PLAYER TRACKING CAMERA

By

Aksh Gupta

Oreoluwa Sunmola

Shivang Charan

Final Report for ECE 445, Senior Design, Fall 2021

TA: Feiyu Zhang

8 December 2021

Team No. 21

Abstract

The project, Player Tracking Camera, seeks to create a functional player tracking system in which a camera is able to follow the movements of an athlete across a playing surface. Bluetooth Low Energy (BLE) data regarding location information allows for the camera to quickly and precisely track the movements of the player. We, Team 21, were able to create a fully functional prototype that met our high level objectives. Our design resulted in a system that is both power-efficient and portable.

Contents

1. Introduction

1.1 Overview

In the digital age people love recording themselves and their surroundings as a way of remembering times in their lives. Whether it be on vacation, at a sporting event, or even at home, it has become commonplace for people to have devices to capture all these moments.

Unfortunately, it is very hard for the general public to record themselves while playing sports. The core issue is that individuals are usually not able to record themselves as they play. Therefore, they are required to ask someone to record them while they are playing. Professional sports broadcasters use bulking and costly equipment to track games; however, there is no product that is convenient and affordable for the common sports enthusiast. This is a major inconvenience and one which can be automated away.

1.2 Solution

In order to solve this problem, we intend on creating a moving stand that will keep the intended target in the view of their camera. This stand will be adjustable to fit most existing cameras. By setting up small beacons around the playing area we can calculate the position of the target person running - which would then be used to rotate the camera to capture the subjects' movements in real-time. Functionally, this would eliminate the need to have a cameraman follow the game's movements and would automate the filming process.

1.3 High Level Requirements

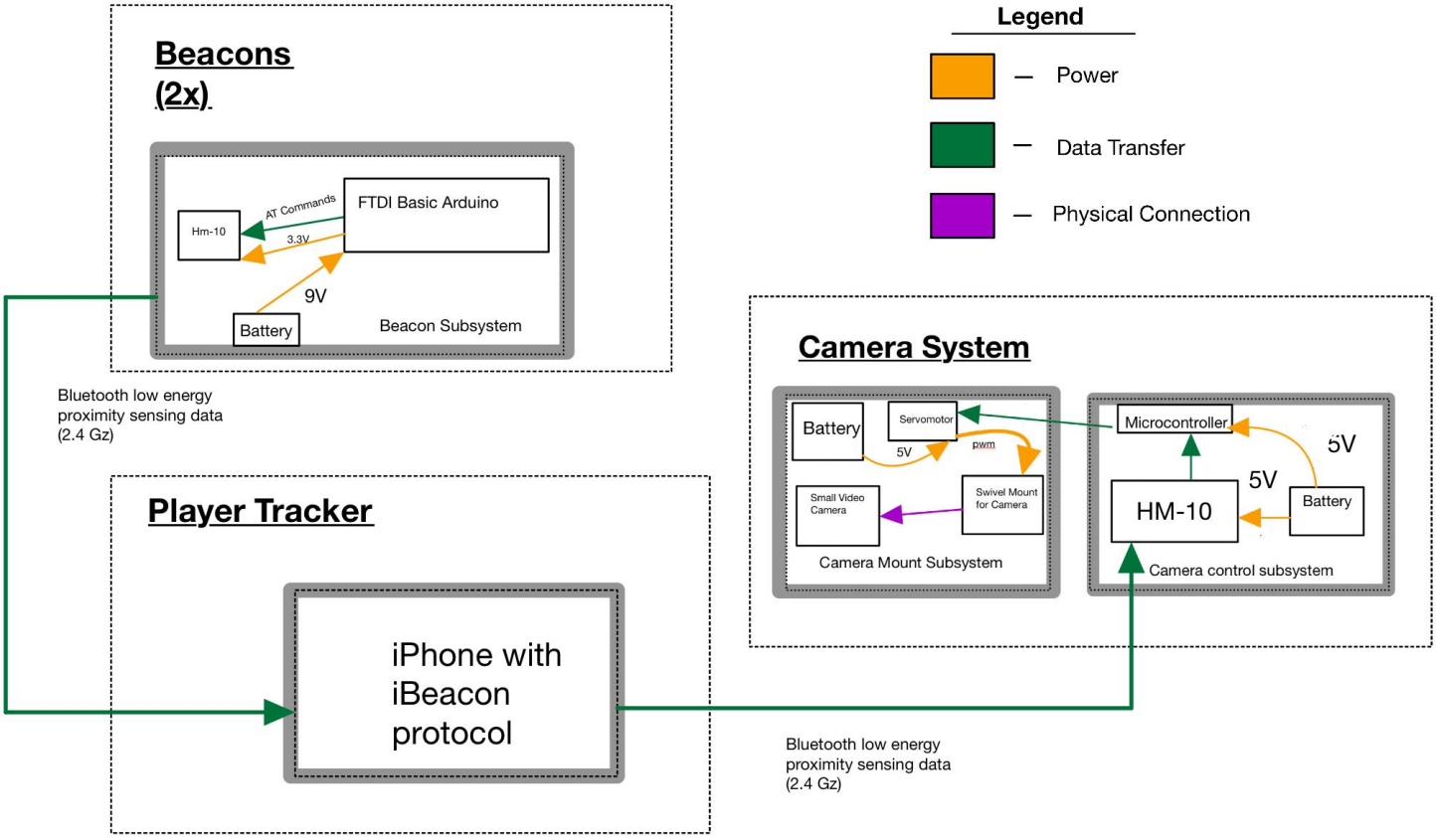
- BLE beacons should be able to create a unique 31-byte data packet and have it constantly transmitting, so that the player tracker is able to pick up the data packet.

- The location is determined using the iBeacon protocol between the BLE Beacons and the target player's iPhone to determine an approximate position within a 10 meters radius.

- The camera system should be able to keep the user within the camera's view for 90 percent of the recording duration and have the person within the frame but not necessarily centered.

2 Design

Our project design consists of four major parts: the beacon, player tracker, camera mount, and the camera control unit. The beacons and the player tracker work in collaboration via bluetooth to determine the player's location on the court and send that data to our camera control unit through bluetooth as well which moves the camera on the mount to capture the player.



2.1 Beacon

In our project we need a way to calculate the relative position of the user running around the court. However, since this project has the ability to be used for different sized courts or fields it would be difficult to make our project modular, so to make it modular we need beacons to create a perimeter around the court or field. The right side beacon represents the initial zero point on our coordinate axis and the left side beacon is the end point (28 meters).

In order to accomplish our high level requirements we had to develop two Bluetooth Low Energy beacons. Our Bluetooth Low Energy beacons are cheaper, faster, and more power consumption efficient than traditional Bluetooth beacons. Fast transmission speed was an integral part of our project.

Throughout a normal basketball game individuals are running around quickly and changing positions fast and we need our beacons to quickly transmit their individual 31 byte data packets. We chose BLE beacons because the typical speed for BLE is ~3ms where a normal Bluetooth is ~100ms. [3] The next part is cost since we want our product to be affordable for the normal consumer. We would like to use parts that are not as expensive. BLE chips are also 60 - 80 percent cheaper than regular Bluetooth chips. [7]

BLE Beacons would connect to the Player Tracker Subsystem, and then the Player Tracker application can triangulate the user's current position. Throughout the project we realized that we do not need two BLE beacons. Using the principles of SAS which account for knowing two sides of a triangle and one angle you are able to figure out the other two angles. [1] The sides we know are a which is a hardcoded distance from the camera subsystem to the BLE beacon, and b is the distance from the user to the BLE beacon. The γ is also 45 degrees because we are taking it as an arbitrary angle because it lets us simplify the equation since the angle because the user and BLE beacon is not relevant since the user still stays in frame.

$$c^2 = a^2 + b^2 - 2ab * cos(\gamma) \quad (1.10)$$

$$b^2 = a^2 + c^2 - 2ab * cos(\beta) \quad (1.11)$$

$$\beta = arccos(\frac{a^2+c^2-b^2}{2ac}) \quad (1.12)$$

where β is how much the mount should turn in order to keep the user in frame.

2.1.1 Bluetooth Transceiver (HM-10) and Arduino Uno

The reason we are using a HM-10 breakout board and Arduino Uno to create our beacon is because we were able to use Arduino's serial monitor to send a series of AT commands. These AT commands are basically used to program the given HM-10 chip to act as a BLE beacon. To make it easier to program the HM-10 we used the Arduino IDE to use the serial monitor to send AT commands. [2] In order for HM-10 breadboard to be a BLE beacon it needs to transmit a packet of 31 bytes of a unique code sent. In order to use iBeacon protocol for the Player Tracker subsystem our BLE beacon needs to broadcast: iBeacon prefix (9 bytes), Proximity UUID (16 bytes), Major (2 bytes), Minor (2 bytes), TX Power (1 byte). [7] Later we had a discussion about removing the Arudino because it was not necessary; however, we kept it convenient incase we needed to program it again.

The most important part of the broadcasting data packet is the TX power byte which is also known as RSSI Value. The RSSI Value is the connection strength value. This is critical for our calculating user's distance from the beacon.

2.1.2 Power Supply

We powered the beacon using a 9V commercial battery connecting it to the arduino's vin and ground port, and connecting the HM-10 to the 3.3V and ground pin.

2.2 Player Tracker

The mobile app is a critical component of our project which serves to interface between the BLE beacons and the camera control subsystem. Moreover, it is required to communicate with the BLE beacon to receive distance data and send this information to the HM-10 chip on the breadboard. In practice, this allows our project to work as intended by helping make sure that the user is in frame of the camera as they play their sport.

2.2.1 iPhone App

When designing this mobile application we had a few key requirements in mind. First and foremost, the app had to effectively and efficiently transfer data. Due to the nature of athletics, players move at fast speeds and the app had to make sure that the camera could keep up. Furthermore, we wanted to make the data transfer as seamless as possible as we knew that when we inevitably had to debug our code, that it was as simple as possible. Another aspect that we considered when designing the app was ease of use. On the consumer end, it was important for us to consider how much effort the user was willing to put in to get the whole system working. We knew that it would take a decent amount of time to get the whole camera system setup so we wanted to make the app experience as seamless as possible.

Our first few forays into making the app were largely unsuccessful, although we learned a lot along the way. We maily tried to build off some open source projects we found online. This was a difficult process as we had to learn the structure of the original project before we built off on it. Along the way we learned many valuable lessons in how we needed our data transfer to work. One of these lessons was about how callbacks work. A callback is essentially a piece of code which sends a ready signal and then awaits a “callback” response. An example of this in our project was that we needed a way to indicate to the beacon that we were ready to receive updated location data. Another important aspect of the app was understanding peripheral connections on the iPhone. When the iPhone is configured as a peripheral it sends out location information, almost as if it were a BLE beacon. Conversely, when configured as a peripheral manager, the iPhone would be able to receive BLE data from external connections [8].

For our final design we built our app off of source code provided to us from Apple [8]. This code was built to allow the iPhone to act as either a peripheral or peripheral manager. We took this basic functionality and overhauled the app to allow for our desired functionality. There were two central challenges.

First, we had to allow for multiple connections. More specifically, we had to make sure that the iPhone would behave as a peripheral manager in reference to the BLE beacon and as a peripheral to the HM-10 on the PCB. This is because the app needs to read data from the beacon and write data to the PCB. We accomplished this by essentially restructuring the callback structure of the app. By considering our dataflow in Figure 2, we were able to make these necessary changes.

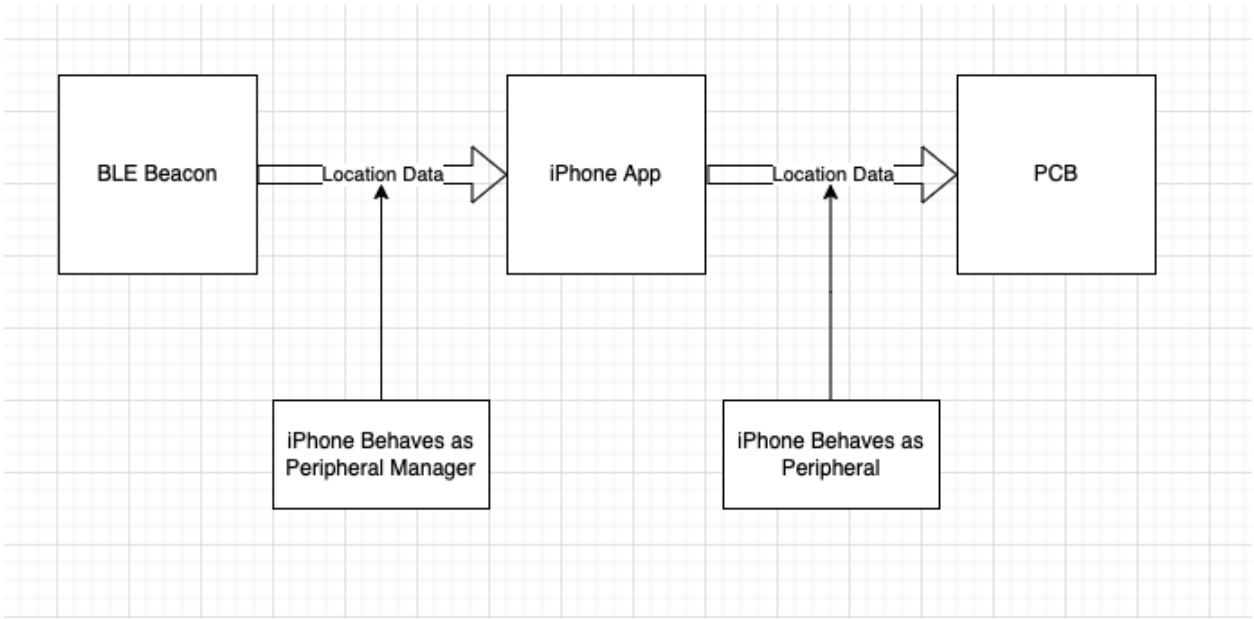


Figure 2

The second major change we had to make was to make sure that the same location data that was being sent from the BLE beacon and received by the iPhone app was the same location data being sent by the mobile app and received by the PCB. We were able to get this functioning by basically including lots of waiting intervals. What this means is that we give pre-defined buffer periods for data to transfer such that the data is sent in known intervals and we can transfer data in a known and methodological manner. Without these waiting signals we were running into errors regarding the camera constantly moving and the data transfer being inconsistent.

2.3 Camera Mount

The camera mount consists of a plastic protective casing that encapsulates the camera control unit, and its power system, a 9V battery. A phone holder is mounted on the servo motor which is installed at the top of the capsule which stands on an adjustable tripod.

2.3.1 Servo Motor

The servo motor will be rotating the phone mounted on the holder to capture the player during operation. We decided to go with the standard parallax servo as the motor for our project. It has the ability to provide 38 oz-in torque during operation from any position between 0 and 180 degrees.

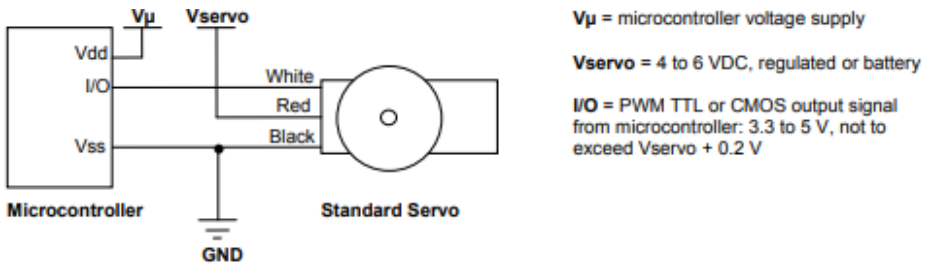


Figure 3: Servo motor and sample microcontroller input signals

The motor has three input pins (signal, power supply, and ground) which are connected to the camera control unit. The servo motor is controlled through a pulse width modulation signal with a peak-to-peak voltage of 5V, which moves or holds the position of the servo shaft by receiving a pulse every 20ms as shown in Figure 4. We decided to power the servo with a 5VDC power supply signal which is within the range of 4VDC to 6VDC [6].

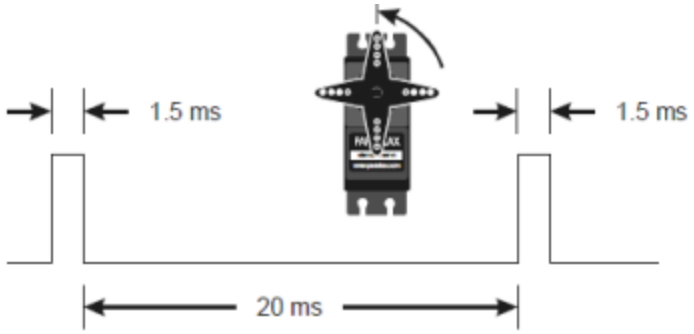


Figure 4: Sample pulse width modulation for center position of the motor

2.3.2 Protective Casing

The protective casing is a 4.53 x 2.56 x 2.17 in outdoor enclosure with a clear cover. I worked together with Glen Hedin from the ECE machine shop to modify the casing by removing a section at the top highlighted in red in Figure 5c, where the servo is fastened into the case by 2.25mm screws, and its bottom where the locking mechanism that connects the casing to the tripod highlighted in yellow in Figure 5c. We also attached a 9V battery holder to the back of the casing, shown in Figure 5b, so the user can easily replace the battery without much difficulty.

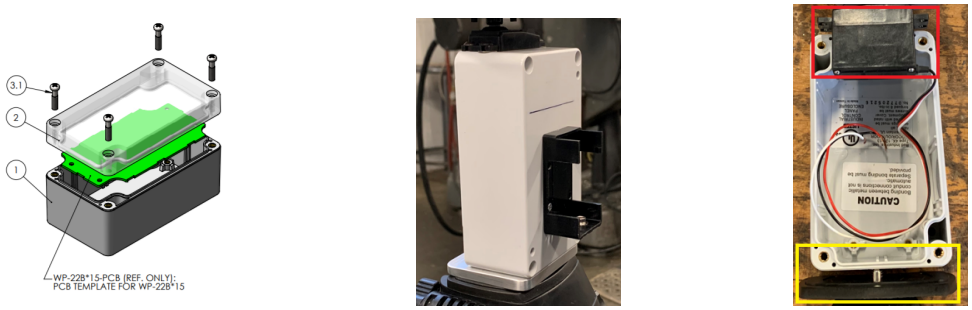


Figure 5: (a) 3D Casing Modelling; (b) Back of the casing; (c) Front of the casing

2.3.3 Adjustable Tripod

A major requirement for our final product was to be modulor and give the user the ability to set up the system at any height to give them their desired video results. That is why we put a tripod attachment underneath the Camera Mount Subsystem so it can attach to any tripod.

2.3.4 Phone Holder

Not everyone has the same size of phone and in order to prevent the user from buying extra accessories to use our product we have an adjustable phone mount. The phone has easy adjustable support beams that let users move them to keep the phone secure. However, the bottom of the phone holder made it quite difficult to have the phone kept in a horizontal position because it would make one side more heavy than the other so when the motor was spinning the phone would sometimes shift have the video would be angled. In the future we would like a phone holder that is able to keep the phone secured hortonizally and vertically.

2.4 Camera Control Unit

This subsystem is responsible for receiving the data that contains the location of the user from the IPhone through bluetooth using the HM-10 breakout board which it sends to the microcontroller through UART communication protocol. From the microcontroller, it sends the control signal to the servo motor. The whole subsystem is contained in a 1.75 x 1.42 in PCB.

2.4.1 Microcontroller

We used an ATtiny84-20 SSU as our microcontroller as it met all the requirements we had for the control unit. It is a 8-bit high performance microcontroller with an in-system programmable flash with 12 programmable pins,7 of which are used and the unused pins routed to female connectors for future use. The microcontroller is connected to a 6-pin in-circuit serial programming header where we could program and flash our code using the arduino integrated development environment. It is also connected to the HM-10 breakout board female connector and the servo motor.

2.4.2 Bluetooth Receiver

We first tried to acquire a RN48731 for our PCB because we thought we should use a regular beacon instead of BLE. The reason we settled on a regular beacon in the beginning was that we were not completely set on what size of data would be transferred from the Player Tacker Subsystem to the Camera Control Subsystem and did not want to limit ourselves. However, as the project progressed we first were not able to find RN48731 anywhere because it was out of stock. We found a RN4870U-V/RM118; however, this specific chip was not the same layout as the original RN48731 so it would not fit our PCB.

The bluetooth chip on the PCB was causing too many problems related to redesigning our most types of bluetooth chip is out of stock. We instead decided to reduce the size of data being transferred from the Player Subsystem to the Camera Control Subsystem. After reducing the size we were able to use a HM-10 chip because it was accessible.

The bluetooth receiver is quite important because without the data being transferred to the microcontroller the standard parallax servo motor would not know when or where to turn.

2.4.3 Power Supply

We also used a 9V battery to power the camera control unit which we connected to a 5V linear regulator that powers the parallax standard servo motor, HM-10 breakout board and the microcontroller.

3. Design Verification

3.1 Beacon

Overall our bluetooth low energy beacons were able to fulfill all of our high level requirements specifically for beacons. The important verification was that our beacon was able to have a unique 31 byte and where the last byte was the RSSI value. In order to

1. Bluetooth Low Energy Beacons should constantly broadcast a unique 31 bytes so that the player tracker can connect to it.
2. The Arduino and HM-10 should be able to properly communicate with AT commands in order to send the data packet.
3. The arduino UNO must have a steady power supply of 9V while the HM-10 bluetooth model will be tapping into the 5V output ports provided by the arduino

3.1.1 Bluetooth Transceiver (HM-10) and Arduino Uno

The standard HM-10 does not come enabled with HM-10 as a iBeacon. In order to program the HM-10 we use the Arduino IDE to send specific AT commands. To make sure that we are properly setting up the HM-10 we used the Arduino IDE because if the AT command was received the serial monitor would return an OK value as shown in Figure 3.

Figure 6: Serial Monitor output when programming HM-10

Next critical stage was to make sure that an external application is able to communicate with the HM-10 chip. We downloaded an external application from the Apple App store called BLE scanner version 3 from Bluepixel Technologies. This application gives us a visual representation about the internal

information being broadcasted from our beacon. Since we were able to connect to our beacon it proved two of three requirements because upon connecting to it shows the beacon is broadcasting a unique 31 byte packet and the AT commands were properly able to program it.

3.2 Player Tracker

3.2.1 iPhone App

The iPhone app's key deliverable was to transfer data correctly. More specifically, we wanted to send location information correctly to and from the app. Thus the tests for these were two fold.

To see if location data was being accurately received we had someone stand with two iPhones. With one of the iPhones the app connected to the beacon and on our laptop console we were able to view the location data coming in. With the other iPhone we were able to use an off the shelf BLE scanner app to verify our location data was being sent correctly.

In order to verify that the data was being sent correctly we repeated the same steps as outlined previously, except this time we connected the HM-10 on our PCB to a separate laptop. From there we were able to open the Arduino IDE and verify that the location data coming into the PCB was the same as the location data leaving the BLE beacon.

Additionally, because the app is so integral to the project and something we created in the latter half of the semester. we were able to determine that the app was indeed moving data correctly as the final outcome was as intended. The camera system was able to follow the player to a degree which satisfied our high level requirement (Player within the frame of the camera at least 90% of the time) so we implicitly know that all of our data was moving soundly.

3.3 Camera Mount

3.3.1 Servo Motor

In order to properly test if the servo motor is able to give our desired output we wrote a small test program for the microcontroller as displayed in Figure 7. The following code connects to the pin attached with the standard parallax motor and first spins it 90 degrees, to 180 degrees, and back to 0 degrees. After that it slowly rotates 1 degree every loop from the range 0 degree to 180 degrees and back 180 degree to 0 degrees. During the rotations, we also measured the waveforms that were generated and found it to be the pulse width modulation we expected shown in Figure 8. This gave us the opportunity to make sure that our standard parallax mtor was able to get the full range for motion that is critical to keep the user in view.

```
test_motor
#include <Servo.h>
#include <SoftwareSerial.h>

Servo myservo;

int angle = 0;
void setup() {
  // put your setup code here, to run once:
  myservo.attach(PIN_PA7); //PIN on Board
}

void loop() {
  myservo.write(90);
  delay(1000);
  myservo.write(180);
  delay(1000);
  myservo.write(0);
  delay(1000);
  // Sweep from 0 to 180 degrees:
  for (angle = 0; angle <= 180; angle += 1) {
    myservo.write(angle);
    delay(15);
  }
  // And back from 180 to 0 degrees:
  for (angle = 180; angle >= 0; angle -= 1) {
    myservo.write(angle);
    delay(30);
  }
  delay(1000);
}
```

Figure 7: ATtiny84-20 ssu code to test Standard Parallax motor

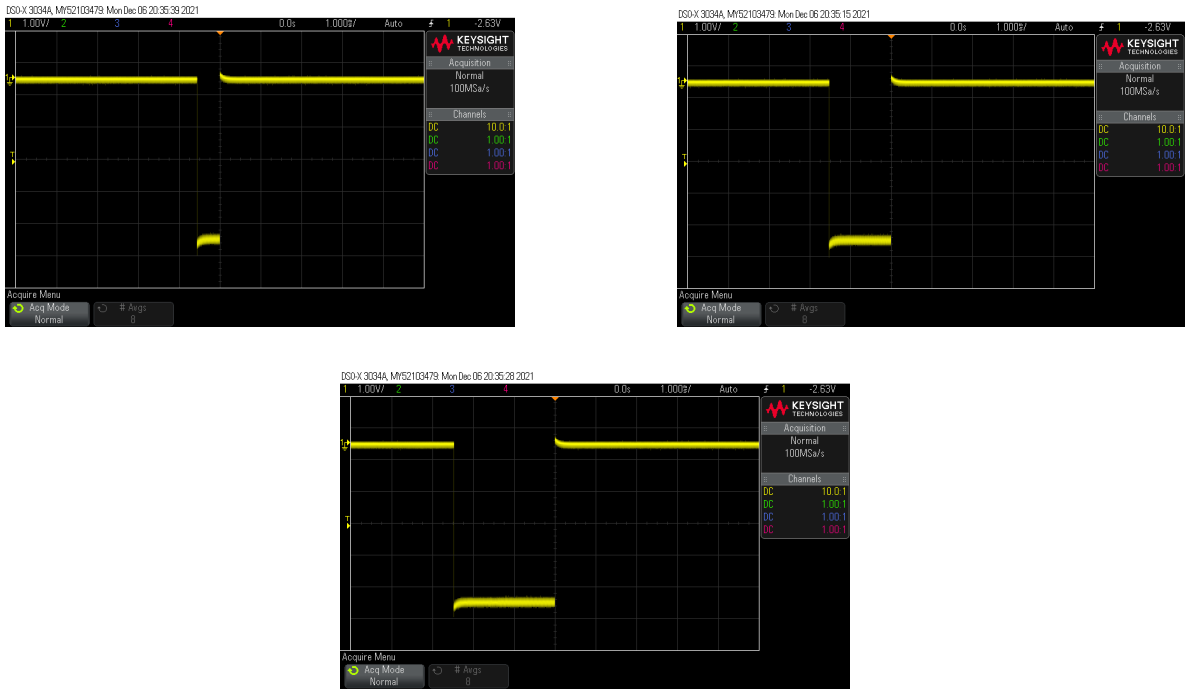


Figure 8: The waveforms generated during the motor test for 0, 90, and 180 degree rotation

3.4 Camera Control

3.4.1 Microcontroller

SInce we were able to successfully test that the servo motor was operating as it should be, it served as a confirmation that our microcontroller was operating properly. We also used the multimeter to perform a continuity test on every pin we soldered to ensure they were all routed correctly.

3.4.2 Bluetooth Receiver

In order to test if the HM-10 is collecting data, we wrote a program that would take input from an external application that we can write data from. In the code we had it set up so that the moment the HM-10 buffer had any data it would print out to the Arduino IDE's serial monitor. This way we were able to make sure whatever we were passing into the HM-10 through bluetooth was actually being stored in the buffer.

3.4.3 Power Supply

After we tested all the peripheral devices connected to the camera control unit, we tested the whole system and measured the voltage and the current drawn from the battery as it was operating. The values we measured are within the range of values we outlined on our verification table.

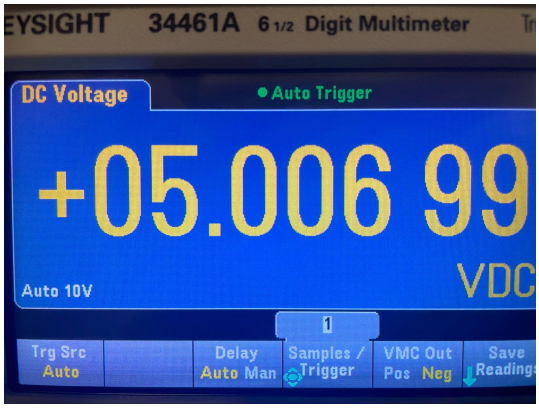


Figure 9: (a) Voltage drawn from the battery; (b) Current drawn from the battery

4. Costs and Schedule

4.1 Cost

4.1.1 Labour

Our team consists of two computer engineers and an electrical engineer. We will be using the average salary an ECE graduate from the University of Illinois makes to determine our hourly wages of $42.5/hour. We invested about 8 hours each week into our project, which brings the total cost of labor to:

$42.5/hour x 8 hours x 3 partners x 10 weeks = $10,200

4.1.2 Parts

The breakdown of the parts we used in our project, the vendor, and part number is shown in Table 1. The retail cost of the components is also included as well as the cost we actually incurred in the last columns of the table.

Table 1: Cost breakdown of project components

Part	Vendor	Part Number	Qty	Retail Cost ($)	Actual Cost ($)
Outdoor Enclosure	Polycase	WC-22*1508	1	13.54	13.54
Bluetooth Chip	Arrow	RN4871	1	13.31	Personal
Microcontroller	Digikey	ATTINY84-20 SSU-ND	6	4.94	4.94
ISP Header	Digikey	732-5394-ND	3	1.38	1.38
Connector Rcpt 3Pos	Digikey	A31081-ND	4	0.84	0.84
Connector Header 3Pos	Digikey	WM4201-ND	4	1.12	1.12
Connector Header 2Pos	Digikey	900-0022232021-ND	4	0.80	0.80
5V Linear Regulator	Digikey	488-LM1117MPX-50NO	4	2.40	2.40
3.3V Linear Regulator	Newark	12AJ7152	5	3.25	3.25
(0201) 4.7K Ohm Resistor	Digikey	P4.7AECT-ND	10	3.20	3.20
(0201) 330 Ohm Resistor	Digikey	P123304CT-ND	10	3.20	3.20
(0201) 10K Ohm Resistor	Digikey	P123222CT-ND	10	3.20	3.20
(0201) 1uF Capacitor	Digikey	1276-6441-2-ND	10	3.50	3.50
(0201) 4.7uF Capacitor	Digikey	490-13230-1-ND	5	2.20	2.20
Diode	Digikey	3757-SS0520_R1_00001	5	0.25	0.25
Blue LED	Digikey	754-202202-ND	2	0.38	0.38
Red LED	Digikey	754-2028-2-ND	2	0.26	0.26
PCB	JLCPCB	N/A	5	13.80	Personal
Arduino Uno	ECEB	N/A	2	43.80	43.80
Parallax Standard Servo	ECEB	N/A	1	17.95	Donated
HM-10 Breakout Board	Amazon	ML-HM-10	2	21.98	Personal
Tripod	Amazon	PLTRI72	1	29.99	Personal
Phone Holder	Amazon	MB000313_V2	1	9.95	Personal

Female Header 4Pos	Digikey	S7037-ND	2	0.94	Personal
Female Header 3Pos	Digikey	S7036-ND	2	0.74	Personal
Female Header 2Pos	Digikey	S7036-ND	2	0.66	Personal
(0805) 10K Ohm Resistor	Digikey	RMCF0805JT10 OCT-ND	10	0.20	Personal
Solderless Breadboard	Mouser	485-2463	1	0.95	Personal
Shipping and Fees	Misc	Digikey & Mouser	N/A	21.52	Personal
Total Cost				**202.30**	**88.26**

4.2 Schedule

Table 2: Schedule of work for our team

Week	Work
10/4	Shivang - Design review Aksh - Design review, Research BLE data transfer with arduino and HM-10 Ore - Design review, Order parts
10/11	Shivang - Finalize PCB Aksh - Finalize PCB Ore - Finalize PCB, Order PCB
10/18	Shivang - start writing code to program the BLE Beacons Aksh - start writing/researching AT commands to program the BLE Beacons Ore - Work on wiring the BLE Beacon
10/25	Shivang - Finish testing the output of the BLE Beacon Aksh - Setup the iBeacon protocol and test to see if the iPhone can pick up a constant signal Ore - If PCB arrives start working on soldering the PCB or finish building the second BLE Beacon
11/1	Shivang - Worked on building the stand and the PCB Aksh - Test to see if data gathered from the phone can be sent to the bluetooth chip on the microcontroller Ore - Worked on building the stand and the PCB
11/8	Shivang - Test the whole system and make changes to fine tune Aksh - Test the whole system and make changes to fine tune

	Ore - Test the whole system and make changes to fine tune
11/15	Shivang - Use feedback from mock demo to improve final presentation Aksh - Use feedback from mock demo to improve final presentation Ore - Use feedback from mock demo to improve final presentation
11/22	Shivang - Fine tune the stand for phone Aksh - Fine tune the stand for phone Ore - Fine tune the stand for phone
11/29	Shivang - Demo / Presentation Aksh - Demo /Presentation Ore - Demo / Presentation
12/6	Shivang - Write final paper Aksh - Write final paper Ore - Write final paper

5. Conclusion

The process of brainstorming, researching, and developing this project was a truly unique experience. Particularly as seniors, we were required to leverage many topics we had learned over the past few years in the ECE department to create our project - The Player Tracking Camera. This made our experience in the ECE department come full circle and we are happy with the final outcome of all our time and effort this semester.

5.1 Accomplishments

At the end of the semester we are proud to have a fully functioning project that met all of our high level objectives. The first objective was to have functional data transfer across all of our subsystems. This was achieved as the BLE beacons are able to send distance data to the mobile app and this data is subsequently sent to the PCB on the camera control subsystem. The next objective was to calculate the user's position within at least a 20 meter radius. We know that this objective is met as we were able to run functional tests across a basketball court with a length of 28 meters. The last objective we had set out to accomplish was to keep the user within the camera's view for 90 percent of the recording duration. This objective was achieved as during our tests we found that we stayed in frame about 93 percent of the time. We were able to accomplish a successful project by meeting all of our initial objective criteria.

5.2 Uncertainties

Fortunately for us, we did not have elements of our projects that did not meet our objectives. With that being said, there are certainly some things that did not work out as planned. One idea that we had that didn't pan out was our phone orientation. When we were testing out our motor we found that when we placed our phone in the holder in a horizontal orientation there tended to be some shakiness to the camera and, occasionally, were instances where the phone was on the brink of falling out.

Another uncertainty that we had was with the speed of the camera motor. As shown in our final demonstration, the camera serves its purpose of capturing the user at a high rate. Unfortunately, the camera moved very rapidly so the final viewing experience ended up a bit choppy - particularly when the user moves very fast the final video can be a little jarring to watch.

5.3 Ethical considerations

When reviewing ethical considerations we decided to consult the ACM code of ethics and the IEEE code of ethics. These two codes are most relevant to our subject matter and provide the most relevant ethical considerations. After reviewing both documents closely there were two specific sections that were applicable to our project.

The most relevant ethical concern is related to ACM code of ethics standard 1.6 - about respecting data privacy [5]. This issue is pertinent to people's personal data being secure and being obtained and used responsibly. In our project we do collect tracking data but we avoid any issues related to data privacy. We do this in two ways. First, we do not store the data for long periods of time. The data input is used instantly and cleared as new data comes in. There is not any sort of memory of location that can be

accessed by those with bad intentions. Additionally, if the instantaneous data were to be captured, the data being collected does not disclose any new information as the person turns on the camera and is standing right in front of it so there is no issue of location data being revealed.

As a result of physical devices being used, there is a possibility of them interfering with human movement. According to the IEEE Code of Ethics section I subsection 1. we must "hold paramount the safety, health, and welfare of the public" [4]. We can adapt the usage of our project to not interfere with the game or the players in a way that causes harm. We can suggest to the user to place the BLE beacons behind the hoop/net/playing surface so that they don't come in contact with the players. We also have the camera on a mount so that it can be placed a safe and reasonable distance away from the players such that it does not interfere with the game and cause anyone any bodily harm. Given the extensive range of BLE and Bluetooth users are able to operate the technology without risk of interference and potential injury through collision.

5.4 Future work

In terms of future work, the things we would like to work on largely stem from our current uncertainties. These things being the orientation of the phone on the camera mount and the speed of the motor on the camera mount.

Being able to orient the phone horizontally would serve as a major upgrade to the camera system. For one, this adds modularity for the user to set up the phone however they want. In addition, it helps when the player tracking camera is used on large playing surfaces, such as a soccer field. This is because a horizontal orientation allows for wide frame video capture - reducing the burden on the beacon and phone's distance accuracy. Additionally, when it comes to practical applications, popular video sharing platforms such as Instagram and YouTube generally have their videos filmed horizontally so this would help the user have a more integrated experience. In terms of brainstorming ideas to fix this, one simple fix would be to have a top notch on the phone holder. This would provide a four-way hold on the phone camera such that it won't get loose and fall out.

Another area of improvement could be the motor speed. As we found out when testing our project, the motor moves pretty quickly. This results in a video that can be a little hard to watch at times. Upon further investigation, it became apparent that his motor did not have adjustable speed controls. Therefore, going forwards we would like to spend time finding a new motor. This motor would have to fit our size and power constraints but, if it exists, it would allow us to have a smoother viewing experience for the final video and overall a more refined project.

References

[1] The SAS Theorem. CUNY New York City College of Technology, 5 Sept. 2021, https://math.libretexts.org/@go/page/34125.

[2] “How to USE HM-10 Ble Module with Arduino to Control an LED Using Android App.” Circuit Digest, 4 July 2019, circuitdigest.com/microcontroller-projects/how-to-use-arduino-and-hm-10-ble-module-to-control-led-with-android-app.

[3] “Bluetooth Low Energy (BLE) Beacon Technology Made Simple: A Complete Guide to Bluetooth Beacons.” Beaconstac RSS, blog.beaconstac.com/2018/08/ble-made-simple-a-complete-guide-to-ble-bluetooth-beacons/.

[4] “IEEE Code of Ethics.” IEEE, www.ieee.org/about/corporate/governance/p7-8.html.

[5] “The Code Affirms an Obligation of Computing Professionals to Use Their Skills for the Benefit of Society.” Code of Ethics, www.acm.org/code-of-ethics.

[6] “Parallax Standard Servo, datasheet.” Parallax Inc., October 2011. Available at: https://cdn.sparkfun.com/assets/9/7/6/3/1/DS-16049.pdf

[7] “What Is Ibeacon? All You Need to Know.” Bleesk, bleesk.com/ibeacon.html.

[8] “Turning an IOS Device into an IBeacon Device.” *Apple Developer Documentation*, https://developer.apple.com/documentation/corelocation/turning_an_ios_device_into_an_ibeacon_device.

Appendix A Requirement and Verification Table

Table 3: Camera Control Subsystem Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Provide 5V +/- 10% from a 9V power	1. Measure the output voltage using an oscilloscope and ensure it maintains a steady voltage of 9V allowing a 10% deviation while it is connected to the control subsystem. Have the probes across the battery while connected to the control system.	Yes
2. The power source can supply output current of 250mA-400mA for at least an hour of continuous use	2a. Measure the output current using an oscilloscope and ensure it Is within the range mentioned on the table. Connect the probes in series with the battery. 2b. Record the time to ensure the test satisfies our conditions for at least an hour.	Yes
3. Maintain the servo motor temperature below 50° C and PCB temperature below 75° C	3. During verification of all the tests, use an IR thermometer to measure the temperature of the PCB and the servo motor over 5-minute intervals	Yes
4. Servo motor should be able to rotate a minimum of 17 oz load in a clockwise and anticlockwise motion between 0°- 180° within a second	4a. We will be using a function generator to send PWM waves to the signal input of the servo and using a power generator to power the servo. 4b. Every 20ms a high pulse must be sent to the signal input (0.75ms to keep at 0° and 2.25ms to rotate it 180°). 4c. We will be alternating sending the signal pulses and measuring the time the servo takes to make the rotation.	Yes
5. Power supply must provide a stable voltage of 5V +/- 10% (power coming from camera control subsystem PCB	5a. Mount the camera phone to the servo mount 5b. The motor will be receiving its three signals from the servo inputs connector on the PCB 5c. Measure the output voltage using an oscilloscope and ensure it maintains a steady voltage of 5V allowing a 10% deviation while it is connected to the control subsystem.	Yes

	Have the probes across the power supply connectors while the servo is in operation	

Table 2: Beacon Requirements and Verifications

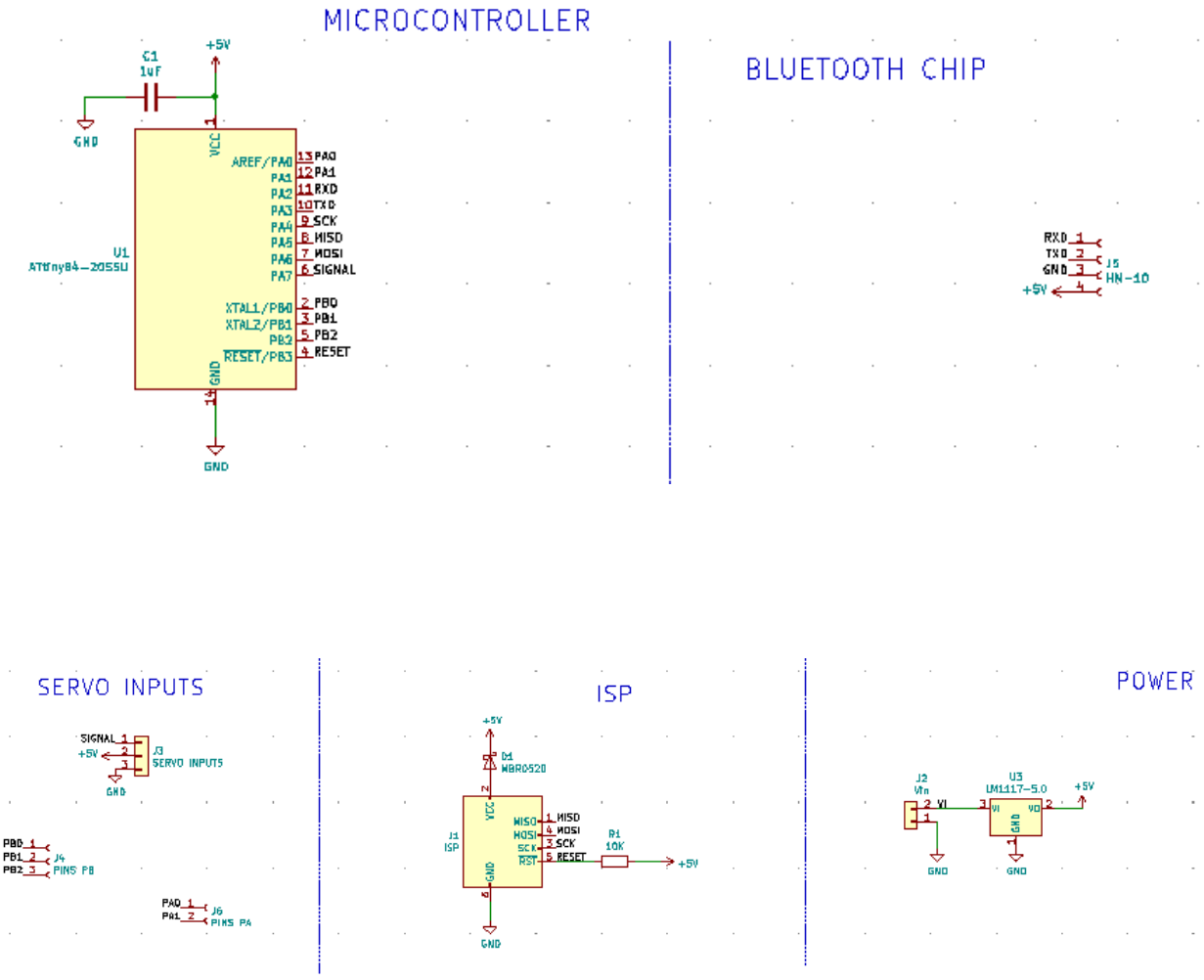
Requirement	Verification	Verification status (Y or N)
1. BLE Beacon should be able to create a 31 byte unique code and have to constantly send so that the player tracker can pick it up.	1. Will set a test program to listen for what the BLE Beacon is sending and confirm that the correct code is being sent. Also, to calculate the position from each beacon, we can use a measuring tape to make sure the distance is approximately the same.	Yes
2. The Arduino and HM-10 should be able to properly communicate with AT commands in order to send the data packet.	2. Will write output statements to the console while coding in order to see if the correct AT command is being sent from the HM-10 chip to Arduino.	Yes
3. The arduino UNO must have a steady power supply of 9V while the HM-10 bluetooth model will be tapping into the 3.3V output ports provided by the arduino	3a. Measure the output voltage using an oscilloscope at the power input of the Arduino UNO and ensure it maintains a steady voltage of 9V. 3b. Measure the output of the 3.3V output port to ensure the HM-10 can properly run off the supply	Yes

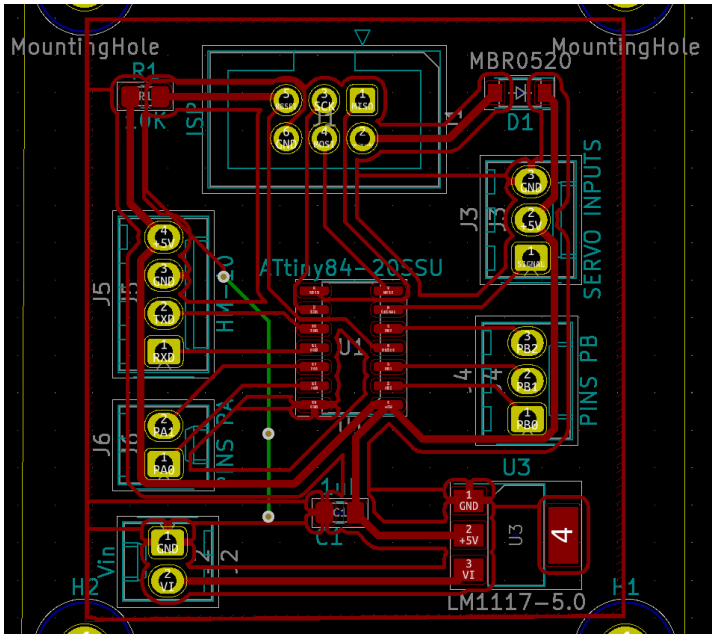
Table 3: Player Tracker Requirements and Verifications

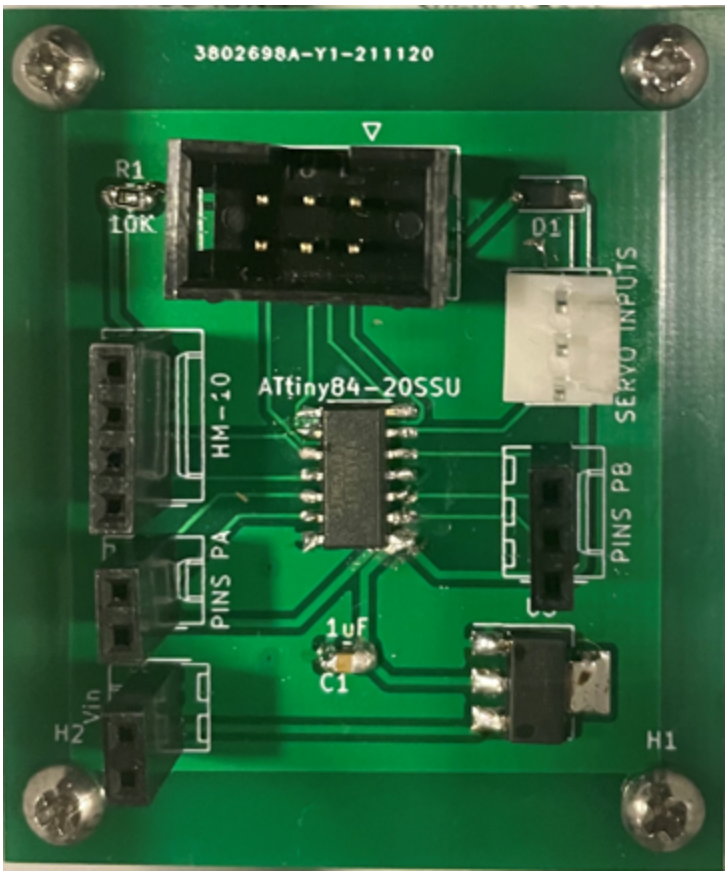
Requirement	Verification	Verification status (Y or N)
1. The app on the phone should be able to gather data from at least one BLE Beacon and calculate the cartesian coordinates and transmit them to the Camera Control Subsystem.	1. Set the beacons to a specific constant 31 byte and distance so that we can test if the code is correctly calculating the distance. Then change the beacon to give different coordinates and make sure that output is correct based on the formula we have for calculating distance.	Yes

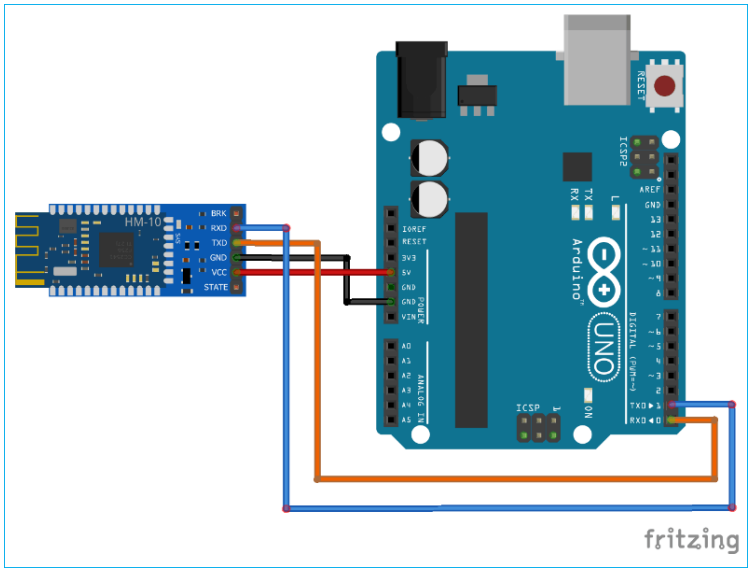
2. Phone must be constantly listening to the BLE Beacons and constantly sending the changing coordinates to the Camera Control Subsystem.	2. We will walk around everywhere on the court to make sure that it is able to pick up the BLE Beacon constant code from within the BLE Beacon range. We will walk out of range of the BLE Beacon range to make sure that the phone is still searching for the 31-byte data packet.	Yes

Appendix B Circuit Schematics




MountingHole
MountingHole
R1
10K
MBR0520
D1
SERVO INPUTS
ATtiny84-20SSU
U1
PINS PB
U3
LM1117-5.0
H2
H1


3802698A-Y1-211120
R1
10K
D1
SERVO INPUTS
ATtiny84-20SSU
HM-10
PINS PB
PINS PA
1uF
C1
H2
H1


HM-10
BRK
RXD
TXD
GND
VCC
STATE
Arduino
UNO
RESET
ICSP2
ICSP
POWER
ANALOG IN
DIGITAL (PWM~)
fritzing