Worcester Polytechnic Institute

ECE 2305 – Final Project Proposal
SPAS (Search and Pick A Seat), Go Online And pick your Table DESK (GOAT) GOAD

Anastasia Karapanagou (640938922) (25%)
Fivos Kavassalis (124919167) (25%)
Ioannis Alexiou (631078066) (25%)
Josue Contreras (851386562) (25%)

Supervised by
Dr. Alexander Wyglinski

https://github.com/FivosJKavassalis/ECE2305_Team14

https://www.youtube.com/watch?v=R3chP828VOM&t=15s

May 1, 2018

Table of Contents

Abstract……………………………………………………………………….……...2

Introduction………………………………………………………………….……….....2

Description of Proposed Solution and Brief Overview of Implementation……..……..…....2

Experimental Results…………………………………………………………...….7

Summary of Challenges Encountered and Lessons Learned…………………………...….. 13

Discussion……...……………………………………………………………………13

Conclusion………………………………………………………………………...14

Appendix………..……………………………………………….………………...14

Abstract

An IoT system was designed and implemented that comprises a wireless network capable of connecting multiple low complexity devices (sensors) to the Internet. The objective was to obtain information about which Gordon Library cubicles are empty, and therefore available for use, at any particular time. Each cubicle possessed a simple sensor connected to an embedded processor. The desired task was to detect whether the cubicle was occupied or not. This information would be transmitted via the Internet to the members of the WPI community seeking an empty cubicle space to work. Our system addresses the need of the WPI community for efficient use of time and resources, especially in periods of high demand for quiet space in the library.

Introduction

The Gordon Library, which offers space to students that wish to find a quiet place to study, is one of the most popular places on the WPI campus. Unfortunately, at peak times, students have a hard time finding a seat in the cubicle desk spaces located in the library. As the population of both undergraduate and graduate students on campus increases, demand for space on campus increases as well. Usually, most students seek places around campus to study and focus on work but looking for space can be a waste of time and energy. This is a great problem for WPI students' tight schedules. Efficient use of time in a busy and demanding academic environment is fundamental for the timely completion of work requirements, for students and faculty alike.

The team, therefore, decided to design a system by which any member of the WPI community may know at any given time whether the particular place/environment they are looking for, is occupied or empty, and, in this way, make an informed decision about its availability and use. Our project was designed to identify empty cubicle desk spaces in the Gordon Library and communicate it to the WPI community via the Internet. The team designed and implemented a wireless network capable of connecting multiple sensors to the Internet in order to obtain information about which desk cubicles are empty, therefore available to use, at any particular time. We feel we have contributed in this way towards addressing an important need for the WPI community.

Description of Proposed Solution and Brief Overview of Implementation

STEP 1: Set-up of the Physical Layer

For data processing, we decided to use the Arduino Nano V3.0 and Arduino Uno microcontrollers due to their small dimensions and cost-effectiveness. Specifically, the Arduino Nano interfaced with the ultrasonic sensor HC-SR04 we decided to include. This microcontrollers' array has a variety of ports to choose from so we could supplement our design with additional modules (shown in Figure 1).

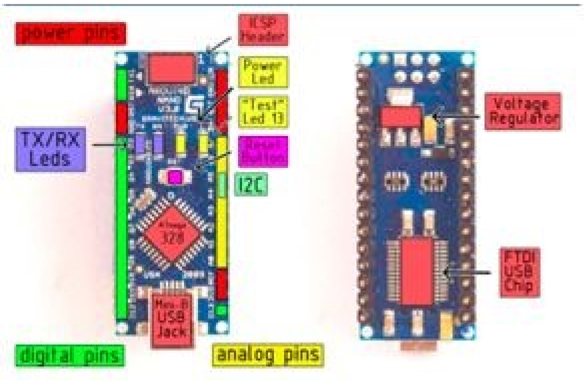


Figure 1: Arduino Nano V3.0 Microcontroller.

The first step involved is the interface of the microcontroller with the HC-SR04 ultrasonic sensor(s) (shown in Figure 2). The ultrasonic sensor operates at a frequency of 40KHz and has a range from 2 cm to 4 m which is suitable for our system. The status of the cubicle was reported every T seconds. We designed it according to the TTL (Transistor-Transistor Logic) communication protocol so that by a 10us pulse (high state) to the trigger pin, the module sent out an 8-cycle sonic burst. Then, the echo pin received this burst and outputted the time in microseconds. We have employed the following equation (shown in Figure 3) to find the distance between an object and the ultrasonic sensor, which helps us understand whether the cubicle is occupied.

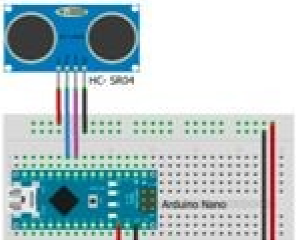


Figure 2: HC-SR04 Ultrasonic Sensor

$$\mathbf{distance = (duration*0.034)/2}$$

Figure 3: HC-SR04 Ultrasonic Sensor Distance Equation

Our sensor communicates to the aggregator node via the 433MHz RF front-end RX and TX modules. The transmitter possesses three pins: GND, Vcc and DATA. We used the DATA pin that is connected to a Digital pin on the arduino to transmit the signal, in this case the sensor data, to our receiver. On the other hand, the receiver possesses four pins: GND, Vcc and two DATA pins. We used only one DATA pin (shown in Figure 4). An antenna allowed our system to propagate the signal at greater distances, therefore solving any attenuation/path loss that the signal could encounter. However, we were also careful not to overamplify the signal as this could cause interference with other signals. We also realized that we had to optimize the various antenna configurations and distances ourselves. We used a coiled-up antenna and 17.3 cm of straight antenna to the receiver and the transmitter. The ASK (Amplitude Shift Keying) employed which controls the carrier wave was encoded and decoded but also limited the amount of data communicated. Another issue we worked on was the reduction of noise. We addressed this by using the Radiohead library.

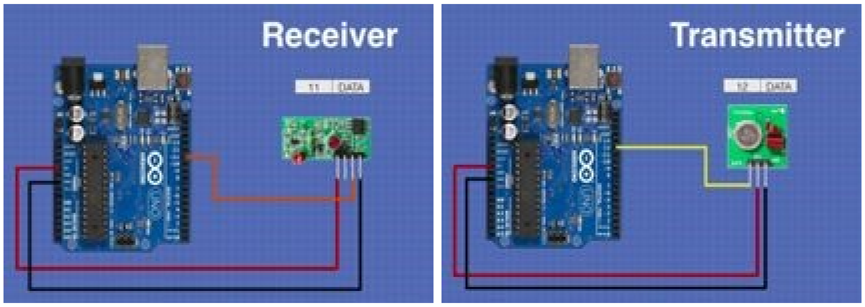
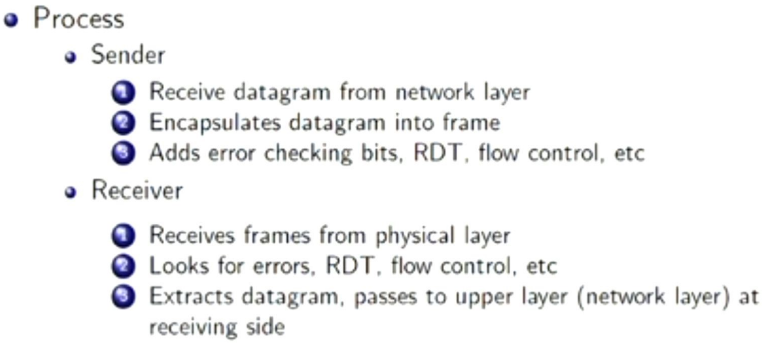


Figure 4: Data Pin on Receiver (left) and Transmitter (right).

The physical layer was completed by the addition of an LCD and LEDs for debugging purposes. For the former, the Crystal library was used to show the status of the modules and whether data is being received. The LEDs served as indicators that the modules are working correctly.

STEP 2: Medium Access Control Layer

We proceeded by working on the Medium Access Control Layer to be implemented in Arduino. To identify the sensor sending data and its location, we had to frame a datagram and add overhead recognizing the sensor number and the embedded processor name (Figure 5). The sender, in this case the sensor node, added only its sensor number node as overhead. This way when the aggregator node received that data, it knew how to filter out the data into the right channels. Each message sent out was a nine-character long string. The first two letters of the string were the name, for example "N1" for node one. Then, the next seven characters were either one for an occupied seat, zero for a free seat, or a comma in order for the aggregator to know where to separate the data.



- Process
  - Sender
    1. Receive datagram from network layer
    2. Encapsulates datagram into frame
    3. Adds error checking bits, RDT, flow control, etc
  - Receiver
    1. Receives frames from physical layer
    2. Looks for errors, RDT, flow control, etc
    3. Extracts datagram, passes to upper layer (network layer) at receiving side

Figure 5: Datagram Construction and Deconstruction

This information was then sent to the aggregator node where the datagram was filtered through the right channels to be executed. By correlating data with identifiable sensors and embedded processors, we were able to test the reliability of our set-up and choose our data display. In addition, we had to consider how we would address handling multiple sensor nodes communicating with one aggregator node. We reasoned we can solve this either by the Time Division Multiple Access (TDMA) or by the Code Division Multiple Access (CDMA) approach. We chose to use the TDMA. This channel access method divides the signal into different time slots, users transmit rapidly one after the other while using a single frequency (433 MHz) and in this way, use only part of the available capacity. In our case, only one sensor node was able to transmit during each time slot. The aggregator served as a master that transmitted the sensor node name that it wanted to communicate with. Then, the corresponding sensor node sent out the frame to the aggregator and once it finished processing the data, the aggregator node called on the next sensor node it wanted to communicate with. By using multiple time slots, the system communicated an increased amount of data. Its implementation required that our sensor nodes and aggregator node have a pair of transmitter and receiver modules for each node. Additionally, the system was able to check for corrupted files and data not received with a parity checking process.

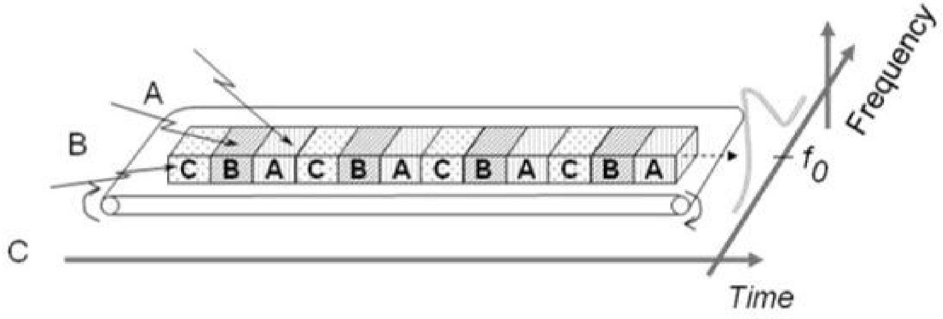


Figure 6: Simplified Diagram of TDMA Showing 3 Users Transmitting in Succession, first A, then B, finally C and so on (from https://www.tutorialspoint.com/cdma/tdma_technology.htm)

As with every system, there are advantages (as outlined above) and disadvantages with TDMA. Some of the caveats involve the fact that if time slots are short, then complicated signal processing should be employed, and complex equalization may be needed.

If we had opted for the CDMA method, we would have added complexity to the code implementation of the aggregator node as the aggregator would have to focus only on one sensor node transmitting. In this way, ideally, we could implement transmitters on the sensor nodes and one receiver on the aggregator node, thus enabling us to have an error/corruption checking scheme, namely the CRC scheme. This option comes with the caveat that data cannot be re-transmitted because of the one transmitter-one receiver implementation mode.

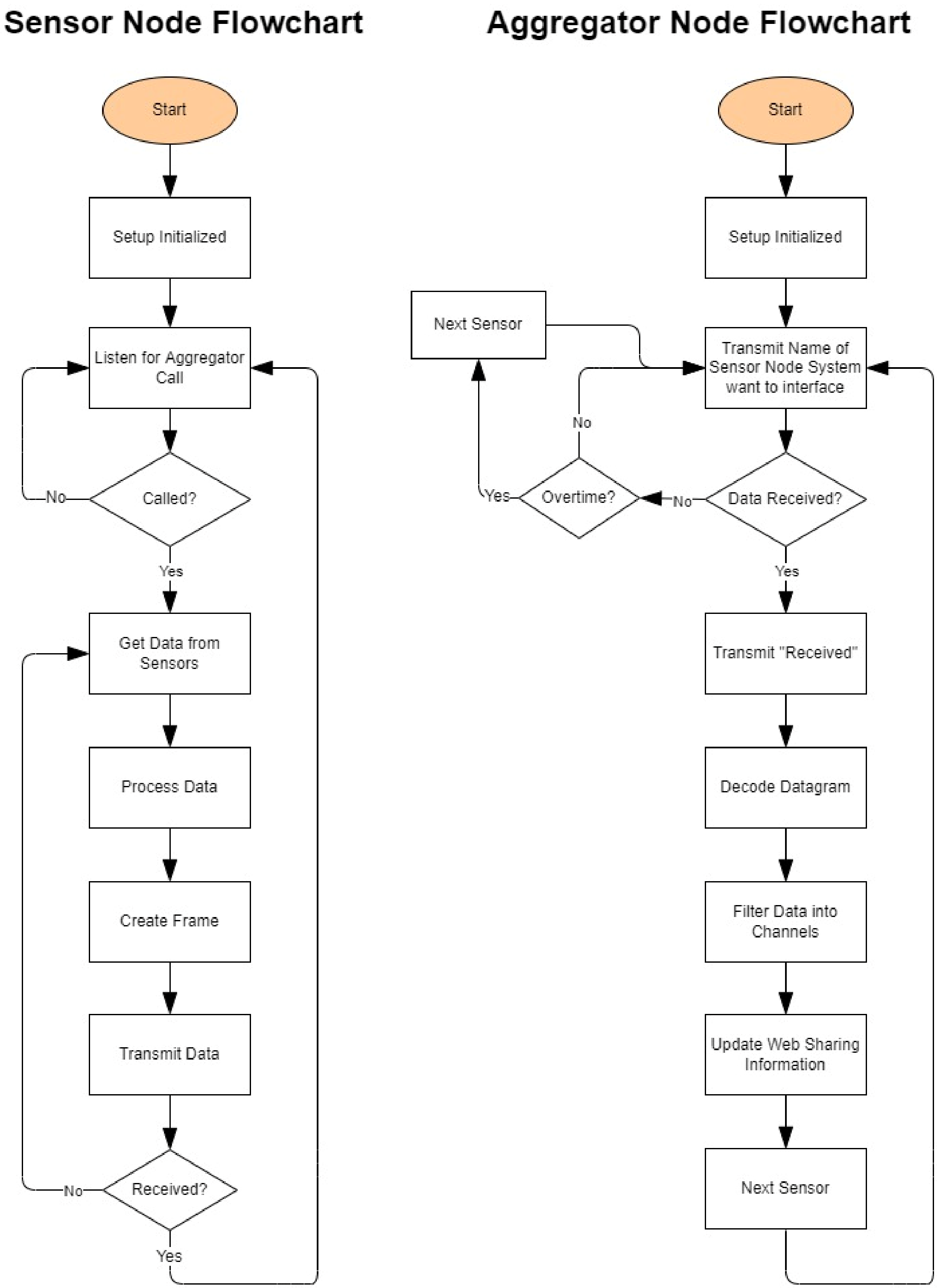


Figure 7: Sensor Node (left) and Aggregator Node (right) Flowcharts.

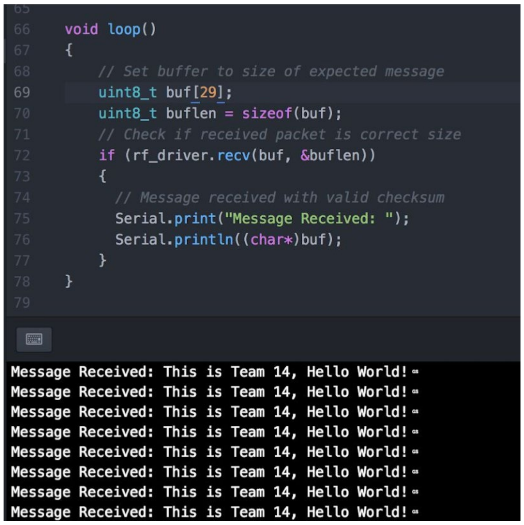
STEP 3: Connectivity to the Internet and Web Sharing Information

For the connectivity step, we used the Blynk Library with a unique authentication token to interface between the Arduino Uno and our smartphone application. Our effort was directed towards taking data from the Application layer of the aggregator and sending it to a user across the Internet. The Blynk Application would allow us to display our data, like the occupancy of cubicles in various areas of the library, in a user-friendly graphical user interface (GUI),

Experimental Results

Throughout implementation of this IoT device we made sure to test everything we added carefully, but due to WPI's fast-paced 7-week term and other classes we were not able to test the final product in its entirety.

We started this project by testing out the one-way communication between our Tx and Rx modules. This can be seen by the successful message in the figure below. Something interesting that our team did not notice at this stage of our project was the "garbage" that can be seen attached after the message received. This gave us problems during the implementation and will be discussed later in this paper.



```
void loop()
{
    // Set buffer to size of expected message
    uint8_t buf[29];
    uint8_t buflen = sizeof(buf);
    // Check if received packet is correct size
    if (rf_driver.recv(buf, &buflen))
    {
      // Message received with valid checksum
      Serial.print("Message Received: ");
      Serial.println((char*)buf);
    }
}
```

```
Message Received: This is Team 14, Hello World!␝
Message Received: This is Team 14, Hello World!␝
Message Received: This is Team 14, Hello World!␝
Message Received: This is Team 14, Hello World!␝
Message Received: This is Team 14, Hello World!␝
Message Received: This is Team 14, Hello World!␝
Message Received: This is Team 14, Hello World!␝
Message Received: This is Team 14, Hello World!␝
```

Figure 8: The first message received to test all the Tx and Rx modules

During this stage we also tested the communication range of the modules. Our results showed that 8 feet was the maximum range before the packages started dropping at a fast rate. We also must take into account that this was a controlled, peaceful environment, which is the complete opposite from where this device would be located, which is the library. We noticed that as a human body got closer to the modules, they would drop packages even more and would lose connection rapidly. It was interesting that when the receiver was pointed towards (this being the green box part as the front of the transmitter) the receiver did not receive the signal. However, when pointed the other way or sideways, it would receive the signal. This experiment was done in Line of Sight (LOS), meaning without any obstruction, because of the attenuation/path loss as the signal traveled from the Tx to the Rx module.

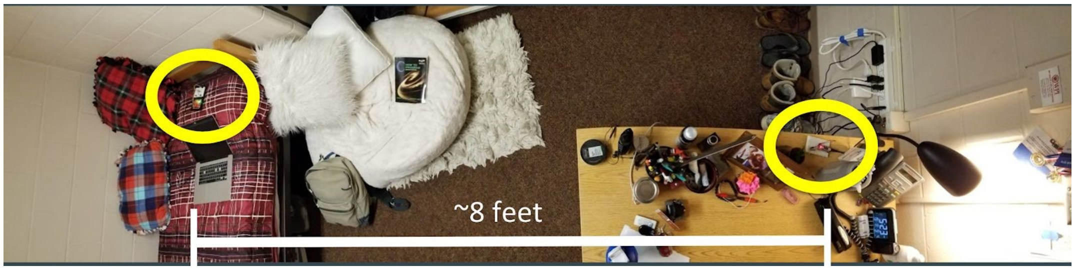


Figure 9: Range without antenna test

Finally, during this stage, we tested out the communication between multiple transmitters and one receiver even though we knew there was going to be collision. We got the following results: When transmitting two sensors at a time without any protocol, the serial port immediately stopped printing values that were being transmitted. This meant that because two signals where transmitting at the same time and in the same frequency, collision occurred at the receiver module. After turning off one of the sensor nodes, it took an average of 2 seconds to start transmitting again, considering that there was no error checking or protocol.

On the next stage of the implementation process, we added the antennas. We decided to use a 17.3 cm antenna on both the receiver and transmitter modules. We came to this length by taking a fraction of the 433MHz wavelength (Figure 10).

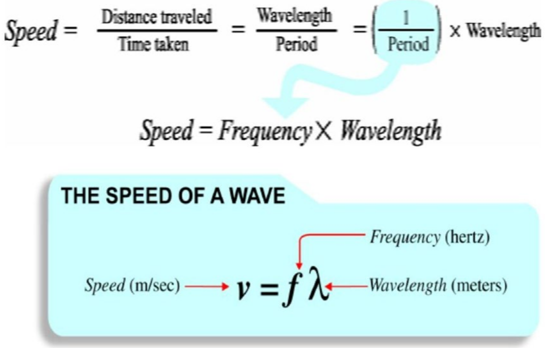

Speed = Distance traveled / Time taken = Wavelength / Period = (1 / Period) × Wavelength
Speed = Frequency × Wavelength
THE SPEED OF A WAVE
Frequency (hertz)
Speed (m/sec)
v = f λ
Wavelength (meters)

Figure 10: Wavelength equation

After adding an antenna, transmission distance increased to about 19.812 meters (Figure 11). As shown in figure 12, we were able to successfully transmit and receive messages at that distance. However, this was before the transmission started slowing down and losing packets.

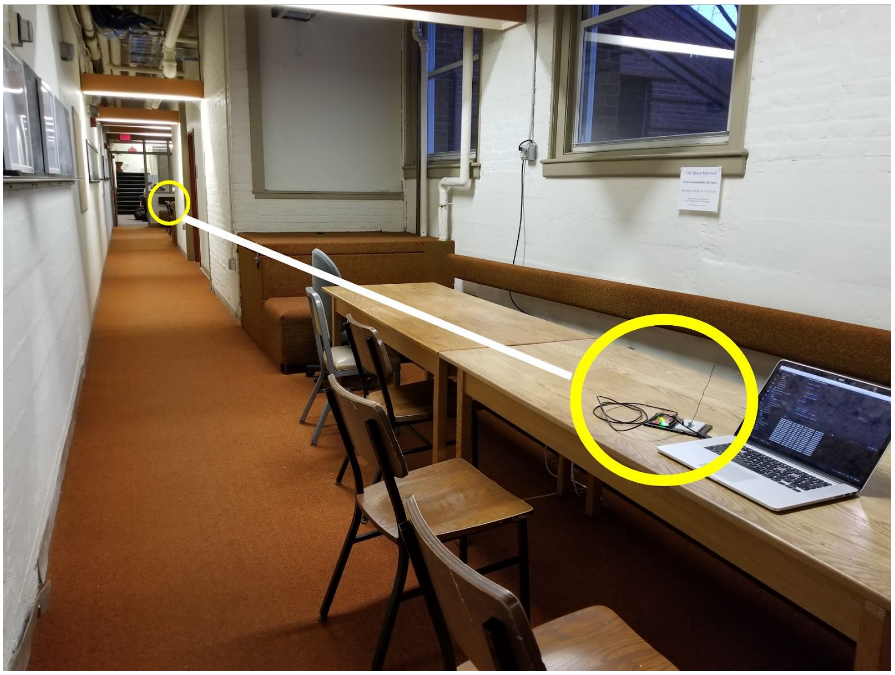
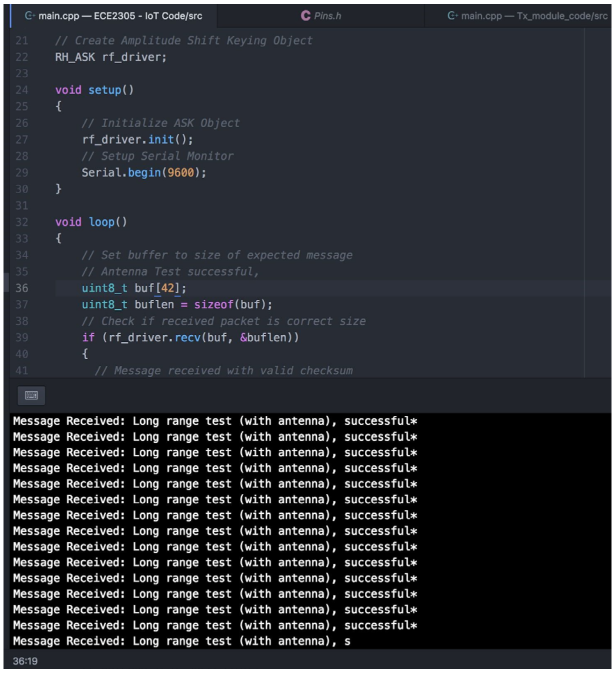


Figure 11: Transmission physical distance with Antenna LOFS



main.cpp — ECE2305 - IoT Code/src
Pins.h
main.cpp — Tx_module_code/src

```cpp
// Create Amplitude Shift Keying Object
RH_ASK rf_driver;

void setup()
{
    // Initialize ASK Object
    rf_driver.init();
    // Setup Serial Monitor
    Serial.begin(9600);
}

void loop()
{
    // Set buffer to size of expected message
    // Antenna Test successful,
    uint8_t buf[42];
    uint8_t buflen = sizeof(buf);
    // Check if received packet is correct size
    if (rf_driver.recv(buf, &buflen))
    {
      // Message received with valid checksum
```

```
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), successful*
Message Received: Long range test (with antenna), s
```

Figure 12: Transmission successful message

Regarding the Multiple Access Layer and Link Layer experimental results, we ended up not doing error detection since it was included in the Radiohead library. When running the system, we had a 66% success rate meaning that only 32% of errors passed. This is not the best, but for this system it works out perfectly fine. Most of the packets dropped where when one sensor node would take time to transmit its sensor readings and would go over its allocated time. There was sometimes uncertainty towards what sensor readings the aggregator node would receive. There were cases where the aggregator node would receive the same sensor data twice in a row. Even though this happened, we found out that because the Web Layer did not update immediately, the false readings did not show up that many times. This is important because that way the users can receive the right readings most of the time. Additional features could be added to the code in order to encapsulate these edge cases.

Furthermore, our datagram changed, and instead of the ultrasonic sensor number, we used the microprocessor name. We did this because during our experimentation process, we realized that as the package was smaller the signals dropped would be less. We then realized that we could further implement a way of using hex values in order to transmit smaller packages for better communication.

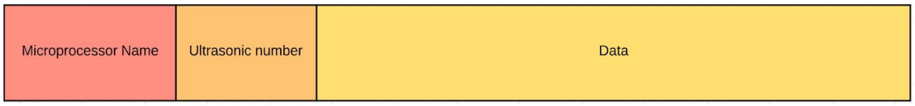
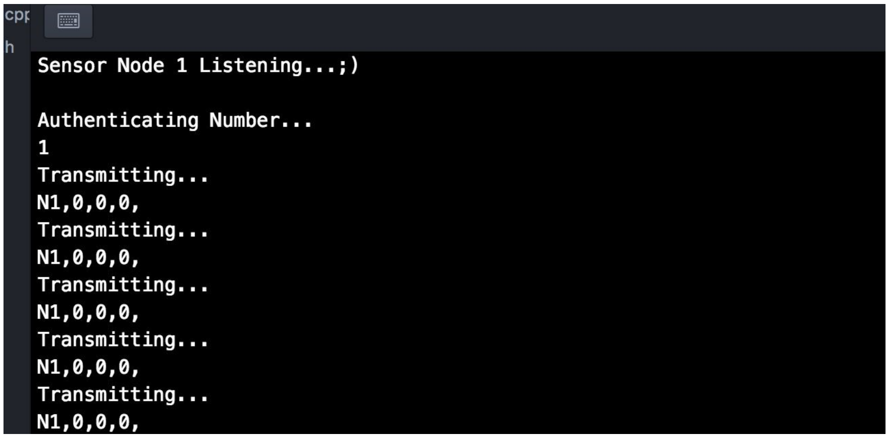


Figure 13: Original datagram

Next, we worked on the Web Sharing Layer. We first tried to use Google Docs to print the messages, but we were unable to since it printed readings from only two sensor nodes instead of 4, which is what we wanted. Therefore, we ended up using the Blynk app, which had the advantage of printing messages on the phone. Blynk uses an authentication token and is very easy to use and setup. To install and set up Blynk, the following steps were taken:

- First, install the libraries on Arduino
- Second, run on terminal the following commands (mac version)
    - Go to your arduino libraries through terminal
    - cd Arduino/
    - Cd libraries/
    - Cd Blynk/
    - Cd scripts
    - ./blynk-ser.sh
    - brew install socat
    - Then select the right port and it should start working
- Third, open app and press the play button

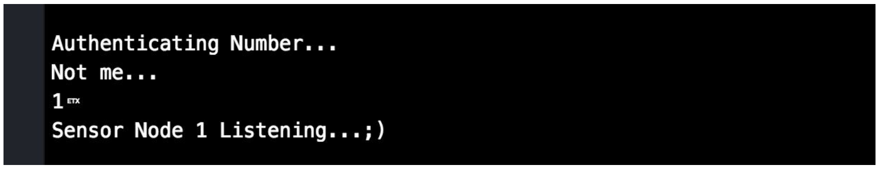
During our implementation, we encountered a transmission issue. When the sensor node first receives the message it authenticates it perfectly, but in the second iteration it does not. There seems to be something added which prevents correct authentication (Figures 14 and 15).



```
Sensor Node 1 Listening...;)

Authenticating Number...
1
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
```

Figure 14: Message received when transmitting correctly.

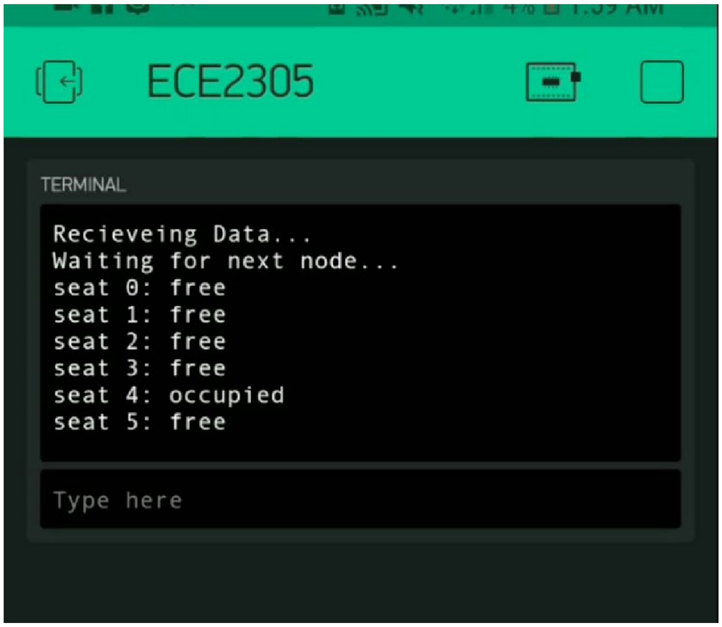


```
Authenticating Number...
Not me...
1␃
Sensor Node 1 Listening...;)
```

Figure 15: Second Iteration problem.

The maximum sensor nodes per aggregator node were 9. This restriction was because of how the code uses the atoi() function to convert an integer to a char. Further implementation could help with expanding the ratio of sensor nodes per aggregator nodes, but for this system's purposes, 9 is more than sufficient.

On the last stage of the implementation process, we tested out various API's. We tried to implement the web layer with PushingBox and Google Docs, but soon we realized that we could only get two sensor values working and we needed a better way of showing our data to our users. After doing some research we found the Blynk library that was simple to use and could display our data in a readable way. Our final GUI can be seen on Figure 16. This platform is also great if we would like to scale the number of nodes we wanted to display.



```
Recieveing Data...
Waiting for next node...
seat 0: free
seat 1: free
seat 2: free
seat 3: free
seat 4: occupied
seat 5: free
```

Figure 16: Web layer phone application GUI

The final iteration of the project is shown in Figure 17. The red light on the left node indicates that the sensor is sending messages, while the yellow light on the right node indicates that the sensor is listening for messages. During our experimentation we realized the transportation of the sensor nodes affected our time spent debugging the circuits because we would disconnect wires. We decided to fix this problem by laser cutting a box to enclose the whole circuit. This can be seen In Figure 18, the final product.

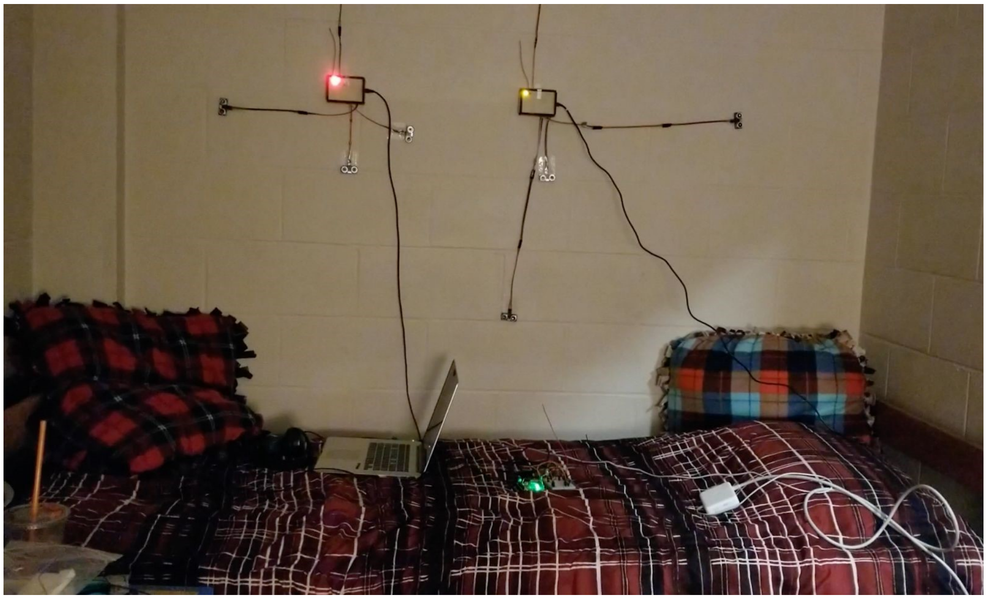


Figure 17: Final Outcome of Project

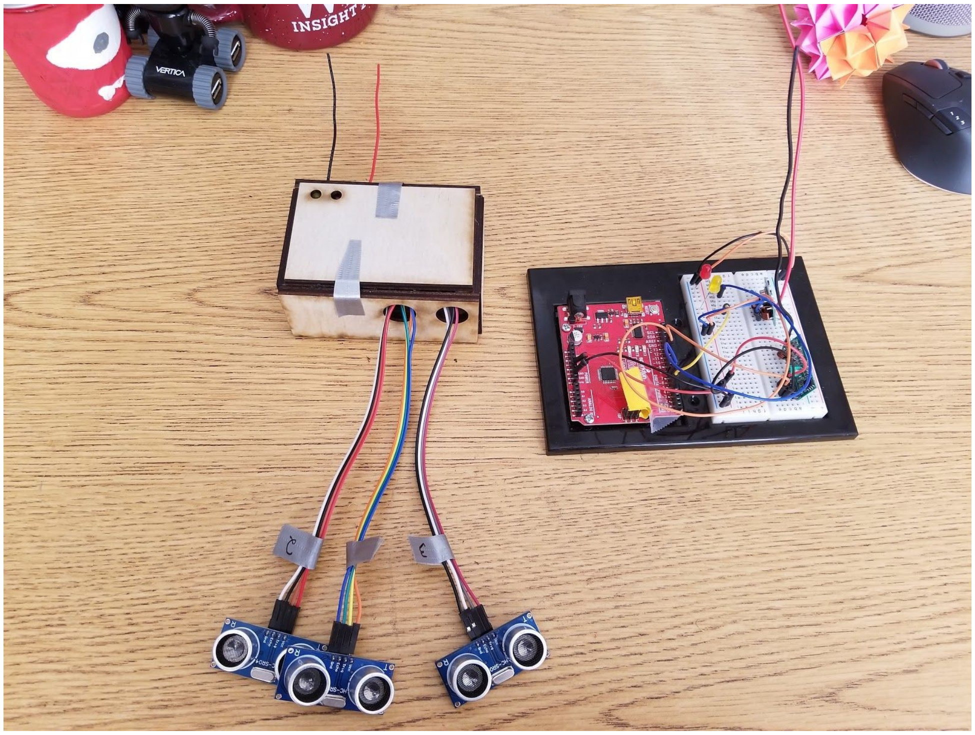


Figure 18: Final Product

Summary of Challenges Encountered and Lessons Learned

The first and most important issue we had to deal with was understanding and implementing technical challenges with TDMA, especially relevant to the testing, debugging and packaging of data. After failing many times, we managed to have it working and transmitting data.

Another mistake we made was the fact that we soldered a 12.5 antenna to the modules while we should have used a 17 cm wire. We also had problems with the second iteration of the TDMA, possibly due to variability of sensors. We thought of removing the additional interference ("noise") from the received message so that the receiver can authenticate the transmitting nodes. We reasoned that we could either implement a numerical filter or reset the variable.

We believe that by working on this project we understood and implemented a major and complex technology used in communications, the TDMA. It is quite complex since it requires precise synchronization between the transmitter and the receiver. We also acquired experience in designing a prototype where we realized the different phases of design and actual implementation. We feel the results of this project could contribute to a specific, identifiable need of the WPI academic community. We also feel that given more time and therefore, more improvement in its implementation, our prototype could be used in real-life situations, outside of our university. Finally, we also realized that technical challenges can occur in unpredictable ways sometimes, and when they occurred, we had to go back and re-evaluate our strategy with the help of our TA. We believe that this enhanced our understanding of basic principles of communications and networks.

Discussion-Future Prospects

The methodology followed in the development and deployment of a prototype application has included the following stages:

1. Breaking the concept in subsystems, designing the different subsystems of the prototype (Sensor Nodes, Aggregator, Web Interface)
2. Defining goals and constraints, creating a hierarchy of design requirements and extracting the overarching system architecture
3. Developing each subsystem as a set of modules (input/output, sensory/processing, action)
4. Integrating subsystems in a whole
5. Testing the functionality of the prototype
6. Usability testing

The resulting prototype should be considered at an early alpha stage in the software release alpha cycle (Reference: Producing Open Source Software - How to Run a Successful Free Software Project https://producingoss.com/en/index.html chapter 7)

This is a very early prototype that needs formal testing and further development. A method of evolutionary prototype may contribute to collect more feedback from the users and adapt the requirements analysis to the needs of the real users while advancing quickly in the path of normal system development (Reference: http://www.cs.nott.ac.uk/~pszjg1/FSE12/FSE_7.pdf)

Conclusion

Thankfully, our project worked! The web layer was launched successfully, and the nodes are communicating correctly with each other. However, every now and then, the nodes failed to communicate momentarily either due to network collisions, where two nodes were trying to transmit at the same time, or due to failure to send packets. We ended up implementing the essential idea of our proposal. We did not apply the idea of adding photoresistors, but we still ended up with an effective, elementary system using only ultrasonic sensors. One of the greatest takeaways from this project was that we learned a lot about TDMA and its implementation. Moreover, we learned more about how IoT works, by connecting physical devices through the network. Finally, our biggest problem was the hardware and especially the receivers. Nevertheless, this is acceptable due to their low cost.

References

Chesbrough, H. W. (2003). Open Innovation: The New Imperative for Creating And Profiting from Technology. Harvard Business Review Press.

Christie, E.J., Jensen, D.D., Buckley, R.T., Menefee, D.A., Ziegler, K.K.,Wood, K. and Crawford, R. (2012). Prototyping strategies: Literature review and identification of critical variables. ASEE Annual Conference and Exposition, Conference Proceedings

Fujimoto, S. T. (2000). The Effect of "Front-Loading" Problem-Solving on Product. Journal of Product Innovation Management, 17, 128-142.

Koen, P.A., Bertels, H. M. J. and Kleinschmidt E.J.(May-June, 2014). Managing the Front End of Innovation—Part II. Research-Technology Management

Li, Y. and Bartos, R. A survey of protocols for Intermittently Connected Delay-Tolerant Wireless Sensor Networks (2014) Journal of Network and Computer Applications 41: 411-423

Miao, G., J. Zander, J., K.W. Sung, K.W. and Slimane, B. (2016). *Fundamentals of Mobile Data Networks*. Cambridge University Press. ISBN 1107143217.

Peisert, S., Talbot, E., and Kroeger, T. (2013) Principles of Authentication, in Proceedings of the 2013 New Security Paradigms Workshop, pp. 47-56, available at https://dl.acm.org/citation.cfm?doid=2535813.2535819

Terho H., et al (2017) Understanding the Relations Between Iterative Cycles in Software Engineering, paper presented at the 50th Hawaii International Conference on System Sciences, available at http://aisel.aisnet.org/hicss-50/st/agile_development/7/

Wenger-Trayner, E. A. (April 15, 2015). Communities of practice – a brief introduction. Wenger-Trayner. Retrieved from http://wenger-trayner.com/introduction-to-communities-of-practice/

Yang , S-H. (2014) Wireless Sensor Networks Principles, Design and Applications, Springer, available at

ftp://doc.nit.ac.ir/cee/m.zahabi/BOOKS/wsn%20book/Shuang-Hua%20Yang%20(aut h.)-Wireless%20Sensor%20Networks_%20Principles,%20Design%20and%20Applic ations-Springer%20London%20(2014).pdf

Datasheets

1. Arduino Nano V3:
    1. http://roboromania.ro/datasheet/Arduino-Nano-roboromania.pdf 2.
   Ultrasonic Sensors:
    1. http://www.micropik.com/PDF/HCSR04.pdf
    2. https://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf
    3. With LCD- http://raspoid.com/download/datasheet/HCSR04

3. 433MHz RF front end RX and TX modules
    1. http://arduinobasics.blogspot.com/2014/06/433-mhz-rf-module-with-arduino-tutorial.html
    2. https://www.pjrc.com/teensy/td_libs_VirtualWire.html
    3. https://www.youtube.com/watch?v=b5C9SPVlU4U&t=1092s
4. LCD
    1. https://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay

Appendix

Design proposal Document

Abstract

One of the most visited places on Campus is the Gordon Library which offers space to students that wish to find a quiet place to study. However, at peak times, many students have a hard time finding a seat in the cubicle desk spaces located in the library. Efficient use of time is essential for successful accomplishment of course assignments. We decided to address this need of the WPI community by designing and implementing a wireless network capable of connecting multiple sensors to the Internet in order to obtain information about which desk cubicles are empty, therefore available to use, at any particular time. Each cubicle will possess a simple sensor and an embedded processor detecting whether the space is occupied or not. This information will be transmitted via the Internet to the members of the WPI community seeking an empty cubicle space to work.

Introduction to the problem statement

Have you ever been to the library on a busy week day and have not found where to study? How about in finals week when there is no time to lose? Or have you ever been to the Campus Center or even the Dining Hall during lunch or dinner and found it to be packed? Yes, these places are highly sought around campus, especially during the busy weeks of the term. As the population of both undergraduate and graduate students on campus increases, demand for space on campus increases as well. Usually, most students seek places around campus to study and focus on work, but it is getting harder to find them fast, sometimes taking up to 20 minutes or more. This is a great problem for WPI students' tight schedules. There has to be a better way to let the WPI student population know if they should make a run to the Gordon Library to successfully find a place to study or to the Campus Center to get something to eat. Efficient use of time in a busy and demanding academic environment is essential for successful completion of work requirements, for students and faculty alike. Especially for students, any measure to facilitate better organization of their time is indispensable. For all these reasons we decided to design a system by which any member of the WPI community may know at any given time whether the particular place/environment they are looking for, is crowded or empty, and therefore, make an informed decision about its availability and use. Our project will focus on identifying empty cubicle desk spaces in the Gordon Library and communicating it to the WPI community via the Internet.

Description of proposed solution (including rationale)

Our objective is to design and implement an IoT system, which comprises a wireless network capable of connecting multiple low complexity devices (sensors) to the Internet in order to obtain information about which Gordon Library cubicles are empty, therefore available to use, at any particular time. Each cubicle will possess a simple sensor connected to an embedded processor. The desired task will be to detect whether the cubicle is occupied or not. This information will be transmitted via the Internet to the members of the WPI community seeking an empty cubicle space to work.

Brief overview of implementation

The first step in the implementation process of our IoT system will be the Physical Layer. For the processing of data we will be using the Arduino Nano V3.0 Microcontroller and Arduino Uno. This cost effective microcontroller has small dimensions and possesses the input and output pins for our system. The main purpose of this microcontroller will be to interface with the Ultrasonic sensor we have decided on using. Furthermore, we will be able to add more modules to our system as we see fit because of the microcontrollers' array of ports to choose from (shown in Figure 1).

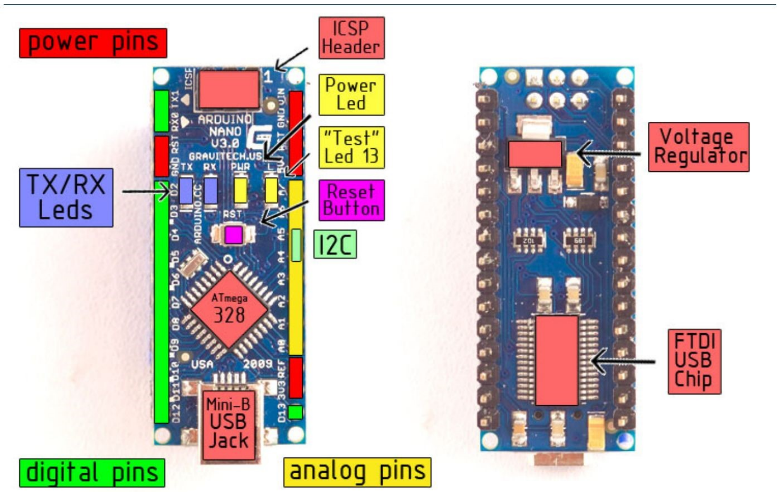


Figure 1. Arduino Nano V3.0 Microcontroller.

As mentioned above, the microcontroller will interface with the HC-SR04 Ultrasonic Sensor/s (shown in Figure 2) in order to monitor the status of a cubicle. The Ultrasonic Sensor operates at a frequency of 40KHz, which is beyond the human hearing capacity. In addition, it has a range from 2 cm-4 m which is perfect for the purpose of our system. The Ultrasonic sensor will be placed directly above the cubicle to be monitored and report its status every T seconds. The Echo and Trigger pins are the main pins we will be using in the Ultrasonic Sensor. By sending a 10us pulse (high state) to the trigger pin the module will send out an 8 cycle sonic burst. Then, the echo pin will receive this burst and output the time in microseconds. This communication protocol is referred as TTL (Transistor-Transistor Logic) which means that the level will always be logic high or logic low (a square wave). Therefore, we can use this to find the distance the object (human) is from the Ultrasonic Sensor with the equation shown below.

$$distance = \underline{duratio2n{*}0.034}$$

Figure 2. HC-SR04 Ultrasonic Sensor Distance Equation

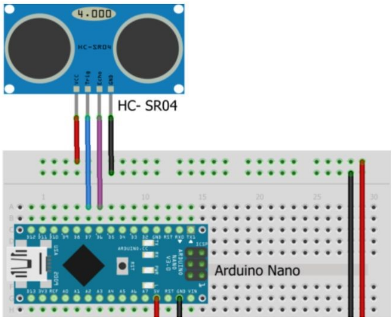


Figure 3. HC-SR04 Ultrasonic Sensor

In order to connect our sensor nodes to our aggregator node we will use the 433MHz RF front end RX and TX modules. Apart from being inexpensive, they are really easy to use and make our system have a minimalistic design. The Transmitter posses three pins: GND, Vcc and DATA. We will be using the DATA pin that is connected to a Digital pin on the arduino to transmit the signal, in this case the sensor data, to our Receiver. On the other hand, the Receiver possesses four pins: GND, Vcc and two DATA pins. We will be using only one DATA pin (shown in Figure 3). According to our research, adding an antenna to the Tx and Rx modules would enable the signal of our system to be propagated at greater distances, therefore solving any attenuation/path loss that the signal could encounter. However, it was brought to our attention that over amplifying the signal could cause interference with other signals and could possibly get us in trouble with authorities, therefore we will be careful to keep the signal within range and consult with the professor when in doubt. Apart from that, we also realized from various resources that we had to test the distance ourselves with the various antenna configurations for better results. Ideally, we will use an antenna of half or a quarter of the wavelength (17.3 cm) because these modules transmit at a frequency of 433 MHz with a wavelength of 69.24 cm. We will start by adding 17.3 cm of coiled up antenna to the receiver and 25 cm of straight antenna to the transmitter. This should give us a pretty good range to start with. Furthermore, these modules use an ASK (Amplitude Shift Keying) which modulates the carrier wave and can be easily encoded and decoded, but is limited to the amount of data it can send. To deal with noise, we will use the "radio head" library that takes care of keying to reduce noise and provides us with an array of functions to transmit/receive data.

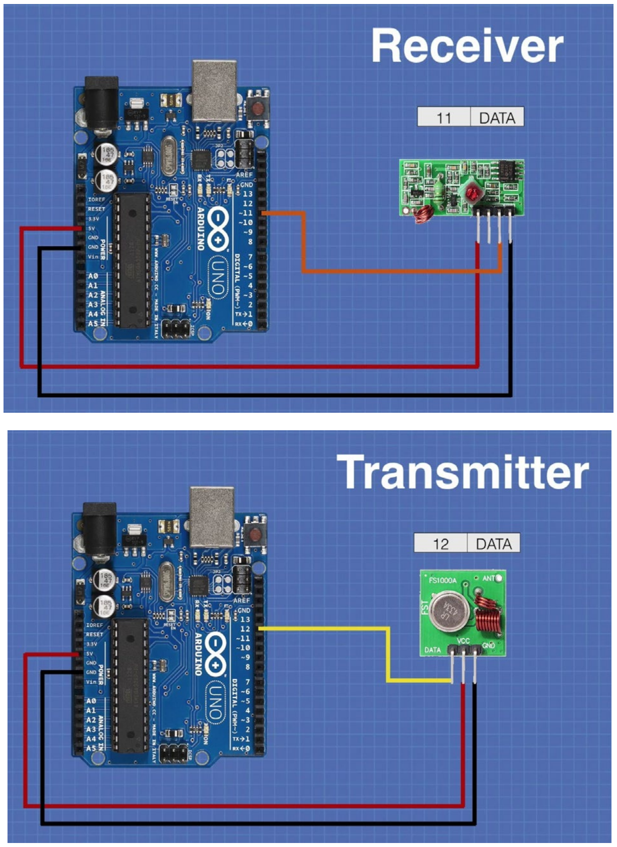


Figure 4. Data pin on Receiver (left) and Transmitter (right).

The final components we will add to the physical layer are a LCD and LEDs. For the LCD we will use the “Crystal library” to show the status of the modules and whether data is being received. The LEDs will serve as indicators that the modules are working correctly. These two components will make debugging the system a lot easier. Our team may decide to add more sensors to the system as we see fit.

Estimated Parts List With Individual Costs		
Name	**Quantity**	**Cost per unit**
Ultrasonic Module HC-SR04 Distance Sensor	TBD	$2.00
Arduino Nano V3.0, Elegoo Nano board CH340/ATmega328P	3	$4.29

UNO R3 Board ATmega328P ATMEGA16U2	1	$10.90
Gowoops 433Mhz RF Transmitter and Receiver	5 pairs	$1.80
LCD Display	1	$3.90
LED Lights	TBD	$0.04

The second step in the implementation process is the Medium Access Control Layer. This will be mainly implemented in Arduino and MatLab code. We will have to keep track of which sensor is sending data and from what area of the library (microprocessor) it is being sent. This can be done by framing a datagram and adding overhead that specifies the sensor number and the embedded processor name. This packet will then be sent to the aggregator node were the datagram will be filtered through the right channels to be executed. We will be able to correlate data to a specific sensor and embedded processor that will allow us to implement a transport layer to test the reliability of the data received. This will also allow our user interface to distinguish how to display the data. Furthermore, there are two possible ways of how we will address handling multiple sensor nodes that communicate with one aggregator node. The first possible implementation is the Time Division Multiple Access (TDMA) approach. This multiple channel access method will allow the system to communicate over one frequency (433 MHz). Time will be divided into slots and only one sensor node will be able to transmit during each slot. For this case the aggregator will serve as a master that calls (transmits) the sensor node name that it wants to communicate with. Then the sensor node will send out the frame to the aggregator and once it finishes processing the data, the aggregator node will call on the next sensor node it wants to communicate with. This will require that our sensor nodes and aggregator node have a pair or transmitter and receiver modules for each node. Additionally, the system will be able to check for corrupted files and data that is not received with a parity checking process. On the other hand, the second multiple channel access method we could implement is the Code Division Multiple Accesses (CDMA). This method will add complexity to the code implementation of the aggregator node. The aggregator node will have to filter out all the sensors nodes that are transmitting and focus on the one it is listening for. The sensor nodes will therefore be transmitting continuously and could be accessed at any time. The reason behind this, is that ideally in the physical layer of our system, this will allow us to use only transmitters on the sensor nodes and one receiver on the aggregator node. This will be able to have an error/corruption checking scheme, namely the CRC scheme, but unfortunately it cannot re-transmit data because of the one transmitter-one receiver implementation. Therefore, that data will be lost.

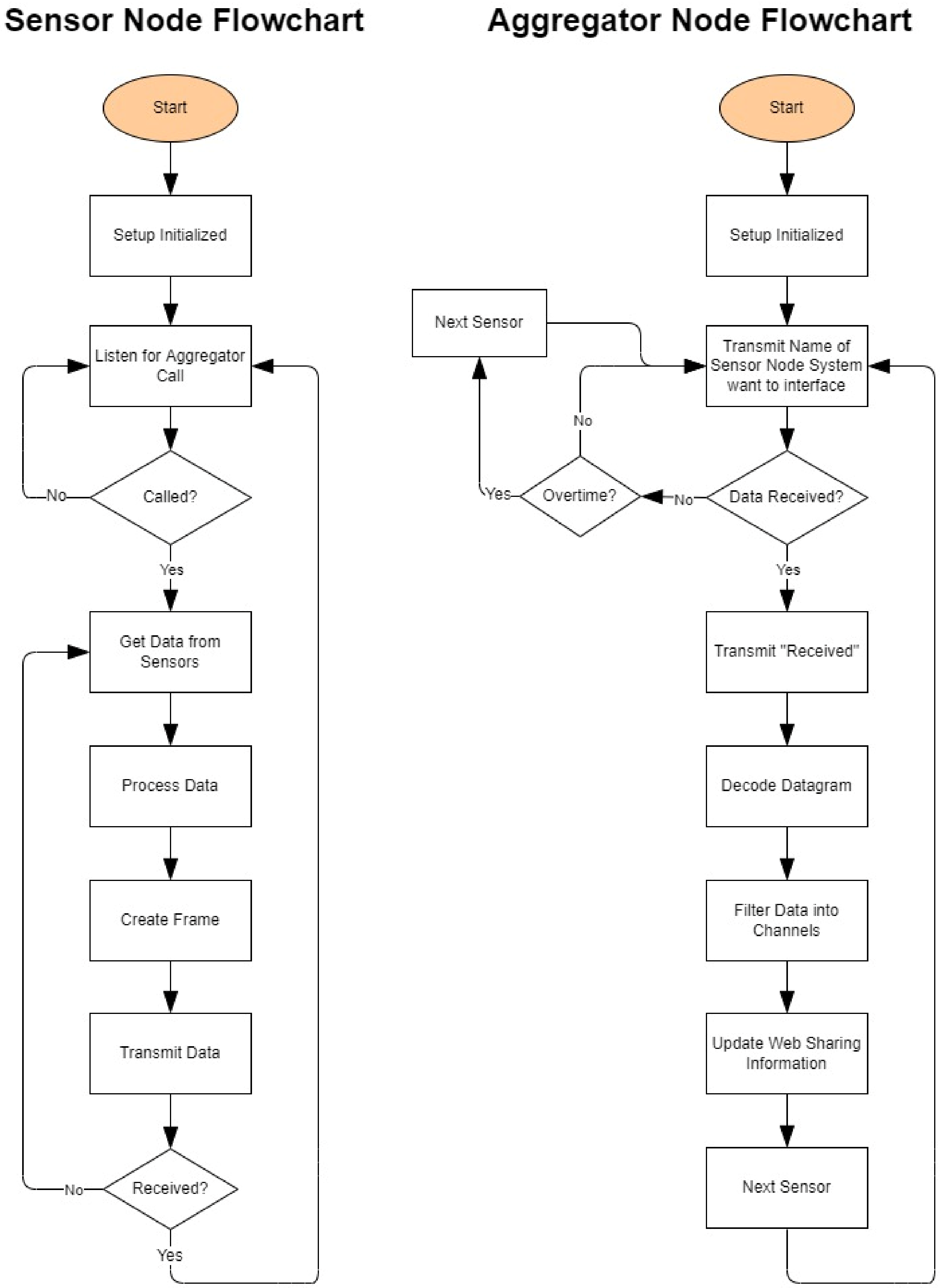


Figure 5. Sensor Node (left) and Aggregator Node (right) Activity Diagrams.

The third and fourth step of the implementation process are somewhat connected, the layers being the connectivity to the Internet and the Web Sharing Information. For the Connectivity to the Internet Layer we will be using the Blynk Library with a unique authentication token in order to interface between the arduino and the application on our phones. This will allow us to take that data from the Applications layer of the aggregator node and send it across the web to a user. This brings us to the Web Sharing of Information Layer where we will use the Blynk Application as mentioned before. This application will allow us to display data from the sensors across the internet with an easy-to-read graphical user interface (GUI). Some of the features that this GUI will display are the status of seats and cubicles in different areas of the library. Another program that our system could interface with is “ThinkSpeak MATLAB”. Our team will further implement this section of the project as we see fit for our system.

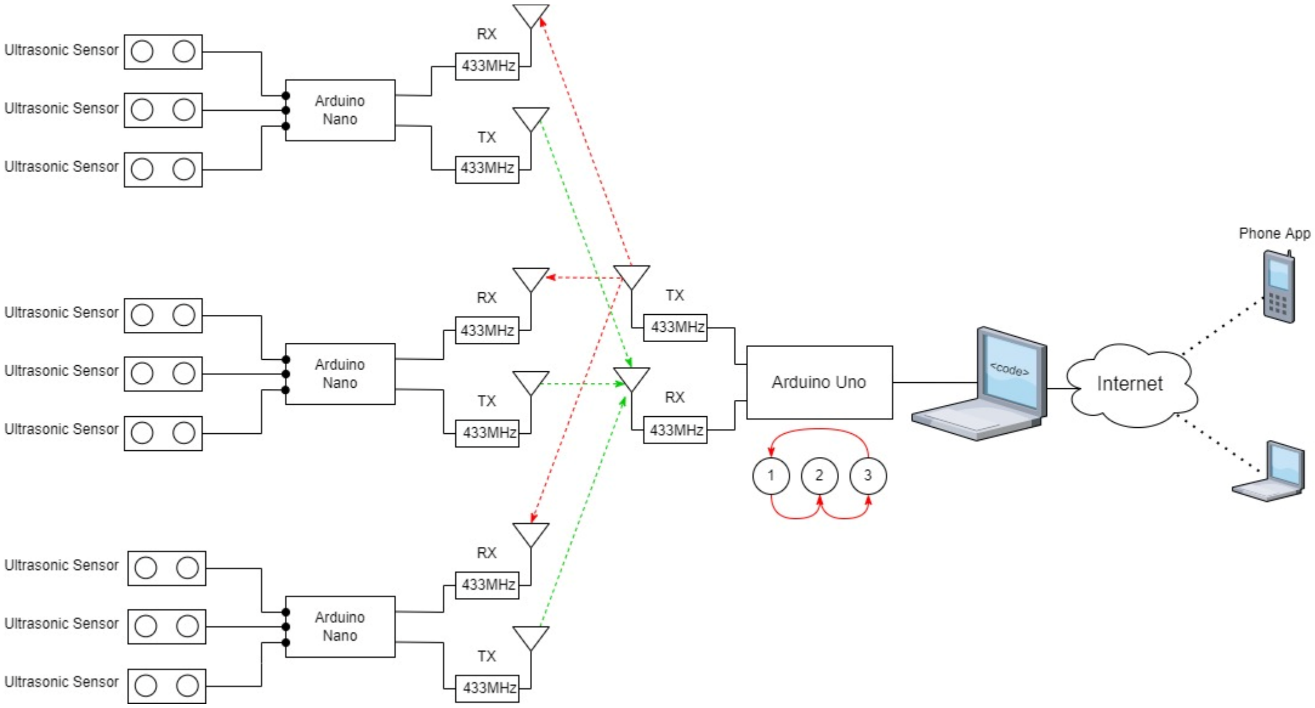


Figure 6. Illustration of the IoT architecture for a wireless sensor network used to detect the presence of humans.

.

Prototype development methodology and evaluation strategies

We will use the following prototyping methodology:

1. Break the concept in subsystems, design the different subsystems of the prototype (Sensor Nodes, Aggregator, Web Interface)
2. Define goals and constraints, create a hierarchy of design requirements and extract the overarching system architecture

3. Develop each subsystem as a set of modules (input/output, sensory/processing, action)
4. Integrate subsystems in a whole
5. Test the functionality of the prototype
6. Usability testing

We will implement the following steps for the completion and evaluation of our prototype

1. We will make sure the prototype is correct and faithful to the design. As we will build the prototype, we will make the necessary changes and fix errors that may come up. In this way we will employ a strategy that identifies and solves problems in early phases of the development of our prototype (otherwise called "front loading" problem solving).
2. We will conduct multiple rounds of usability testing with increasing levels of fidelity.
3. We will consider the limitations of our prototype, thus defining what we can and cannot test.
4. We will create a "community of practice" (CoP) which would be a group of people from the WPI community (undergraduate and graduate students, faculty, administrators) that share the same need of solving the problem we are attempting to solve, i.e. the need of knowing which common spaces on campus are available to use.. We will explain our prototype to them and ask them to use it and share their views and comments on its usability. From the literature it is clear that employment of CoPs in prototype evaluation strongly correlates with innovation.
5. We will create a video that we will upload to YouTube, in order to showcase our prototype.

Project logistics (e.g., timeline, milestones, task specification)

TIMELINE	MILESTONES AND TASK SPECIFICATION
March 30	• Complete Proposal Document
April 6	• Setup client and complete functionality of individual board without networking
April 13	• Set up local communication between clients and “server” • Initial setup for web-interface
April 19	• Complete web interface
April 27	• Fix bugs
May 1	• Final Report

Source Code & video

The source code and video can be found in the following link:

https://github.com/FivosJKavassalis/ECE2305_Team14
https://www.youtube.com/watch?v=R3chP828VOM&t=15s