Creation of a Sport Backstop Using Axiomatic Design

A Major Qualifying Project Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirement for the
Degree of Bachelor of Science by:

Bailey Berg
Jordan DeDonato
Michael Keable
Thien Q. Nguyen
Griffin St. Onge

Report Submitted to:

Professor Christopher A. Brown
Mechanical Engineering
Worcester Polytechnic Institute

Professor Walter T. Towner
Industrial Engineering
Worcester Polytechnic Institute

A Major Qualifying Project submitted to the faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science. Submitted May 5, 2021.

Table of Contents

Abstract 6
Acknowledgments 7
Key Definitions 8
Key Acronyms 9
1. Introduction 10
1.1. Objective 10
1.2. Rationale 10
1.3. State of the Art 10
1.3.1 Similar Products 11
1.3.2 Design Methods 12
1.3.3 Speed Measurement 12
1.3.4 Selecting Hardware 12
1.4. Approach 12
1.5 Methods 13
2. Design Decomposition and Constraints 14
2.1. Objective Functional Requirement and Design Parameter 14
2.2. Project Constraints 15
2.3. Functional Requirements 15
2.4. Design Parameters 16
2.5 Conditional Functional Requirements and Design Parameters 16
2.6 Design and Decomposition Equations 16
2.6.1 CEME Min Analysis 17
2.7 Design Matrix 17
3. Physical Integration and Prototype 18
3.1 Physical Integration of the Design Parameters 18
3.1.2 Housing Unit Materials 19
3.2 Iterations of the Housing Unit 19
3.2.1 Iteration 1 19
3.2.2 Iteration 2 20
3.3 The Computer Program 21
3.3.1 Creating the Environment 21
3.3.1.1 Preparations for the Environment 21
3.3.1.2 Dependencies 23
3.3.1.3 Python Virtual Environment 23
3.3.1.4 OpenCV 4 25
3.3.1.5 Pre-made Utilities for OpenCV 25
3.3.2 Radar Code 25
3.3.3 Camera Code 26
3.3.4 Feedback Via a Graphic User Interface (GUI) 26
3.4 Selection of DPs 27
3.4.1 Measurement Hardware 27
3.4.2 Feedback Hardware 28

3.5 Cost 28
4. Testing of Design 29
4.1 Running the Software 29
4.2 Initial Testing 29
4.3 Testing Results 30
5. Quality 32
5.1 Quality 4.0 32
5.2 Applying Quality Practices 33
5.3 This Product's Quality 33
6. Broader Impacts 37
6.1 Engineering Ethics 37
6.2 Societal and Global Impact 37
6.3 Environmental Impact 37
6.4 Codes and Standards 37
6.5 Economic Factors 38
7. Discussion 39
7.1 Axiomatic Design Analysis 39
7.2 Analysis of Design Objectives 40
7.3 Future Work 40
8. Conclusions 42
9. References 43
10. Appendices 47
Appendix A. Research on Sensors 47
Radars 47
Lasers 48
Pressure sensors 49
Ultrasonic Sensors 50
Photogates 51
Appendix B. Candidate DPs Selection Criteria Matrix 52
Appendix C. Additional Information on Axiomatic Design and C-K Theory 55
Appendix D. Full Axiomatic Design and Design Matrix for The Product 56
Appendix E. Full Code 61
Open-Source Code Used as Reference 61
Radar 63
Camera Vision 64
Gui Code 65
Appendix F. Camera Vision Failures 66
Appendix G. Full Test Results 67
Appendix H. Reflections 69
Bailey Berg 69

List of Tables

Table 1: Summary of Similar Products 11
Table 2: Prototype Cost Analysis 28
Table 3: Equations Used to Develop Quality Control Charts 34
Table 4: Constants Used to Develop Quality Control Charts 34
Table 5: Results of Control Chart Equations 35
Table 6: Speed Hardware Selection Criteria Matrix 52
Table 7: Position Hardware Selection Criteria Matrix 53
Table 8: Visual Feedback Hardware Selection Criteria Matrix 53
Table 9: Audio Feedback Hardware Selection Criteria Matrix 54
Table 10: Functional Requirements 56
Table 11: Design Parameters 57
Table 12: Metrics and Equations on Axiomatic Design 58

List of Figures

Figure 1: Chart on Accuracy vs Expense in Existing Products 10
Figure 2: Summary of Project Timeline 13
Figure 3: Coordinate System 15
Figure 4: Electrical Block Diagram of Physical Integration 18
Figure 5: Final Radar Housing Unit 18
Figure 6: Final Camera and Raspberry Pi Housing Unit 18
Figure 7: CAD model of Iteration 1 of Housing Unit 19
Figure 8: CAD model of Iteration 2 of Housing Unit 20
Figure 9: Image SD Card with Raspbian 22
Figure 10: Raspbian Buster Flashed with Etcher 22
Figure 11: Expanded Disc Space Verification 23
Figure 12: Dependencies for the Environment 23
Figure 13: Installing "pip" 24
Figure 14: Updated /.bashrc with Virtualenvwrapper Settings 24
Figure 15: Feedback GUI 27
Figure 16: Terminal Running the Project 29
Figure 17: Camera Vision Test GUI 1 31
Figure 18: Camera Vision Test GUI 2 31
Figure 19: X Bar Control Chart for Speed 35
Figure 20: R Control Chart for Speed 36
Figure 21: Lidar Data Set (Sawicki, 2017) 49
Figure 22: Lidar Block Diagram (Sawicki, 2017) 49
Figure 23: Photogates (Nelson) 51
Figure 24: Design Matrix 60
Figure 25: Reference Code for Environment (Cuni & Duran, 2016) 63
Figure 26: Dependencies Code 63
Figure 27: Radar Code 63
Figure 28: Camera Vision Code for Recording the Video 64
Figure 29: Camera Vision Code for Trimming Video 64
Figure 30: Camera Vision Code for Finding the Ball 64
Figure 31: Camera Vision Code to Determine if the Strike Status 64
Figure 32: Code for GUI 65

Abstract

The objective of this project was to design and physically develop an affordable, meaning under $500 retail price, device that can accurately provide feedback on speed and position of a pitch. Current products like this are usually either expensive, inaccurate or both. Additionally, they often lack feedback on position of the pitch over homeplate. Due to these limitations, customers are often dissatisfied with the smart backstop product they have purchased. Axiomatic Design theory and concept-knowledge theory were used in the design of a new smart backstop product. Our team has come to a solution that can provide more accurate results for speed and position of a pitch and provide feedback in a clear and user-friendly manner.

Acknowledgments

We would like to extend thanks to the following people, as they were essential to the completion of this project. We would like to thank our advisers, Professor Christopher A. Brown and Professor Walter T. Towner for their advice and guidance throughout the yearlong project. We would also like to thank Gary Sowyrda for sponsoring this project and David Leandres and Cole Noreika for writing large parts of the computer program.

Key Definitions

Word or Phrase	Definition
Accurate	The term accurate refers to the closeness of a measured value to a standard or known value (NCSU, 2020). In the scope of this project, this word is used when referring to a range of accuracy. For this specific project, the team compared the speed given by this device to a OPS241-A Doppler Short Range Radar Sensor from OmniPreSense. To maintain similar standards to market competitors, this device should have an overall accuracy within +/- 1 miles per hour for speed and +/- 4 inches in the X or Z direction for position.
Affordable	Affordable means that the product has a cost that is not too high (Merriam-Webster, 2020). In reference to this project, being affordable means being less expensive than competitors in the market while maintaining the same abilities as the competition. For this project specifically, the limit for what is considered affordable is $500 market price. Anything at or below $500 is affordable.
Precise	The term precise refers to the closeness of two or more measurements to each other (NCSU, 2020). In terms of this project, precise means there are multiple readings within the acceptable range, the system is referred to as precise.

Key Acronyms

Acronym	Word or Phrase
AD	Axiomatic Design
ASQ	American Society for Quality
CAD	Computer-Aided Design
CEME min	Minimum number (of functional requirements) that are Collectively Exhaustive and Mutually Exclusive
C-K	In reference to C-K Theory, or Concept-Knowledge Theory
CN	Customer Need
COVID	Corona Virus Disease 2019 (prevented interpersonal work in 2020 and 2021 due to its transmission and infection rate)
DP	Design Parameter
FR	Functional Requirement
GUI	Graphical User Interface
ISO	International Organization for Standards
OC	Optimization Criteria
SC	Selection Criteria

1. Introduction

1.1. Objective

The objective of this project was to design and physically develop a smart backstop that can measure speed and position on a vertical 2D plane of a baseball or softball passing over a home plate. The backstop will provide feedback on the speed and position to the user visually and audibly. The backstop will be affordable, which is defined as under $500.

1.2. Rationale

Creating an affordable sports backstop that measures speed and position is important in many ways. Technically, this project can output accurate (+/- 1 mph) speed and position readings for future products. Socially, this project can provide entertainment for people or be used as a training equipment for people who want to improve their throwing techniques. Commercially, this product can be marketed as a device used to challenge friends, get better at sports, and encourage physical activity. It can also create an affordable version of the existing smart backstops. Current models for a digital smart backstop are either inaccurate (+/- >1 mph) in their measurements or expensive or both (Sports Sensors, Inc) (Pocket Radar). The graph below shows the accuracy and cost of current models compared to our objective accuracy and cost.

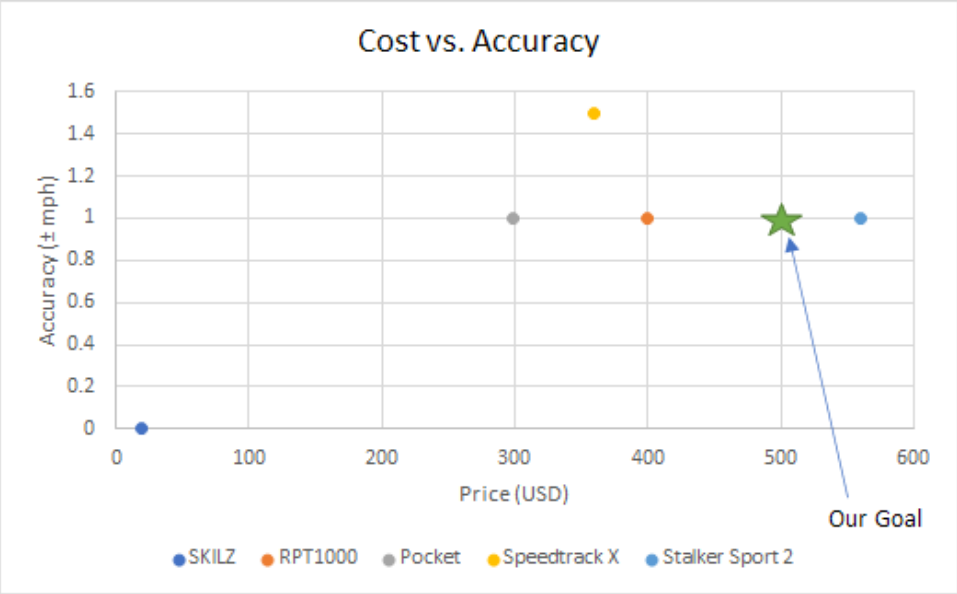

Figure 1: Chart on Accuracy vs Expense in Existing Products

1.3. State of the Art

There are multiple existing products that relay pitch speed to the user. Few simultaneously provide feedback on pitch location. With most products using the same type of sensor (radar) the cost and performance vary greatly.

1.3.1 Similar Products

Table 1: Summary of Similar Products

Product	Performance	Measurement Method	Accuracy	Reading Range	Power	Price
SKLZ Bullet Ball	Inconsistent readings Weight was not the same as regular baseball Never shuts off	Speed sensor	Not accurate ± 50 MPH	Up to 120 MPH	1 LR-41 Battery	$18.49
Radar Pitching Trainer 1000	Good accuracy and display Moves when speed is over 50 mph Materials ware out	Radar	± 1 MPH	20-99 MPH	4 D Batteries	$399.00
Pocket Rader	Consistent readings Immediate feedback Easy to operate and transport	Radar	± 1 MPH	25-130 MPH	2 AAA Batteries	$300-$400
Speedtrack x	Poor quality and accuracy Does not read high speeds	Radar	± 2 MPH	9-150 MPH	4 C Batteries	$360.00
Sports Sensor Swing Speed	Moderate accuracy Easily portable	Doppler Radar	Within 1%	20-200 MPH	3 AA Batteries	$119.95
Stalker Sports 2 Radar Gun	High level accuracy Able to read at multiple angles	Doppler Radar	± 1 MPH	5-150 MPH	6 AA Batteries	$560
Statcast (MLB)	High level accuracy Reads all speeds and spin rates	Radar and camera	Extremely minimal ± 0.5 MPH	All speeds if ball is picked up	Hardwired	$50,000 +

1.3.2 Design Methods

This project uses a design method called Axiomatic Design along with some aspects of C-K (Concepts and Knowledge) theory. To see more about Axiomatic Design and the C-K theory, refer to Appendix C.

1.3.3 Speed Measurement

This project uses speed measurement devices that can calculate the speed of a baseball. The pre-existing devices that record speed use different variations of radars and sensors. For example, the Sklz ball uses a speed sensor and picks up the speed as soon as it leaves the pitchers' hand. The Pitching Trainer, Pocket Radar, Speedtrac, Sports Speed Sensor all use stational radars using the doppler effect. The MLB Statcast system is more complicated by using optical tracking sensors and radar sensors. Refer to Appendix A for information on how doppler effect and radar works using position and time.

1.3.4 Selecting Hardware

To choose the best hardware for the design, a matrix was created for each measurement and feedback type the design warranted. Speed measurement had one matrix, position measurement had another and there were also matrices for audio and visual feedback. To see the chosen hardware for this project, refer to Section 3.4. To see the matrices that compare different hardware options, refer to Appendix B.

1.4. Approach

This project used Axiomatic Design (AD) to develop a smart backstop that measures speed and position and provides feedback on those measurements to the user. The theory of AD states that all good design solutions must comply with two axioms: maintaining the independence of the functional elements and minimizing the information content (Suh, 1990). Functional requirements (FRs) are composed to satisfy customer needs (CNs) while design parameters (DPs), or physical solutions, are selected to fulfill FRs (Suh, 1990). FRs and DPs are mapped in hierarchical decompositions, from abstract to detailed. DPs are then checked against FRs for independence using an independence matrix. CNs for this project were developed using the sponsor's description of the product as well as research on what similar products fulfill. More information on AD can be found in Appendix C.

This project's AD contained multiple candidate DPs for various FRs. Using selection criteria, the candidates were compared, and the best option was chosen. The DPs were either purchased and adapted to fulfill the FRs, or computer-aided designs (CAD) were used to develop physical integrations of the DPs. The physical prototype is the combination of the backstop (purchased), the sensors used for measurement (purchased and adapted through coding), and the housing that contains and protects the sensors (developed by team).

1.5 Methods

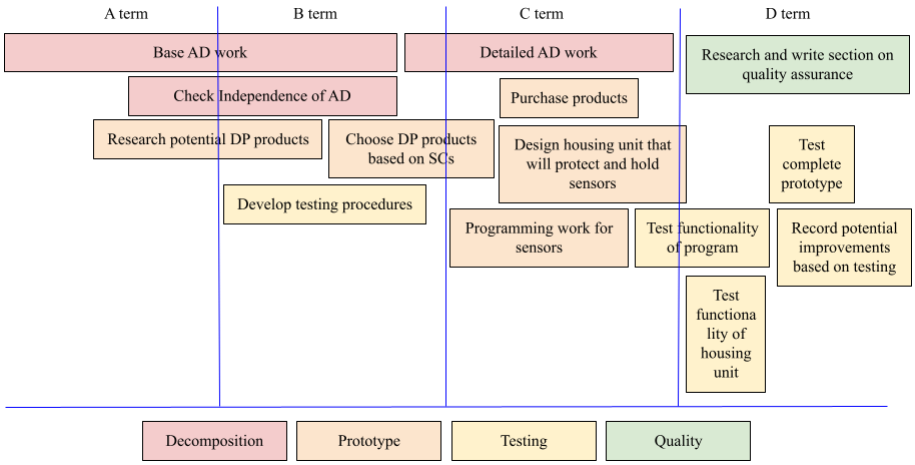

Figure 2: Summary of Project Timeline

Axiomatic Design was used to design a smart backstop. The Axiomatic Design began by defining our CNs through a sponsor-provided description and research on similar products. There were multiple options for certain DPs that could be used to fulfill each FR. Each candidate DP was compared using a selection criteria (SCs) matrix (see Appendix B). A design matrix was used to determine independence of the design.

Once the Axiomatic Design decomposition was complete, the team bought hardware that could be used for measurement needs and began designing a housing unit for the sensors. The physical backstop was purchased. While the team, and an outsourced computer programmer, worked on programming the sensors, multiple iterations of the housing unit were produced. The housing unit iterations were designed on CAD. When the team and advisers finalized the design and CAD model of the housing unit, it was 3D printed for testing. The whole prototype of the sensors and the housing unit were tested against the FRs they needed to fulfill. The tests were done by the team and not by an outside group due to COVID prevention guidelines. After testing, potential improvements for the design and prototype were noted and the results were written out. Additionally, how to apply quality assurance to this product was researched and discussed.

2. Design Decomposition and Constraints

As mentioned above, Axiomatic Design involves a process of defining FRs and DPs and decomposing these items in order to design a product. After decomposing is complete, metrics are added for how an FR can be measured. Relations between an FR and its respective DP are added to confirm that the sum of the children is equal to the parent. This section will go over why FRs, DPs, and metrics were chosen and discuss the relation between them. To see the full Axiomatic Design, refer to Appendix D.

2.1. Objective Functional Requirement and Design Parameter

The CNs are defined as:

1. Measure speed of ball
2. Measure position of ball on X-Z plane
3. Easy to store
4. Easy to transport
5. Provide feedback on speed
6. Provide feedback on position
7. Affordable
8. Weather resistant
9. Stop balls
10. Minimize potential injuries
11. Can be used by anyone: children or adults

Additionally, the sponsor had ideas for the backstop that were not needs, but rather ‘wants’ if feasible. These included:

1. Measure spin rate of ball
2. Measure position of ball in 3D space over Homeplate
3. Measure velocity of ball in 3D space

Many of these were translated to selection criteria (SCs) and optimization criteria (OCs) as they did not require a specific DP but rather would be considered when choosing a physical integration of the DPs. The OC was to minimize cost. The project sponsor specified that he wanted the product to be affordable. From this information, the team deducted that the cost should be minimized, while maintaining the integrity of the product and fulfilling all the desired CNs. The general SCs were that the products used to protect the hardware would be weather and sunlight resistant and that any sensors used could measure within the range needed accurately and precisely. When choosing products that would act as the DPs, all the criteria mentioned was considered.

From the CNs, an objective FR was developed: to develop product that determines speed and position and stops balls. From the objective FR, an objective DP was developed: a product that measures speed and position of a ball and gives feedback to user. These objective statements give a broad overview of what the sponsor wanted and what the product is. When looking at the additional wants of the customer, the team developed conditional FRs that could be added if desired but would not be detrimental to the product if not added.

2.2. Project Constraints

The following constraining factors were taken into consideration while completing the design for this project:

1. Budget: The team had to stay within the budget provided by the sponsor ($15,000), as well as the money allocated to each of the students by the ME department at WPI ($1250). The total money for the project totaled $16,250. The team did not intend to spend all this money. Rather the large budget acted as a barrier to avoid feeling constrained by money when producing a prototype.
2. Construction: When completing the prototype production, the team had limited access to machines and each other due to COVID.

2.3. Functional Requirements

Before going into the FRs, it is important to define the coordinate system as it is mentioned frequently in the decomposition. In the figure below is the defined coordinate system for this product.

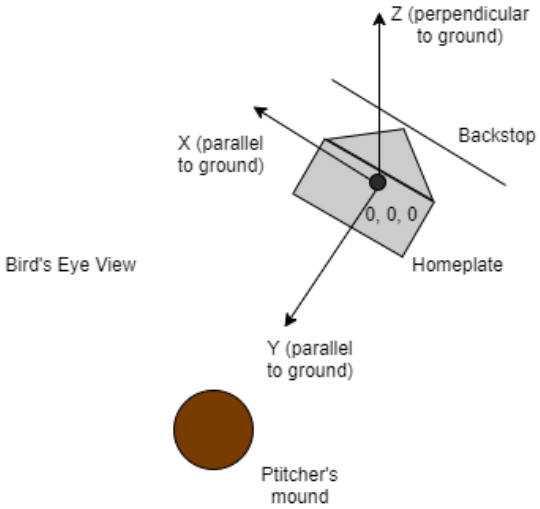

Figure 3: Coordinate System

In Axiomatic Design theory, FRs set the tone for the whole design. No design can be better than its FRs. FRs determine what the design needs to do based on the CNs. The top-level FR for the decomposition is to develop product that determines speed and position and stops balls. Under this, there are six sub FRs. These can be seen in Appendix D.

FR 5 is conditional on if the team has the time and resources to fulfill this FR. Additionally, when considering the prevention of injuries (FR 6.3), it should be noted that this FR is to minimize and not to prevent because it is impossible to prevent all potential injuries while still maintaining the functional properties needed for the housing unit. For example, the housing unit can be painted a bright color to minimize the people who might trip over it, but there will always be a possibility of someone tripping over something that has volume.

2.4. Design Parameters

In Axiomatic Design theory, DPs describe what the product will look like or contain. They coordinate with the FRs one to one, for example, FR 2.1 matches with DP 2.1. The top-level DP for the decomposition is a smart backstop that measures speed and position of ball and gives feedback to user. Under this, there are seven sub-DPs each with their own sub-DPs. These can be seen in Appendix D.

After researching the physical options for each DP, there were multiple options for many of the DPs in this decomposition. A matrix, similar to a decision matrix, was created containing different products that could fulfill that DP on the top row and the criteria on the left-most column. The criteria included the selection and optimization criteria as well as specific criteria the team deemed important when looking at DP options. While doing this, research on the types of sensors and their typical uses was conducted and can be seen in Appendix A. This research gave the team a better picture on the capabilities of the products being considered. The chosen products are discussed in Section 3.4. The full matrices used to select the hardware can be seen in Appendix B.

2.5 Conditional Functional Requirements and Design Parameters

This design had a lot of flexibility because of its simplicity. Because of this, there were FRs that could be added based on how much the product should cost. For example, FR 5: Determine the spin rate, is an additional feature that can be added to the design if the customer is willing to pay extra for it. It costs extra because the sensors needed to measure the spin rate are more expensive and require a more sophisticated program. The team also had to utilize outside help for the computer programming parts of this project. Because of the programmer's time restraints, measuring the spin rate was conditional. Additionally, FR 1.1: measure distance that ball travels in the x direction, could be improved upon to become a vector in three dimensions rather than just measuring the range in the x direction. This measurement could result in velocity reading rather than just speed. This function depended on the ability of the sensor to distinguish the difference in locations in three dimensions.

2.6 Design and Decomposition Equations

When looking at verifying that the FRs are good, the team looked at what the minimum amount of FRs that are collectively exhaustive and mutually exclusive (CEME min) be. This means that the number of FRs is the minimum number of independent items that completely describe what the design needs to do. To determine if FRs are CEME min, metrics and design equations are added to a decomposition.

Metrics describe how well the FR is done. For example, if the FR is to measure speed, the metric could be measurability. These metrics go into equations that describe the FR. For example, measurability could be the sum of accuracy and precision of the system. To determine that the FRs are CEME min, the sum of the children FRs should be equal to the parent FR for each aspect of the decomposition.

This process of adding metrics and design equations is obscure and not well studied. Additionally, the team did not have much experience in this process and thus this part of the decomposition is less reliable.

2.6.1 CEME Min Analysis

The AD theory states that the FRs for the design needs to be collectively exhaustive and mutually exclusive using the minimum number of FRs (CEME min). Collectively exhaustive means that the CNs of the design have been fully fulfilled by the FRs and everything that the design needs has been laid out. By defining all the parts of the design functions, they can be controlled. Mutually exclusive FRs means that none of the functions are saying the same thing. If there are overlapping functions, there will be coupling in the design matrix. The coupling and independence analysis of this design is discussed in Section 2.7 below.

The team believes all the FRs that are not specified as coupled in Section 2.7 are CEME min. As shown in the full AD in Appendix D, the children FR's metrics go into the parent FRs. One caveat to this is that the relations between the children's and parent's metrics is usually Σ children = parent or the square root of the children = parent. This is not always the case in this decomposition.

2.7 Design Matrix

A design matrix was made to determine the FRs that the DPs must influence. It is used to check if the design fulfills axiom one of AD or what is called coupling. It is a simple matrix format, in which a 0 is placed in the box if the DP of that column must affect the corresponding FR in that row (meaning they are coupled). Ideally, the matrix should follow a downward diagonal pattern in which each DP only influences the corresponding FR (DP 1.1 influences FR 1.1). Realistically, due to constraints, it is acceptable to not have a diagonal matrix. Additionally, there are multiple FRs and DPs taken out because of repetition. If a DP influences FR 1.1, it inherently influences FR 1. Thus, only the lowest levels of each section are left on the matrix. This also applies to DPs because if DP 1.1 influences an FR, DP 1 will inherently influence it as well.

As shown in the matrix (shown in Appendix D), there is coupling outside of the diagonal. When coupling occurs below the diagonal, this is acceptable and represents that there is an order to do things. For example, DP 1.3 influences not only FR 1.3 but also 1.4. This means FR 1.4 should be performed after 1.3 so that 1.3 is not altered when changing 1.4. Because the matrix contains coupling outside of the diagonal, there is room for future work in the future to limit or remove non-diagonal coupling. To see potential future work that could do this, refer to Section 7.4. To see the whole Design Matrix, refer to Appendix D.

In the breakout matrix (that corresponds to the breakout decomposition), there is coupling. DP 6.1 affects FR 6.5 and FR 6.6. This is because 6.1 is a battery and 6.5 and 6.6 both rely on power to work. This shows that there must be a battery for the indication of the sensors being on or off and for the data to be transmitted to the feedback system. This was determined to not be a big problem for the design because the battery should not affect the quality of data transfer or the LED sensors from working.

3. Physical Integration and Prototype

3.1 Physical Integration of the Design Parameters

The physical integration of the design includes sensors that are purchased, a program that utilizes the sensors, a backstop that is purchased, and a housing unit to hold the sensors. A physical integration of a design involves all the DPs put together into a prototype design. Shown below is the block diagram of the prototype's electronic DPs.

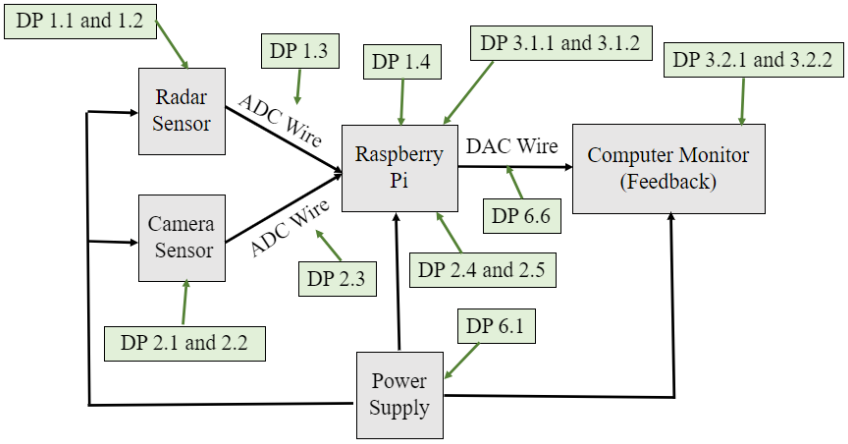

Figure 4: Electrical Block Diagram of Physical Integration

There are a few DPs not shown in the diagram above either because they were not integrated into the current prototype or because they are not directly part of the electronic system. DP 5: system to measure spin rate, DP 6.4: on/off switch, and DP 6.5: LED light to indicate on or off, were not integrated into the current prototype due to time restraints.

The housing unit was another major aspect to the design as it holds and protects the sensors as well as allowing the sensors to be at the best height and angle for accurate and consistent readings. See below for the final housing units.

Figure 5: Final Radar Housing Unit

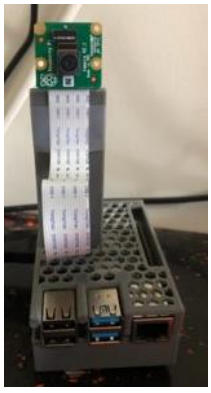

Figure 6: Final Camera and Raspberry Pi Housing Unit

The housing unit was designed to minimize potential injuries and protect the hardware from being damaged. The radar housing unit clips onto the side of the backstop using screws and the transmitter and receiver can transmit and receive radar signals from the window. The plug that goes from the radar to the Raspberry Pi fits into the slit on the bottom of the housing unit. The camera sits on top of the housing unit that contains the Raspberry Pi. The housing for the Pi contains holes to prevent overheating of the Pi and areas for plugs to be inserted. The team ended up prioritizing functionality of the housing units for testing purposes. A such, it does not fulfill all the protective and preventative aspects the AD called for. Further development of the housing unit would be beneficial. Below discusses the materials used in the housing unit and why it was chosen.

3.1.2 Housing Unit Materials

For housing for the Raspberry Pi and Radar the team chose PLA as the 3D printing material. PLA was chosen because it has good strength and stiffness to protect the hardware if hit by stray balls. It is also low cost which is beneficial for if large quantities of the product are produced. Some of the other materials considered were ABS, Polyamide, and PETG.

3.2 Iterations of the Housing Unit

This section discusses the iterations of the housing unit that were developed as well as how and why they were improved.

3.2.1 Iteration 1

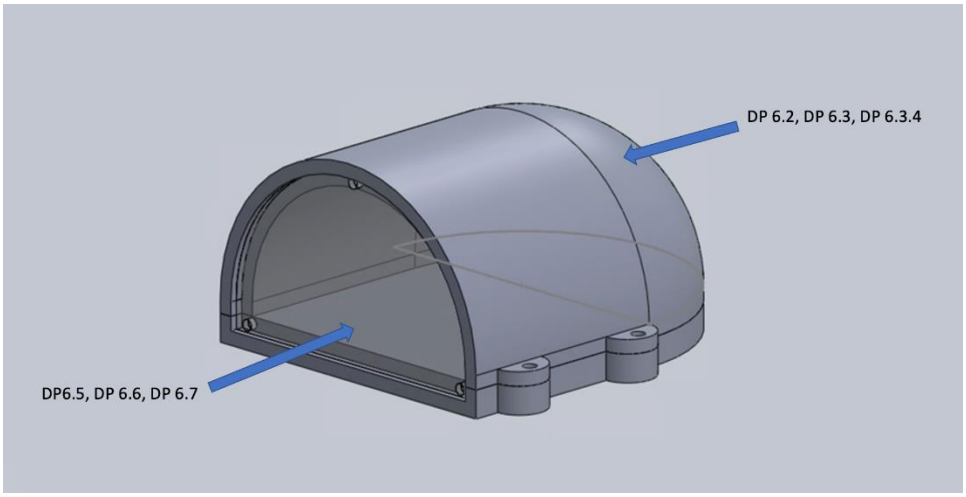

Figure 7: CAD model of Iteration 1 of Housing Unit

The first iteration of the housing unit, shown above, contains a window so that the hardware can “see” and sense the ball’s movement. It is mostly rounded to prevent injuries. It is 9 in long (flat end to dome end) x 6 in wide (side to side) x 3.5 in tall (bottom to top). After this, the project advisers suggested the team think of all the ways a user could be hurt using this product and try to prevent those in the next design while maintaining the integrity of the functions. The function of this container is to contain the sensors and power supply and allow them to work as they should.

The list of potential injuries and obtrusions and potential solutions to them is below. The potential solutions to these problems were brainstormed by the team in a meeting.

1. Someone could run it over with a lawn mower
    1.1. Paint it a bright color
    1.2. Make material hard enough to not be destroyed by lawn mower
2. Someone could trip over it
    2.1. Paint it bright color
3. Someone falling on it
    3.1. Completely rounded, dome w/ half of it clear material, or like hot dog shape
    3.2. Two half domes latching together in the middle
4. Net falling over onto it
    4.1. Keep net weighted down
    4.2. Make tripod legs attached to the ground w/ hooks or
5. Child consumes parts
    5.1. Use large pieces that aren't easily swallowed or choked on
6. Someone could be cut on sharper parts
    6.1. Use latches instead of screws or nails
    6.2. Use rounded shape
7. Flammability
    7.1. Use nonflammable materials

The problems with this iteration, outside of or pertaining to potential injuries, are the flat end, having too much space on the inside, and nothing to keep equipment in place.

3.2.2 Iteration 2

The second iteration of the housing unit is shown below.

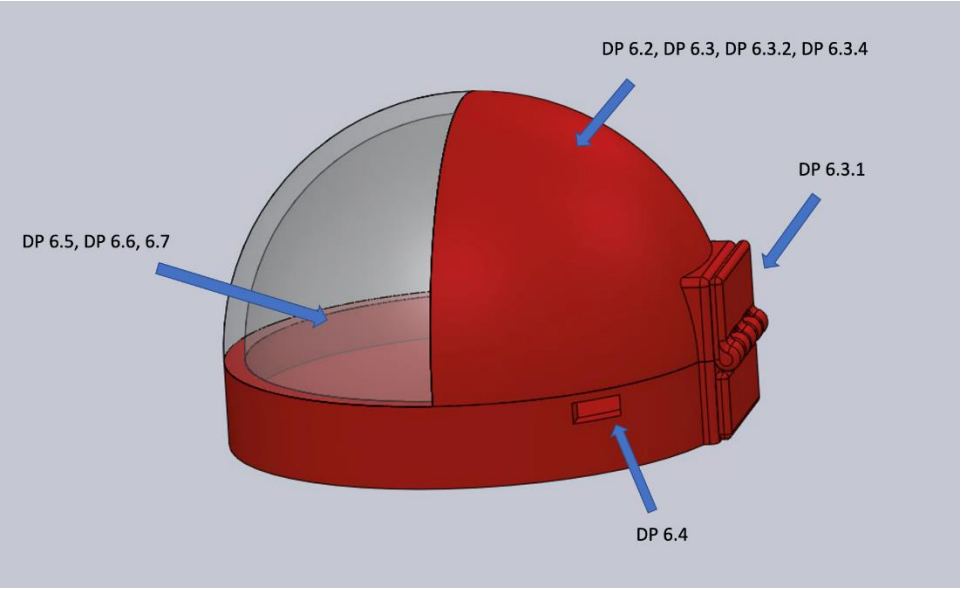

Figure 8: CAD model of Iteration 2 of Housing Unit

The second iteration of the housing unit was designed with a similar concept to the first. The main feature that was added to the second iteration was the rounded housing. This was added to the housing to reduce injuries in case anyone fell on the housing. We also added a latching system for the dome for easy access in case any of the hardware needed to be replaced. Another feature that was added was a slot for the on/off button and holes for ventilation of the hardware. While this did a better job at preventing potential injuries, it did not fulfill the functions needed for the sensors to work. For one, the sensors needed to be kept at two different locations, the camera directly behind the backstop, a few feet off the ground, and the radar on the side of the backstop, also a few feet off the ground. Due to time constraints, the team decided to design the final iteration for testing use only. The team suggests, further development of the housing unit.

3.3 The Computer Program

Open-source code was referenced throughout the project. Creating the virtual environment was based on an online tutorial. This tutorial can be found in appendix E (Cuni & Duran, 2016).

3.3.1 Creating the Environment

In order to establish the code for the sensors, an ideal environment is set up to handle everything at once. The environment is established to allow for the applications to be run in not only a user-friendly way, but to allow for all the necessary applications to be able to run at the same time. This process of picking the right environment was the most time-consuming part of the coding process. Being able to run multiple applications located in one place involved trying different methods, using different kinds of software, and finally finding a process that works for the desired outcome.

3.3.1.1 Preparations for the Environment

The first step was to set up the operating system within the Raspberry Pi. The next step is to image the Raspberry Pi's SD card. The system used for this used a Raspbian Buster. For this project, the team selected to use the Raspberry Pi OS Full (32-bit). Once the Raspbian Buster file was downloaded, it was flashed onto the micro-SD card using Etcher.

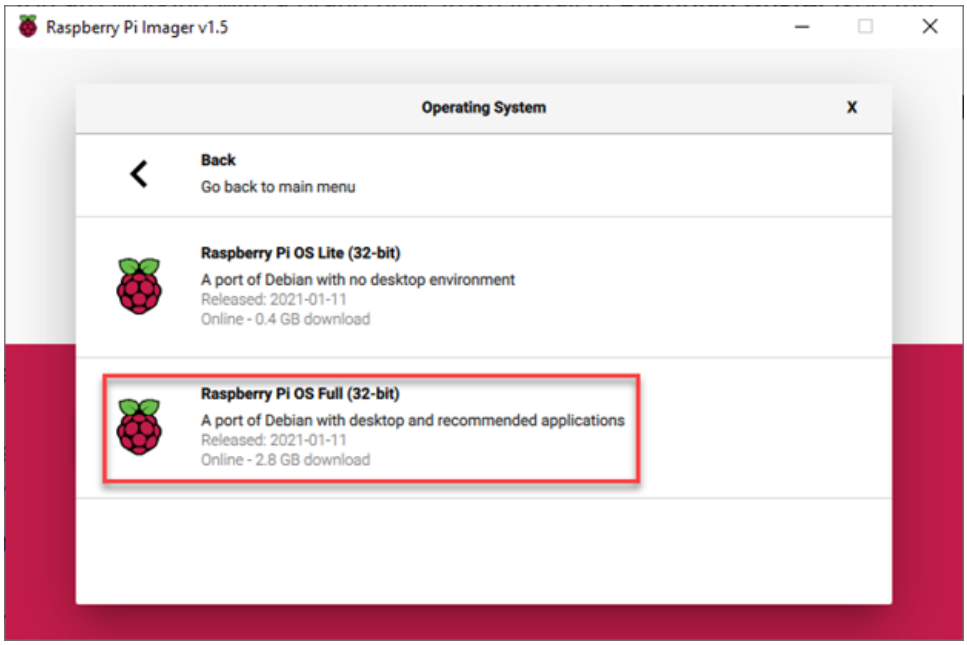

Figure 9: Image SD Card with Raspbian

Figure 10: Raspbian Buster Flashed with Etcher

The next step in the process was to expand the file system to use the entire SD card on the Raspberry Pi. This would allow for maximum output of the Raspberry Pi while running the applications in the environment. The first step in this process was to run “raspi-config” to expand the filesystem. From the menu, the “7 Advanced Options” was selected in the menu items of the Raspberry Pi. From there, the “A1 Expand Filesystem” option was selected, the program was run, and “<finish>” to end this process. Finally, to update the Raspberry Pi, it was rebooted using the command “sudo reboot”. Once rebooted, the filesystem was expanded to allow for all available disc space on the misco-SD card of the Raspberry Pi to be used.

Install OpenCV 4 on Raspberry Pi 4 and Raspbian Buster

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G  5.3G   23G  20% /
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  8.6M  1.9G   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mmcblk0p1  253M   40M  213M  16% /boot
tmpfs           386M     0  386M   0% /run/user/1000
```

Figure 11: Expanded Disc Space Verification

3.3.1.2 Dependencies

Updating and upgrading the existing packages was the next step in creating the environment in the Raspberry Pi. The commands explained in this section will reference the letter next to the commands in the figure below. This was done using the command labelled a below. The next step is to install some useful developer tools, including CMake, which aids in the configuration of the OpenCV building process. This was done using b. Next the image I/O and video I/O packages were installed using the commands in c, d, and e. These installations allow the environment to read various different files of photos and videos. The command lines d and e, which are so the environment can read videos, also allows for live stream videos to be used. Lines f, g, and h were installed next in order to allow for sub-modules to be compatible with the environment. This allows for basic GUIs to be used, which will be talked about later in this section. Line i was added in the dependencies next, which allowed for optimizations within OpenCV. Next lines j and k were run to further aid the environment in working with outside GUI systems. Finally, line l was run to install Python 3 header files, which will help the environment in compiling OpenCV resources with Python bindings.

```
a. sudo apt-get update && sudo apt-get upgrade
b. sudo apt-get install build-essential cmake pkg-config
c. sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev
d. sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
e. sudo apt-get install libxvidcore-dev libx264-dev
f. sudo apt-get install libfontconfig1-dev libcairo2-dev
g. sudo apt-get install libgdk-pixbuf2.0-dev libpango1.0-dev
h. sudo apt-get install libgtk2.0-dev libgtk-3-dev
i. sudo apt-get install libatlas-base-dev gfortran
j. sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
k. sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
l. sudo apt-get install python3-dev
```

Figure 12: Dependencies for the Environment

3.3.1.3 Python Virtual Environment

The method of creating the environment for this project used Python Virtual Environments, since it is proven to be compatible with Python, which is the coding language the Raspberry Pi reads in. The Python Virtual Environment allows for other Python packages to be managed inside the virtual

environment. This virtual environment will allow for isolated development, testing, and production on the system.

The Python Virtual Environment manager used for this project is “pip”. This virtual environment manager is extremely compatible with Python, allowing for ease of management within the system. Pip was installed using the commands in the figure below.

```
Install OpenCV 4 on Raspberry Pi 4 and Raspbian Buster
1.  $ wget https://bootstrap.pypa.io/get-pip.py
2.  $ sudo python get-pip.py
3.  $ sudo python3 get-pip.py
4.  $ sudo rm -rf ~/.cache/pip
```

Figure 13: Installing "pip"

Two other virtual environment managers, Virtualenv and Virtualenvwrapper, were also installed to allow for maximum flexibility while fine tuning the code. Virtualenv and Virtualenvwrapper were installed using the command “sudo pip install virtualenv virtualenvwrapper”. Once installed, the command “~/.bashrc” was ran to open the file. The following lines were appended below. Once appended, the changes were applied to the current bash session using the command “source ~/.bashrc”.

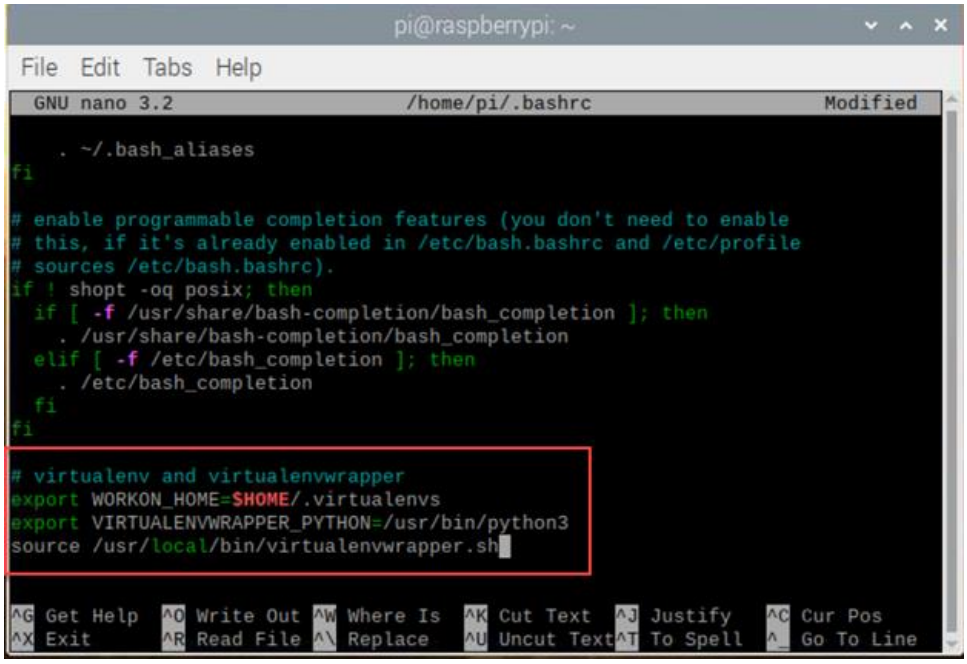

Figure 14: Updated /.bashrc with Virtualenvwrapper Settings

Finally, in order to confirm the Python 3 Virtual Environment, the command “mkvirtualenv cv –p python3” was run. From now on while coding, the virtual environment created will use the tag “cv”.

Since the project will be using a Raspberry Pi camera module, PiCamera API was installed next to the environment. This was installed using the command “pip install “picamera[array]””. This will be useful in the future.

3.3.1.4 OpenCV 4

The next addition to the virtual environment is to install OpenCV. This software is essential in creating an environment that uses real-time computer vision. OpenCV is an open source computer vision library that will allow for the application of detecting the baseballs via the camera. Not only can this open source library aid this project in object detection, its uses are endless such as being useful in motion tracking, robotics, facial recognition, and more.

Since the cv virtual environment is established, installing OpenCV 4 into the environment is simple. Using the command “pip install opencv-contrib-python==4.1.0.25”, OpenCV is compatible with the cv environment.

3.3.1.5 Pre-made Utilities for OpenCV

To allow for ease of usage with OpenCV, an additional add-on was installed into the environment. This add-on improves the environment at reacting to both live-feed and photos/videos better. In the environment, the pip command “pip install –upgrade imutils” was run. Next, the folder: “/home/pi/.virtualenvs/cv/lib/python3.7/site-packages/imutils” was opened, and a new Python file was created, labeled “range-detector.py”. In this file, open source github code was copied into it. This code can be found in Appendix E.

3.3.2 Radar Code

The radar sensor uses basic code that runs the OPS 241-A radar sensor. It simply reads in MPH and KPH down to 2 decimal places. To ensure that the radar is accurate and not collecting unnecessary data, a minimum speed of 20 mph is set. This allows for objects other than a baseball to not be detected, such as other people walking around. If the user requires a lower minimum threshold, the minimum speed can be changed accordingly.

The radar speed is also used as a trigger switch for the camera. In order to not overload the Raspberry Pi with data, the radar sensor ensures that minimal data is collected. When the radar picks up data, it in turn makes the camera collect a 2 second recorded video. From the time stamp of the radar, it will take a recorded video 1 second before and 1 second after the time the radar read the speed. More about this topic is explained in Section 3.3.3.

Finally, the radar data is sent to the GUI to be displayed to the user. It is set to only read the highest speed collected each pitch. This allows for another point of deleting unnecessary data from the Raspberry Pi for maximum output.

The code referenced in this section can be found in Appendix E.

3.3.3 Camera Code

As stated in the section above, the camera selectively records videos to be analyzed. The camera is always actively on and recording, but unneeded data is deleted. Once the timestamp from the radar speed is sent to the camera code, the Raspberry Pi simply collects the recorded video 1 second before and 1 second after the timestamp. This allows for the code to only analyze 2 second trimmed videos opposed to 2+ seconds.

Once the trimmed 2 second video is saved, the code then analyzes every fifth frame of the video. It then goes on to pick the frame with the baseball with the biggest diameter (signifying the ball is closest to the camera) to be analyzed.

The frame picked by the code now goes through the object detection phase. The code is trained to pick up on circles so the analysis is narrowed down tremendously. In order to accurately pick up the baseball, the color range is selected according to the kind of ball used. In our case, we used a yellow ball for the highest performance. Both the upper and lower limits of the color of the ball was picked on a basic RGB scale from 0 to 255. This allowed for accurate readings under any light (or dark) settings. The color scale can be changed according to what

Now that the baseball is detected by the software, it is then overlayed with a strike zone box to determine if it is a ball or a strike. There are 4 points on the detected baseball that are analyzed with the strike zone box. Using a compass for comparison, a point at north, east, south, and west are analyzed. If any of these points are within the strike zone box, the GUI will read strike. If all 4 are outside the box, it will read ball.

The code referenced in this section can be found in Appendix E.

3.3.4 Feedback Via a Graphic User Interface (GUI)

A user interface was created in order to give feedback to the user. This is also known as a GUI (graphical user interface). There are many open-source GUIs out there, but since our project is built to use a Raspberry Pi, the GUI selected was from the Tk GUI toolkit. From this set of GUI interfaces, the Tkinter package was used to create the GUI for the project. Tkinter was also made for Python, which made it a good choice.

The radar sensor data is sent directly to the GUI during the process and read on the screen. The camera uses the frame previously discussed, makes the whole background dark, and overlays a white strike zone box. The ball will be shown as a red circle on top of this picture. Finally, this picture is also sent to the GUI to be displayed next to the pitch speed. The result of the pitch (ball or strike) is also displayed on the GUI. Shown below is an image of the GUI.

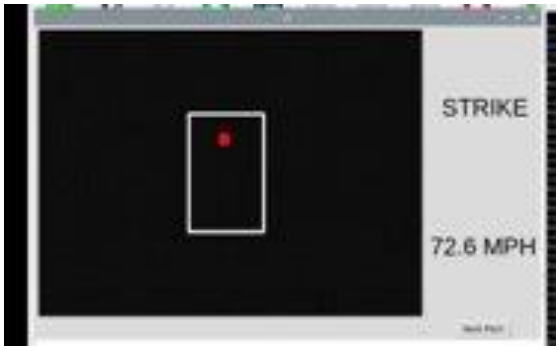

Figure 15: Feedback GUI

The code referenced in this section can be found in Appendix E.

3.4 Selection of DPs

When looking at which hardware to buy to fulfill the DPs, the team created a selection criteria matrix that compared multiple different sensors against the criteria the team deemed important. This section will go over the hardware that was chosen. To see the other hardware that was considered, refer to Appendix B, which contains tables comparing the different hardware options.

The team chose Raspberry Pi 4 as the computer for the system because of the team's familiarity with coding on a Raspberry Pi. The specifications of the Raspberry Pi are:

- Max Current Draw: 600mA
- Power Supply needed: 5.1V, minimum 4.63 to operate

3.4.1 Measurement Hardware

When choosing the speed measurement hardware, the team decided to use the OPS241-A Doppler Short Range Radar Sensor from OmniPreSense. This was chosen because it had open-source coding that could be used by the team. This radar sensor worked the way it needed to with few changes in the original source code.

When choosing the position measurement hardware, the team decided to use camera vision hardware. The team chose the Camera Module V2 from Raspberry Pi. This camera was for testing to see if it would be possible to use cameras for this project. This specific camera was chosen because it was specially designed to work well with the Raspberry Pi. The Raspberry Pi has a built in CSI camera connector that easily connects all 15 pins of the camera into it with ease. Using this specially made camera for the Raspberry Pi was an attempt to allow maximum data transfer between the camera and the Pi.

The specifications of this camera are:

- Still Resolution: 8 Megapixels
- Modes: 1080p30, 720p60, 640x480p60/90
- Resolution: 3280x2464

- Horizontal field of view: 62.2 degrees
- Vertical field of view: 48.8 degrees

3.4.2 Feedback Hardware

When choosing the feedback hardware, the team decided to use a desktop to display the measurements. The team chose this because it has adaptability for users and can be placed in the position the user finds most helpful. The display can be either a computer or just a monitor. If a monitor is used, a mouse and keyboard need to be attached to the Raspberry Pi.

Connecting this display to the system is an easy process. The only necessary step is plugging the display into the Raspberry Pi via an HDMI to USB adapter. Once this display is plugged into the Raspberry Pi, the program can be run accordingly.

All the data collected from a pitching outing is stored on the Raspberry Pi. The files get backed up onto the Raspberry Pi's hard drive for analysis later. The radar speed is saved as text files, and the location via the camera is saved in mp4 files.

3.5 Cost

The current device is under $500 and thus affordable under the terms defined earlier. There is room to grow the product as well. Assuming future production includes a better-quality (more expensive) camera, bigger batch sizes which allow for better prices on items, renting workspace, and paying employees to produce and further develop the product it is still feasible to maintain an affordable product depending on the customer demand. Below is the cost calculation for the prototype.

Table 2: Prototype Cost Analysis

Item	**Cost**
Raspberry Pi 4 kit	$120.00
OmniPreSense 241-A Radar Sensor	$160.00
RPi camera	$25.00
USB Cable	$10.00
Miscellaneous Hardware	$5.00
3D Printing	$10.00
Total	$330.00

The tripod used for testing purposes is not included in the estimated cost.

4. Testing of Design

4.1 Running the Software

In order to run the program, the virtual environment needs to be initialized. After the Raspberry Pi is turned on, the file explorer is opened. The path to open the virtual environment is: /home/pi/.virtualenvs/cv/backstop-cam/src. Once here, a terminal is then opened in this location. In this terminal, this next step is to enter the correct directories with all of the project files. This is done using the command: cd /home/pi/.virtualenvs/cv/backstop-cam/src. Now the virtual environment with all of the dependencies are opened next. This is done using the command: cd workon cv. Finally, in order to turn on the sensors and begin to take readings, the following command is entered: python3 backstop –gui.py. The terminal should look like the figure below.

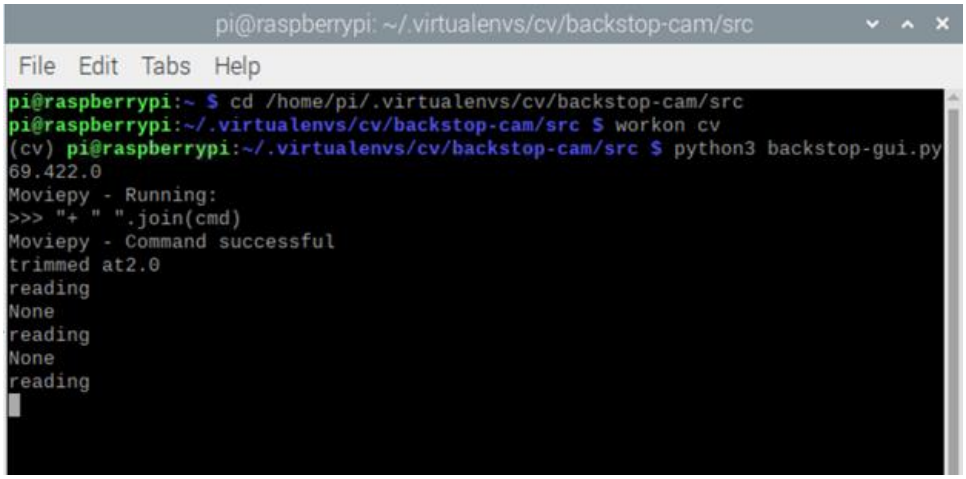

Figure 16: Terminal Running the Project

4.2 Initial Testing

While conducting the initial testing of the Omni Sense 240a radar sensor, readings were often inaccurate and inconsistent when taken from behind a netted backstop. One of the issues the team thought could have caused it was interference between the radar sensor and the radar gun. Another consideration was that the radar was picking up the movement of the backstop netting after impact rather than the movement of the ball. To solve the issue of interference we used only one device at a time to measure velocity. After some test and comparing the values obtained by both under similar circumstances, it was determined that the radar sensor was measuring velocity to an accuracy of approximately +/- 2 mph compared to the radar gun. To eliminate the issue of the radar detecting the net instead of the ball was to place the radar sensor above the frame of the backstop. This eliminated the detection of the net, but the radar was having trouble picking up readings for balls that were about two feet off the ground or lower. We attempted to fix this by angling the sensor downward, but the issue persisted. Our next attempt was directly off to the side of the backstop frame about three feet off the ground. This position detected balls thrown at various heights off the ground and did not detect unwanted values. From these discoveries, we determined that the housing unit for the radar needed to allow the radar to be directly to the side of a backstop

and have adjustable heights and angles to ensure correct readings. These functions were added to the Axiomatic Design.

When testing the camera vision, multiple issues came up. The first issue occurred when selecting the measurements for the color of a baseball. The vision software read in terms of RBG. Each color was on a scale of 0 to 255. Within each color of the RGB scale, a minimum and maximum was selected for each one. Since a typical baseball is white with red seams, the software did not capture the ball consistently. This was solved by using all yellow balls instead. Though using a yellow ball helped with the consistency of readings, it was still only picking up partial parts of the ball. The next fix involved adding a blur effect to the software. This allowed for unwanted noise in the background of the images to be removed. This blur effect helped with the edge detection software to perform its job more accurately. There was also a reoccurring problem with blue. Objects that were blue would sometimes get picked up in addition to the ball. To read more about the failures encountered and solutions and work arounds used while testing the camera vision, see Appendix E.

4.3 Testing Results

The team set up the camera sensor on a tripod six feet behind the backstop and three feet off the ground, with a 3D printed housing to hold it to the tripod. The radar sensor was attached to the side of the backstop using a 3D printed part. Both the sensors were connected to the Raspberry Pi computer and the RPi was connected to a monitor. The program to analyze the data and the GUI used to show feedback was pulled up on the monitor. To test, the team threw yellow balls that the camera vision had been programmed to detect at the backstop.

While testing, many complications came up. While both the camera and the radar work, there are stipulations. The radar has trouble reading from on top of and behind the backstop. The radar does work from the side and had been proven to work before through the initial testing. Because it had worked before, the team determined that the problems with the radar were due to the hardware and not the program. After further testing, the team believes a power problem when running off the RPi causes inconsistent readings. The radar's accuracy and quality are discussed further in Section 5.3. When the radar was able to take the reading, the program can show a visual feedback of the speed on the computer GUI.

The camera vision currently only works with videos loaded into the analysis system and not with live data. So, a video recorded using the camera can be taken, but the analysis done on it has to be done by uploading the video to the program. Once it has been analyzed, the program shows the visual feedback of the ball on the strike zone through the computer GUI. The computer program used needs to be altered to be able to read live videos rather than prerecorded videos. Currently, the GUI of the camera vision output is correct when using a screenshot from a prerecorded video. There was not enough testing to confidently define the accuracy of the camera vision. Additionally, the camera vision processing in the future should handle live readings which could change the overall accuracy of the system.

Shown below are photos of the screenshot from the prerecorded video and the GUI representing where the ball is in comparison to the strike zone. This shows that the GUI feedback system works in these circumstances.

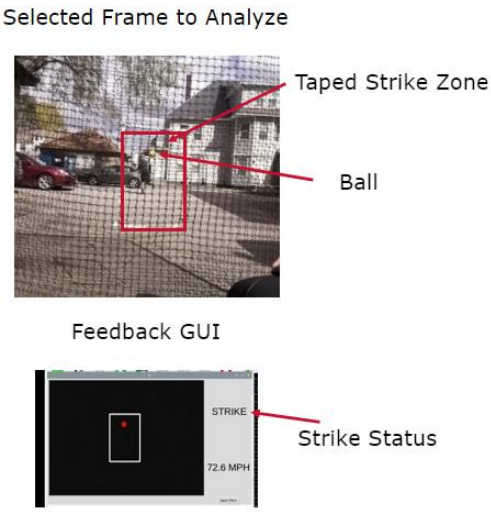

Figure 17: Camera Vision Test GUI 1

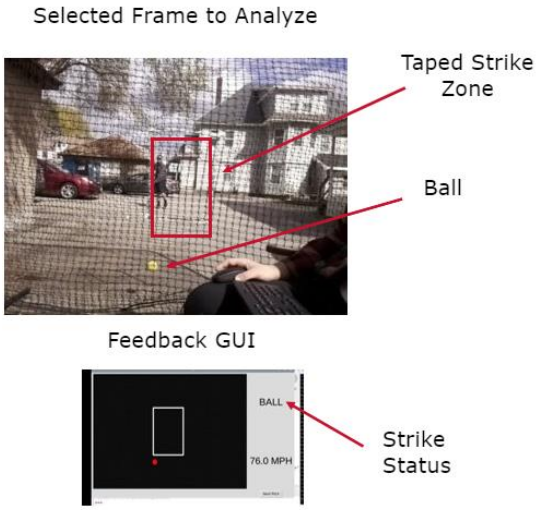

Figure 18: Camera Vision Test GUI 2

5. Quality

Quality does not fall under one simple definition. Depending on who you ask, there are many different definitions. It is described as the "ratio of the perceptions of performance to expectation," or how well the performance stands up to the expectations (Besterfield, 2013). Quality is also defined as a combination of multiple dimensions, as shown below (Besterfield, 2013).

1. Reputation - Past performance and other intangibles
2. Performance - Primary product characteristics
3. Features - Secondary characteristic
4. Conformance - Meeting specifications or industry standards
5. Reliability - Consistency of performance over time
6. Durability - Useful life
7. Service - Resolution of problems and complaints
8. Response - Human-to-human interface
9. Aesthetics - Sensory characteristics
10. Reputation - Past performance and other intangibles

By identifying these dimensions in a specific product, one can identify the quality of said product. It is not a specific calculation, but rather what you define it as. The International Organization for Standards (ISO) has multiple standards on quality management. The ISO 9000 series is a standardized quality management system that covers fundaments and vocabulary, requirements, and improvement guidance (Besterfield, 2013). ISO 14000 is a standardized environmental management system for processes. The overall goal of ISO 14000 is to balance support of environmental protection and prevention of pollution and socioeconomic needs (Besterfield, 2013).

5.1 Quality 4.0

Quality 4.0 is a set of criteria or "axes" that allow operations to be more efficient and higher quality (Marr, 2019). It combines aspects of people, technology, and process to create better products and systems (Marr, 2019). The axes of Quality 4.0 include data, analytics, connectivity, collaboration, app development, scalability, management systems, compliance, culture, leadership, competency (Marr, 2019). Industry 4.0 references the new, or 4th, "industrial revolution" which introduces new technology into the working world that allows computers that run machines to create a network and communicate between each other (Rigert). In this sense, Industry 4.0 is simply technological, but it will bring changes to people in the company and the processes that are used. Quality 4.0 is relevant to Industry 4.0 because companies can use the technology from Industry 4.0 to bring better quality and less defects thus, bringing on quality 4.0. Quality 4.0 also relates to people because the people behind the scenes are still very much part of the process and can cause or prevent defects. In other words, people are behind the scenes creating and analyzing the technology, such as codes, data, and machines, that are part of the processes in a company. The tools used in Quality 4.0 include artificial intelligence, big data, blockchain, deep learning, enabling technologies, machine learning, and data science.

5.2 Applying Quality Practices

When considering how to apply quality to a product or process, it is important to define quality assurance and quality control. Quality assurance is the 'how' of how a process is performed or a product is made (Coleman). There is a focus on "providing confidence that quality requirements will be fulfilled" (Coleman). Quality control is a subsection of Quality assurance. Control focuses on checking that the requirements are being fulfilled (Coleman). To ensure requirements are being fulfilled, there needs to be inspection and auditing on the process and products. There are multiple strategies used to investigate the status of quality in a process or product.

Failure Mode & Effect Analysis is a technique that recognizes potential failures and its effects, actions that that could prevent and reduce the chance of failures occurring, and document the process (Besterfield, 2013). Quality Function Deployment identifies and sets priorities based on customer satisfaction for process improvement (Besterfield, 2013). It works by employing the voice of the customer to make sure all aspects of the company meet or exceed customer expectations (Besterfield, 2013). Another strategy used to determine if your process fulfills your expectations on quality is benchmarking. Benchmarking is the idea of comparing your process to a different company's process that you think of as doing better (Besterfield, 2013). The comparison can be used to make your process better (Besterfield, 2013). Quality by Design is the practice of using a multidisciplinary team to work on concept, design, and production planning at the same time (Besterfield, 2013). This would be different from the traditional way of doing things where one department completes their task then hands it on with little to no communication with the following or previous departments.

Six Sigma is another tool and philosophy to use when looking at the quality of a process or product. It looks at the number of nonconformities, or defects, within a process and has a method to decrease that number (Besterfield, 2013). The six sigma itself refers to being +/- six standard deviations from the mean, which means 99.9999998% of the process will fall within the specifications (Besterfield, 2013). Because 99.9999998% of the process is conforming, only 0.002 parts/million (ppm) will be considered nonconforming, or defective (Besterfield, 2013). Because of shifts of the mean (usually assumed to be +/- 1.5σ), the actual defective rate of a Six Sigma process is 3.4 ppm. Ideally, this product's accuracy would follow the Six Sigma theory in terms of inaccurate readings.

5.3 This Product's Quality

Quality for this specific product's procedure would be assessed by measures such as, how many defective products are produced, how quickly the process runs, the cost of quality. Because the team is somewhat multidisciplinary, this project is most closely related to quality by design strategies and would work well by adding benchmarking. Since the team is inexperienced in developing a production process and there are already multiple similar products on the market, benchmarking would allow the team to gain a basic understanding of the way this product is produced on a larger scale and improve upon it. The team could also utilize Quality 4.0 systems when developing a production process rather than trying to implement it into a system that already exists. It is important to note that when developing the prototype for this product, the team bought premade items and assembled them. When looking into a production process, it would be

beneficial to do a cost benefit analysis on buying premade products versus breaking down the design and making the products in house.

When there is a production process in place, it will be important to ensure a culture of quality as well. It is important to set employees up to succeed through training and having proper systems in place. There should be set lists of things to measure for quality assurance workers and clear procedures easily available throughout the workplace. Employees, no matter their level, should bring any complaints to top-level management who should take them seriously.

Because an important objective of this product is to have good accuracy and precision, the team used X bar and R charts to look at the variation and range in the measurement system. Because the camera was not able to work live, the team only looked at the speed measurements. The speed given by the prototype and the speed given by the OPS241-A Doppler Short Range Radar Sensor from OmniPreSense will be compared and the difference between them will be the X. There will be five samples per subgroup and six subgroups. Each subgroup takes the average of the X's and that becomes that subgroup's X bar. The X bars are then put into a chart and control limits are calculated. The range will also be measured by find the R from the maximum difference in each subgroup minus that same subgroup's minimum difference. The Rs are also compiled into a chart and control limits are calculated. If the camera vision system worked, the same process would follow, but the prototype reading will be compared to readings found through markings on the backstop. The equations and constants used to develop three sigma control limits are shown below, where UCL is the upper control limit and LCL is the lower control limit. Z refers to the statistical Z-value, which is the measure of how many standard deviations below or above the population mean a raw score is. Z upper refers to the standard deviations above the population mean, and Z lower refers to the standard deviations below.

Table 3: Equations Used to Develop Quality Control Charts

UCL xbar = X double bar + A2*Rbar
LCL xbar = X double bar - A2*Rbar
UCL R = D4*Rbar
LCL R = D3*Rbar
Z upper = abs(USL - X double bar) / Stand.dev.p(all differences between readings)
Z lower = abs(LSL - X double bar) / Stand.dev.p(all differences between readings)
Defects per Million Opportunities = 1 million * (tail proportion Z lower + tail proportion Z upper)
Sigma level = lower of the two Z values

Table 4: Constants Used to Develop Quality Control Charts

n	5
D3	0
D4	2.114
A2	0.577

Below are the results of the equations. The full set of test readings can be found in Appendix G.

Table 5: Results of Control Chart Equations

Subgroup	LCL X bar	LSL X bar	X bar	USL X bar	UCL X bar	LCL R	R	USL R	UCL R
1	-4.10	-1	-0.314	1	0.44	0	4.14	2	8.33
2	-4.10	-1	-2.318	1	0.44	0	4.1	2	8.33
3	-4.10	-1	-0.906	1	0.44	0	3.28	2	8.33
4	-4.10	-1	-3.184	1	0.44	0	5.1	2	8.33
5	-4.10	-1	-3.214	1	0.44	0	1.64	2	8.33
6	-4.10	-1	-1.05	1	0.44	0	5.37	2	8.33
	X double bar:		-1.83		R bar		3.94		
	Z upper (Z_u)		1.544		Z lower (Z_l)		0.453	<-Sigma level	
	Tail proportion Z_u		0.0618		Tail proportion Z_l		0.3264		
	Percent Nonconforming		0.3882		Defects Per Million Opportunities		388200		

Below are the results of the control charts.

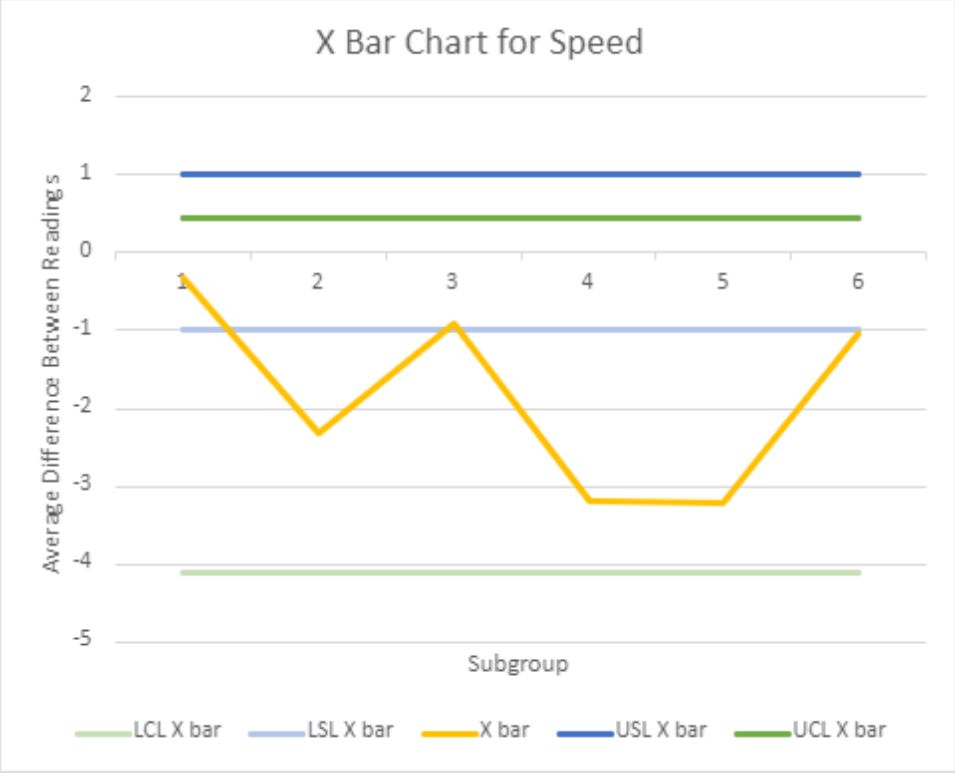

Figure 19: X Bar Control Chart for Speed

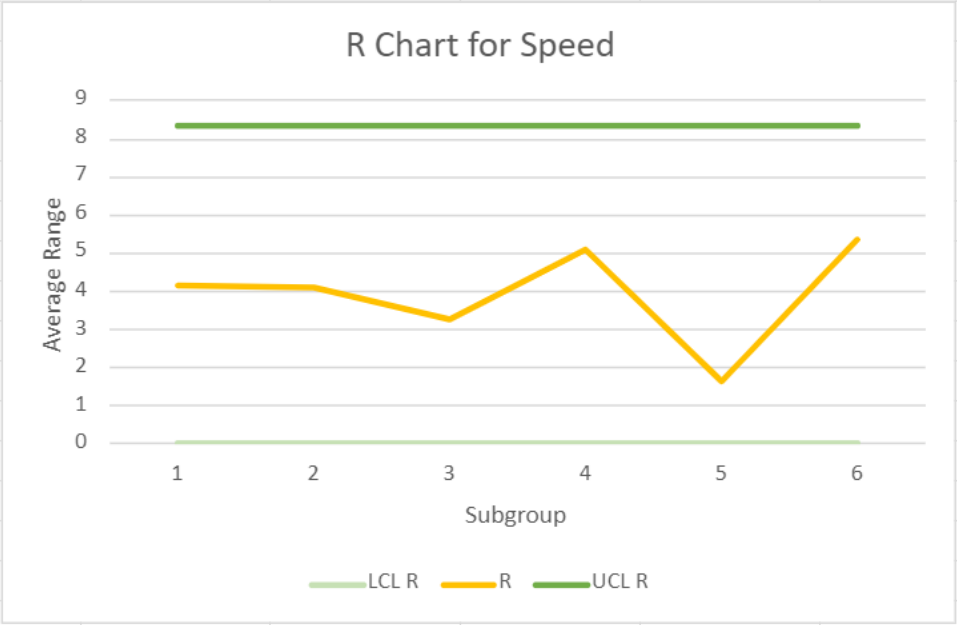

Figure 20: R Control Chart for Speed

As you can see in the speed control chart, both the charts are in control, or within the control limits, shown by the green lines. This means the process is acceptably precise and stable, meaning the readings are close to each other. But the process is not very accurate, as it is outside the specification limits, defined as +/- 1 MPH, shown by the blue lines (USL and LSL on the X bar chart). The radar sensor currently averages at about 1.82 MPH below the real speed, with the total range being –5.55 MPH to +1.92 MPH. This is not good, but a good starting point for future work. The accuracy can be improved with better hardware.

The sigma level was calculated to be 0.453385743. This means the process is not capable, as it is below 3.0. The process is not of very good quality, but it is stable which makes it easier to improve in the future.

6. Broader Impacts

When looking at any new device, it is essential that one considers the broader social, ethical, economical, and environmental impacts. Not only should one consider the device itself, but the manufacturing process, distribution process, and discarding of the device.

6.1 Engineering Ethics

When considering the ethics of a device, a good baseline to reflect upon is the Mechanical Engineering Code of Ethics. The Code of Ethics include the following.

1. Using their knowledge and skill for the enhancement of human welfare;
2. Being honest and impartial, and serving with fidelity the public, their employers and clients; and
3. Striving to increase the competence and prestige of the engineering profession.

This project reflects these principles in many ways. For one, this project enhances human welfare by introducing a cheaper option for a product that already exists. Honesty has been shown throughout the project by describing the failures encountered throughout the project, possible future work, and describing the abilities and limits of the product clearly.

6.2 Societal and Global Impact

Although this product is not the most influential product, there are still many consequences. Some intended impacts of this product are being able to have fun with friends, practice pitching, and have fun competitions with people. Some unintended consequences might be that it might make things too competitive and take the fun out of pitching. It could also cause injuries if used incorrectly, although the team has tried to mitigate as many potential injuries as possible.

6.3 Environmental Impact

When manufacturing any product, there will be an environmental impact. Because the housing will most likely continue to be made from plastic because it is inexpensive and easy to form, there will be negative environmental consequences such as air and water contamination. Additionally, plastic is often not recyclable or ends up not getting properly recycled. The energy needed to produce a large number of products is also a negative impact as the energy used in manufacturing processes is often not sustainable. Another consideration would be that users often do not recycle or dispose of products correctly. Since this product has electrical components, not disposing it correctly, will have negative impacts on the environment.

In the future, to lessen the environmental impacts, the device could be made out of a more sustainable plastic alternative. Additionally, the manufacturers could introduce a take-back program for old, broken, or unwanted products so that they can discard of them properly.

6.4 Codes and Standards

There are not a lot of specific codes for this type of product, but rather manufacturing process best practices and electronic standards. Since there is not a manufacturing procedure designed yet, this will not be discussed in this report. The electronics must follow National Fire Protection Agency electrical systems codes.

6.5 Economic Factors

Some economic factors to consider is making sure the customer gets the quality they expect for the product they are buying. Meeting or exceeding expectations is what keeps customers satisfied. Maintaining a reliable quality and price compared to the market competitors is also important.

Some negative economic impacts might be that umpires would be less desirable because over time, it would be more expensive. The team is not very worried about this because Major League Baseball uses a complex and accurate system to track baseballs and umpires are still used to verify the readings. Additionally, if baseball teams want to use this product, it might raise the price of joining the team to account for buying the product.

7. Discussion

The completion of this project revealed several notable conclusions. First, that Axiomatic Design is a viable design theory that encourages new ideas and organization of complex systems. Through the Axiomatic Design process, the team was successful at creating a device that provides users with measurement and feedback on the speed and position of a baseball. Future improvements can be made to the complexity of the measurements taken.

7.1 Axiomatic Design Analysis

By using the method of Axiomatic Design, the team was able to keep the system requirements independent of each other in the form of functional requirements and design parameters. By thinking of the FRs before the DPs, the team could think of multiple innovative ideas and choose the best one to fulfill the functions, rather than thinking of one possible solution. Because each of the FRs were satisfied through their corresponding DP, the team could ensure a good design before developing a physical prototype. Although the goal was to maintain that DPs only affect their corresponding FR, there were some instances of coupling, described below.

- DP 1.3, wires to transfer sensor data to computer for speed measurement, affect FR 1.4, calculate speed, because the data needs to be transferred for the calculation in the program to take place. This also follows for DP 2.3 and FR 2.4 and DP 5.3 and FR 5.4 which is the same but for the position measurement and spin rate measurement, respectively.
- DP 2.3, wires to transfer sensor data to computer for position measurement, affect FR 2.5, determine strike status of pitch, because the position measurement needs to be known for the strike status to be determined. This FR is also affected by DP 2.4, program that interprets coordinates of pitch on X-Z plane, because the coordinates need to be used to determine the strike status.
- DP 3.1.2, code that translates data from serial monitor to feedback system, affect FR 3.2.1, show feedback visually, and FR 3.2.2, announce feedback audibly, because there needs to be data to give feedback on.
- DP 6.1, power supply for sensors, affect FR 6.5, indicate whether sensors are on or off, and FR 6.6, allow for data to be transmitted to feedback system, because there needs to be power for the indicator light to work and for transmitting to take place.

Because all the coupling occurs below the diagonal, there was no need to rearrange the matrix and coupling can be disregarded by maintaining an order of events. For example, turning on the power source before using the feedback system.

Outside of the coupling, there are also aspects of the AD that did not completely make it into the physical prototype, as listed below. These are aspects that the team could not add or complete due to lack of time. The team recommends adding these aspects to further developments.

- DP 2.4, the program that interprets coordinates of pitch on X-Z plane, cannot do live interpretations.
- DP 3.1.2, the code that translates data from serial monitor to feedback system, is missing audible translation of feedback.
- DP 4.2, the gutter system to return balls, is not currently a part of the prototype.

- DP 5, a spin rate measurement system, is not currently a part of the prototype.
- DP 6.2, protective aspects for the sensors through the use of housing units, are not complete and requires the housing unit to be redesigned.
- DP 6.3, preventative safety measures (for users) for housing unit, are not complete and requires the housing unit to be redesigned.
- DP 6.4, an on/off switch for the sensors, is not currently a part of the prototype.
- DP 6.5, an LED light to indicate whether the sensors are on or off, is not currently a part of the prototype itself. But the sensors themselves come with light indicators to show if they are activated or not.

7.2 Analysis of Design Objectives

The objective of this project was to design and physically develop a smart backstop that can

- Measure speed of a pitch on a vertical 2D (X-Z) plane
- Measure position of a pitch on a vertical 2D (X-Z) plane
- Provide feedback on the speed and position to the user visually
- Provide feedback on the speed and position to the user audibly
- Be affordable, which is defined as under $500
- Be accurate
    - +/- 1 MPH for speed
    - +/- 4 inches for position

The team succeeded in fulfilling the objective of creating a design solution that measures the speed, through radar, and position, through camera vision, on the X-Z plane. The system gives visual feedback though computer GUIs. The cost of the product is under $500.

The team was not able to determine the accuracy of the total device because the sensors were not able to do proper readings while together. Despite this, the team did calculate the radar's accuracy alone to be 1.82 MPH below the real speed on average, with the total range being –5.55 MPH to +1.92 MPH. This is not within the +/- 1 MPH specification that was desired. Since the readings are in control, this accuracy can be improved in the future with better hardware. The position is not currently able to work live, but the GUI output from prerecorded videos was correct for all the times it was tested. It was not tested enough to confidently state the accuracy, though. Additionally, the feedback does not currently give audible readings. With further development, all the objectives can be met.

7.3 Future Work

Future work for this project can include adding more complex measurement abilities to the system, such as allowing speed and position to be measured in 3D space rather than in 2D space. The spin rate of the ball after it has been pitched can also be introduced through further use of camera vision. Using better-quality hardware for the device would help produce more consistent readings. The hardware parameters also need to be altered in the program to ensure the readings register every time. Further testing would also be beneficial. Using outside test groups and testing in different weather conditions would expand the understanding of this prototype's abilities. Further

development of the housing unit would also be beneficial. Additionally, the manufacturing process and business plan for this product could be developed further.

8. Conclusions

1. The team's diverse knowledge of different engineering skills allowed them to develop a product that utilized not only mechanical skills, but electrical, computer science, and quality analysis.
2. The team was able to complete most of the project goals despite meeting in person being restricted.
3. The team was able to follow a general schedule set in the beginning of the year despite the process and design itself being very open ended. The specifics of the schedule changed as the design increased in complexity, but the main goals were completed on time. This gave the team confidence in managing larger projects.
4. The team was able to investigate applying for a utility patent provisional which developed their entrepreneurial spirit.
5. Axiomatic Design provided a design method that was successful in creating a system that fulfills the customer needs.
6. The design solution created was then produced through the purchase of items, 3-D printing, and programming.
7. Functional requirements were analyzed to be collectively exhaustive and mutually exclusive using metrics and equations.
8. Design parameters were chosen by developing and comparing selection and optimization criteria. The selection and optimization criteria were developed by analyzing the customer needs and seeking knowledge on the potential DPs.
9. Potential coupling was analyzed using a design matrix. Coupling that did occur was designated as minor.
10. The team created a device that measures speed and position of a baseball in motion and provides feedback on those measurements through a laptop.
11. Through testing, the team determined the device needs to be made up of better-quality hardware and the programming work needs to be furthered to produce more consistent readings.
12. Future work on this project could include improvements in the complexity of measurements taken by the sensors. For example, the position could be measured in 3D space and the speed could be measured as a velocity vector in 3D space. Additionally, spin rate could be measured and the DPs that did not make it to the current prototype could be added.
13. Further testing would improve the understanding of the capabilities of the prototype.
14. After further work, the axiomatically designed smart backstop device will allow people to get feedback on their baseball pitches for fun and for use in athletic improvements.

9. References

APCI-3300, Pressure Measurement Board for PCI. *A-Tech Instruments Ltd.* Available at: http://www.a-tech.ca/Product/Series/464/APCI-3300_Pressure_Measurement_Board_for_PCI/?tab=2 [Accessed September 27, 2020].

Barfield, Jr., J.R., 2018. Pressure sensor assisted position determination. Available at: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=9967701.PN.&OS=PN/9967701&RS=PN/9967701

Berg, R.E., 2017. Ultrasonics. *Encyclopædia Britannica*. Available at: https://www.britannica.com/science/ultrasonics [Accessed September 22, 2020].

Besterfield, D.H., 2013. *Quality improvement: (formerly entitled quality control)* 9th ed., Upper Saddle River, New Jersey: Pearson.

Burnett, R., 2020. Understanding How Ultrasonic Sensors Work. *MaxBotix Inc.* Available at: https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm [Accessed September 22, 2020].

Coleman, L.B., Quality Assurance & Quality control. *ASQ*. Available at: https://asq.org/quality-resources/quality-assurance-vs-control [Accessed March 25, 2021].

Cuni, A, Duran, A.S. (2016) range-detector (Version d9264d0) [Source code]. https://github.com/jrosebr1/imutils/blob/master/bin/range-detector

C-K Theory. Available at: https://www.ck-theory.org/c-k-theory/?lang=en [Accessed October 24, 2020].

Delta DRS1000 Non-contact Speed Sensor. *GMH Engineering DRS1000 Speed Sensor*. Available at: http://www.gmheng.com/speed_sensor.php [Accessed September 11, 2020].

ETC Educational Technology Connection (HK) Ltd, Light gate / Photo gate. *ETC Educational Technology Connection Shop*. Available at: https://www.etchkshop.com/products/lightgate [Accessed September 22, 2020].

Hoel, K. ed., Radar Basics. *Radartutorial*. Available at: https://www.radartutorial.eu/01.basics/Distance-determination.en.html [Accessed September 14, 2020].

Hohne, F., 2002. Radar distance measuring device. Available at: https://patents.google.com/patent/US6373427B1/en

Jacob, D., *What is Quality 4.0?* Available at: https://blog.lnsresearch.com/quality40 [Accessed January 24, 2021].

Jedlovec, B., 2020. Introducing Statcast 2020: Hawk-Eye and Google Cloud. *Medium*. Available at: https://technology.mlblogs.com/introducing-statcast-2020-hawk-eye-and-google-cloud-a5f5c20321b8 [Accessed September 22, 2020].

Koval, L., Vaňuš, J. & Bilík, P., 2016. Distance Measuring by Ultrasonic Sensor. *IFAC-PapersOnLine*, 49(25), pp.153–158.

Keyence Corporation., 2020. Detection based on "Ultrasonic Waves" What is an Ultrasonic Sensor?. Available at: https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/ [Accessed September 10, 2020]

Le Masson, P., Weil, B. and Hatchuel, A., 2017. *Design theory*. Springer International Publishing AG.

Marr, B., 2019. What is Industry 4.0? Here's A Super Easy Explanation For Anyone. *Forbes*. Available at: https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/ [Accessed January 24, 2021].

Merriam-Webster, 2020. Dictionary by Merriam-Webster: America's most-trusted online dictionary. Available at: https://www.merriam-webster.com/ [Accessed October 23, 2020].

Nathan, A. M., The Physics of Baseball Alan M. Nathan University of Illinois. Baseball Physics. Available at: http://baseball.physics.illinois.edu/index.html [Accessed September 11, 2020].

NCSU, 2020. NC State University. Available at: https://www.ncsu.edu/ [Accessed November 1, 2020].

Nelson, R., Light Gates. Data Harvest | Light Gates - Learn more. Available at: https://store.data-harvest.co.uk/light-gate [Accessed September 22, 2020].

Ogier Electronics, 2020. How Radars Work. *Ogier Electronics*. Available at: http://ogierelectronics.com/how-radar-works.php [Accessed September 11, 2020].

OMEGA, 2017. Theory of Operation of Pressure Sensors. Available at: https://www.omega.co.uk/technical-learning/sensor-theory-of-operation.html [Accessed September 14, 2020].

Pendergast, R.L. et al., 2019. Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino. Available at: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/ [Accessed September 12, 2020].

Pereira, R., 2020. 8 Dimensions of Quality. *Gemba Academy*. Available at: https://blog.gembaacademy.com/2008/05/28/8-dimensions-of-quality/ [Accessed January 24, 2021].

Pocket Radar, Pocket Radar Smart Coach/Bluetooth App Enabled Radar Gun Allows Remote Display and Speed in Video. *Amazon.* Available at: https://www.amazon.com/Pocket-

Radar-Bluetooth-Enabled-Display/dp/B07H7SN831/ref=sr_1_1_sspa?dchild=1&keywords=speed+radar+sports&qid=1600032618&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzTjVPRDUzR0dLWFcyJmVuY3J5cHRlZElkPUEwODYxMTUyOE83SzZUVU9LOVVUJmVuY3J5cHRlZEFkSWQ9QTAzNjQyMTIzTUVCWU40Qjg3RVUmd2lkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJlY3QmZG9Ob3RMb2dDbGljaz10cnVl [Accessed September 27, 2020].

Prep Room, Light Gate. *Prep Room - The Online Science Prep Room*. Available at: https://www.preproom.org/equipment/eq.aspx?eqID=5071 [Accessed September 22, 2020].

The Complete Pocket Radar Review. *TreadAthletics*. Available at: https://treadathletics.com/pocket-radar-review/ [Accessed September 28, 2020].

Quality 4.0. *ASQ*. Available at: https://asq.org/quality-resources/quality-4-0 [Accessed January 24, 2021].

Radar Pitching Trainer RPT1000. *Radar Pitching Trainer*. Available at: http://www.radarpitchingtrainer.com/pitching-trainer.html [Accessed September 28, 2020].

Rigert, M., Why Quality 4.0 Should Be a Top Priority for Quality Managers. *MasterControl*. Available at: https://www.mastercontrol.com/gxp-lifeline/the-quality-leaders-guide-to-quality-4.0/ [Accessed January 24, 2021].

Robinson, G. & Robinson, I., 2016. Radar speed gun true velocity measurements of sports-balls in flight: application to tennis. Physica Sc*ripta*, 91(2), p.023008.

Roehr, S., Gulden, P. & Vossiek, M., 2008. Precise Distance and Velocity Measurement for Real Time Locating in Multipath Environments Using a Frequency-Modulated Continuous-Wave Secondary Radar Approach. *IEEE Transactions on Microwave Theory and Techniques*, 56(10), pp.2329–2339.

Sawicki, D.S., Introduction to Laser Radar. Cop Radar. Available at: https://copradar.com/chapts/chapt5/ch5d1.html [Accessed September 22, 2020].

Schneider, R., 2001. Distance measurement of moving objects by frequency modulated laser radar. *Optical Engineering*, 40(1), p.33.

SKLZ Bullet Ball Baseball Pitching Speed Sensor. *Amazon*. Available at: https://www.amazon.com/SKLZ-Bullet-Baseball-Accurately-Measures/dp/B002XT9A7K [Accessed September 28, 2020].

SpeedTrac X Sports Radar. *Net World Sports*. Available at: https://www.networldsports.com/speedtrac-x-sports-radar.html [Accessed September 28, 2020].

Sports Sensors, Inc, Sports Sensors Swing Speed Radar. *Amazon*. Available at: https://www.amazon.com/Sports-Sensors-Swing-Speed-Radar/dp/B00124P7SI/ref=sr_1_3?dchild=1&keywords=speed+radar+sports&qid=1600029223&sr=8-3 [Accessed September 27, 2020].

Sports Speed Sensor. *Sport speed measurement Doppler radar resource for the manufacturer, engineer, fabricator*. Available at: https://www.stalkerradar.com/stalker-speed-sensor/sport-speed-sensor/sport-speed-sensor.shtml [Accessed September 26, 2020].

Sridykhan, C.T. (2020). *A tour of Video Object Tracking — Part I: Presentation*. [online] Medium. Available at: https://medium.com/@cindy.trinh.sridykhan/a-tour-of-video-object-tracking-part-i-presentation-8a8aa9da9394 [Accessed 20 Sep. 2020].

Statcast. *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Statcast [Accessed September 28, 2020].

Stelzer, A., Jahn, M. & Scheiblhofer, S., 2008. Precise distance measurement with cooperative FMCW radar units. *2008 IEEE Radio and Wireless Symposium*.

Suh, N., 1990. *Principles of design*, New York: Oxford Univ. Press.

TestTube 101, 2015. *How Do Radars Work?*, Available at: https://www.youtube.com/watch?v=f78yIn0rFzU [Accessed September 9, 2020].

Woodford, C., 2019. How radar works: Uses of radar. *Explain that Stuff*. Available at: https://www.explainthatstuff.com/radar.html [Accessed September 10, 2020].

Zeng, Y., Xu, J. & Peng, D., 2010. Radar Velocity-Measuring System Design and Computation Algorithm Based on ARM Processor. *Traffic and Transportation Studies 2010*.

10. Appendices

Appendix A. Research on Sensors

Radars

Radar stands for Radio Detection and Ranging (Woodford, 2019). In order to work, radars need "something to generate radio waves, something to send them out into space, something to receive them, and some means of displaying information so the radar operator can quickly understand it" (Woodford, 2019). Radars work by emitting waves and waiting for them to come back to the receiver (TestTube 101, 2015).

Radio waves are "invisible electromagnetic waves that have no mass" (Ogier Electronics, 2020). As opposed to ultrasonic, or sound, waves, radio waves can travel at the speed of light, or around 300,000,000 metres per second (Ogier Electronics, 2020). The high velocity of radio waves makes them ideal for measuring distant objects quickly (Ogier Electronics, 2020).

Doppler radars measure the time taken to return and the phase the waves return with to determine the distance and direction of the object (TestTube 101, 2015). If the phase returns negative, the object is likely moving away from the radar, and if it's positive, it's likely moving towards the radar (TestTube 101, 2015). The speed of the object can be determined from the amount of shift (TestTube 101, 2015).

Radars measure velocity of an object by using the Doppler frequency shift (Ogier Electronics, 2020). The Doppler effect "causes moving objects to shift the frequency of reflected radio waves based on the speed of the object" allowing velocity of the object to be calculated (Ogier Electronics, 2020). One caveat with using the Doppler effect to measure velocity is that a Doppler is only "seen for objects moving radially, that is, directly toward or away from the radar" (Ogier Electronics, 2020). The plus side of using Doppler measurement is that it is "effective at detecting moving targets and ignoring targets that don't move" (Ogier Electronics, 2020).

The Doppler frequency, which is the increase or decrease of frequency value, will be used to determine the speed of the object using the following equation (Zeng, Xu, & Peng, 2010):

$$f_d = 2 * v_r * f_0 / C \text{ or } v_r = (f_d * C) / (2 * f_0)$$

Where f_d is the Doppler frequency, v_r is the velocity of the target object, f_0 is the launch frequency of the radar wave, and C is the speed of light.

Since this project will only deal with relatively short distances, we will not have to worry about the radar horizon, caused by the curve of the earth.

Like how radar measures speed of an object, they can measure the distance from the radar to an object by emitting radio waves and measuring how long it takes for the waves to return (Hoel). The radio wave is shot off with high pulse power in the direction of the antenna (Hoel). An oscilloscope in the radar antenna "moves synchronously with the transmitted pulse at a luminous point and leaves a trail" (Hoel). When the radio wave returns after hitting the object, another pulse is shown. The radar can then translate the distance between the two pulses into the distance between the antenna and the object (Hoel). The distance from the radar and the object is known as

the slant range, versus the ground range which only measures the horizontal distance between the two (Hoel).

The slant range can be determined using the following equation (Hoel):

$$R = C_0 * t / 2$$

Where R is the slant range antenna - aim [m], C_0 is the speed of light [m/s], and t = measured runtime [s].

Lasers

A laser distance meter works by measuring the time it takes a pulse of laser light to be reflected off a target and returned to the sender. This is known as the "time of flight" principle, and the method is known either as “time of flight” or “pulse” measurement.

Laser radars or lidars transmit pulsed infrared laser light to measure target range. The time it takes a pulse to travel at the speed of light from the lidar to the target and back is used to compute range. The change in range over time is used to calculate speed. Laser radars typically require 0.3 to 0.7 seconds sample time to get one speed reading (Sawicki, 2017).

Pulse round trip time is measured to determine range using the following equation:

$$R = C * t / 2$$

Where R is the range from the lidar to the target, t is the pulse round trip travel time, and C is the speed of light.

The lidar collects a series of range measurements during a sample time period before calculating speed. To distinguish legitimate echoes from false returns, each echo pulse width must be within the expected pulse width limits. If the pulse widths are within the correct limits, it can be used as a valid data point. An entire data set must meet variance and standard deviation constraints to be considered valid. A valid data set is usually processed with a linear least squares line algorithm to line fit the change in range for computing speed.

Early lidars discarded an entire data set if only one echo was missing or out of place. Modern lidars can process speed even if a limited number of echoes are missing or out of expected bounds (Sawicki, 2017).

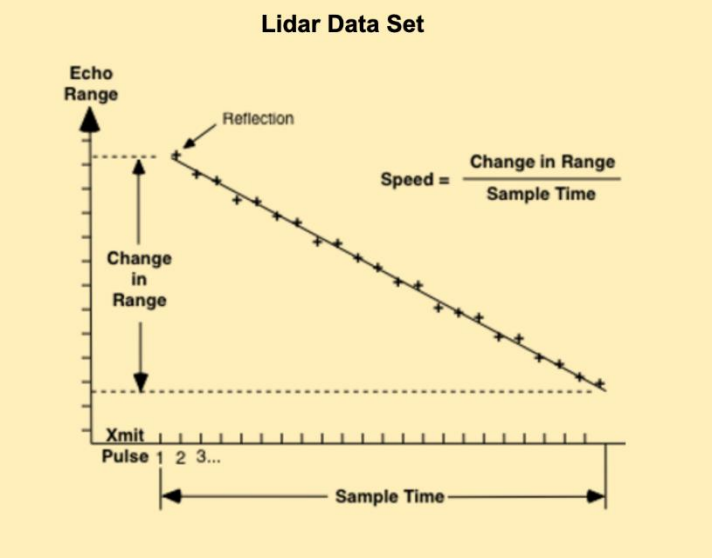

Figure 21: Lidar Data Set (Sawicki, 2017)

After a data set is determined valid, the lidar calculates velocity from the change in range divided by sample time. Decreasing ranges indicate the vehicle is moving toward the lidar, increasing ranges the vehicle is moving away. Under ideal conditions the lidar could display a speed reading after only one sample period.

Laser systems use diodes to generate a laser pulse for transmission to an optical aperture. A diode detector sensitive to IR converts the reflections to electrical signals. A computer processes the reflection's amplitude, pulse width, and time of arrival to determine if the reflection is legitimate (Sawicki, 2017).

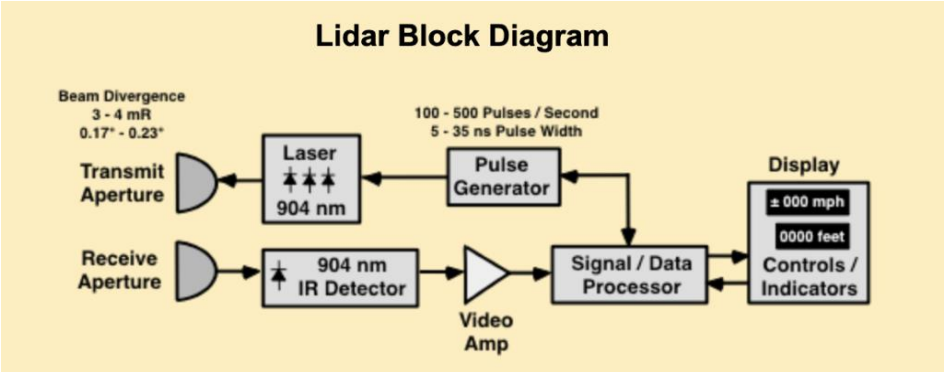

Figure 22: Lidar Block Diagram (Sawicki, 2017)

Pressure sensors

Pressure sensors, or pressure plates, are simple devices that work using a strain gauge. The strain gauge works through variable resistance when a force is applied. The measurement of the strain gauge is done with an electrical circuit, specifically a Wheatstone bridge, designed to detect the small changes in the gauge's resistance. The sensor itself has a diaphragm that deflects as the force

is applied to it, affecting the output of the Wheatstone bridge, allowing for the pressure to be measured (OMEGA, 2017).

There are a few problems with a pressure plate for the use in this project. After researching, we decided not to look further into using a pressure sensor for measuring velocity. Since the pressure plate would most likely be in a vertical position, the ball would have to be moving perpendicular to the plate and contact it in such an orientation to be able to measure the full force of the ball's impact. The problem with this is that most pitches in baseball and softball are moving left, right, or down. This motion would cause the reading of the pressure plate to be incorrect as part of the ball's force is not perpendicular to the plate. Additionally, if the ball were to miss the pressure plate, there would be no feedback available. Because one of this product's main purposes is to give feedback to people trying to improve their pitches, the pressure plate would need to be large enough to detect pitches that are inside and outside the strike zone.

There are multiple pitch types, such as curveball, that may be a strike when it reaches Homeplate, but is outside the strike zone after passing home plate.

Overall, a pressure plate would have to accommodate a large enough area that would increase the cost of a product without providing any better accuracy than our other options.

Ultrasonic Sensors

Ultrasonics are vibrational waves that are out of the range audible for humans. They are in the upper limit of frequencies, typically around 20 kilohertz. The phrase "ultrasonic" is derived from the term "ultrasound", used to depict sounds of high amplitudes (Berg 2017).

Ultrasonic sensors are a type of sensor used frequently when measuring position of objects. They are also commonly used to measure the distance where an object is from the sensor. The sensor itself has two main components, an emitter and a sensor. An ultrasonic sensor emits ultrasonic waves through air space that bounce off an object and are received from the sensor. It can then measure the distance by measuring the time it takes between emission and reception (Keyence 2020).

Ultrasonic sensors work by using the simple distance equation:

$$L = 1 / 2 * t * C$$

Where L is the distance an object is from the sensor, t is the time it takes in between emission and reception of the ultrasonic waves, and C is the speed of sound (Keyence 2020).

When measuring levels of liquids in a closed container, an ultrasonic sensor can be placed in the container to notify when that specific level is reached. This data can then be used to notify factory personnel to act depending on the scenario (Burnett 2020). Ultrasonic sensors are also commonly used to detect the presence of people at businesses and stands. Businesses commonly use ultrasonic sensors to detect when humans come close to the stands or check out desks. Because ultrasonic sensors can often range within a 10-meter length, using it for people is beneficial since humans are large objects, so ranging out at farther distances is easy for this specific type of sensor. In this application, the sensor isn't used to measure how far away people are from the sensor, but for if they are within the desired range of the sensor (Burnett 2020).

Relating to this project, ultrasonic sensors can be used for distance measurement. Ultrasonic sensors have many marine applications. Submarines and boats have many ultrasonic sensors to locate other boats and submarines. It is also used when mapping out the bottom of a sea floor (Berg 2017). Depending on the medium of the space the sensor is in, the speed of sound must be adjusted.

Photogates

A photogate, or light gate, is a digital switch-type sensor that is most commonly used in timing experiments. It usually consists of a transmitter and receiver both mounted on a frame with a gap between them (the gate). The transmitter is a small diode (LED) that emits an infrared beam of light which is then detected by the receiver, a device sensitive to infrared light. When a solid object is passing through the gap/gate, a red Off/On indicator will light up when the beam is blocked (Prep Room, Light Gate).

Figure 23: Photogates (Nelson)

A photogate usually comes with a steel mounting rod which can be screwed into the base or side of the photogate and clamped into a suitable holding frame. Photogates have increased accuracy compared to stopwatches as they can be set to react quickly when the beam is interrupted, which eliminates human reaction time errors (ETC Educational Technology Connection (HK) Ltd).

Appendix B. Candidate DPs Selection Criteria Matrix

Table 6: Speed Hardware Selection Criteria Matrix

Velocity Selection Criteria	Delta DRS1000 Non-contact Speed Sensor	Vernier photogate	Pasco smart gate	Caldwell Ballistic Precision Chronograph	TruSpeed S Laser Speed Detector	OPS241-A Doppler Short Range Radar Sensor
Type	Doppler Radar	Photogate	Photogate	Ballistic Chronograph	Laser Sensor	Doppler Radar
Arduino or Raspberry Pi Compatible	N/A	Yes	N/A	N/A	No	Yes
Precision (close to real value)	±0.34% at 1mph, Error increased 0.0023% for every 1 mph increase in speed	N/A	N/A	0.25%	± 2 km/h (± 1 mph)	±0.5%
Cost	N/A	$49.00	$55	$100.95	N/A	$169.00
Weather/Temperature resistance	Yes, has resistant enclosure, 0 to 140° F	N/A	N/A	N/A	-30° C to +60° C, Water-resistant; NEMA 4 and IP 55	-40° C to +85° C,
Number needed to fulfill design	multiple sensors required at different locations for best result	1	1	1	multiple sensors required at different locations for best result	1
Detection range	Up to 300 meters	Within Photogate	Within Photogate	Within Chronograph	Up to 2000 feet	3-82 feet 78° beam width
Speed range	0.5 to 300 mph	N/A	N/A	5 - 9,999 fps	0-200 mph	0-138 mph
Notes		used mostly for experiments	used mostly for experiments	used for measuring gunshot velocity, 1 star rating, providing a readout of velocity in feet per second or meters per second on a large LCD screen	speed gun used for traffic purposes	open-source code available

Table 7: Position Hardware Selection Criteria Matrix

Position Selection Criteria	HC-SR04 Ultrasonic Sensor	TeraRanger One ToF Rangefinder Type B	LIDAR-Lite 3 Laser Rangefinder	SEN-09376 Pressure Pad	FSR X 406	Raspberry Pi Camera Module V2
Type	Ultrasonic	Optical	LIDAR Radar	Pressure	Pressure	Camera
Arduino or Raspberry Pi Compatible	Yes	Yes	Yes	Yes	Yes	Yes
Precision (close to real value)		up to ± 4cm in precision mode	+/- 2.5cm	+/- 2%	+/- 2%	N/A
Cost	$2-4	$69	$129.99	$12	$5	$25
Weather/Temperature resistance	No, will need protection	N/A	-20 to 60° C	-30 to 70 C	Yes	N/A
Number needed to fulfill design	4-pin module, multiple sensors required	at least 2 (1 for x, 1 for y)	at least 2 (1 for x, 1 for y)	An array of them	an array	At least 1
Distance range	0.02m-4m	Up to 14m (At least 5 to 6m in sunlight)	0-40 meters	N/A	N/A	N/A
Field of view	15 degree cone	3°	Optical aperture: 12.5 mm	43.69mm x 43.69 mm	39.6mm x 39.6mm	Horizontal 62.2° Vertical 48.8°
Notes	a 2D plane can be constructed with multiple sensors	based on infrared Time-of-Flight (ToF) technology, supposedly "much faster than ultrasound and far smaller and lighter than laser-based systems"		Very small	Interlink Electronics can custom make pressure pads to any size. They can also create sensors that can withstand up to 50N	

Table 8: Visual Feedback Hardware Selection Criteria Matrix

Visual Feedback Selection Criteria	Adafruit LED Matrix	Wesiri LED Matrix	App on Phone	Hook up to laptop
Cost	$24.95	$23.99	$0	$0

Screen size	16x32 LEDs	16x16 LEDs	Depends on phone	Depends on computer
Product size	7.6" x 3.8" x 0.5"	6.25" x 6.25" x 0.07"	N/A	N/A
Arduino or Raspberry Pi Compatible	Yes	Yes	N/A	Yes
Notes	Lighting with different color LEDs	Lighting with different color LEDs	Requires a lot of coding	

Table 9: Audio Feedback Hardware Selection Criteria Matrix

Audio Feedback Selection Criteria	InduSKI speakers	Gikfun Speakers	App on Phone	Hook up to laptop
Cost	$7.89	$9.99	$0	$0
Impedence	8 Ohms	4 Ohms	Depends on phone	Depends on computer
Speaker Maximum Output Power	3 Watt	3 Watts	Depends on phone	Depends on computer
Product size		2 in diameter	N/A	N/A
Arduino or Raspberry Pi Compatible	Yes	Yes	N/A	Yes
Notes			Requires a lot of coding	

Appendix C. Additional Information on Axiomatic Design and C-K Theory

The process of Axiomatic Design was created by Nam P. Suh in the 1970s. It is based on the theory that all good designs have two axioms in common: The Independence Axiom and the Information Axiom. The Independence Axiom states that the independence of the functional requirements (FRs) must be maintained or maximized. This means that each design parameter (DP) should be linked to only one FR. The Information Axiom states that the information content of the design must be minimized. The Information Axiom provides a metric of the probability that a DP will fulfill its corresponding FR. This means simplifying the instructions and intent behind each DP. A design is considered Axiomatic if it complies with these axioms.

There are five domains in Axiomatic design: customer, functional, physical, process, and constraints. The customer domain defines the customer needs (CNs) and what adds value to the design. The functional domain defined the FRs and what the design does. The physical domain defines the DPs and what the design looks like. The process domain defined the process variables and how the design can be made. The constraints domain defines the constraints (CONs) or what needs to be avoided/limited.

In the Functional and Physical domains, there are hierarchies, branches, and levels. The hierarchies allow for each specific FR to be linked to a specific DP (e.g., FR1.1 and FR1.2 that relate to DP1.1 and DP1.2 respectively).

The branches organize each specific FR and DP under a parent (e.g., FR1.1 and FR1.2 are children to FR1). The levels allow the FRs and DPs to start with a general idea and become more specific as the level gets higher (e.g., FR1 is level 1 while FR1.1 and FR1.2 are in level 2).

Axiomatic design is done by building the decomposition then physically integrating the design. To decompose the design, one must determine CNs, FRs that fill those needs, DPs that fulfill the FRs, and process variables that allow for the DPs to be physically integrated.

Axiomatic Design is based on the theory that all good designs have two axioms in common: The Independence Axiom and the Information Axiom. The Independence Axiom states that the independence of the FRs must be maintained or maximized. The Information Axiom provides a metric of the probability that a DP will fulfill its corresponding FR.

In C-K (Concept and Knowledge) Theory, the C-space is formed from various ideas that are all connected to the initial concept in a tree-like structure (Le Masson, Weil, Hatchuel, 2017). The K-space “manage[s] the knowledge available at the beginning of exploration and reveal knowledge elements that might be hidden” (C-K Theory).

Appendix D. Full Axiomatic Design and Design Matrix for The Product

Table 10: Functional Requirements

FR 0:	**Develop product that determines speed and position, and stops balls**
FR 1	**Determine speed of ball**
FR 1.1	Measure distance that ball travels in the x direction OR measure the vector that ball travels in the x, y, and z directions
FR 1.2	Measure time
FR 1.3	Transfer data to program for calculation
FR 1.4	Calculate speed
FR 2	**Measure position of baseball on X-Z plane**
FR 2.1	Allow for measurement on an X-Z plane
FR 2.2	Measure coordinates on X-Z plane
FR 2.3	Transfer data to program for calculation
FR 2.4	Determine coordinates of pitch
FR 2.5	Determine strike status of pitch
FR 3	**Deliver feedback to the user**
FR 3.1	Analyze data from measurement
FR 3.1.1	Translate the data from sensors to serial monitor
FR 3.1.2	Translate the data from serial monitor to user-friendly feedback
FR 3.2	Announce feedback
FR 3.2.1	Show feedback visually
FR 3.2.2	Announce data audibly
FR 4	**Stop ball**
FR 4.1	Take energy from ball
FR 4.2	Transfer balls to bucket
FR 5*	**Determine spin rate of ball**
FR 5.1	Determine point on ball to analyze
FR 5.2	Measure rotation of point
FR 5.3	Transfer data to program for calculation
FR 5.4	Calculate spin rate
FR 6	**Contain sensors**
FR 6.1	Provide power to sensors
FR 6.2	Protects sensitive equipment
FR 6.2.1	Protects sensitive equipment from weather
FR 6.2.2	Protects sensitive equipment from movement within containment
FR 6.3	Minimize injuries that could occur
FR 6.3.1	Prevent pieces small enough for child to eat
FR 6.3.2	Indicate location of container
FR 6.3.3	Cushion potential fall

FR 6.3.4	Prevent flammability
FR 6.4	Allow for turning on an off
FR 6.5	Indicates whether sensors are on or off
FR 6.6	Allows for data to be transmitted to feedback system
FR 6.7	Allow for adjustability in angles and height

Table 11: Design Parameters

DP 0	**Smart backstop that measures speed and position of ball and gives feedback to user**
DP 1	**System to measure speed**
DP 1.1	Sensor that measures range traveled by object OR sensor that measures range traveled by object in multiple directions
DP 1.2	Sensor with timer
DP 1.3	Wires to transfer sensor data to computer
DP 1.4	Program to calculate speed
DP 2	**System to measure position**
DP 2.1	Projected coordinate array on X-Z plane
DP 2.2	Camera vision sensor
DP 2.3	Wires to transfer sensor data to computer
DP 2.4	Program that interprets coordinates of pitch on X-Z plane
DP 2.5	Program that takes position coordinates and compares them to a strike zone coordinate zone
DP 3	**Vision and audio system that shares position and speed of ball**
DP 3.1	Program
DP 3.1.1	Serial monitor
DP 3.1.2	Code that translates data from serial monitor to feedback system
DP 3.2	Laptop graphics display
DP 3.2.1	Laptop screen
DP 3.2.2	Laptop speaker
DP 4	**Backstop**
DP 4.1	Suspended net with frame
DP 4.2	Gutter system that is 1 ft in y direction and covers bottom of backstop
DP 5*	**System to measure spin rate**
DP 5.1	Sensor that pinpoints place to analyze
DP 5.2	Sensor that measures rotation
DP 5.3	Wires to transfer sensor data to program
DP 5.4	Program to calculate spin rate
DP 6	**Container that allows for sensors to still work**
DP 6.1	Power supply
DP 6.2	Protective aspects
DP 6.2.1	Waterproof surfaces and joints

DP 6.2.2	gimble OR Velcro OR exact measurements of sensors built into container OR slots that sensors can slide into OR strap sensors down OR glue sensors down
DP 6.3	Rounded housing unit
DP 6.3.1	Big pieces and use of latches attached to component
DP 6.3.2	Bright colored paint
DP 6.3.3	Cushioned cover for dome (non-clear parts)
DP 6.3.4	Inflammable materials
DP 6.4	On/off switch
DP 6.5	LED light to indicate on or off
DP 6.6	Wires that connect to feedback system
DP 6.7	Clip that can go onto backstop and can adjust the angles

Table 12: Metrics and Equations on Axiomatic Design

Number	Metrics	Relation: Design Equations, Graphs, Tables
1	Measureability of speed [m_s]	$m_s = (q_1 * (c_d + c_t + c_s)) + (q_1 * (p_d + p_t + p_s))$
1.1	Accuracy [c_d] and precision [p_d] of distance	c_d = 0.9<=our value/radar value<=1.1 and p_d = stand.dev(measurements)
1.2	Accuracy [c_t] and precision [p_t] of time	c_t = 0.9<=our value/radar value<=1.1 and p_t = stand.dev(measurements)
1.3	Quality of data [q_1] (% correct) of speed position data to program	q_1 = 100%
1.4	Accuracy [c_s] and precision [p_s] of speed measurement	c_s = 0.9<=our value/radar value<=1.1 and p_s = stand.dev(measurements)
2	Measureability of position [m_p]	$m_p = d_{xz} * m_c * q_2 * C_p * C_s$
2.1	Range covered in X and Z position (d_{xz})	d_{xz} = 5 ft x 5 ft
2.2	Measureability of coordinates [m_c]	m_c = 100%
2.3	Quality of data [q_2] (% correct) from calculation to feedback	q_2 = 100%
2.4	Correctness of position reading (C_p)	C_p = 100%
2.5	Correctness of strike reading (C_s)	C_s = 100%
3	Quality of data [q_3] (% correct) of feedback	q_3 = 100%
3.1	Quality of data [q_4] (% correct) from calculation to feedback	q_4 = 100%
3.1.1	Quality of data [q_5] (% correct) from sensors to serial monitor	q_5 = 100%
3.1.2	Quality of data [q_6] (% correct) from serial monitor to user friendly feedback	q_6 = 100%
3.2	Comprehension level [cl] of feedback	cl = 0 if r =0 and dB < 60, = 1 if r =1 and dB > 60
3.2.1	Readability [r] of visual feedback	r = 0 if did not read numbers correctly, = 1 if correct
3.2.2	Decibels [dB] of audible feedback	dB > 60 decibels
4	Speed [v] (ft/second)	v = 0

4.1	Distance ball bounces back [D] from	0 in <= D <= 1 ft
4.2	Percentage of times that ball ends up in gutter [G]	G = 100%
5*	Measureability [m_r] of rotation	$m_r = c_r + p_r$
5.1	Usability [u]	$u = c_r + p_r$
5.2	Accuracy [c] and precision [p]	c = 0.9<=our value/radar value<=1.1 and p = stand.dev(measurements)
5.3	Quality of data [q_7] (% correct)	q_7 = 100%
5.4	Accuracy [c_r] and precision [p_r]	c_r = 0.9<=our value/true value<=1.1 and p_r = stand.dev(measurements)
6		$W = o * t_s * e * q_9 * LOI * (P) / I * m_{vc}$
6.1	Power [P] (Watts)	P = power needed by sensors
6.2	Protection of sensors [t_s] (% that sensors are protected)	$t_s = (t_w + t_m)^{-1}$
6.2.1	Protection of sensors from weather [t_w] (% of weather that gets in)	t_w = 0
6.2.2	Protection of sensors from movement [t_m] (% of damage from movement)	t_m = 0
6.3	Injury occurrence [I] (% of people who get hurt)	$I = I_e + I_t + I_f$
6.3.1	Eating injury occurrence [Ie]	I_e = % of people who get hurt by eating piece = 0
6.3.2	Tripping injury occurrence [I_t]	I_t = % of people who get hurt by tripping on it = 0
6.3.3	Fall injury occurrence [I_f]	I_f = % of people who get hurt by falling on it = 0
6.3.4	Limiting oxygen index (LOI)	LOI = flame resistance > 21%
6.4	Ease of use [e]	e = rating of how easy it is to use based on testing = 100%
6.5	On/Off [o] (binary)	o = 0 for off, 1 for on
6.6	Quality of data [q_9] (% correct)	q_9 = 100%
6.7	Moveability of container [m_{vc}]	m_{vc} = 0

Design Matrix

DPs	1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4	2.5	3.1.1	3.1.2	3.2.1	3.2.2	4.1	4.2	5.1	5.2	5.3	5.4	6.1	6.2.1	6.2.2	6.3.1	6.3.2	6.3.3	6.3.4	6.4	6.5	6.6	6.7
FR 1.1	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 1.2	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 1.3	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 1.4	0	0	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 2.1	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 2.2	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 2.3	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 2.4	0	0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 2.5	0	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 3.1.1	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 3.1.2	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 3.2.1	0	0	0	0	0	0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 3.2.2	0	0	0	0	0	0	0	0	0	0	X	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 4.1	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 4.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 5.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 5.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0
FR 5.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0
FR 5.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	0	0	0
FR 6.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0
FR 6.2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0
FR 6.2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0
FR 6.3.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0
FR 6.3.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0
FR 6.3.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0
FR 6.3.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0
FR 6.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0
FR 6.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	X	0	0
FR 6.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	X	0
FR 6.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X

Figure 24: Design Matrix

Appendix E. Full Code

Open-Source Code Used as Reference

Because of the inexperience of the team in computer programming, the use of open-source code was often used. The code used as reference is shown below (Cuni & Duran, 2016).

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# USAGE: You need to specify a filter and "only one" image source
#
# (python) range-detector --filter RGB --image /path/to/image.png
# or
# (python) range-detector --filter HSV --webcam

import cv2
import argparse
from operator import xor


def callback(value):
    pass


def setup_trackbars(range_filter):
    cv2.namedWindow("Trackbars", 0)

    for i in ["MIN", "MAX"]:
        v = 0 if i == "MIN" else 255

        for j in range_filter:
            cv2.createTrackbar("%s_%s" % (j, i), "Trackbars", v, 255, callback)


def get_arguments():
    ap = argparse.ArgumentParser()
    ap.add_argument('-f', '--filter', required=True,
                    help='Range filter. RGB or HSV')
    ap.add_argument('-i', '--image', required=False,
                    help='Path to the image')
    ap.add_argument('-w', '--webcam', required=False,
                    help='Use webcam', action='store_true')
    ap.add_argument('-p', '--preview', required=False,
                    help='Show a preview of the image after applying the mask',
                    action='store_true')
    args = vars(ap.parse_args())

    if not xor(bool(args['image']), bool(args['webcam'])):
        ap.error("Please specify only one image source")
```

```
44
45      if not args['filter'].upper() in ['RGB', 'HSV']:
46          ap.error("Please speciy a correct filter.")
47
48      return args
49
50
51  def get_trackbar_values(range_filter):
52      values = []
53
54      for i in ["MIN", "MAX"]:
55          for j in range_filter:
56              v = cv2.getTrackbarPos("%s_%s" % (j, i), "Trackbars")
57              values.append(v)
58
59      return values
60
61
62  def main():
63      args = get_arguments()
64
65      range_filter = args['filter'].upper()
66
67      if args['image']:
68          image = cv2.imread(args['image'])
69
70          if range_filter == 'RGB':
71              frame_to_thresh = image.copy()
72          else:
73              frame_to_thresh = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
74      else:
75          camera = cv2.VideoCapture(0)
76
77      setup_trackbars(range_filter)
78
79      while True:
80          if args['webcam']:
81              ret, image = camera.read()
82
83              if not ret:
84                  break
```

```
85
86              if range_filter == 'RGB':
87                  frame_to_thresh = image.copy()
88              else:
89                  frame_to_thresh = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
90
91          v1_min, v2_min, v3_min, v1_max, v2_max, v3_max = get_trackbar_values(range_filter)
92
93          thresh = cv2.inRange(frame_to_thresh, (v1_min, v2_min, v3_min), (v1_max, v2_max, v3_max))
94
95          if args['preview']:
96              preview = cv2.bitwise_and(image, image, mask=thresh)
97              cv2.imshow("Preview", preview)
98          else:
99              cv2.imshow("Original", image)
100             cv2.imshow("Thresh", thresh)
101
102         if cv2.waitKey(1) & 0xFF is ord('q'):
103             break
104
105
106 if __name__ == '__main__':
107     main()
```

Figure 25: Reference Code for Environment (Cuni & Duran, 2016)

Dependencies

```
import tkinter as tk

from PIL import ImageTk, Image
from ops241.radar import OPS241Radar
import argparse
from collections import deque
from imutils.video import VideoStream
import numpy as np
import cv2
import imutils
import time
from datetime import datetime, timedelta
from moviepy.video.io.ffmpeg_tools import ffmpeg_extract_subclip
from multiprocessing import Process
```

Figure 26: Dependencies Code

Radar

```
# Get the speed data from radar sensor
def read_radar():
    with OPS241Radar() as radar:
        while True:
            data = radar.read()
            print("Data: " + data)
            if (data != 0.0) :
                file1 = open("speed_readings.txt","a")
                file1.write(data + " "+ str(datetime.now()) +"\n")
                file1.close()
                if float(data) > minimum_speed:
                    continue_reading = False
```

Figure 27: Radar Code

Camera Vision

```
def film_clip():
    cap = cv2.VideoCapture(0)
    vid_cod = cv2.VideoWriter_fourcc(*'XVID')
    output = cv2.VideoWriter("recording.mp4", vid_cod, 20.0, (640,480))
    while(True):
        ret,frame = vid_capture.read()
        cv2.imshow("My cam video", frame)
        output.write(frame)
        if cv2.waitKey(1) &0XFF == ord('x'):
            break
    vid_capture.release()
    output.release()
    cv2.destroyAllWindows()
```

Figure 28: Camera Vision Code for Recording the Video

```
def trim_video(timestamp):
    start_time = timestamp - 1
    end_time = timestamp + 1
    ffmpeg_extract_subclip("recording.mp4", start_time, end_time, targetname="trimmed.avi")
```

Figure 29: Camera Vision Code for Trimming Video

```
def locate_ball(pic):
    bg = cv2.imread("img/black.jpeg")
    bg = imutils.resize(bg, width=640, height=480)
    # Define upper and lower bounds of our ball color (neon yellow) in the HSV color space
    greenLower = (16, 129, 113)
    greenUpper = (255, 255, 255)
    pts = deque(maxlen=64)
    # allow the camera or video file to warm up
    time.sleep(2.0)
    center = None
    biggest_radius = 0
    # keep looping
    gui_pic = False
    while True:
        frame = pic
        cv2.imwrite("te.jpg", frame)
        frame = imutils.resize(frame, width=600)
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Figure 30: Camera Vision Code for Finding the Ball

```
def determine_result(center, radius):
    top_center = [center[0], center[1] - radius]
    bottom_center = [center[0], center[1] + radius]
    right_center = [center[0] + radius, center[1]]
    left_center = [center[0] - radius, center[1]]
    points = [center, top_center, bottom_center, right_center, left_center]
    for point in points:
        if (bbx_min < point[0] < bbx_max) & (bby_min < point[1] < bby_max):
            return True
    return False
```

Figure 31: Camera Vision Code to Determine if the Strike Status

Gui Code

```
def __init__(self, master=None):
    super().__init__(master)
    self.master = master

    self.result_text = tk.StringVar()
    self.result_text.set(results[strike])
    self.speed_text = tk.StringVar()
    self.speed_text.set(str(radar_data[0]) + " " + units[unit])

    strikebox_img = Image.open("img_final.jpg")
    strikebox_img = strikebox_img.resize((640, 480), Image.ANTIALIAS)
    strikebox_img_obj = ImageTk.PhotoImage(strikebox_img)
    tk.Label(master, image = strikebox_img_obj).grid(row = 0, column = 2, columnspan = 2, rowspan = 2, padx = 5


    self.result_label = tk.Label(master, text = self.result_text.get(), font=("Arial", 30))
    self.result_label.grid(row = 0, column = 5, columnspan = 1, rowspan = 1, padx = 5, pady = 3)
    self.speed_label = tk.Label(master, text = self.speed_text.get(), font=("Arial", 30))
    self.speed_label.grid(row = 1, column = 5, columnspan = 1, rowspan = 1, padx = 5, pady = 3)
```

Figure 32: Code for GUI

Appendix F. Camera Vision Failures

1. Setting up the virtual environment was the biggest hurdle in completing this project. Many different methods were tried in order to set up an ideal environment to continue working.
2. Object detection methods were changed when it was realized how long it took the Raspberry Pi to analyze each picture. It would have taken months of nonstop running the code to teach it effectively.
3. The camera we used is very cheap, which made it difficult to pick up white baseballs. Therefore, a yellow ball is used in testing.
4. The code failed to be able to live analyze feed. Even if it was able to analyze live feed, there was no chance the Raspberry Pi could handle analyzing that amount of data. This is assumed since it takes almost a minute to analyze a 2 second video.
5. Initializing the software from the GUI was also not completed. Ideally, there is a start button to run the program. Instead, a couple command codes are necessary to run the virtual environment.

Appendix G. Full Test Results

Below are all the data recorded during radar accuracy testing.

Subgroup 1	Radar Speed	Prototype Speed	Difference
Attempt 1	39	38.02	-0.98
Attempt 2	35	35.03	0.03
Attempt 3	31	30.68	-0.32
Attempt 4	38	39.92	1.92
Attempt 5	31	28.78	-2.22
		avg:	-0.314
		range:	4.14

Subgroup 2	Radar Speed	Prototype Speed	Difference
Attempt 1	48	46.98	-1.02
Attempt 2	49	47.52	-1.48
Attempt 3	52	50.51	-1.49
Attempt 4	50	47.52	-2.48
Attempt 5	54	48.88	-5.12
		avg:	-2.318
		range:	4.1

Subgroup 3	Radar Speed	Prototype Speed	Difference
Attempt 1	47	47.52	0.52
Attempt 2	49	46.71	-2.29
Attempt 3	43	43.99	0.99
Attempt 4	46	43.72	-2.28
Attempt 5	46	44.53	-1.47
		avg:	-0.906
		range:	3.28

Subgroup 4	Radar Speed	Prototype Speed	Difference
Attempt 1	43	39.37	-3.63
Attempt 2	48	44.26	-3.74
Attempt 3	42	41.55	-0.45
Attempt 4	46	43.45	-2.55
Attempt 5	49	43.45	-5.55
		avg:	-3.184
		range:	5.1

Subgroup 5	Radar Speed	Prototype Speed	Difference
Attempt 1	48	45.35	-2.65
Attempt 2	50	46.16	-3.84
Attempt 3	49	46.16	-2.84
Attempt 4	46	43.45	-2.55

Attempt 5	49	44.81	-4.19
		avg:	-3.214
		range:	1.64

Subgroup 6	**Radar Speed**	**Prototype Speed**	**Difference**
Attempt 1	45	46.71	1.71
Attempt 2	51	50.78	-0.22
Attempt 3	53	51.05	-1.95
Attempt 4	52	48.34	-3.66
Attempt 5	53	51.87	-1.13
		avg:	-1.05
		range:	5.37

Appendix H. Reflections

Bailey Berg

I think if I was to continue this project, I would need to do more research on coding sensors. I have a basic knowledge of a couple of coding languages, but this project introduced me to a lot of new features that I am not familiar with. I think it would be very helpful for me to gain more knowledge about different coding languages and different uses. Additionally, I think my knowledge of Axiomatic Design and how to implement it could be furthered. This was a great introductory project to use it, but I found myself having problems with the thought process of AD as it was very new to me. Developing and using the metrics and equations was especially difficult to me and I need more practice in them.

When it comes to working on a team during COVID-19, I found it very difficult. In previous projects I have been in, I find that I am most productive and most connected to my team when we can meet and work in person often. This project was difficult because we only met in person a handful of times, and it was hard to feel connected to the group. When meeting on Zoom, it is often a case of making sure everyone is up to date on where we are and what everyone needs to do, then separating and doing our parts and texting to keep each other updated between meetings. Nobody wants to hang out silently on Zoom and do work together if they don't need to.

Throughout this project, I found myself taking a project management role. Although I have previous experience managing smaller projects, I learned a lot about myself and how to work in a team. For one, the scheduling of the project itself was a much harder task than I had realized. Not only were there a lot of moving parts but trying to schedule things that had not even been conceptualized was difficult. Our Gantt chart changed almost weekly at certain points in the project because we had learned new things about what we were doing, our strategy changed, or something got held up. Another thing I noticed was that trying to manage and schedule a project that you have relatively little experience in is hard. I came into this project with little to no knowledge of sensors, coding, and AD. So, when I tried to guide others and help on certain things, I felt like I was not as guiding or helpful as I could be. As the project progressed and I gained more knowledge, it did become easier to do these things, but I would not say I'm an expert in many of the things we used for this project.

Overall, I learned a lot about myself, how to work in a group, and how to manage larger projects throughout this MQP project.