosure is a cellular antenna for transmitting data, a solar panel to keep the battery charged, a barometric temperature and pressure sensor, and the underwater sensors. This candidate design adds new possible components that will need to be tested before a prototype can be built. One of these new components is an underwater pressure and temperature sensor that was discovered late into the project. This would resolve the issue we had with collecting water pressure data while also consolidating two types of data collection into one sensor. Of these new components the barometric sensor, cellular antenna, and new underwater sensor will most likely be integrated first, with solar charging possibly at a later date.

This work will also require a database to to house the collected data. The ultimate goal for this database is for it to be updated in real time and made available to the public. As this research continues, goals will very likely change or evolve in an effort to make this solution adaptable to a variety of environmental research scenarios.

V. Conclusion

Arduino sensor networks can be developed for the purpose of collecting environmental data. Arduinos come in a variety of customizable features with an abundance of coding libraries that can extend their capabilities. The work described in this paper illustrates that the Arduino shows promise for being used as a low cost environmental data collection tool. By drawing upon the previous work done with WSNs and applying that work to an Arduino based sensor network, a powerful and flexible, low cost sensing solution can be developed. The future of this research is promising, and with more time and development, could prove to be a real asset to anyone hoping to collect environmental data.

Acknowledgment

Thank you to the Authors of all the papers referenced in the paper as well as everyone who worked on this project.

References

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[2] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "The hitchhiker's guide to successful wireless sensor network deployments," *Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08*, pp. 43–56, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1460412.1460418

[3] S. Lee, J. Jo, Y. Kim, and H. Stephen, "A Framework for Environmental Monitoring with Arduino-Based Sensors Using Restful Web Service," *2014 IEEE International Conference on Services Computing*, pp. 275–282, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6930544

[4] Z. I. Chowdhury, M. I. Rahaman, and S. I. Chowdhury, "Autonomous Monitoring of River Level with Real Time Event Prediction," *Computer and Information Science*, vol. 7, no. 3, pp. 67–80, 2014. [Online]. Available: http://www.ccsenet.org/journal/index.php/cis/article/view/35905

[5] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2008.06.003