**D**igital

**A**ccuracy

**R**eal-time

**T**racking

and

**S**coring

EN.525.743 – Embedded Systems Development Laboratory

Kevin Parlak

Contents

[Figures 4](#_Toc150947371)

[Tables 5](#_Toc150947372)

[Equations 5](#_Toc150947373)

[Project Description 6](#_Toc150947374)

[Problem 6](#_Toc150947375)

[Solution 6](#_Toc150947376)

[Capabilities 6](#_Toc150947377)

[Limitations 7](#_Toc150947378)

[Functional Description 8](#_Toc150947379)

[Imaging System 8](#_Toc150947380)

[Configuration 8](#_Toc150947381)

[Training 9](#_Toc150947382)

[Recognition 10](#_Toc150947383)

[Translation 10](#_Toc150947384)

[Mapping 12](#_Toc150947385)

[State Machine 16](#_Toc150947386)

[Scoring System 18](#_Toc150947387)

[Database 18](#_Toc150947388)

[User Interface 19](#_Toc150947389)

[Gameplay 23](#_Toc150947390)

[State Machine 27](#_Toc150947391)

[Interface Description 32](#_Toc150947392)

[Imaging System 33](#_Toc150947393)

[Scoring System 33](#_Toc150947394)

[Material Requirements 34](#_Toc150947395)

[Resource Requirements 35](#_Toc150947396)

[Development Plan and Schedule 37](#_Toc150947397)

[Milestones and Schedule 37](#_Toc150947398)

[Risk 38](#_Toc150947399)

[Assembly 40](#_Toc150947400)

[Performance 41](#_Toc150947401)

[References 42](#_Toc150947402)

[Appendix 43](#_Toc150947403)

[Acronyms 43](#_Toc150947404)

[Source Code 43](#_Toc150947405)

[Imaging System State Machine 43](#_Toc150947406)

[Scoring System State Machine 43](#_Toc150947407)

[‘501’ Game 43](#_Toc150947408)

[‘Around the World’ Game 43](#_Toc150947409)

# Figures

[Figure 1: Home Dartboard 6](#_Toc150947410)

[Figure 2: Functional Block Diagram 8](#_Toc150947411)

[Figure 3: Dartboard Setup 8](#_Toc150947412)

[Figure 4: Camera View 9](#_Toc150947413)

[Figure 5: Training Logic 9](#_Toc150947414)

[Figure 6: Object Detection Box 10](#_Toc150947415)

[Figure 7: Dartboard Source Features 10](#_Toc150947416)

[Figure 8: Dartboard Destination Features 11](#_Toc150947417)

[Figure 9: Dartboard Polar Coordinates 13](#_Toc150947418)

[Figure 10: Standard Dartboard Distances 14](#_Toc150947419)

[Figure 11: Standard Dartboard Angles 15](#_Toc150947420)

[Figure 12: Imaging System State Machine Diagram 16](#_Toc150947421)

[Figure 13: Imaging System Startup Logic 16](#_Toc150947422)

[Figure 14: Imaging System IDLE START Logic 17](#_Toc150947423)

[Figure 15: Imaging System WAIT THROW Logic 17](#_Toc150947424)

[Figure 16: Imaging System FIND DART Logic 17](#_Toc150947425)

[Figure 17: Imaging System MAP DART Logic 18](#_Toc150947426)

[Figure 18: Scoring System Database Entity Relationship 19](#_Toc150947427)

[Figure 19: Startup Display 20](#_Toc150947428)

[Figure 20: Profile Display 20](#_Toc150947429)

[Figure 21: Create Profile Display 21](#_Toc150947430)

[Figure 22: Select Game Display 21](#_Toc150947431)

[Figure 23: Select Players Display 22](#_Toc150947432)

[Figure 24: Load Profile Display 22](#_Toc150947433)

[Figure 25: Scoreboard Display 23](#_Toc150947434)

[Figure 26: "501" Game Logic 24](#_Toc150947435)

[Figure 27: "501" Scoreboard 25](#_Toc150947436)

[Figure 28: “Around the World” Game Logic 26](#_Toc150947437)

[Figure 29: "Around the World" Scoreboard 27](#_Toc150947438)

[Figure 30: Scoring System State Machine Diagram 27](#_Toc150947439)

[Figure 31: Scoring System IDLE START Logic 28](#_Toc150947440)

[Figure 32: Scoring System START Logic 28](#_Toc150947441)

[Figure 33: Scoring System UPLOAD PROFILE Logic 28](#_Toc150947442)

[Figure 34: Scoring System WAIT PLAY Logic 29](#_Toc150947443)

[Figure 35: Scoring System SELECT GAME Logic 29](#_Toc150947444)

[Figure 36: Scoring System LOAD PROFILE Logic 29](#_Toc150947445)

[Figure 37: Scoring System IDLE TURN Logic 30](#_Toc150947446)

[Figure 38: Scoring System NEW DART Logic 30](#_Toc150947447)

[Figure 39: Scoring System WAIT DART Logic 30](#_Toc150947448)

[Figure 40: Scoring System UPDATE GAME Logic 31](#_Toc150947449)

[Figure 41: Scoring System FINISH GAME Logic 31](#_Toc150947450)

[Figure 42: Interface Diagram 32](#_Toc150947451)

[Figure 43: Network Diagram 32](#_Toc150947452)

[Figure 44: Roadmap Example 37](#_Toc150947453)

# Tables

[Table 1: Dartboard Ring Mapping 14](#_Toc150947454)

[Table 2: Dartboard Number Mapping 15](#_Toc150947455)

[Table 3: Hosted Games 23](#_Toc150947456)

[Table 4: "501" Unit Tests 24](#_Toc150947457)

[Table 5: “Around the World” Mapping 26](#_Toc150947458)

[Table 6: "Around the World" Unit Tests 27](#_Toc150947459)

[Table 7: Communication Matrix 32](#_Toc150947460)

[Table 8: LOCATION Message 32](#_Toc150947461)

[Table 9: Bill of Materials 34](#_Toc150947462)

[Table 10: VS Code Extensions 35](#_Toc150947463)

[Table 11: Python Libraries 35](#_Toc150947464)

[Table 12: Third-Party Applications 35](#_Toc150947465)

[Table 13: Milestone Schedule 38](#_Toc150947466)

[Table 14: Risk Impact/Probability 38](#_Toc150947467)

# Equations

[Equation 1: Top Source Equation 11](#_Toc150947468)

[Equation 2: Center Source Equation 11](#_Toc150947469)

[Equation 3: Bottom Source Equation 11](#_Toc150947470)

[Equation 4: Left Source Equation 11](#_Toc150947471)

[Equation 5: Right Source Equation 11](#_Toc150947472)

[Equation 6: Top Destination Equation 12](#_Toc150947473)

[Equation 7: Center Destination Equation 12](#_Toc150947474)

[Equation 8: Bottom Destination Equation 12](#_Toc150947475)

[Equation 9: Left Destination Equation 12](#_Toc150947476)

[Equation 10: Right Destination Equation 12](#_Toc150947477)

[Equation 11: Source Array Equation 12](#_Toc150947478)

[Equation 12: Destination Array Equation 12](#_Toc150947479)

[Equation 13: Homographic Matrix Equation 12](#_Toc150947480)

[Equation 14: Translation Equation 12](#_Toc150947481)

[Equation 15: Radius Equation 13](#_Toc150947482)

[Equation 16: Quadrant 1 Angle Equation 13](#_Toc150947483)

[Equation 17: Quadrant 2 Angle Equation 13](#_Toc150947484)

[Equation 18: Quadrant 3 Angle Equation 13](#_Toc150947485)

[Equation 19: Quadrant 4 Angle Equation 13](#_Toc150947486)

# Project Description

## Problem

Today, modern dartboards have built-in ways to automatically calculate user scores as players take turns throwing. However, traditional steel-tipped darts use cork dartboards which do not have a means of keeping score. This means players must know how a specific game is scored and manually keep score as the game progresses.



Figure : Home Dartboard

## Solution

Dartboards are unique in that they are perfect circles and utilize different colored sections to indicate score location. This makes the game of darts a great candidate for using a computer vision solution to detect dart location. DARTS is a system designed to identify darts and score games automatically. This system implements core capabilities from a computer vision/machine learning perspective that can be improved with additional data. Capabilities and limitations are defined that describe what will and will not be possible with the system designed.

## Capabilities

The system has the following capabilities to compliment core functionality:

* Imaging System
  + Capable of detecting number, ring, radius, and angle of hit
    - Number and ring used for scoring purposes
    - Radius and angle of hit user for statistical purposes
* Scoring System
  + Capable of holding up to 10 players information
  + Capable of updating displays after each dart is thrown and identified
  + Capable of pulling statistics
    - Win %
    - Hit % by number
    - Hit % by double ring, triple ring, bull, bullseye

## Limitations

System capabilities will be limited based on development time and resources available. The system has the following limitations:

* Imaging System
  + Limited to tracking only a single dart per turn
    - Players must retrieve dart before throwing subsequent dart
* Scoring System
  + Limited to database access on home network
  + Limited to two games
    - Framework available for future games
  + Limited to two-players per game
  + Limited to play only on user interface

# Functional Description

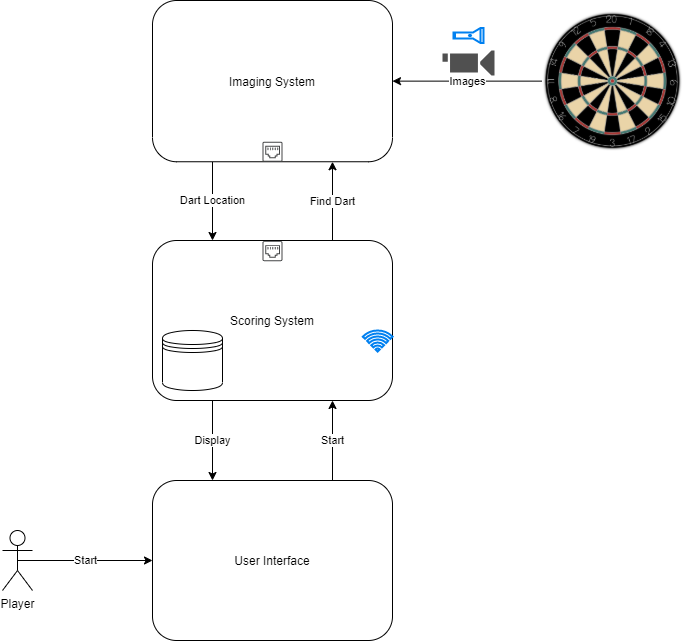


Figure : Functional Block Diagram

Figure 2 illustrates how DARTS works. The system functions as follows:

1. Player(s) interact with the user interface to select their profile and a game of choice
2. The Scoring System will load profiles and wait for player input
3. Before throwing, player will select their profile
4. The Scoring System will indicate to the Imaging System that a dart is incoming
5. The Imaging System will start looking at the dartboard, find dart, and report location back to the Scoring System
6. The Scoring System will update the game and player statistics

This process repeats as players take turns throwing until a winner is declared or the game is ended.

## Imaging System

The Imaging System is the subsystem responsible for identifying and locating darts on the dartboard. This system is dependent on the [Jetson Inference](https://github.com/dusty-nv/jetson-inference/tree/master) project and tools from Nvidia.

### Configuration

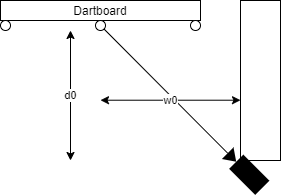


Figure : Dartboard Setup

Figure 3 shows how the Imaging System camera is setup. The camera is mounted on a wall aligned with the center of the dartboard. The camera sits at distance of 18 inches and of 14.5 inches from the center of the dartboard.

A dart board with a dart in the center

Description automatically generated

Figure : Camera View

Figure 4 shows the camera view when looking at the dartboard. The camera faces in line with the 11-point value and perpendicular to the 20-point value.

### Training

The Imaging System requires object detection training to properly identify darts. The [Jetson Inference – Collecting your own Detection Datasets](https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect-detection.md) project provides tools to properly capture, train, and detect custom objects.

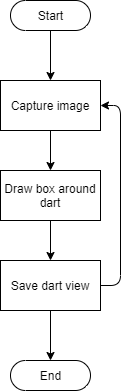


Figure : Training Logic

Figure 5 shows how the system is trained to identify darts. Images are captured, tagged, and saved to teach the system different positions of dart locations.

The system uses a [Single Shot Detection MultiBox Detector](https://roboflow.com/model/mobilenet-ssd-v2) machine learning architecture alongside PyTorch to train for dart recognition. DARTS is trained on 2200 base data points. 1400 data points include 18 positions at the triple ring + 18 positions between the double and triple ring + 18 positions at the double ring + 16 positions between the bull and double ring \* 20 numbers. 800 data points include additional lighting with 6 positions at the triple ring + 18 positions between the double and triple ring + 6 positions at the double ring + 10 positions between the bull and double ring \* 20 numbers. The model will continue to be trained as time permits.

### Recognition

The Imaging System uses object detection to identify darts. Once identified, the system draws a box around the object with coordinates corresponding to the image location.

A dart in the center of a dart board

Description automatically generated

Figure : Object Detection Box

Figure 6 shows what the system sees once an object is identified. The camera is mounted on the right side of the dartboard meaning darts land with the pointed end on the bottom left portion of the object detection box. The system uses this position to set the image location.

### Translation

The Imaging System camera is at a look angle, meaning the captured image shows the dartboard in the shape of an ellipse due to geometric distortion. The system uses features of the dartboard to create scale factors for putting the dart at a non-distorted position.

To account for projection distortion and squeezing of dart locations above the origin point and stretching below, the system uses homographic projections to translate pixel locations according to a destination perspective. The system uses [OpenCV](https://opencv.org/) to manipulate images to get generic outlines so geometric shapes can be fit. Once images are fit with shapes, data points for the transformation matrix are computed.



Figure : Dartboard Source Features

Figure 7 shows the creation of features on the source image. Fitting an ellipse yields the origin , major axis , and minor axis in the pixel frame.

Equation : Top Source Equation

Equation : Center Source Equation

Equation : Bottom Source Equation

Equation : Left Source Equation

Equation : Right Source Equation

Equation 1 through Equation 5 show how the system calculates source points for the homographic projection. The center of the ellipse that fits the dartboard does not have the center at the bull. A fudge factor was used to manually move the center point of the ellipse to the bull.



Figure : Dartboard Destination Features

Figure 8 shows the creation of features on the destination image. Fitting a circle yields the origin and radius in the pixel frame.

Equation : Top Destination Equation

Equation : Center Destination Equation

Equation : Bottom Destination Equation

Equation : Left Destination Equation

Equation : Right Destination Equation

Equation 6 through Equation 10 show how the system calculates destination points for the homographic projection.

[OpenCV – Homography](https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html) is used to compute the homographic translation matrix. The source and destination points are used to form input arrays for the matrix computation.

Equation : Source Array Equation

Equation : Destination Array Equation

Equation : Homographic Matrix Equation

Equation 11 through Equation 13 show the formation of the arrays and function for computing the homographic projection matrix.

Equation : Translation Equation

Equation 14 shows how the detected dart location is translated to normal.

### Mapping

The Imaging System uses polar coordinates to map dart hits appropriately. Standard dartboard distances are used to identify the various rings and numbers of the dartboard.

A circular object with a circle and arrows

Description automatically generated

Figure : Dartboard Polar Coordinates

The system uses the bullseye as the origin of the polar coordinate system. Figure 9 shows how the system maps the x and y locations of the dart hit to polar coordinates. The y-axis is in line with the camera look angle with the x-axis perpendicular.

Equation : Radius Equation

Equation 15 shows how the radius is computed from x and y distances. [Pythagoras Theorem](https://www.mathsisfun.com/pythagoras.html) is used to calculate a dart hit radius from the center. The system computes angles in degrees depending on the quadrant of the hit to ensure consistency when mapping to areas on the board.

X > 0 & Y > 0 (Quadrant 1):

Equation : Quadrant 1 Angle Equation

X < 0 & Y > 0 (Quadrant 2):

Equation : Quadrant 2 Angle Equation

X < 0 & Y < 0 (Quadrant 3):

Equation : Quadrant 3 Angle Equation

X > 0 & Y < 0 (Quadrant 4):

Equation : Quadrant 4 Angle Equation

Equation 16 through Equation 19 show how angle is computed from x and y distances. Inverse tangent functions are used to compute the angle relative to the x-axis or y-axis depending on quadrant of the hit.

A diagram of a dart board

Description automatically generated

Figure : Standard Dartboard Distances

Figure 10 shows standard dartboard distances in millimeter units of length. All calculations occur in millimeters to make comparisons easier. Standard distances are used as references for determining rings hit.

The system uses radius to determine the ring hit.

|  |  |  |
| --- | --- | --- |
| **Radius (mm)** | **Ring** | **Ring Map** |
| >225.5 – 170 | Outside dartboard | Z |
| 170 – 162 | Double ring | A |
| 162 – 107 | Between double and triple ring | B |
| 107 – 99 | Triple ring | C |
| 99 – 16 | Between triple ring and bull | D |
| 16 – 6.35 | Bull | X |
| 6.35 – 0 | Bullseye | Y |

Table : Dartboard Ring Mapping

Table 1 shows how the system maps radius to ring hit. The rings are broken into subsections to streamline game implementation.

A graph of a function

Description automatically generated with medium confidence

Figure : Standard Dartboard Angles

Figure 11 shows standard dartboard angles in degree units of angle. All angle calculations occur in degrees to make comparisons easier. Standard angles are used as references for determining the number hit.

The system uses angle to determine the number hit. Since the camera is mounted 90 degrees from the top of the dartboard, the top of the dartboard starts with number 11.

|  |  |
| --- | --- |
| **Angle (deg)** | **Number** |
| 351 – 9 | 20 |
| 9 – 27 | 5 |
| 27 – 45 | 12 |
| 45 – 63 | 9 |
| 63 – 81 | 14 |
| 81 – 99 | 11 |
| 99 – 117 | 8 |
| 117 – 135 | 16 |
| 135 – 153 | 7 |
| 153 – 171 | 19 |
| 171 – 189 | 3 |
| 189 – 207 | 17 |
| 207 – 225 | 2 |
| 225 – 243 | 15 |
| 243 – 261 | 10 |
| 261 – 279 | 6 |
| 279 – 297 | 13 |
| 297 – 315 | 4 |
| 315 – 333 | 18 |
| 333 – 351 | 1 |

Table : Dartboard Number Mapping

Table 2 shows how the system maps angle to number hit. Bull, bullseye, and hits outside of the dartboard map to number 0.

The system reports both ring and number hit after each dart is detected. For example, if a player hits the double ring of 20, the system will report number 20, ring A. If a player hits the bull the system will report number 0, ring X. If a player hits the bullseye the system will report number 0, ring Y. If a player hits outside the dartboard the system will report number 0, ring Z.

### State Machine

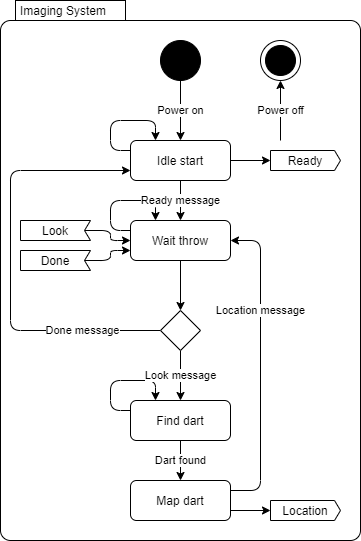


Figure : Imaging System State Machine Diagram

Figure 12 shows how states flows within the Imaging System. This system is comprised of image recognition logic. The state machine functions as follows:

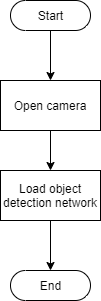


Figure : Imaging System Startup Logic

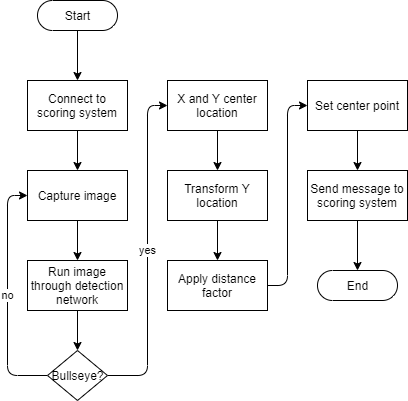


Figure : Imaging System IDLE START Logic

1. After power-up, the application will open and enter the IDLE START state. The system will block until connected to the Scoring System. Once connected, the system will load the detection model and attempt to recognize the bullseye. Once the bullseye position is identified, the system will send a message to the Scoring System and transition to the WAIT THROW state.

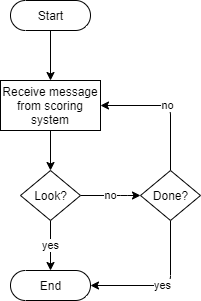


Figure : Imaging System WAIT THROW Logic

1. The system will remain in WAIT THROW state until a message is received from the Scoring System. If the system receives a LOOK message, it will transition to the FIND DART state. If a DONE message is received, the system will transition to the IDLE START state.

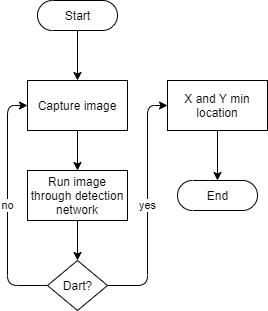


Figure : Imaging System FIND DART Logic

1. In the FIND DART state, the system will capture an image of the dartboard and run it through the detection network. Once a dart is recognized, the system will transition to the MAP DART state.

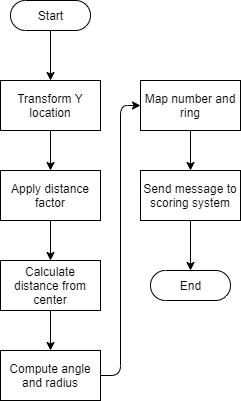


Figure : Imaging System MAP DART Logic

1. In the MAP DART state, the system will map the image location of the dart to a hit point on the board. Once positions are computed, the system will send a message to the scoring system and transition to the WAIT THROW state.

The system will continue to recognize images based on triggers from the Scoring System until powered off or manually stopped.

## Scoring System

The Scoring System is the subsystem responsible for updating user scores based on game mode as well as controlling user interaction. The user interface and database are part of the Scoring System.

### Database

The Scoring System implements a database for storing player information, playable games, and player hit records.

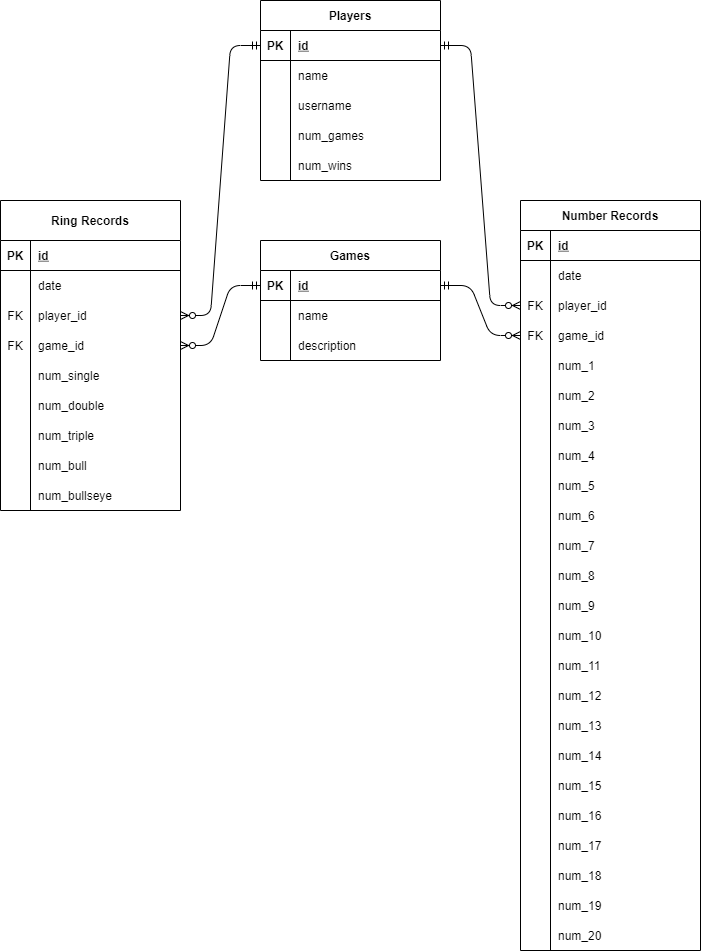


Figure : Scoring System Database Entity Relationship

Figure 18 shows how the system entities relate within the database context. The system logs hit locations as players execute games. Win and hit percentages by ring and number are available for query.

### User Interface

The user interface for the scoring system consists of nested displays based on states defined in Figure 30.

A screen shot of a game

Description automatically generated

Figure : Startup Display

Figure 19 shows what will display on the user interface when in the IDLE START state. ‘Help’ will show directions on how to start.

A screenshot of a video game

Description automatically generated

Figure : Profile Display

Figure 20 shows what will display on the user interface when in the START state. ‘Help’ will show directions on what each button option does. ‘Cancel’ will return to the startup display as seen in Figure 19.

A screenshot of a login screen

Description automatically generated

Figure : Create Profile Display

Figure 21 shows what will display when in the CREATE PROFILE state. ‘Help’ will show directions on what each button option does. ‘Cancel’ will return to the profile display as seen in Figure 20.

A screenshot of a video game

Description automatically generated

Figure : Select Game Display

Figure 22 shows what will display when in the SELECT GAME state. ‘Help’ will show summaries of each game. ‘Cancel’ will return to the profile display as seen in Figure 20.

A screenshot of a computer

Description automatically generated

Figure : Select Players Display

Figure 23 shows what will display when in the SELECT PLAYERS state. ‘Help’ will show directions on what each button option does. ‘Cancel’ will return to the select game display as seen in Figure 22.

A screen shot of a login screen

Description automatically generated

Figure : Load Profile Display

Figure 24 shows what will display when in the LOAD PROFILE state. ‘Help’ will show directions on what each button option does. ‘Cancel’ will return to the select players display as seen in Figure 23.

A screenshot of a game

Description automatically generated

Figure : Scoreboard Display

Figure 25 shows the scoreboard that each game will base their scoreboard display on when in the IDLE TURN and subsequent states. The dartboard will appear in grayscale to see hit markers. Hit markers will display analogous to the player throwing (red or black). Usernames will appear above the player button for the game. Scores will appear below the button.

### Gameplay

|  |  |
| --- | --- |
| **Scoring Games** | **Knockout Games** |
| 501 | Around the World |

Table : Hosted Games

The system will be capable of hosting scoring and knockout games. Table 3 shows what dart games will be available.

Figure 40 shows how the system will score games. The UPDATE GAME state will provide a framework for implementing various dart games. The system will use game specific logic when calculating scores.

#### “501” Game

The goal of “501” is to reach a score of 0. Players start with a score of 501 and count down as areas of the dartboard are hit. Players throw darts to any position, sum each throw, and subtract from 501. Double rings count as double multipliers while triple rings count as triple multipliers of each numbers’ value. The bull counts as 25 points while the bullseye counts as 50 points. Players win by hitting the double ring to reach a score of 0 even.

For example, if a player has 20 remaining, they must hit a double 10 to win. If they hit a double 5, they need a double 5 on their next throw to win.

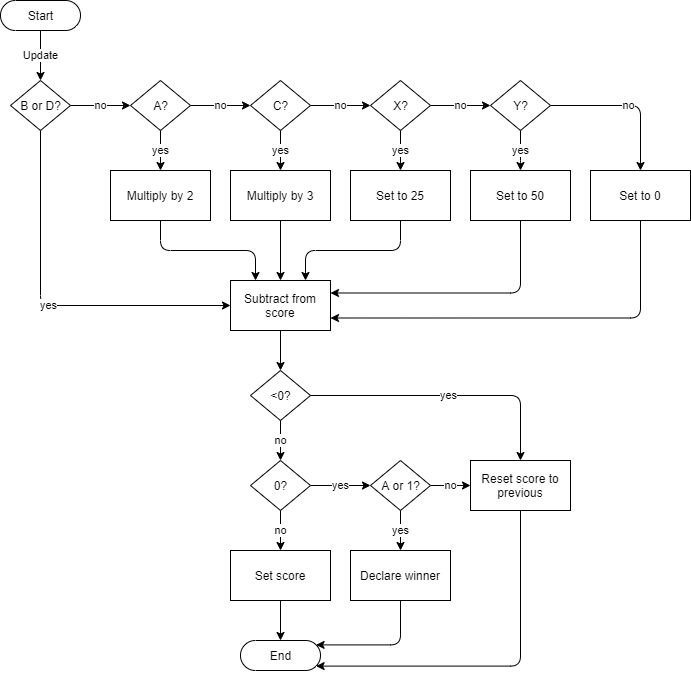


Figure : "501" Game Logic

The system uses mappings as seen in Table 1 to determine multipliers and scores to tally player scores throughout the game. Figure 26 shows how “501” logically works. When calculating the score, the system checks what ring was reported by the Imaging System. The ring determination yields a multiplier, straight value, or nothing. The system then subtracts the modified value from the loaded score. If the score goes below 0, the system resets the score to the loaded value. If the score equates to 0, the system checks if a double ring was hit. If a double ring was hit, the system declares a winner. Otherwise, the system resets the score to the loaded value. If the score is above 0, the system sets the computed score.

|  |  |  |  |
| --- | --- | --- | --- |
| **ASSERT** | | | **EXPECT** |
| **Number** | **Ring** | **Current Score** |
| 20 | A | 501 | 461 |
| 20 | B | 501 | 481 |
| 20 | C | 501 | 441 |
| 20 | D | 501 | 481 |
| 20 | X | 501 | 476 |
| 20 | Y | 501 | 451 |
| 20 | Z | 501 | 501 |
| 20 | A | 20 | 20 |
| 20 | B | 20 | 20 & NO WINNER |
| 10 | A | 20 | 0 & WINNER |

Table : "501" Unit Tests

Table 4 shows how the game logic for “501” is unit tested. The scenarios listed test the calculations and conditionals for each leg of the logic chain.

A screenshot of a game

Description automatically generated

Figure : "501" Scoreboard

Figure 27 shows what the “501” scoreboard will look like. Scores will update after each throw, displaying the remaining points.

#### “Around the World” Game

The goal of “Around the World” is to hit each number on the dartboard in sequential order. Players throw darts starting at position 1 and move clockwise around the board. Double and triple rings count as singles towards the number hit. The bull and bullseye count as nothing. Players win by knocking out all numbers, finishing with 20.

For example, if a player hits 1, 4, 18, they knockout 1 but are only on 4 because 18 was the last sequential number hit.

|  |  |
| --- | --- |
| **Current Number** | **Next Number (A, B, C, or D)** |
| 0 | 1 |
| 1 | 18 |
| 18 | 4 |
| 4 | 13 |
| 13 | 6 |
| 6 | 10 |
| 10 | 15 |
| 15 | 2 |
| 2 | 17 |
| 17 | 3 |
| 3 | 19 |
| 19 | 7 |
| 7 | 16 |
| 16 | 8 |
| 8 | 11 |
| 11 | 14 |
| 14 | 9 |
| 9 | 12 |
| 12 | 5 |
| 5 | 20 - WINNER |

Table : “Around the World” Mapping

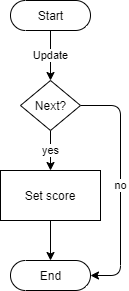


Figure : “Around the World” Game Logic

The system uses a mapping technique to check the current and next number to determine sequential knockout of the dartboard. Figure 28 shows how “Around the World” logically works. When calculating the score, the system checks what ring was reported by the Imaging System. This value is graded against the next number associated with the current number as seen in Table 5. If the number is the next number, the system sets the current number to the new number.

|  |  |  |  |
| --- | --- | --- | --- |
| **ASSERT** | | | **EXPECT** |
| **Number** | **Ring** | **Current Score** |
| 1 | A | 0 | 1 |
| 1 | B | 0 | 1 |
| 1 | C | 0 | 1 |
| 1 | D | 0 | 1 |
| 1 | X | 0 | 0 |
| 1 | Y | 0 | 0 |
| 1 | Z | 0 | 0 |
| 20 | A | 0 | 0 |
| 5 | A | 12 | 5 |
| 18 | A | 1 | 18 |
| 4 | A | 18 | 4 |
| 13 | A | 4 | 13 |
| 6 | A | 13 | 6 |
| 10 | A | 6 | 10 |
| 15 | A | 10 | 15 |
| 2 | A | 15 | 2 |
| 17 | A | 2 | 17 |
| 3 | A | 17 | 3 |
| 19 | A | 3 | 19 |
| 7 | A | 19 | 7 |
| 16 | A | 7 | 16 |
| 8 | A | 16 | 8 |
| 11 | A | 8 | 11 |
| 14 | A | 11 | 14 |
| 9 | A | 14 | 9 |
| 12 | A | 9 | 12 |
| 5 | A | 12 | 5 |
| 20 | A | 5 | 20 & WINNER |

Table : "Around the World" Unit Tests

Table 6 shows how the game logic for “Around the World” is unit tested. These scenarios test the mapping logic for moving sequentially around the dartboard.

A screenshot of a game

Description automatically generated

Figure : "Around the World" Scoreboard

Figure 29 shows what the “Around the World” scoreboard will look like. Scores will update after each throw, displaying the next number the player needs to hit.

### State Machine

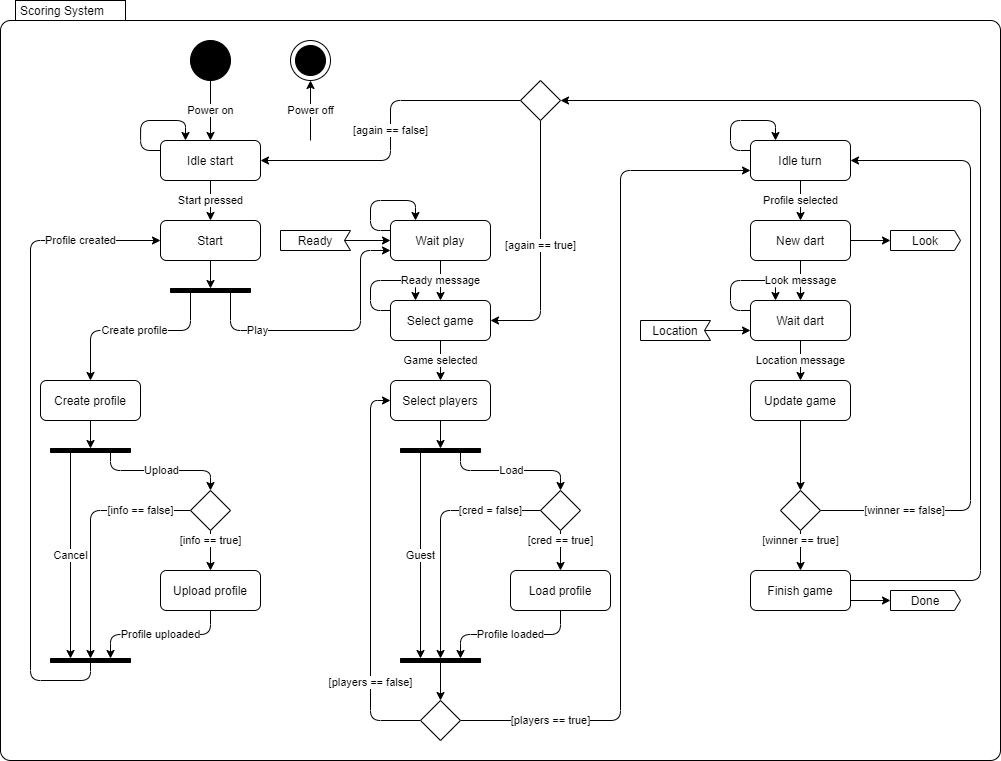


Figure : Scoring System State Machine Diagram

Figure 30 shows how states flow within the Scoring System. This system is comprised of game and user interface logic. The state machine functions as follows:

A diagram of a check button

Description automatically generated

Figure : Scoring System IDLE START Logic

1. After power-up, the application will open and enter the IDLE START state. The system will remain in this state until the player starts the application from the user interface. After starting the application, the system will transition to the START state.

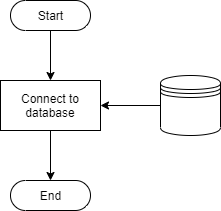


Figure : Scoring System START Logic

1. In the START state, the system will connect to the database and wait for the player to choose to create a profile or play. Once chosen, the system will transition to the appropriate state.

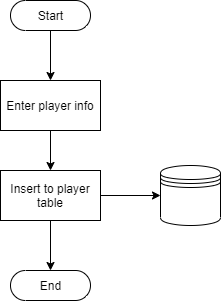


Figure : Scoring System UPLOAD PROFILE Logic

1. If the player chooses to create a profile, the system will enter the CREATE PROFILE state. Players will enter information to upload or cancel the create a profile request. Upon upload, the system will check to ensure information is filled out and transition to the UPLOAD PROFILE state if valid. Once uploaded, the system will transition back to the START state.

A diagram of a message

Description automatically generated

Figure : Scoring System WAIT PLAY Logic

1. If the player chooses to play, the system will enter the WAIT PLAY state. In this state, the system will connect to the Imaging System. The system will remain in this state until a message is received indicating the Imaging System is ready. Upon receipt of this message, the system will transition to the SELECT GAME state.

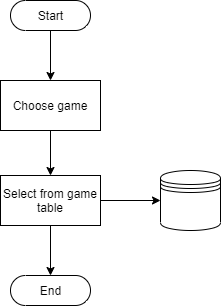


Figure : Scoring System SELECT GAME Logic

1. The system will remain in the SELECT GAME state until a game is selected. Once a game is selected, the system will transition to the SELECT PLAYERS state.

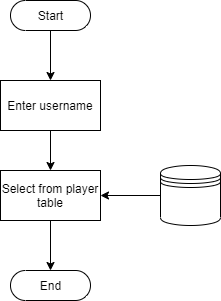


Figure : Scoring System LOAD PROFILE Logic

1. The system will remain in the SELECT PLAYERS state until the appropriate number of players are ready for the game specified. Players can play as a guest or load an existing profile. Upon loading a profile, the system will check to ensure credentials are filled out and transition to the LOAD PROFILE state if valid. After the appropriate number of players are selected, the system will transition to the IDLE TURN state.

A diagram of a check button

Description automatically generated

Figure : Scoring System IDLE TURN Logic

1. The system will remain in the IDLE TURN state until the user selects their profile before throwing darts. Once a profile is selected, the system will transition to the NEW DART state.

A diagram of a message

Description automatically generated

Figure : Scoring System NEW DART Logic

1. In the NEW DART state, the system will send a message to the Imaging System indicating a dart is incoming and transition to the WAIT DART state.

A diagram of a system

Description automatically generated

Figure : Scoring System WAIT DART Logic

1. The system will remain in the WAIT DART state until a message is received from the Imaging System indicating it has recognized and located a dart. Upon receipt of this message, the system will transition to the UPDATE GAME state.



Figure : Scoring System UPDATE GAME Logic

1. In the UPDATE GAME state, the system will update player scores based on the game selected. If the system determines a player won, the system will enter the FINISH GAME state. If a winner is not declared, the system will enter the IDLE TURN state.

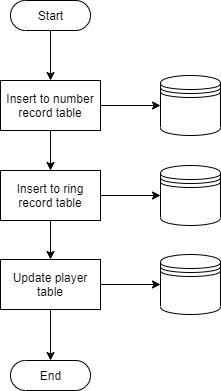


Figure : Scoring System FINISH GAME Logic

1. In the FINISH GAME state all player data will be uploaded to the database. Once complete, the system will check if the player(s) want to play again. If yes, the system will transition to the SELECT GAME state. If no, the system will transition to the IDLE START state.

The system will continue to check for player input until powered off or manually stopped.

# Interface Description

DARTS uses three main interface standards to communicate across subsystems. The first is CSI which is a specification of MIPI. The second is TCP which is an IP standard for communication within a network. The third is USB which is a serial standard for peripheral communication.

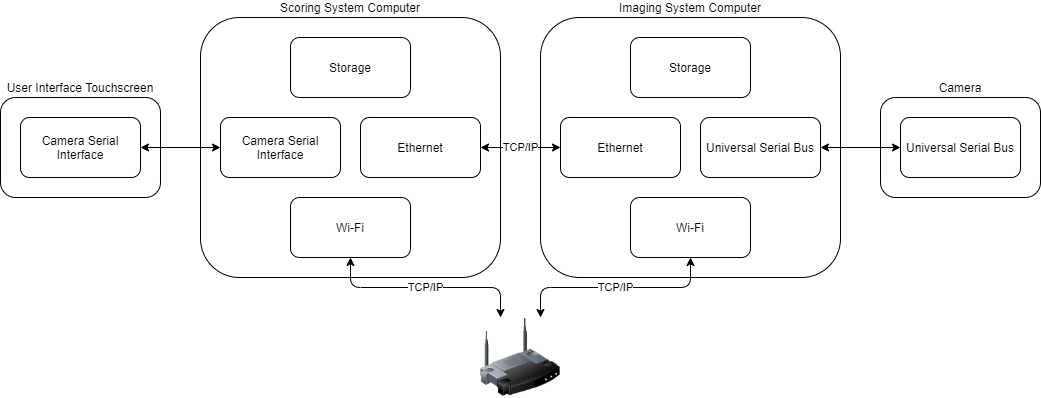


Figure : Interface Diagram

Figure 42 shows how subsystems communicate across the system. The Imaging System uses a USB camera for easier manipulation of picture and physical mount location.

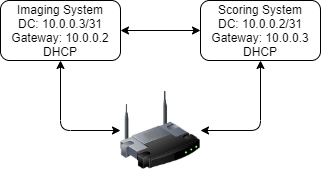


Figure : Network Diagram

Figure 43 shows the DARTS network. The Scoring System is directly connected to the Imaging System via Ethernet. Both the Imagin and Scoring System are also on the ISP subnet of the household. DHCP is used to assign IP addresses, allowing remote access from any household device.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sender** | **Receiver** | **Purpose** | **Content** |
| Imaging | Scoring | Ready to play | “READY” |
| Scoring | Imaging | Start detection | “LOOK” |
| Imaging | Scoring | Dart location | LOCATION message |

Table : Communication Matrix

Table 7 shows how message content flows throughout the system. The Imaging System acts as a TCP server while the Scoring System acts as a TCP client. Port 5000 is used for traffic between the two systems. The system uses SCPI-like syntax to initiate state triggers.

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| int | number |
| str | ring |
| float | radius |
| float | theta |

Table : LOCATION Message

Table 8 shows the contents of the dart location structure the Imaging System populates once a dart is identified. This data is sent to the Scoring System for game progression.

## Imaging System

The Imaging System uses USB to communicate with the offboard camera. This is the main interface for acquiring images of the dartboard. The system uses TCP/IP over Ethernet to communicate with the Scoring System. This allows the system to receive messages to look for an incoming dart and transmit messages containing the dart location.

## Scoring System

The Scoring System uses CSI to communicate with the user interface touchscreen. This is the main interface for user interaction with the application. The system uses TCP/IP over Ethernet to communicate with the Imaging System and TCP/IP over Wi-Fi to communicate with home server.

# Material Requirements

DARTS is comprised of two main computing systems.

The Scoring System runs on a Raspberry Pi 3B+. This microprocessor has built-in Wi-Fi capability in addition to Ethernet communication. The Raspberry Pi also has CSI connectivity which is used to communicate with a touchscreen user interface.

The Imaging System runs on a Nvidia Jetson Nano Developer Kit. This GPU-based microprocessor has built-in Ethernet communication as well as USB and CSI connectivity.

|  |  |  |
| --- | --- | --- |
| **Item** | **Price** | **Website** |
| Raspberry Pi 3B+ | $35.00 | https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/ |
| SanDisk Ultra 64GB MicroSD Card | $8.99 | https://www.amazon.com/SanDisk-Ultra-microSDHC-Memory-Adapter/dp/B08GYBBBBH |
| FREENOVE 5 Inch Touchscreen | $39.95 | https://www.amazon.com/FREENOVE-Touchscreen-Raspberry-Capacitive-Driver-Free/dp/B0B455LDKH?th=1 |
| Raspberry Pi Power Supply | $9.95 | https://www.amazon.com/CanaKit-Raspberry-Supply-Adapter-Listed/dp/B00MARDJZ4 |
| Nvidia Jetson Nano Developer Kit | $149.00 | https://developer.nvidia.com/embedded/jetson-nano-developer-kit |
| SanDisk Ultra 64GB MicroSD Card | $8.99 | https://www.amazon.com/SanDisk-Ultra-microSDHC-Memory-Adapter/dp/B08GYBBBBH |
| Jetson Nano Metal Case w/ Fan | $25.59 | https://www.amazon.com/Metal-Case-Fan-Jetson-Nano/dp/B07Z2MFTYC?th=1 |
| Logitech C270 HD Webcam | $19.90 | https://www.amazon.com/Logitech-Desktop-Widescreen-Calling-Recording/dp/B004FHO5Y6 |
| Wireless NIC Module for Jetson Nano | $23.99 | https://www.amazon.com/dp/B07SGDRG34 |
| Jetson Nano Power Supply | $13.68 | https://www.amazon.com/5V-Power-Supply-Adapter-Universal/dp/B07RTWD725?th=1 |
| **TOTAL** | $335.04 |  |

Table : Bill of Materials

# Resource Requirements

Development for DARTS was completed in the Python language. Python was chosen due to the amount of FOSS libraries and examples that are referenced during project development.

VS Code was used to write Python code for both computing systems. This IDE was chosen because of the FOSS nature of the environment and due to the number of extensions possible for faster development within the Python language.

|  |  |
| --- | --- |
| **Extension** | **Reason** |
| Python | Rich support for language |
| Pylance | Static type checking tool |
| Intellicode | AI-assisted development tool |
| Remote Explorer | View remote machines for SSH |
| Remote-SSH | Open folders on remote machines using SSH |
| SQLite | Explore and query SQLite databases |

Table : VS Code Extensions

Table 10 shows a list of VS Code extensions that were used for faster system development. These extensions will allow remote development to occur on the Raspberry Pi and Jetson Nano processors.

|  |  |  |  |
| --- | --- | --- | --- |
| **System** | **Reason** | **Library** | **Documentation** |
| Scoring | GUI development | pyqt5 | https://pypi.org/project/PyQt5/ |
| Scoring | Graphics | turtle | https://docs.python.org/3/library/turtle.html |
| Imaging | Jetson Inference | \* | https://github.com/dusty-nv/jetson-inference |
| Imaging | Computer vision | opencv-python | https://pypi.org/project/opencv-python/ |
| Imaging | Array computing | numpy | https://pypi.org/project/numpy/ |

Table : Python Libraries

Table 11 shows a list of non-standard Python libraries that were be used to implement DARTS. \*[Jetson Inference – Building the Project from Source](https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md) is required and installs several dependencies for image detection and training. Libraries not listed are assumed to be available within the standard Python installation package.

|  |  |
| --- | --- |
| **Application** | **Reason** |
| VNC Viewer | Remote Raspberry Pi/Jetson Nano access |
| Wireshark | TCP/IP packet observation |
| Cygwin | SSH and X11 forwarding |
| DB Browser for SQLite | SQL database visualization |

Table : Third-Party Applications

Table 12 shows a list of third-party applications that will be used during development. These applications will be used as necessary for debugging purposes.

Existing dart scoring projects were referenced to speed up development and integration of DARTS.

The following projects were referenced for image processing and dart detection:

* <https://developer.nvidia.com/embedded/community/jetson-projects/dart_score>
* <https://github.com/hanneshoettinger/opencv-steel-darts>
* <https://github.com/wmcnally/deep-darts>

Standard dart rules are referenced for implementation of the Scoring System:

* <https://dartsguide.net/guides/dart-games-rules/>

# Development Plan and Schedule

Planning and tracking of the dart scoring system occurred in GitHub. Code will undergo CM in an online repository which can be accessed from the following link:

<https://github.com/kparlak/dart-scoring-system>

GitHub Projects was be used to plan and track progress throughout development. Projects allow milestones to be tracked with related issues. Issues within Projects were tied to merge requests on the main repository. The online project can be accessed from the following link:

<https://github.com/users/kparlak/projects/2>



Figure : Roadmap Example

TODO : Final roadmap with milestones Figure 44 shows a snapshot of the project roadmap in GitHub. Milestones are represented as vertical lines traversing the plan. Issues were given start and end dates within GitHub to align progress with milestone and completion dates.

## Milestones and Schedule

This project will use milestone-based development, meaning a new capability will be introduced when a milestone is completed. Unit testing will be used to drive individual milestone capabilities. This will ensure each subsystem is functional part and apart from the overall system.

The first milestone will be Dart Recognition Development. The goal of this milestone will be to implement computer vision and image processing on the imaging system. After this milestone is complete the system will be able to do the following:

* Recognize a dart and map it to a location on the board
* Transmit dart location via TCP/IP

The second milestone will be Scoring and Game Development. The goal of this milestone will be to implement game logic for “501” and “Around the World” on the scoring system as well as communication with the imaging system. Building on the previous milestone, after this milestone is complete the system will be able to do the following:

* Receive dart location via TCP/IP
* Calculate score based on game being played
* Determine game winner

The third milestone will be Database Development. The goal of this milestone will be to implement a database on the scoring system and store data from games being played. Building on the previous milestone, after this milestone is complete the system will be able to do the following:

* Host database for user profiles and scores
* Upload data to database as game progresses

The fourth and final milestone will be User Interface Development. The goal of this milestone will be to implement user interface designs on the scoring system. This includes hit maps and scoreboards as well as game flow interfaces. Building on the previous milestone, after this milestone is complete the system will be able to do the following:

* Enable user interaction with touchscreen interface
* Display scoreboard and hit maps

Full system functionality will be available following the fourth milestone.

|  |  |
| --- | --- |
| **Date** | **Milestone** |
| 10/2 – 10/23 | Dart Recognition Development |
| 10/23 – 11/6 | Scoring and Game Development |
| 11/6 – 11/13 | Database Development |
| 11/13 – 12/4 | User Interface Development |

Table : Milestone Schedule

## Risk

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Low | Impact | | | High |
| Very Likely |  |  |  |  |  |
| Likelihood |  | **1** |  |  |  |
|  |  |  | **2** |  |
|  |  |  |  |  |
| Not Likely |  |  |  |  |  |

Table : Risk Impact/Probability

Table 14 shows the impact/probability table for risks associated with this project. The current risks for this project are as follows:

1. Loss of home internet connectivity

Mitigation: Switch Raspberry Pi to act as WAP rather than Wi-Fi client when internet connectivity is lost

1. Consistent lighting and angle viewpoints for Imaging System likely non-deterministic

Mitigation: Implement calibration routines that will run when Imaging System is powered on

# Assembly

[Getting Started With Jetson Nano Developer Kit](https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit) should be referenced for introduction to the Nvidia Jetson Nano Developer Kit.

[Jetson Nano Case](https://www.waveshare.com/wiki/Jetson_Nano_Case_(C)) should be referenced for construction of Jetson casing and subcomponents.

The Imaging System must be powered on prior to playing games with the Scoring System.

[Freenove Touchscreen Monitor for Raspberry Pi](https://github.com/Freenove/Freenove_Touchscreen_Monitor_for_Raspberry_Pi) should be referenced for assembly of Raspberry Pi touchscreen.

# Performance

# References

<https://github.com/dusty-nv/jetson-inference/tree/master>

<https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect-detection.md>

<https://roboflow.com/model/mobilenet-ssd-v2>

<https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html>

<https://opencv.org/>

<https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html>

<https://www.mathsisfun.com/pythagoras.html>

<https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md>

<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit>

<https://www.waveshare.com/wiki/Jetson_Nano_Case_(C)>

<https://github.com/Freenove/Freenove_Touchscreen_Monitor_for_Raspberry_Pi>

Other:

<https://www.sqlitetutorial.net/sqlite-python/update/>

<https://snapcraft.io/install/sqlitebrowser/raspbian>

<https://www.tomshardware.com/how-to/forward-x-session-ssh>

# Appendix

## Acronyms

|  |  |
| --- | --- |
| CM | Configuration Management |
| CSI | Camera Serial Interface |
| DHCP | Dynamic Host Configuration Protocol |
| FOSS | Free and Open Source Software |
| IDE | Integrated Development Environment |
| IP | Internet Protocol |
| ISP | Internet Service Provider |
| MIPI | Mobile Industry Processing Interface |
| SCPI | Standard Commands for Programmable Instruments |
| SSH | Secure Shell |
| TCP | Transmission Control Protocol |
| USB | Universal Serial Bus |
| VS | Visual Studio |
| WAP | Wireless Access Point |

## Source Code

Please refer to <https://github.com/kparlak/dart-scoring-system/tree/main> for full system source code.

### Imaging System State Machine

### Scoring System State Machine

### ‘501’ Game

### ‘Around the World’ Game