Robotic Football Indoor Positioning System

Electrical Engineering Senior Design Final Report

By: Matt Domenech, Daniel Ridzik, Peter Ryan, and Aidan Shaughnessy

Table Of Contents

1 Introduction

The Notre Dame Robotic Football Club (RFC) is an on-campus club involved with building and piloting robots to play a game of eight-on-eight football against other universities. The game is meant to mimic collegiate football as closely as possible, including rules, player positions, and designed plays. While there are not many schools with robot football clubs, it is an ever-growing sport, with universities like Notre Dame, Valparaiso, and Purdue fielding teams. The schools compete at several scrimmages throughout the school year, which culminate in an end-of-year tournament to decide the national champion.

Beyond the substitution of robots for people and the decrease in the number of on-field players, the biggest difference between college football and robotic football is the reward for a pass completion. In robot football, a pass is considered a completion if the ball touches the wide receiver in the air; the wide receiver does not have to actively catch a thrown ball. If the offense successfully throws the ball to a wide receiver, the offensive team is awarded two points. This is important, as a team can still score points even when they do not get a touchdown. Passing is thus a vital role in any team's offensive game plan.

The rules committee decided upon this because completing a pass is a complex endeavor. To complete a pass, the player controlling the quarterback will select a wide receiver to throw to. The quarterback will then lock on to and measure the distance between the wide receiver and itself, and must compute the angle and velocity at which

to throw the ball. For the Notre Dame RFC, this information is being gathered through a Pixicam. While the Pixicam system works, it can be easily interfered with if a lineman blocks the camera. Further, it has trouble calculating an accurate distance directly between the chosen wide receiver and the quarterback.

Due to the complexity involved in completing a pass, the quarterback is arguably the most important player on the team. If a team is able to consistently complete a pass, they will be continually accruing points, thus potentially lessening the need to score touchdowns to win a game. However, because completing a pass is so difficult, many teams opt to simply the run the ball for the majority of a game, and attempt a pass only if they are winning (or losing) by a large margin.

The Robot Football Club thus tasked us with creating a new indoor positioning system that can be implemented with their robots, and would eventually replace the need for the Pixicam. Last year, a team of students were able to demonstrate a working indoor positioning system, but it only worked in isolation and could not be easily connected to the robots.

Our team's goal was to improve last year's system. The first step was to improve upon the previous team's board by making it smaller, so it can more easily fit on a robot, and make it I2C compliant, so that it can directly communicate with a robot. These new boards can be placed on any robot, as well as three pylons. For calculating the distance between two points, a Decawave Module 1000 chip (DWM1000) was used, as it can deliver the accuracy and precision required for our project through RF flight time calculations. Using a system of three pylons at designated points around the playing

field, a quarterback and any wide receivers can find their (x,y) coordinates relative to the field through an algorithm known as trilateration.

This process is continually looped through, so the quarterback and each wide receiver will constantly know its position. When the person decides to throw a ball a specific receiver, he or she presses a button on their controller, which will then initiate a communication between the wide receiver and the quarterback. This is done through the DWM1000, as it is IEEE802.15.4-compliant, and the wide receiver will send its position to the quarterback. Then, through simple triangular calculations, the quarterback can find both the distance and angle between itself and the receiver. This information is then relayed to the actual QB by I2C protocol, and the quarterback can theoretically use that information to calculate how fast its wheels must spin, and how high it must aim, in order to complete a pass.

Our plan and design were solid, but unfortunately we were unable to meet all of our desired requirements and goals. The biggest disappointment was with the DWM1000 ranging module, which claimed to have an accuracy of +/- 10 cm. In reality, our measurements were consistently 20 inches or more above the actual distance between two DWM1000s. In addition, there was an error with our I2C code where it responds to every address sent, rather than only acting when a specific address is used. As a result, our system cannot at this time be fully integrated into a robot, but our code is such that when fixed, it will be much easier to implement our system. Our trilateration algorithm works, and is much improved over the previous team's code, but because of the inaccuracy of the DWM1000, it does not correctly calculate the position

of a robot. Individually, all of our subsystems work robustly, and with a few small corrections, they could be seamlessly integrated into a product RFC could use soon.

2 Detailed System Requirements

2.1 Board Requirements

- Boards function properly on USB and lithium-ion battery power
    - Takes input voltage of 5V, and steps it down to approximately 3.3 V
    - Board can charge lithium-ion battery
- Decrease size of board from previous year (at least 1 to 2 square inches)
- Has I2C MOLEX ports to connect to robot
- Configurable switches to designate various boards as “initiator” or “responder”
- Able to be mounted on robots and pylons
- DWM works with the board/microcontroller used

2.2 Pylon Requirements

- Pylons are sturdy - able to withstand hits from a robot throughout a game
    - Weighted down in some fashion
- Platforms will sit around the same height as where the boards will sit upon the robots
    - PCB’s will fit comfortably on the pylons platforms
- Has ventilation to avoid PCB and battery overheating
- Doesn’t interfere with DWM1000 ranging signals

2.3 Ranging - DWM1000

- Accurate to within +/- 10 cm of actual location
- Can track multiple DWM1000 modules
    - Minimum of five - three pylons, one quarterback, one wide receiver
- Determine the angle between the quarterback and the wide receiver
    - Used for the quarterback to have a vector distance and angle to throw.
- Can range out to approximately 50 feet
- Maintains accuracy when two DWM1000s are within approximately 5 feet of one another

2.4 I2C Communication

- Communication position data from PIC32 to Arduino with minimal data error.
- Configure to communicate as a slave, with the arduino as the master.
- Must function in an environment with multiple slaves and one master.
- Must have minimal delays to ensure no lag in passing commands.

3 Detailed Project Description

3.1 System theory of operation

The entire system consists of two or more robot players (one quarterback and at least one wide receiver) and three pylons. On each of these will be a PCB board

housing, among other components, a microcontroller and a DWM1000 module. The pylons will be positioned as shown in Figure 3-1 on the next page. When a person controlling the quarterback decides to throw a pass to the wide receiver, they push a button on their controller. As soon as this happens, the microcontroller on the board on the quarterback will send several signal pulses through the DWM1000 module, configured to specific addresses corresponding to the three pylons. When each of the pylons receive the signal, it will respond with a pulse of its own, intended for the quarterback. As the pulses are received by the QB, the microcontroller will use RF time-of-flight calculations to determine the distance between the QB and each of the pylons.

Once the QB has calculated its distance from each of the pylons, it uses a function known as trilateration, which allows the QB to calculate its position as a series of x- and y-coordinates relative to the field. By doing so, the QB is now able to determine the distance and angle between a wide receiver and itself. As all this is happening, the wide receiver is also continuously sending out pulses to the three pylons, determining its x- and y-position, and updating and storing its location on its microcontroller.

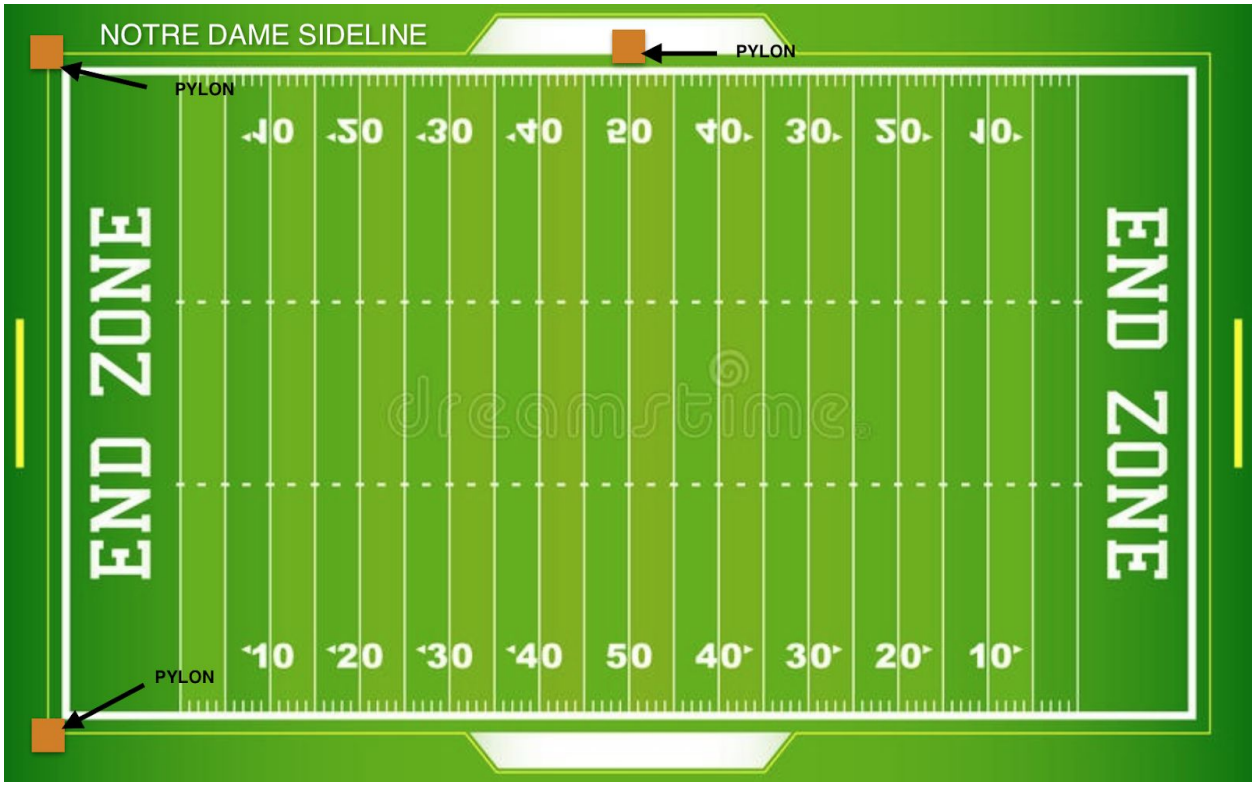

Figure 3-1: Pylon position placement. Pylon 1 is located in the top left corner, pylon 2 is at the 50 yard line, pylon 3 is in the bottom left corner

When commanded, the QB will send another DWM1000 signal, addressed to the wide receiver selected by a player. Unlike the time-of-flight signals, this signal will request the location of the wide receiver. The WR, upon detection of this signal, will send its most recently stored position back to the QB. Once the quarterback has the location of both itself and the wide receiver, it will use a function to calculate the distance and angle between the two. This data is then relayed through an I2C-Arduino interface to the Teensy microcontroller of the QB robot, where it will adjust the wheel speed of the throwing mechanism accordingly. The QB will also adjust its heading to ensure it is facing the wide receiver by comparing the calculated angle and the x- and y-heading calculated by its on-board magnetometer. When the wheel speed is adjusted

to the correct speed and the QB is properly aligned to the WR, the ball will then be thrown, and (hopefully) the WR will successfully catch it.

3.2 System Block diagram

Figure 3-2 is a block diagram detailing how our system fits together. The positioning system box shows everything that is located on the board. The Dip Switches configure the board as an initiator or responder, and set the address of the responder. The Magnetometer is used to orient the initiator. The DWM1000 communicates wirelessly with each pylon to determine distances and informs the PIC32 through SPI. The PIC32 slave gives all position data to the Arduino master through an I2C interface. The boards on each pylon are identical to those on the QB and other bots being located, but do not use their I2C pins to communicate with an outside microcontroller.

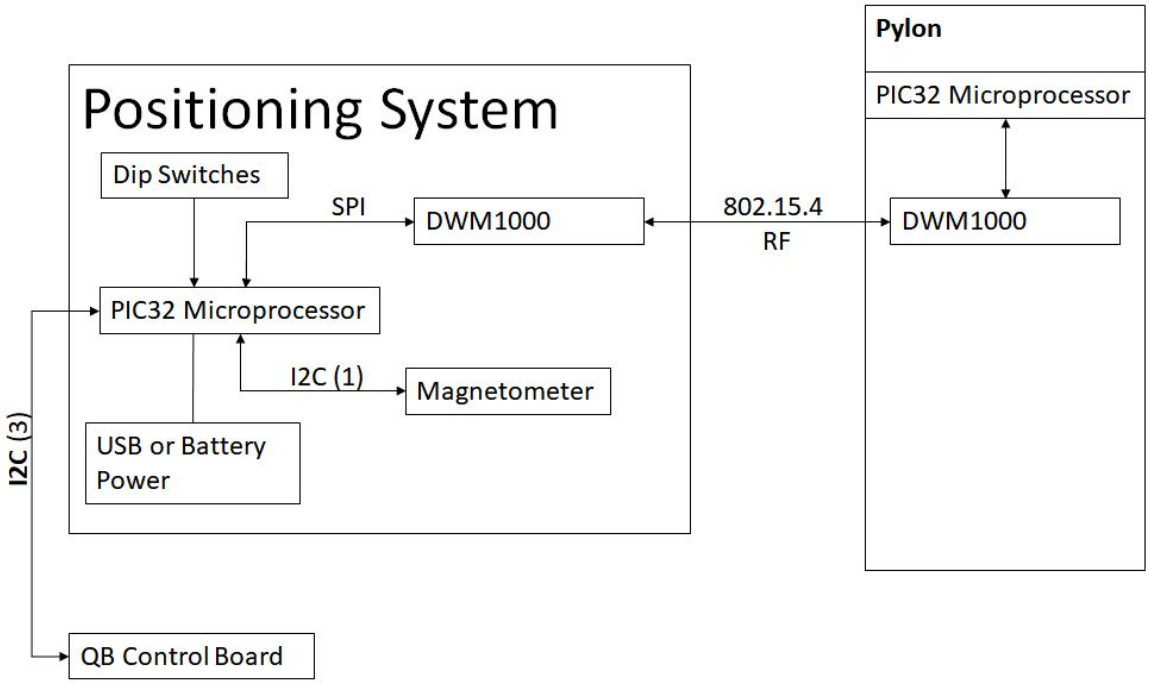

Figure 3-2: Block diagram of the system.

3.3 PCB Board Design

As this project was a continuation of a senior design project from the previous year, one of the first steps of the project was to analyze their board and decide how (if possible) to improve upon it. Below is a comparison between last year's board and this year's board.

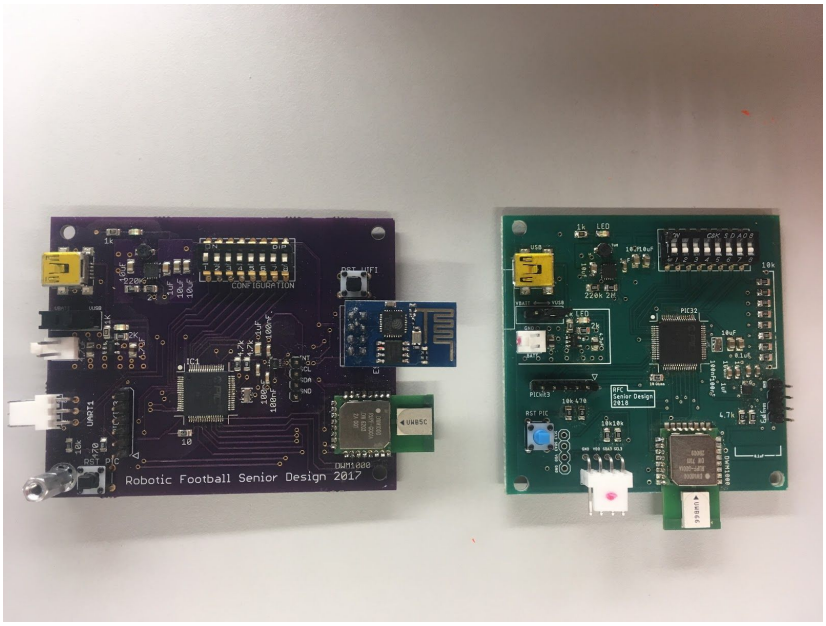

Figure 3-3: Left: last year's board design. Right: our board design.

Our team decided early in the process that an ESP8266 module would not be necessary, so both it and its reset button were removed from our version of the board. As a result, a significant amount of empty real estate now existed. This, combined with rerouting and moving parts around, allowed our group to ultimately decrease the size of the board by 2.5 square inches. This can be seen in figure 3-3.

Both boards share the following major devices: a TPS61200, an MCP38371, a DWM1000, a MAG3110 magnetometer, and a PIC32MX795 microcontroller. The main

function of the TPS was that of a voltage regulator, ensuring that the voltage reaching the rest of the board was roughly 3.3 V, and that the current was no higher than 600 mA. The MCP38731 allowed the board to recharge its 3.7V lithium-ion battery. This was crucial, as the boards must be able to run isolated from a charging cable for about an hour, roughly the length of a game of robot football. In addition, by recharging the battery, it eliminated the need to constantly buy new batteries. Were a game to last longer than an hour, it was also important that the batteries could be easily and quickly swapped out for fresh ones. The current iteration of the board can operate on a mini USB power supply while also charging a battery, or run just on battery power. A toggle switch chooses between the two power supplies

The DWM1000 ranging module is the crux of our indoor GPS system. By utilizing RF time-of-flight data, our project would be able to track the distance between any two PCB boards, whether they were placed on a robot or a pylon. This, combined with a trilateration algorithm, would then theoretically allow for an accurate positioning system. To allow for as little interference as possible from the board, and to avoid accidentally grounding the module, the DWM1000's antenna extends off the board. In addition, the DWM1000, because it is IEEE802.15.4 compliant, should be able to directly send messages to another DWM1000.

The microprocessor our team chose to use on the PCB boards was the PIC32MX795. Stored on it includes all of the code for trilateration, ranging, and communication with a robot, each of which uses a different type of code (SPI or I2C). Therefore, it was very important that the PIC32 is properly powered to 3.3V, and that it

could be programmed with a PICkit. The figure below shows the final routing and placement of all required components on the PCB.

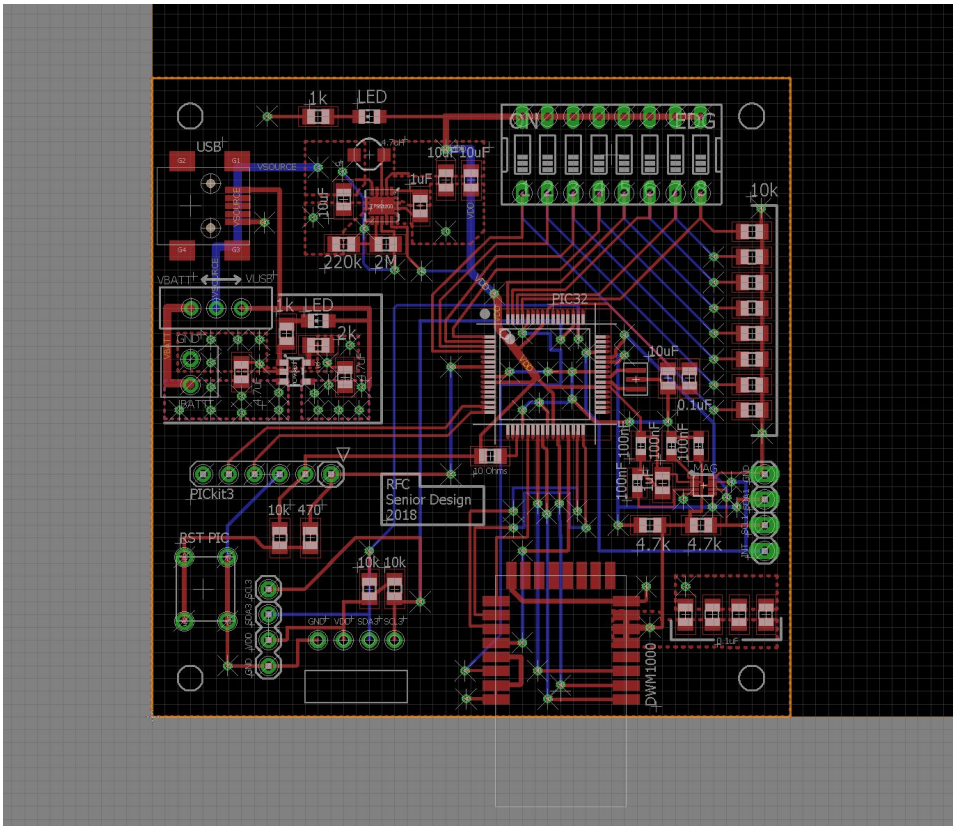

Figure 3-4: Board Layout in EAGLE software

Another revision to the old PCB board is the I2C MOLEX pin connection. This past year, the Robot Football Club reprogrammed their quarterback to be compliant with I2C protocol. They therefore tasked us with ensuring that our board could interface with their quarterbacks through an I2C port. Therefore, we replaced the three-pin serial UART port with a four-pin I2C MOLEX port. In addition, to ease routing problems, a ground plane was added to the top layer.

After looking over the initial design of the board, it was realized that the coupling capacitors were too far from the input to serve their purpose of filtering noise on an

incoming signal. Therefore, they were moved to the bottom side of the PCB, and placed directly underneath the PIC32 microcontroller to shorten the trace distance between the PIC32's inputs and the capacitors as much as possible. It was an original goal of ours to have no parts on the bottom of the PCB, so as to make attaching the board to the robots and pylons as simple as possible. In addition, there was a capacitor with the magnetometer circuit that wasn't necessary, so it was removed in the second revision of the board. The second version's board schematic is shown below.

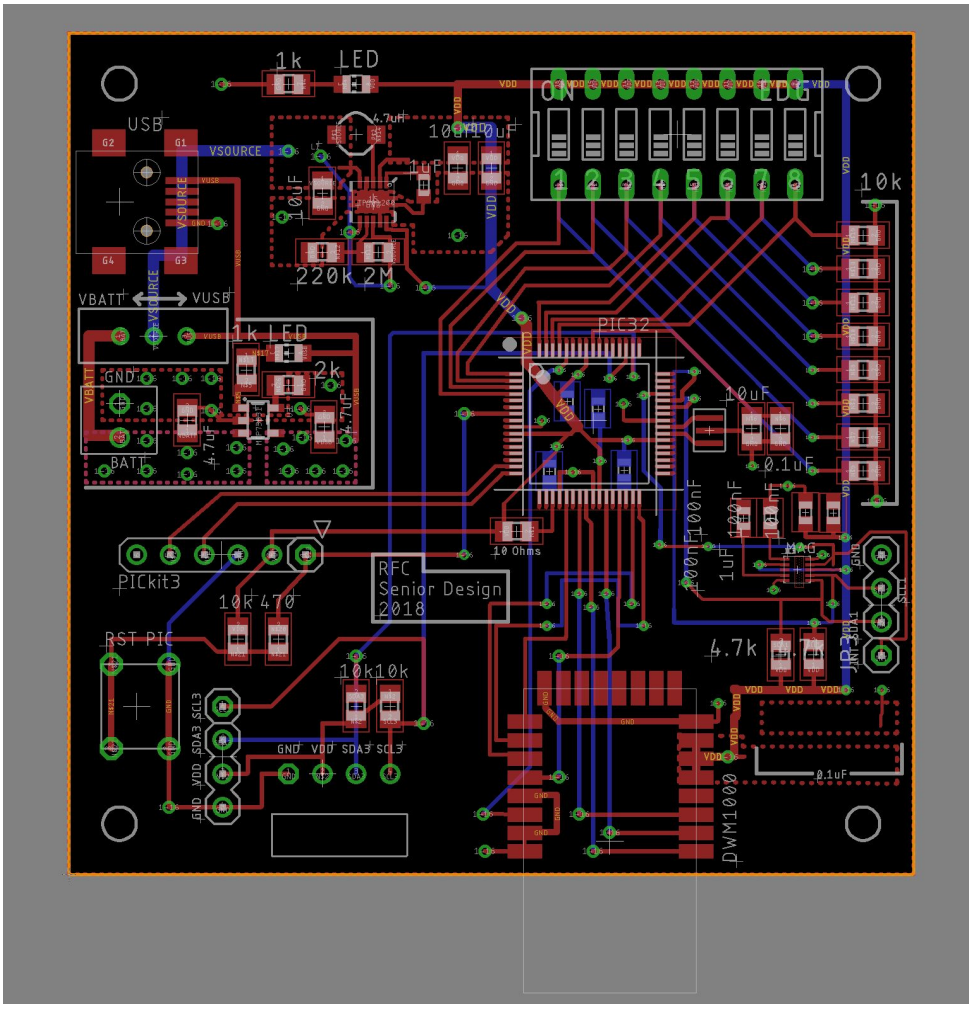

Figure 3-5: Board 2 Revision - the coupling capacitors have been moved, and one extra capacitor in the magnetometer circuit has been removed

Lastly, as seen in Figure 3-5 above, there are drill holes marked on the board, but when the boards were delivered by Osh Park, there were no holes drilled. This is either a mistake in the saved files sent to the fabricators, or if there was not enough room to drill them.

3.4 Microcontroller: PIC32MX795

The team chose to use the PIC32 as the board's microcontroller. This was the same microcontroller last year's team used. This allowed us to utilize much of their code without reconfiguring which pins would be used for each devices on the board. Each board can be configured using dip switches as either an initiator, or as a responder. As a responder, the address of each board can also be set with the dip switches. No one board will be confined to a single role as the boards can be set up at any time to run the uploaded code in a different configuration. The PIC32 is programmed using a PICkit3, and we used MPLabX as the compiler for all code uploaded to this microcontroller.

The PIC32 facilitates all communication between every module on the board. It communicates with the DWM1000 through SPI, the MAG through I2C, and the QB arduino through I2C. We are currently using the pins corresponding to I2C3 for communication with the Arduino, and the pins corresponding to I2C1 for communication with the MAG.

3.5 Detailed Subsystems

3.5.1 Ranging

Ranging between each board is done using the DWM1000 chip from Decawave. The DWM1000 is an 802.4.15 compliant device used for precision indoor location and data communication. It communicates with other devices wirelessly at a data rate of up to 6.8Mb/s, and communicates with the PIC32 using SPI protocol.

Much of the DWM 1000 code is proprietary and available for download off of Decawave's main site. We did not alter any of the code used for determining the distance between two modules. We did, however, begin writing code to send messages containing position data between two DWM 1000s. The prototype for this code is included on our documentation page.

3.5.2 Trilateration

Trilateration is done via the pylons and measures distance without the need to measure an angle when finding a position. It is done by sending out signals from three transmitters and seeing where the radius of each transmitter intersects (see figure 3-6). Trilateration is not meant to be confused with triangulation, as trilateration is implemented by determining distances only, where triangulation determines an angle. A benefit of trilateration over triangulation as well is that trilateration is not confined to the

triangle created by the three pylons; the location of the receiver can be calculated as long as it is within the range of the pylon (see figure 3-7).

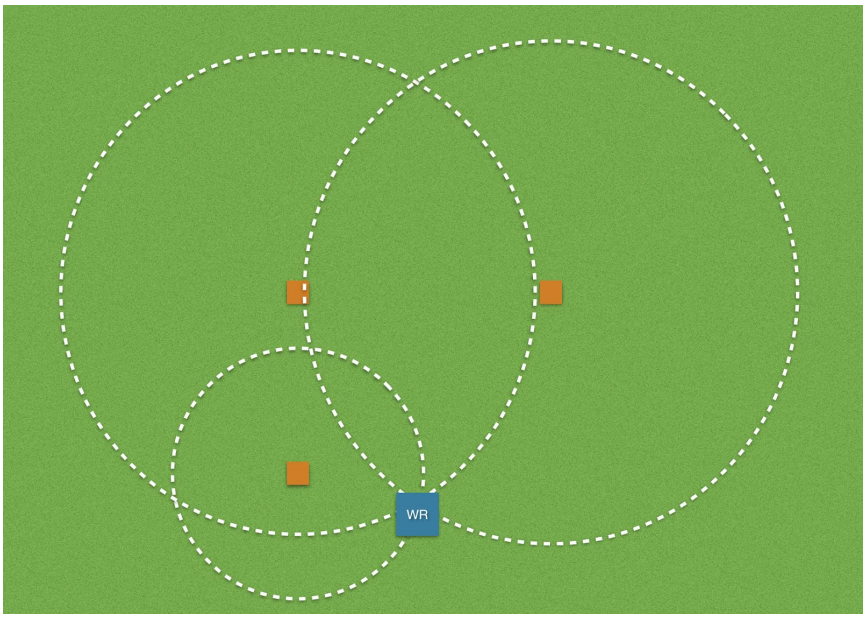

Figure 3-6: Trilateration overview of the three pylons determining where the receiver is currently located

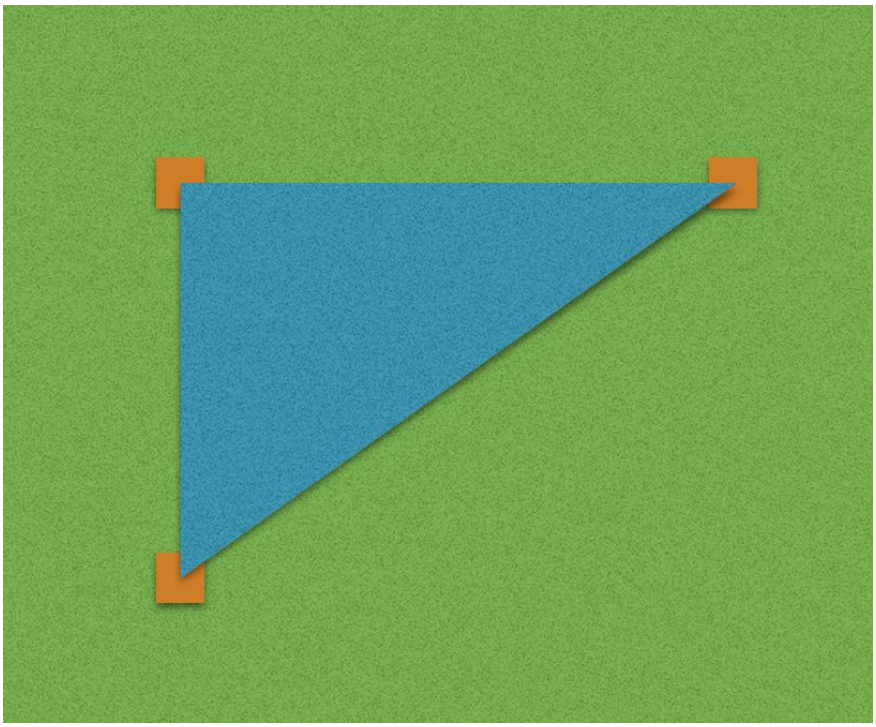

Figure 3-7: In triangulation, determining the location of an object is constrained to the triangle as shown. If the object does not lie within the triangle, the receiver can not be located accurately. Trilateration is not confined to this triangle, which is a large benefit for the RFC.

Trilateration is done by first sending a signal from one pylon to determine how far the distance is from the wide receiver to the pylon. This distance is referred to as the radius of the circle made by the pylons signal, but the receiver could be anywhere on the circle (see figure 3-8). Once this distance is determined, the next pylon sends a signal and the radius of that circle is the distance from said pylon to the receiver. Now both circles intersect at two different locations (see figure 3-9), meaning the receiver could be at either position. To determine which of the two the receiver is at, the third pylon repeats this process and the radius of that circle is the distance from the receiver to said pylon. This determines which of the two former intersections the receiver is located at, thus returning an x and y coordinate of the receiver with respect to the pylons.

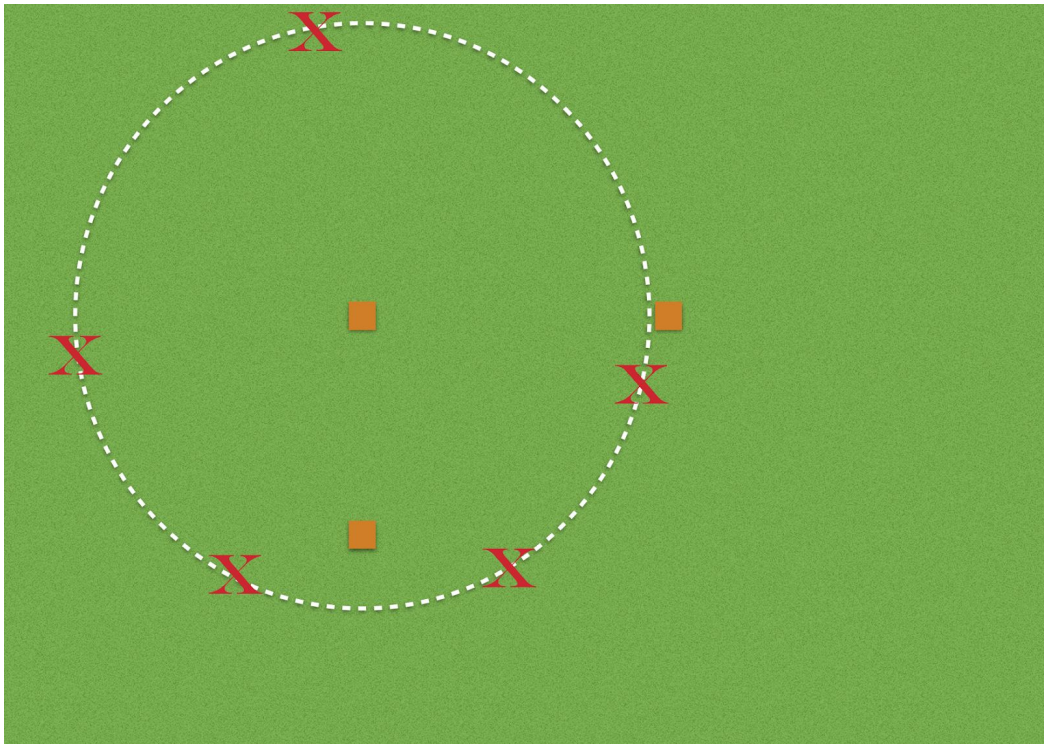

Figure 3-8: Each “X” represents a possible location of where the wide receiver could be after the first pylon determines the radial distance from itself to the receiver.

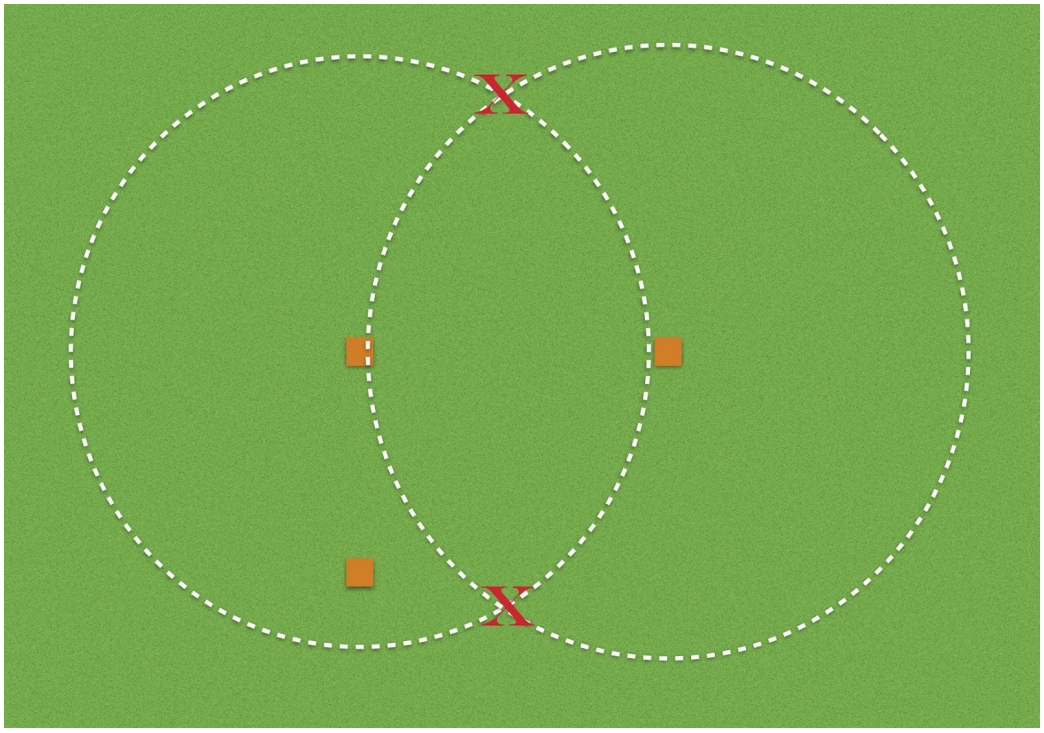

Figure 3-9: After the second pylon sends a signal, there are two intersections where the receiver could be. The third pylon's signal will determine which of the two the receiver is positioned at.

This system works best for this set up because the receivers could be on any end of the field, yet the RFC would not be confined to the area of the triangle that the pylons create. On top of this, the quarterback can also be located even if the receiver happens to lie in the immediate triangle. On figure 3-10, let's assume that the quarterback wants to throw to the wide receiver across the middle of the field. This means we want to measure the distance to the middle wide receiver with respect to the quarterback, and we must follow the ensuing steps: determine the wide receiver's location with respect to the pylons, determine the quarterbacks location with respect to the pylons, take the coordinates of the wide receiver and quarterback and subtract them compute the distance between the distance of the wide receiver with respect to the quarterback.

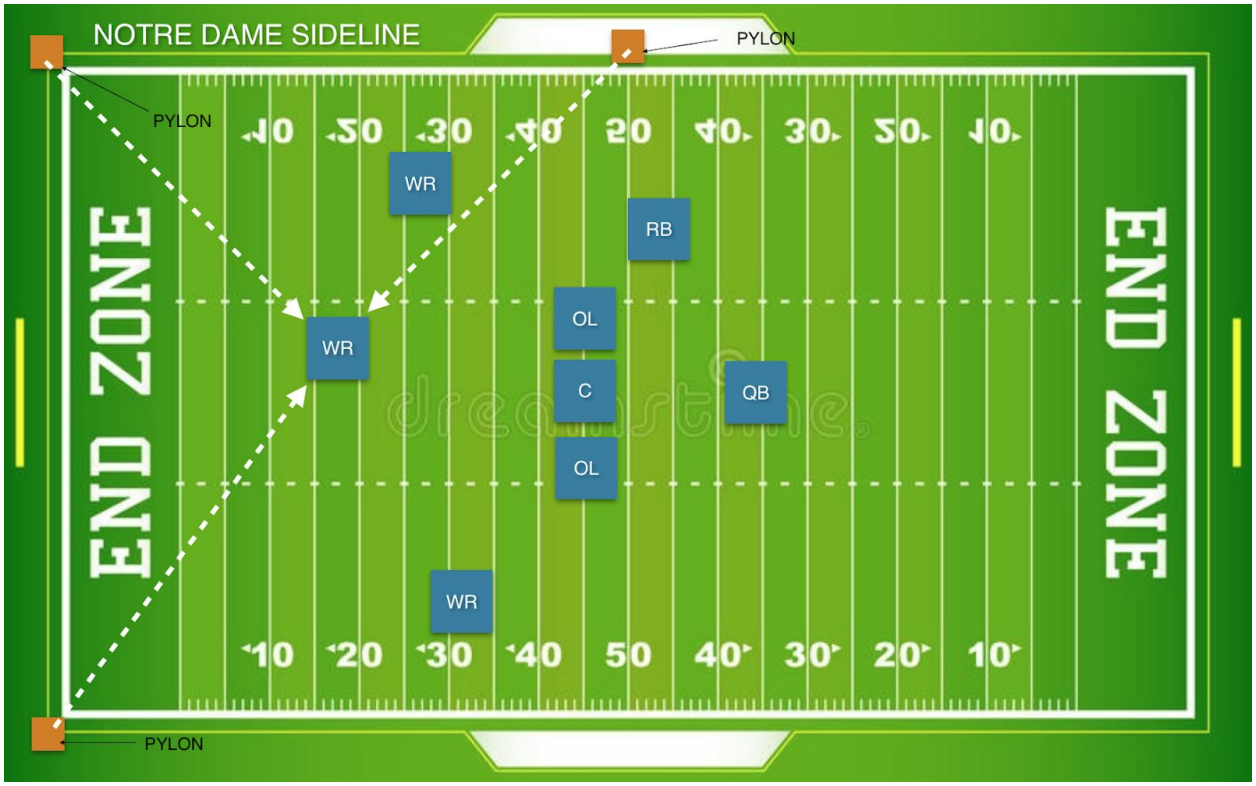

Figure 3-10: the dashed white lines represent the radial distance to the wide receiver from each pylon. The same process will be done for the quarterback.

For example, using figure 3-10 above, let us say that the wide receiver is measured to be located at position x = 25, y = 30 (the coordinate system is defined as the bottom left corner of the end zone is (0,0) and the top right corner of the end zone is (120, 53), also known as the length of a standard American football field). The quarterback is then measured to be located at x = 75, y = 26. We then take the quarterbacks coordinates and subtract them from the wide receiver, which in this example case gives us the position (-50, 4). Since the distance between the two is a vector, we must take the magnitude of the answer, meaning that the wide receiver lies at a position of (50,4) with respect to the quarterback's position. This allows us to no longer need the guess how far the location of the wide receiver is with respect to the quarterback.

Since most applications of trilateration occur in a three dimensional world, the actual calculation of the location is done not by simple circles, but spheres. Yet for our application, the value of z, or the height, is not necessary to calculate, as we only care about the x and y position of the robot. We use a two-dimensional plane to calculate the position of the robots. The basic equation for solving for the circle's radius location is given by:

$$(x - h)^2 + (y - k)^2 = r^2$$

Referring to figure 3-10, pylon 1 will be in the top left, pylon 2 will be on the 50 yard line on the Notre Dame sideline, and pylon 3 will be in the bottom left corner of the end zone. In the case of pylon 1, the values of h and k will be zero, as this is the pylon where the distance will be the anchor for each of the other pylons. This first equation is denoted as:

$$x^2 + y^2 = r_1^2$$

From here, we take the location of the second pylon with respect to the first one. This is determined by the equation:

$$(x - h)^2 + y^2 = r_2^2$$

The value of h is the in the x direction. This is the distance between the two pylons. Once this has been determined, the final equation is denoted as:

$$(x - h_1)^2 + y^2 = r_2^2$$

Finally, the third equation depends on where it stands with respect to the first two pylons. This is denoted by the following equation:

$$(x - h_2)^2 + (y - k)^2 = r_3^2$$

The third pylon may not be exactly directly “in line” with the first pylon (meaning the three pylons form a perfect 90 degree angle) so this is why we take into account the x and y offset from the “line” first two create with each other.

For our applications, we want to determine where all three circles collide, so we have to solve for x and y (since we will have sufficient information to get a precise location). After some rearranging of equations, it was determined that x and y are denoted as:

$$x = \frac{r_1^2 - r_2^2 + h_1^2}{2 \cdot h_1}$$

$$y = \frac{r_1^2 - r_3^2 + h_2^2 + k^2}{2 \cdot k} - \frac{h_2}{k} \cdot x$$

Since all of these values are known, we can easily solve for x and y. This allows us to get an x and y coordinate with respect to the pylons of both the wide receivers and the quarterback.

Using C for implementation in MPLabX, the group was able to simulate a way to determine the distance between two robots. Initially, by using figure 3-10, we calculated tested to see if the code could calculate the distance to the robots from where the pylons are. We hardcoded the radial distances from each pylon to the robots and successfully located where the robots stood on the field in the diagram. We were able to calculate the x and y position for WR1, WR2, WR3, the running back and the quarterback. Conceivably, the algorithm could track any robot on the playing field at any time, not limited to the receivers or quarterback.

From here, we were able to determine the vector distance between the quarterback and each robot along with the angle between them and successfully verified it based on the radial distances we hardcoded in. The next hurdle would be for the DWM1000's to update the radial distances to each robot so the algorithm can work in real time.

3.5.3 I2C Communication Protocol

I2C communication protocol is a synchronous, multi-master, multi-slave, serial computer bus. It needs only 3 lines between master and slave to communicate: SDA, SCL, and a common GND. The Serial Data Line transmits all data to and from each of the devices. The Serial Clock Line is the clock shared by all masters and slaves in the system. One bit of data is sent every clock pulse. Both the SDA and SCL lines must be pulled up to either 5V or 3.3V through ~2.4k pullup resistors. We use a 4th VDD line to take 3.3V from the QB to pull up our SDA and SCL resistors. Our system uses 7 bit addressing, in which the first 7 bits are the address of the slave and the 8th bit tells the slave whether the master is requesting a Read or is Writing to the slave. A typical Write from the master to the slave will work as follows. The master sends a Start command, followed by the 7 bit address, and then a final bit signaling a Write. If the address matches, the slave generates an ACK, and a slave interrupt on the falling edge of the 9th SCL clock pulse as seen in Figure 3.5.3.1. Then the master will send data, and the slave will read it out of the Receive Buffer.

We used this protocol to communicate between the Arduino running the QB, and the PIC32 running our positioning system. The QB is the master and our board is a slave. The QB already has multiple other devices slaved to it, but has a free molex port for us to attach to on the board. This system requires very few lines to function with a large number of slaves compared to other communication protocols. For example, SPI needs a separate Slave Select line for every slave communicating with the master.

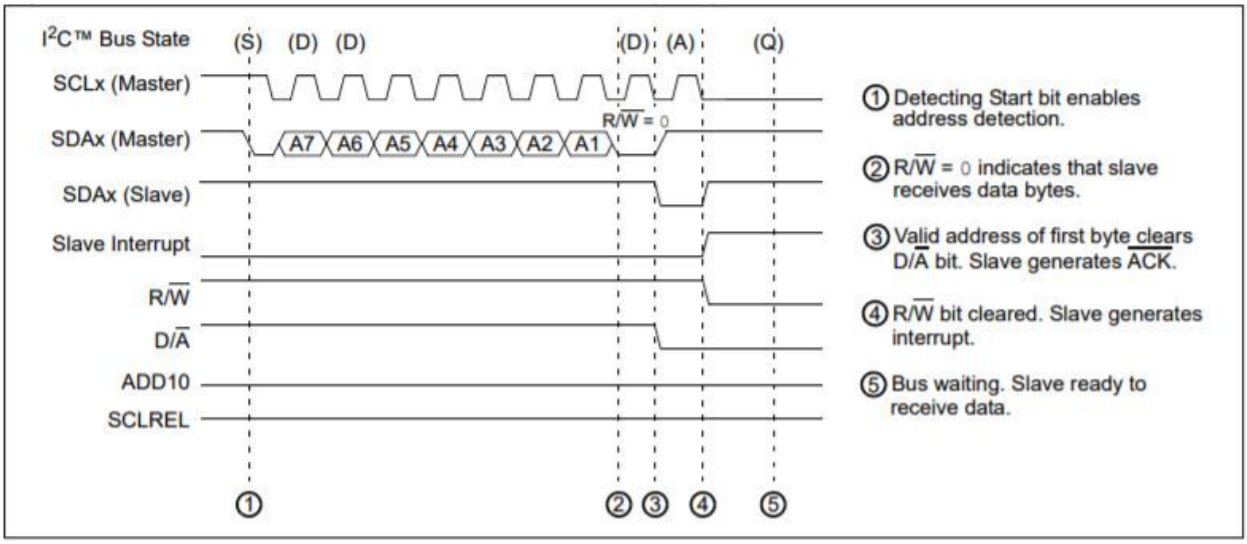

Fig 3.5.3.1

3.6 Pylons

The pylons serve as the main anchors to the project for positioning. The system relies on the pylons to trilaterate the location of the wide receivers and quarterback so that the distance of any receiver could be calculated with respect to the quarterback regardless of where it lies on the playing field.

3.6.1 Pylon Build

The pylons are constructed out of a high density plastic (HDPE solid polymer material, also known as hydro-polyethylene) of 1/4 inch thickness. The benefits of using this material are vital to the project for multiple reasons, the first being that it is both lightweight and very strong. One advantage to using HDPE is that in a high speed activity such as robotic football, we do not have to worry much about potential damages to the pylons due to fact that the material is impact resistant. Should a robot happen to collide with one of our pylons (which they will often), there is a low probability of suffering significant damages as I predict these pylons can be used for years. Should the pylons take too many hits and begin to break down, the Creo files are readily available to manufacture new pylons via the Techno machine in the machine shop of Stinson Remick.

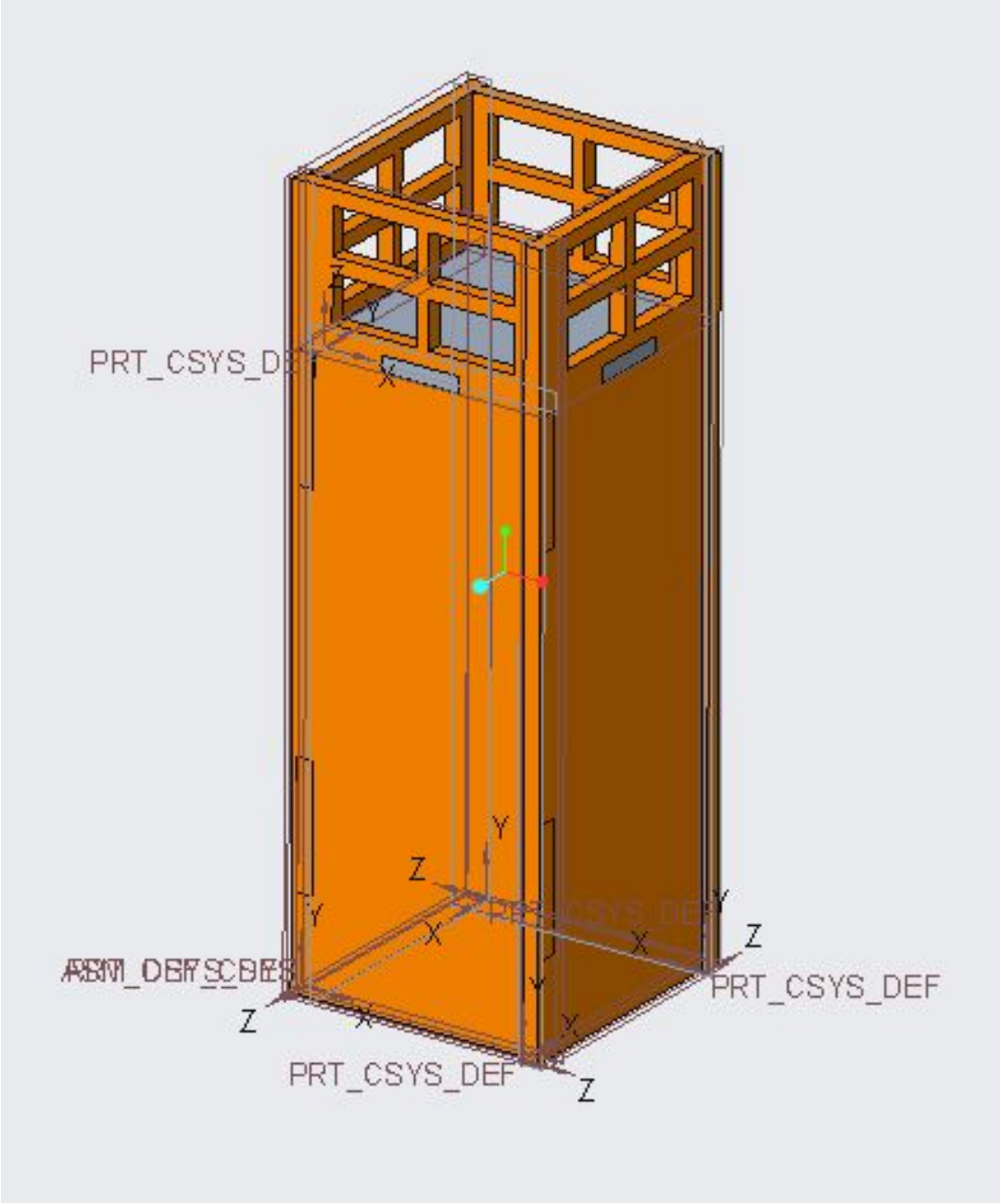

Figure 3-11: The final pylon assembly modeled in Creo 4.0

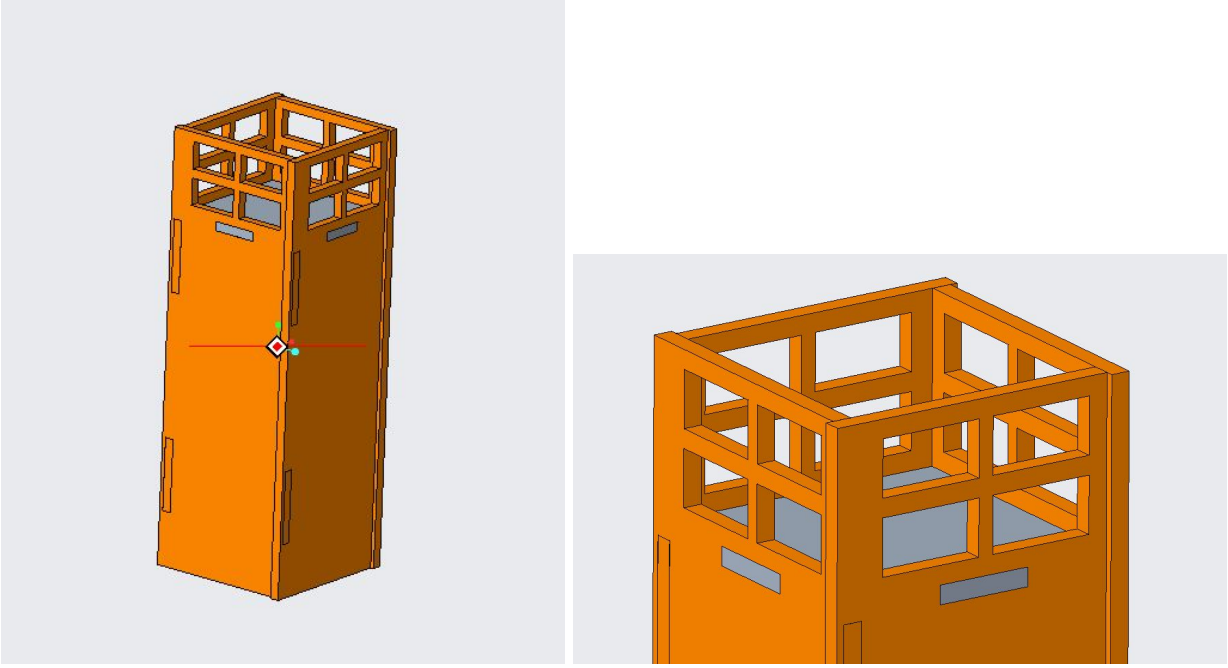

Figure 3-12: Fully pylon structure modeled in creo (left). Top view of the pylon platform with ventilation windows (right).

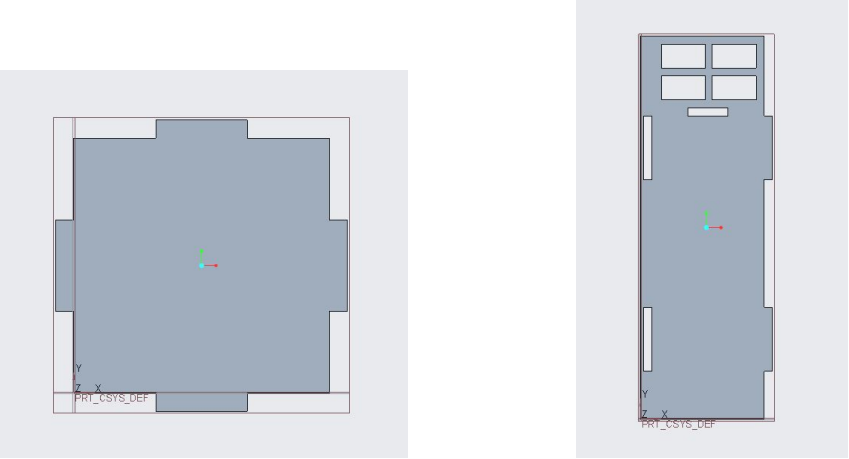

Figure 3-13: View of the pylon platform on its own (left). Pylon wall structure (right).

3.6.2 Pylon Design

As mentioned above, the pylons were designed by Peter Ryan using Creo 4.0, a CAD based computer software for modeling mechanical systems. When fully constructed, the pylons measure at 12 inches by 4 inches by 4 inches in length, width and height respectively. Sitting about 9.5 inches above the base of the pylons is a 4

inch by 4 inch by 1/4 inch platform which acts as the resting place of the electronics. To accomplish this design, three different Creo files were created, one which included the design of the pylon walls, another that modeled the design of the pylon platform, and the third that modeled the assembly of the system as a whole.

Each pylon wall was designed to be 12 inches in height by 4 inches in length, (along with the 1/4 inch width, or thickness due to the material). On the right side of the pylons we have two pieces sticking out 1/4 of an inch ranging from heights 1.5 inches to 3.5 inches and the other ranging from 7.5 inches to 9.5 inches. The left side of each wall contains two holes for the pieces to fit into: one ranging from 1.5 inches to 3.5 inches and the other ranging from 7.5 inches to 9.5 inches, each with a thickness of .25 inches. These holes were designed perfectly so that any pylon wall can fit into any other pylon wall, meaning order does not matter in the construction of the final product.

These blocks that stick out on the right as well as the holes on the left allow the four pylon walls to be assembled together as a puzzle piece system. We took another step by gluing the pylons walls together for more stability but we would recommend for the completion of this project that the pylons be secured down with 90 degree corner brackets instead of glue. Corner brackets holding the pylon walls together would be must more structurally sound and would be able to withstand more impacts than glue would.

At a height of 9.5 inches up the pylon wall, another hole was designed so that the pylon platform would be able to fit inside the main structure. The hole ranges from 9.5 inches to 9.75 inches (in the height direction) and are 1.25 inches in length. This is a

vital component to the functionality of the pylon itself as this is what will support the platform in which the electronics sit on.

At the top of the pylon there are 4 square holes which were drilled simply to ensure that the pylon has an impedance free way to communicate among the robots. The four squares are each of dimensions 1.375 inches by .75 inches. They serve as a means of ventilation for the circuit board which will be placed upon the platform should it begin to heat up. It also serves to give a clear path to communicate between boards. Also, on top of the benefits they serve, they also add a unique look to the structure while also staying true to the football theme. The pylon will not serve as a Faraday cage either because the material we used will not cause interference with the DWM1000.

The platforms were designed without any holes, only 4 pieces sticking out which were to fit into their respective holes near the top of the pylon. The platform was designed to be 4 inches by 4 inches by 1/4 inch with each jagged piece being 1/4 in length. The actual platform without the jagged edges sticking out is 3.5 inches by 3.5 inches to ensure that it fits comfortably after the assembly of each pylon. Each jagged edge is measured at 1.25 inches in length and goes from 1.125 inches to 2.375 inches. This was to ensure that the platform fits completely and perfectly through the hole, meaning no excess on either end. The circuit boards will fit perfectly on top of this platform and be stationary after drilling. Figure 3-16 shows an in depth analysis of how the electronics are secured on the platform of the pylon. Essentially, with the use of inch long spacers, inch and a half long screws, and nuts, we were able to elevate the board

off of the platform in order to help with ventilation, stability, and to create a place to store the battery that will power the electronics.

The original design of the pylon walls had a minor flaw, though, as a circular drill bit cannot create square corners when completing cuts. The main reason why this would be an issue is because when the pylons were to be put together, the pieces would not fit perfectly into each other. This would mean that we would have to file down each hole manually to ensure each pylon wall fits very snug. To combat this error, Peter inserted an extra circular cut in each corner of the two holes on the left side where the pylon walls would fit and the hole where the platform was to be inserted. This allows for each end of the cut to be inserted into their respective holes without having to do manual labor to file them down, and without ruining the aesthetic appearance of the pylons.

3.6.3 Pylon Creation

To create this design, the pylons were broken down into 3 phases (or cuts): wall holes, wall lining and platform cuts. It was essential that the interior holes of the pylon walls were cut before the profile to ensure that the pylons would stay secured as long as possible while the milling machine was cutting our plastic.

The first thing that was completed was securing the HDPE to the table of the Techno. The main purpose for this was to ensure that the plastic would not move while completing the cuts. A piece of wood lied in between the plastic and the base of the table. This was to ensure that the machine would not drill into the table itself during it’s cut. Next we loaded the NC files into the Techno software. Once the files were loaded

and the Techno machine was calibrated in the x, y, and z axes, we pressed the start button and the machine took it from there. Each cut lasted about 11 minutes and where cut with a 1/8 inch drill bit.

After waiting several hours for the Techno to cut 16 pylon walls and 4 pylon platforms, we played the pieces on cardboard outside and spray painted them orange to make them look more like realistic football pylons. Next, after letting the paint dry overnight went ahead and put the pieces together to construct four pylons. After that was complete we secured the circuit boards down on the platforms using screws, bits, and spacers. The spacers functioned as a means of protection for the coupling capacitors on the underside of the board, ventilation for the board should it heat up, and to create a place to store the 3.7V lithium ion battery. The bolt goes through the spacer and through the platform. Underneath the platform is a nut to secure the board down.

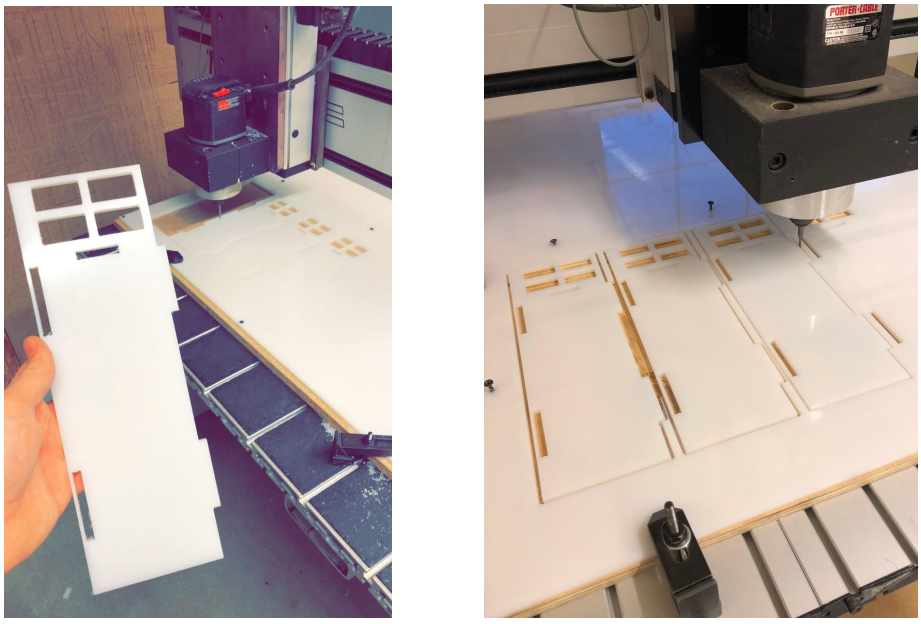

Figure 3-14: Completed cutout of a pylon wall (left). Construction of the pylon walls (right).

Figure 3-15: Left to right: completed construction of pylons (without platforms), final construction with platforms inserted and painted to mimic football pylons.

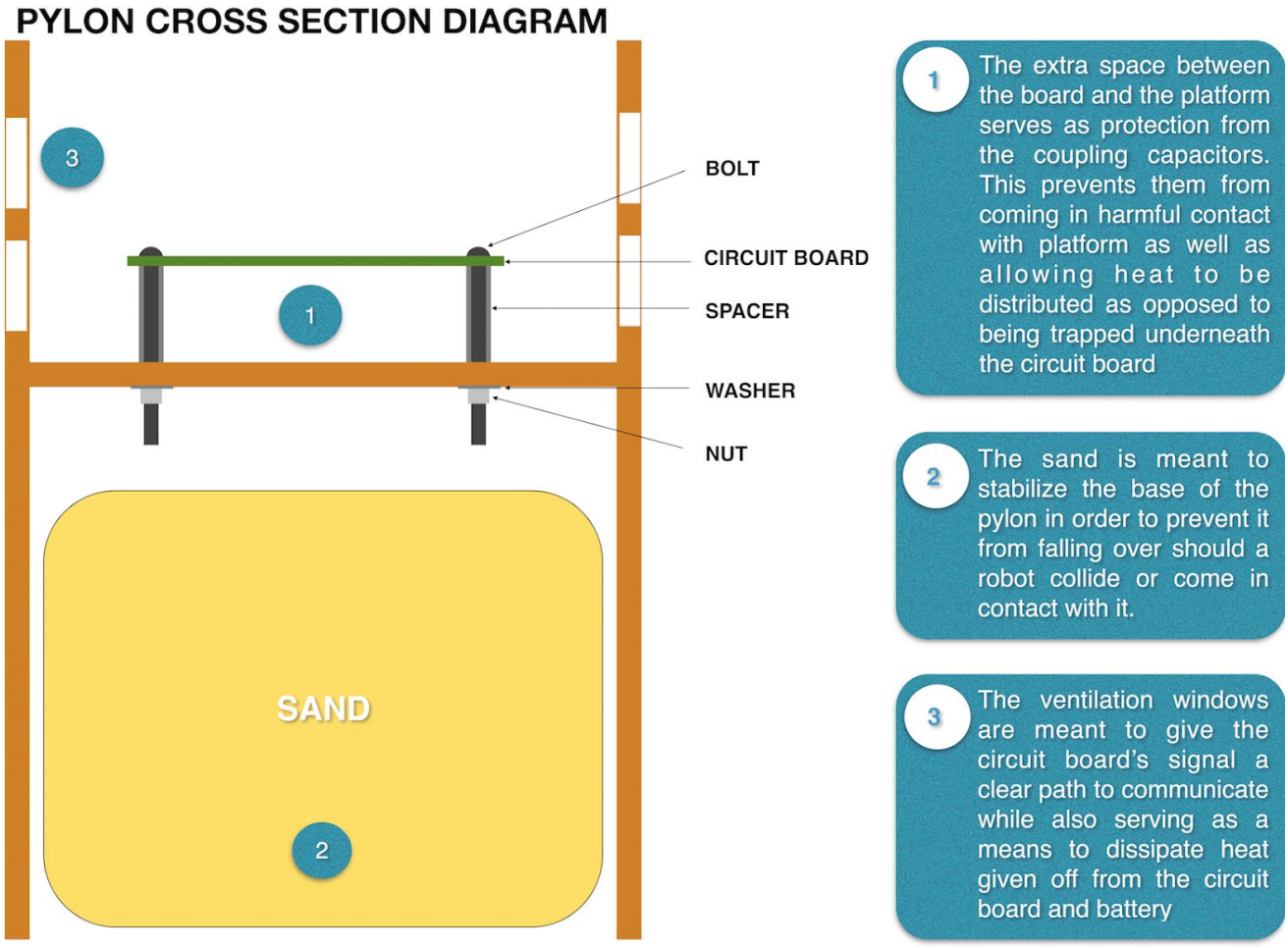

Figure 3-16: Pylon cross section diagram with descriptions.

3.6.4 Pylon Functionality

The pylons are meant to be anchors in this system which communicate directly with the wide receivers and quarterback. They serve in a similar fashion as telephone poles do when locating the position a cellular device. First, each of the pylons will communicate with the wide receiver to determine its distance with respect to the pylons. Once this distance is calculated, it will repeat the same process upon the quarterback. Once these two distances are calculated, the quarterback can compute some simple math to find out within a range of 10 cm where the specified wide receiver is with respect to the quarterback itself.

The pylons are designed to be placed in three locations, one in each corner of an end zone and another on the sideline at midfield. By placing the pylons in each of these locations, it allows for absolute optimization when trying to locate the robots regardless of where they stand on the playing field.

3.7 Interfaces

3.7.1 I2C Protocol

The team primarily used I2C protocol to communicate between the PIC32 microcontroller and the Arduino Teensy 3.5 mounted on the QB. The PIC32 is connected to the Arduino through SDA, SCL, GND, and VCC. SDA is the data line, SCL is the clock, and GND is a common ground. The VCC line is used to pull up the SDA and SCL lines to 3.3V through 2.4k resistors. If we were to redesign our board, we

would remove the VCC line and wire it into our VCC rather than pulling from the Arduino. I2C protocol is also used to communicate with the MAG mounted on the board, but this is within obscured functions and we did not alter any of this code from last year.

3.7.2 SPI Protocol

SPI protocol was used to communicate between the PIC32 and the DWM1000 chips. This communication protocol requires the use of 4 lines: MOSI, MISO, SCLK, and SS. MOSI stands for Master Out Slave In, and MISO for Master In Slave Out. These lines should be wired MISO < - > MOSI. SCLK is the clock line and SS is the slave select.

3.7.3 Wireless Communication

The DWM1000 is an IEEE 802.15.4 compliant device. This means that in addition to doing time of flight calculations between two devices, the user can also send messages utilizing 802.15.4 protocol. The documentation on this communication protocol is extensive (around 700 pages worth) and is available on IEEE's website. Sending messages wirelessly with these modules replaces the need for a separate ESP8266 module. Now, instead of sending distance data to an outside server for position calculations, this data is sent to the QB initiator, and positioning calculations are done using the PIC32.

3.7.4 Configuration Switches

The dip switches on the board are used to configure the board into an initiator, or a responder with four possible address. Switch 1 controls whether the board is an

initiator or responder, and switches 2 and 3 control the address of the responder. When switch 1 is ON, the board is a responder. When switch 1 is OFF, the board is an initiator, and switches 2 and 3 do nothing. When in responder mode, the address is configured using switches 2 and 3 according to the table below.

4 System Integration Testing

4.1 Description of Subsystem Testing

4.1.1 Printed Circuit Board

It was desired that our board be smaller than last year's team, which it is by approximately 2.5 square inches. In doing so, the board has more flexibility on where it is placed within a robot - in its current state, it can be connected on the side of the throwing mechanism with no issues. The board can also be powered by both a battery and a USB cable, which can be demonstrated by simply connecting one at a time, and flipping the power source switch accordingly. Further, the PIC32 can be shown to successfully download code from a computer, and it is possible to reset the PIC32 with the reset button. The I2C ports work as intended, and therefore the board can interface easily with another robot.

4.1.2 PIC32 Microcontroller

The PIC32 microcontroller was a vital component to the project. After verifying that the circuit board powered up properly, the PIC was programmed with the PICkit3. If

the program uploaded successfully, we moved on to testing all the I2C pins. If the PCB did not function properly after this stage, we tested to see if pins on the PIC were improperly soldered onto the board. When all soldering issues were fixed and the code uploaded properly, and the I2C communication sent position data from PIC to Arduino, then we knew the PIC32 was working properly.

4.1.3 DWM chip

When the DWM1000 chips first arrived, we tested them by connecting them to an arduino through SPI and using example ranging code. We were successful in determining the distances between two chips down an entire hallway, verifying that the DWM's did in fact function properly. After ensuring they worked, the chips were soldered onto our newly designed boards using the PIC32 microcontrollers.

In order to make the boards smaller, we had to remove the ESP8266 Wifi modules, which meant we could no longer send data wirelessly using the previous team's code. To work around this, we attempted to use the DWM1000 to send position data wirelessly, as they are 802.15.4 compliant devices. While in theory this should have worked, we did not have enough time to implement this upgrade. The example code provided by Decawave proved useful, but our main problem was properly initializing frames, the standard method of sending data over this communication protocol. Below is one of the tables outlining a standard outline of a communication

frame, taken from IEEE's extensive 802.15.4 guide.

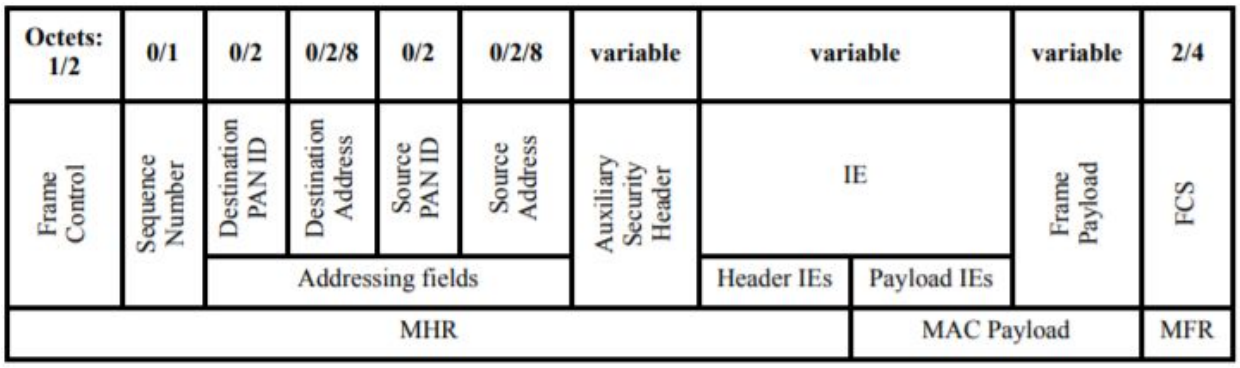

Figure 4-1: IEEE's 802.15.4 communication guide.

4.1.4 I2C Communication Between PIC32 Board and Arduino

We first tested I2C communication using last year's boards and an arduino uno as placeholders. The arduino side of the code was written to request a read when the character "r" was sent to the serial monitor, and to request a write when "w" was sent to the monitor. Any data sent to the arduino was output to the serial monitor. We also used a Logic Analyzer attached to the SDA and SCL pins to see everything being sent back and forth between master and slave, and to see if each bit was being sent on the correct clock cycle. Using this method, we were able to get reading any number of bits working consistently. Writing worked sporadically, and had issues with the slave sending a NACK immediately upon receiving an address.

We tested to see if ranging data was being updated properly by turning on all three pylons and determining the position of an initiator placed on a moving chair. If moving the chair updated the values correctly, we knew that the protocol was working correctly.

When integrating our system with the QB in a multi slave environment, we discovered that address recognition was not working properly on our board. Our slave was attempting to take control of the bus with every address it was given, not just the one it was assigned. This froze up the QB's communication system. We confirmed this by sending the wrong address to our slave and monitoring its response on a logic analyzer. It is likely we were not checking for an interrupt flag properly or made an error when coding the state machine that handled the I2C transactions.

4.1.5 Trilateration

Trilateration was tested with various simulations. The issues with the DWM1000 measurement inaccuracies meant that fully testing the trilateration code with information from the DWMs was futile and led to poor results. However, it was possible to simulate a playing field, and "hard code" both the position of the pylons and the distance of each pylon to a robot. After doing so, we can run the trilateration code and show that we can definitively know the (x,y) position of a robot relative to the field.

To run a simulation, we set the positions of the three pylons exactly where they would be on a real robot football (one at each corner of the ND endzone, as well as a third along the sideline at midfield). These distances are then put into the code and will not be altered - since they will always be at the same position, the location values of the pylons do not change. A fourth board (simulating a robot) is then placed anywhere on the field, and for the purposes of our demonstration, the distance from it to each of the three pylons is manually calculated and put in the code. In our full implementation, these distances from the pylon would be provided by the ranging data of the

DWM1000s. Once this is done, the program can be run, and the (x,y) coordinates are outputted. We can also “hard code” a second robot on the field, and then show how our code calculates the position of both robots, and can find the distance and angle between the two. The exact code used for this simulation can be found on the website, under Documentation.

4.1.6 Pylons

The pylons, not being an electrical system, were not tested like the other subsystems. However, it can be shown that the platform that holds the PCB is at essentially the same height as where the board would sit on the robot, which eliminates the need to take into account height differences for calculations. In addition, the pylons have suitable venting capabilities that prevent the battery and board from overheating. The pylon, when put together with Gorilla Glue, is quite sturdy, and with sandbags within the pylon, it should be able to withstand a hit (or multiple hits) from a robot.

4.2 Meeting Design Requirements

Our testing was done to ensure that all project design requirements were met to the best of our abilities. The only thing that could not be shown was the full integration of all systems and code, though every other subsystem works as expected and to the requirements of our design. The trilateration code worked as intended when simulated, but the inaccuracy of the DWM1000 module prevents a full implementation of the two subsystems. The code is written so that they are already implemented, but the output of the trilateration will be incorrect until the DWM ranging problems are resolved. The PCB

functioned as we expected and required it to, and the pylons served their dual purpose of both protection of the boards and positioning through trilateration. The I2C-Arduino interface is nearly correct - once it correctly responds to only its address, the project will be ready to implemented with the robots, and the RFC should be able to much more consistently complete a pass between a quarterback and a wide receiver.

5 Users Manual/Installation Manual

5.1 How to install your product

For our installation, we are under the assumption that the user is in possession of all fully constructed pylons and circuit boards. We are also under the assumption that the user has both installed and has direct access to MPLabX IDE as well as access to a PICkit3. The code (found on the website) must be built and programmed onto each PCB board, and the code is the same for the quarterback, wide receiver, and pylons. All libraries needed are included in this project file. There is also an assumption that the user has access to Arduino IDE for purposes of programming the quarterback, as well as testing the I2C master-slave code.

5.2 How to setup your product

To set up this system, plug the PICkit3 programmer to your computer and a PCB board, which is connected to a power source. Open MPLabX, and click on the “Make and Program Device Main Project” button (follow the on screen instructions should any pop up during this process). After compiling and uploading the code to the PCB, ensure

that the configuration bits on the board that is communicating with the Arduino (in other words, the Initiator) are all set to zero. Any board you are measuring the distance to (known as Responders) will have the first switch set to 1, and their addresses controlled by the next 2 switches. These addresses can be set from 1 to 4 in binary (00,01,10,11). Once the PCB is set up, switch over to the Arduino software, and upload the example code labeled I2C_Master_Example onto the Arduino. If it builds and uploads correctly, hook up the GND, SDA, SCL, and 5V pins on the Arduino to the respective pins on the PIC32 board (found on the MOLEX connector). Once this is done, hit the “reset” button on each board.

5.3 How the user can tell if the product is working

Once the programs have been successfully downloaded to both the PIC32 and Arduino on each circuit board, the user can open the Arduino serial monitor. By sending the character “r” to the Arduino through the serial monitor, the Arduino should initiate a distance read with the PIC32 based ranging system. If the output on the screen displays the distance between the initiator DWM1000 and responder DWM1000, then the system is functioning properly. The Arduino example code cycles through each of the four boards. The actual RFC QB code should be uploaded to the proper Arduino for installation inside the QB (QB code not posted to protect the intellectual property of the RFC. Please contact RFC directly for permission to use code).

5.4 How the user can troubleshoot the product

Your first step when troubleshooting the boards should be connecting a Logic Analyzer to the SDA, SCL, and GND pins communicating between the PIC32 and Arduino. This way, you can see whether the issue is with the I2C communication protocol, or if something else is causing problems. There may be a problem with both the Master (Arduino) and the Slave (PIC32) thinking they own the communication bus. To solve this, press the reset button on each board.

If the boards seem to be communicating but there is no data being outputted to the screen, make sure that each connection is good, and that the SDA and SCL lines are each being pulled up to 3.3V or 5V through the pullup resistors (see the schematic on the website).

If the issue does not lie with I2C communication, there may be an issue with the DWM1000, or its possible that there is a bad solder joint that is preventing the communication and ranging from working correctly. First, try to rebuild and reload each program onto their respective device. Should that not work, the issue may lie within a device on the board, and consulting the spec sheet and use of a multimeter are then the user's best bet. Also try and check if any pins on the USB connector or PIC32 are not fully pressed down on their respective pads; if this is the issue, use a soldering iron to reheat the solder joints and let them reflow.

6 To-Market Design Changes

A future improve would be to ensure that the boards can be easily swapped between different pylons. Currently, each board can only be paired with a specific pylon, because the boards we ordered did not come with pre-drilled holes. We manually drilled holes into the circuit boards in each corner, and they each only match up with one pylon due to human error in drilling.

The design of the pylons could be enhanced by restructuring the Creo files. Each pylon wall and platform could have an extra circle drilled into the pieces sticking out which fit into the security holes. This would eliminate the need to manually file down the pieces. Also, If we were to manufacture this product in bulk, we would use 90 degree brackets to secure the pylons together rather than using gorilla glue.

In our current design, only the quarterback is configured as an initiator. This means that the QB is the only bot getting its position in relation to the pylons. In order for the trilateration and distance measurement system to work as intended, the QB and each WR being tracked all need to be initiators. The WRs then need to send their position in relation to the pylons to the QB to compute its distance and angle relative to each receiver. The position data will need to be sent using the DWM1000 chips. We have included prototype code for sending a message between two chips in our documents page in order to help next year's team get this working quickly. This would allow the QB to know the position of *any* receiver at any time.

The routing on the current version of the PCB board could be improved to reduce the size of the board. As there is not much room on the quarterback, it would be prudent to continue to decrease the board size. The overall design of the board could also be analyzed to potentially discern a better overall layout for the PCB. Our board was simply a revision of, and a (significant) incremental upgrade to, last year's PCB, but it's possible that a vastly different layout could lead to a much larger decrease in board size. It would also be ideal to remove our 4th I2C line (VDD) being used to pullup our SDA and SCL resistors, and pull them up from our own boards VDD. This will ensure we pull no power from the QB.

Communication in an environment with multiple slaves and one master needs to be fixed in order for this system to fully integrate with the QB. Once address recognition is properly implemented, the rest of the system should work robustly. Consult the I2C section of the PIC32mx795 spec sheet for information as communicating as a slave, and the section on interrupts to ensure the Master Slave Interrupt Flag is functioning properly.

7 Conclusions

Our project goal was to improve upon last year's indoor positioning system, and fully integrate it with the Robot Football Club's Quarterback. This goal was partially met. Our biggest successes were redesigning a smaller board with an I2C molex connector, adding physical pylons to mount the board on, and converting the trilateration code to work on our PIC32 rather than on an external server running on a Raspberry PI. The

biggest improvements that need to be made are fixing the I2C code so that the slave only responds when given the correct address (not to any address), and improving the accuracy of the DWM1000 ranging modules.

8 Appendices

For full schematics of the new board, new positioning code, and I2C testing and example codes, please see our Documents page and download the SeniorDesignSoftware.zip file. Everything needed to install this system is included within this file.

DWM1000 Datasheet:
https://www.decawave.com/sites/default/files/resources/DWM1000-Datasheet-V1.6.pdf

MAG3110 Magnetometer Datasheet:
https://www.nxp.com/docs/en/data-sheet/MAG3110.pdf

PIC32MX6XX/7XX Datasheet:
http://ww1.microchip.com/downloads/en/DeviceDoc/60001156J.pdf

TPS6120x Datasheet:
http://www.ti.com/lit/ds/symlink/tps61200.pdf

MCP73831 Datasheet:
http://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf