Dart Scoring System
Critical Design Review

EN.525.743 – Embedded Systems Development Laboratory

Kevin Parlak

Contents

Figures

Tables

Equations

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 the player must know how a specific game is scored and manually keep score as the game progresses.

Figure 1: 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. The dart scoring system solution will consist of multiple subsystems that interact to identify darts and score games automatically. The goal of this project is to produce the core capabilities needed to implement such a solution. Capabilities and limitations are defined that describe what will and will not be possible with the system being designed.

Capabilities

The system will have the following capabilities to compliment core functionality:

- Scoring system database shall be capable of holding up to 10 player profiles and information
- User interface shall be capable of updating displays after each dart is thrown and identified
    - Virtualizes games for players through hit map/scoreboard updates
- User interface shall be capable of pulling statistics such as win %, hit % by number, double ring hit %, triple ring hit %, outer bullseye hit %, inner bullseye hit % per profile
    - Helps players understand how their game is doing

Limitations

System capabilities will be limited based on development time and resources available. The system will have the following limitations while ensuring core capability is achieved:

- Scoring system database shall only be accessible via home network or WAP
- Imaging system shall only be capable of tracking a single set of darts per turn
    - Requires players to select profile before throwing
- Scoring system shall only host two games (score-based and knockout-based)
    - Provides framework for adding future games
- System shall only be started from the physical user interface
- System shall only have a two-player limit per game
- Scoring system database shall only add new user profiles from the physical user interface

Functional Description

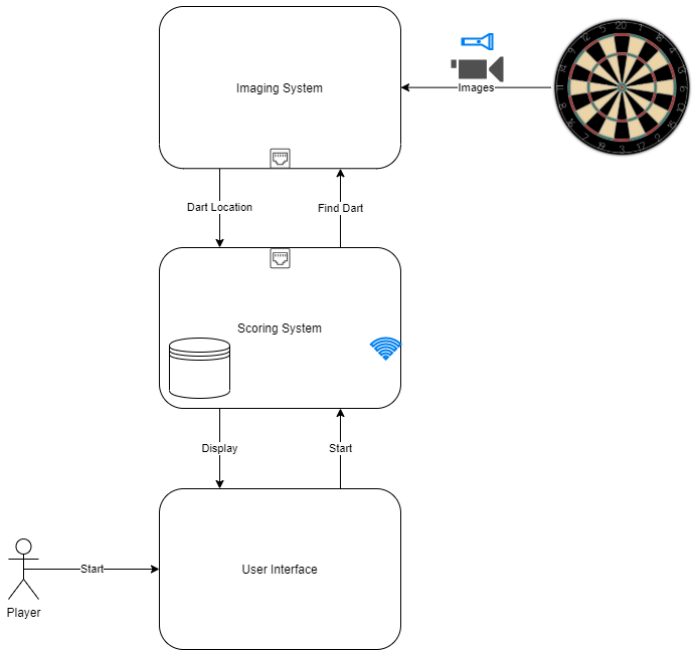

Figure 2: Functional Block Diagram

Figure 2 illustrates how the dart scoring system will work. The system will function as follows:

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

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.

Setup

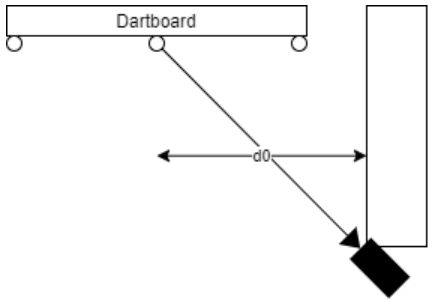

Figure 3: Dartboard Setup

Figure 3 shows how the imaging system camera will be setup. The camera will be mounted on a wall aligned with the center of the dartboard. Distance from the mounting point, $d0$ to the bullseye must be measured and input to the system.

Figure 4: Camera View

Figure 4 shows what the camera will see when looking at the dartboard. The camera will face in line with the 11-point value and perpendicular to the 20-point value.

Translation

The camera will be at a look angle, meaning the image captured will show the dartboard in the shape of an ellipse because of geometric distortion. The system will use features of the dartboard to create scale factors for putting the dart at a relative position from the bullseye to allow easier mapping of the hit. The system will use the image location and ratio the known distances. Let $p0$ be the image location of the bullseye center.

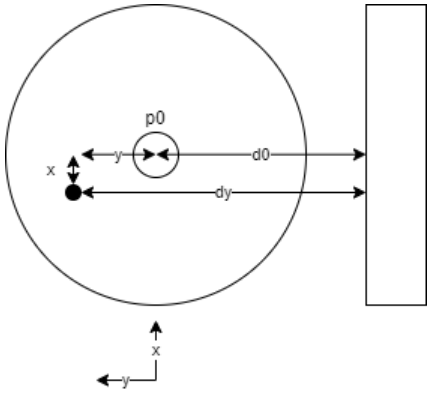

Figure 5: Dartboard Translation

Figure 5 shows how the system will manipulate the positions to identify physical dart location on the board. The camera will be at a look angle lateral to the x-axis, meaning distortion in x-direction does not need to be accounted for. Let dy be the physical distance of the hit from the mounting point.

$$\frac{p0}{d0} = \frac{py}{dy}$$

Equation 1: Hit Distance to Mounting Point Equation

Equation 1 shows how the hit distance to the mounting point will be computed. The system will produce an image location py of the hit and ratio to find dy.

$$y = dy - d0$$

Equation 2: Hit Distance to Bullseye Equation

Equation 2 shows how the hit distance to the bullseye will be computed. The system will subtract the input bullseye distance $d0$ from dy to find y.

Training

The imaging system will require object training to properly identify darts.

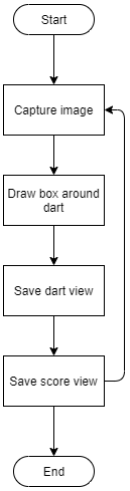

Figure 6: Training Logic

Figure 6 shows how the system will be trained to identify darts. Images will be captured, tagged, and saved to teach the system different positions of dart locations.

There is a total of 82 definitive landing points for the darts (4 rings * 20 numbers + 1 bull + 1 bullseye). The system will train on 4 points from each area + 22 non-scoring points totaling 350 distinct points. This data will be saved as unique dart points and scoring areas for use as the object detection model.

Recognition

The imaging system will use trained object detection to identify what a dart looks like. Once identified, the system will draw a box around the object with coordinates corresponding to the image location.

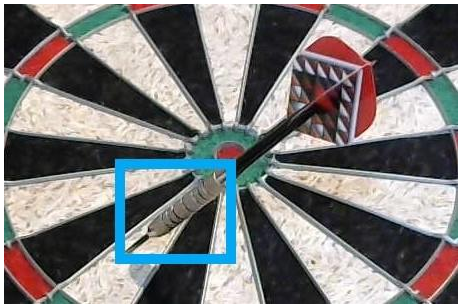

Figure 7: Object Detection Box

Figure 7 shows what the system will show once an object is identified. The camera will be mounted on the right side of the dartboard meaning darts will always land with the pointed end on the bottom left portion of the object detection box. The system will use this position to set the image location.

Mapping

The system will use polar coordinates to map the hit appropriately. Standard dartboard distances will be used to identify the various rings and numbers of the dartboard.

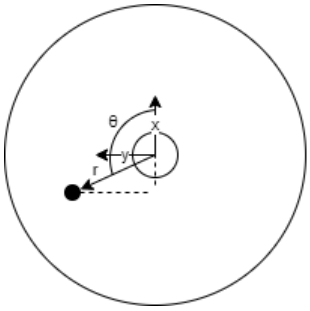

Figure 8: Dartboard Polar Coordinates

The system will use the bullseye as the origin of the polar coordinate system. Figure 8 shows how the system will map the x and y locations of the dart hit to polar coordinates. The y-axis will be in line with the camera look angle with the x-axis perpendicular.

$$r = \sqrt{x^2 + y^2}$$

Equation 3: Radius Equation

Equation 3 shows how the radius will be computed from x and y distances. Pythagoras Theorem will be implemented to calculate a dart hit radius from the center. The system will compute 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):

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

Equation 4: Quadrant 1 Angle Equation

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

$$\theta = \tan^{-1}\left(\frac{x}{y}\right) + 90$$

Equation 5: Quadrant 2 Angle Equation

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

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) + 180$$

Equation 6: Quadrant 3 Angle Equation

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

$$\theta = \tan^{-1}\left(\frac{x}{y}\right) + 270$$

Equation 7: Quadrant 4 Angle Equation

Equation 4, Equation 5, Equation 6, and Equation 7 show how the angle will be computed from x and y distances. Inverse tangent functions will be used to compute the angle relative to the x-axis or y-axis depending on quadrant of the hit.

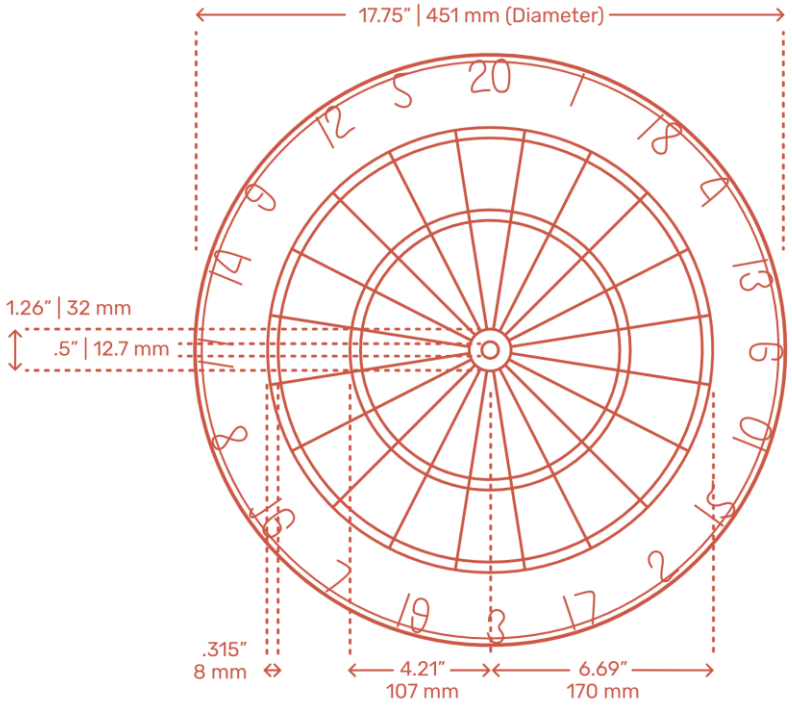

Figure 9: Standard Dartboard Distances

Figure 9 shows standard dartboard distances in millimeter units of length. All calculations will occur in millimeters to make comparisons easier. Standard distances will be used as references for determining rings hit.

The system will use radius to determine the ring hit.

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

Table 1: Dartboard Ring Mapping

Table 1 shows how the system will map radius to ring hit. The rings will be broken into subsections to streamline game implementation.

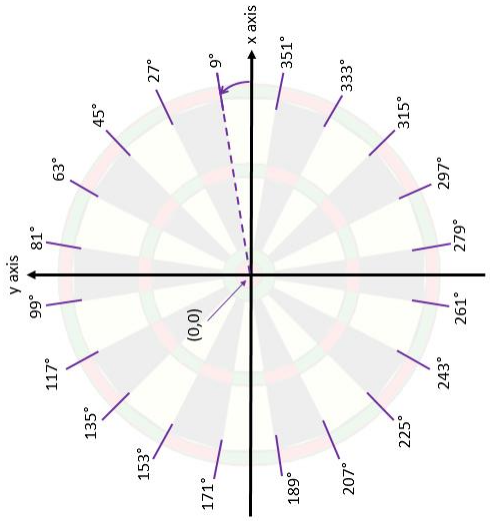

Figure 10: Standard Dartboard Angles

Figure 10 shows standard dartboard angles in degree units of angle. All angle calculations will occur in degrees to make comparisons easier. Standard angles will be used as references for determining the number hit.

The system will use angle to determine the number hit. Since the camera will be mounted 90 degrees from the top of the dartboard, the top of the dartboard will start 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 2: Dartboard Number Mapping

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

The system will report 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 value 20, ring A. If a player hits the bull the system will report value 0, ring X. If a player hits the bullseye the system will report value 0, ring Y. If a player hits outside the dartboard the system will report value 0, ring Z.

State Machine

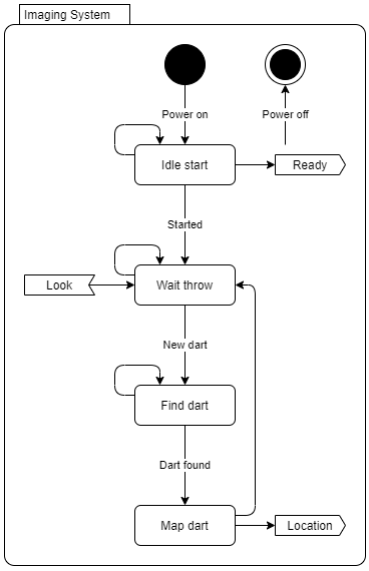

Figure 11: Imaging System State Machine Diagram

Figure 11 shows how states will flow within the imaging system. This system will be comprised of image recognition logic. The imaging system state machine will function as follows:

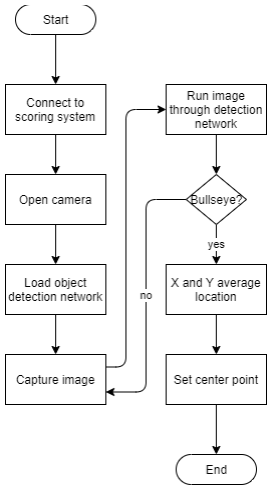

Figure 12: Imaging System IDLE START Logic

1. After power-up, the application will open and enter the IDLE START state. The system will remain in this state until the system completes recognition of 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 13: Imaging System WAIT THROW Logic

2. The system will remain in WAIT THROW state until a message is received from the scoring system indicating it should start looking for an incoming dart. Upon receipt of this message, the system will transition to the FIND DART state.

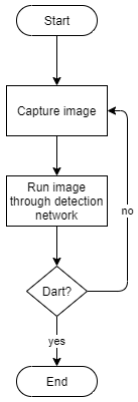

Figure 14: Imaging System FIND DART Logic

3. 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.

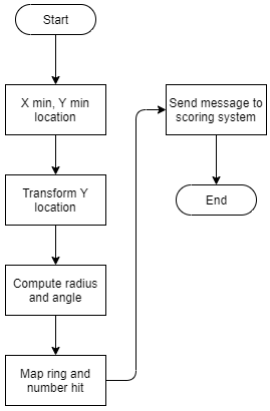

Figure 15: Imaging System MAP DART Logic

4. 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 will implement a database for storing player information.

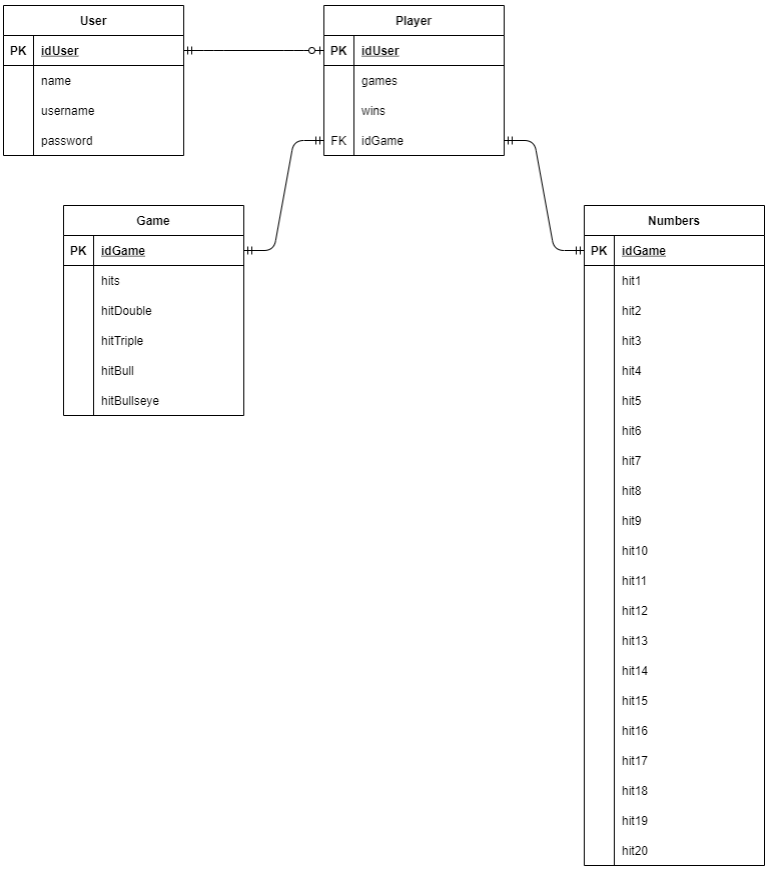

Figure 16: Scoring System Entity Relationship

Figure 16 shows how the system will relate player profiles to statistics. The system will log hit locations as players execute games. Win and hit percentages by ring and number will be available for query.

State Machine

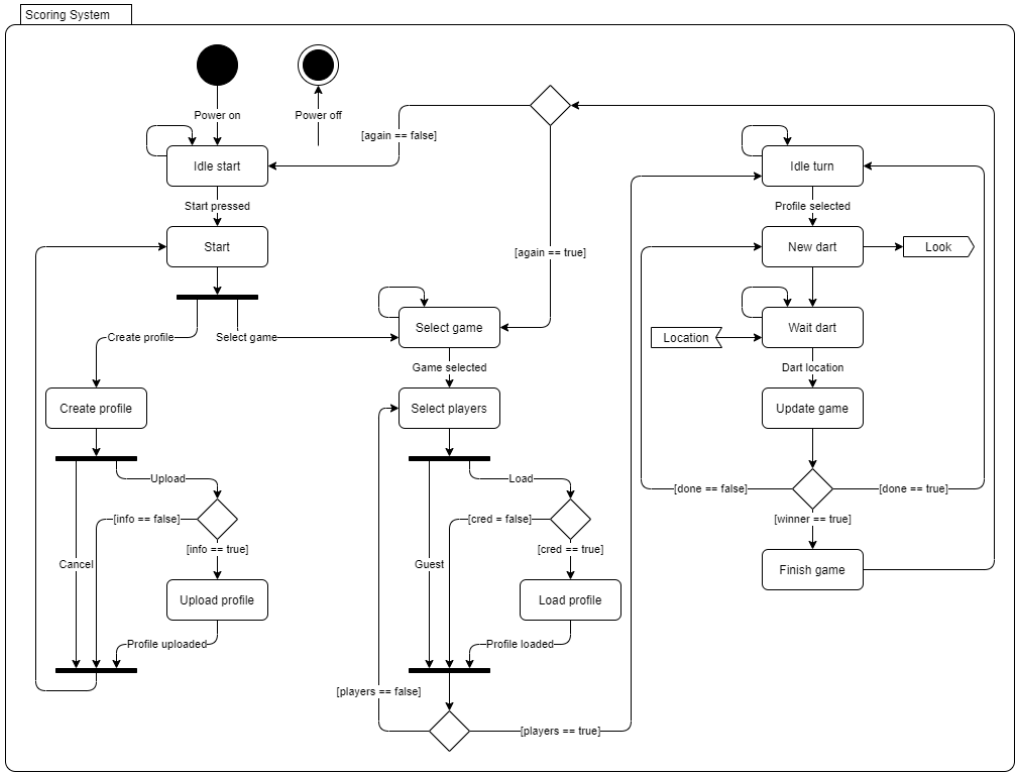

Figure 17: Scoring System State Machine Diagram

Figure 17 shows how states will flow within the scoring system. This system will be comprised of game and user interface logic. The scoring system state machine will function as follows:

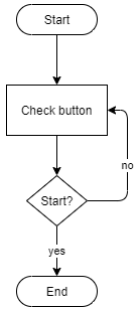

Figure 18: Scoring System IDLE START Logic

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 enter the START state.
2. The system will remain in the START state until the player creates a profile or selects a game. Once chosen, the system will transition to the appropriate state.

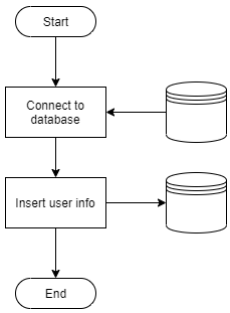

Figure 19: Scoring System UPLOAD PROFILE Logic

3. If the player creates a profile, the system will enter the CREATE PROFILE state. In this state, the player will enter information to upload or cancel the create request. Upon upload, the system will check to ensure information is filled out and enter the UPLOAD PROFILE state if valid. Once uploaded, the system will transition back to the START state.
4. If the player selects a game, the system will enter the SELECT GAME state. In this state, the system will wait for the user to select a game.

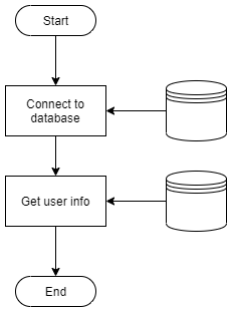

Figure 20: Scoring System LOAD PROFILE Logic

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

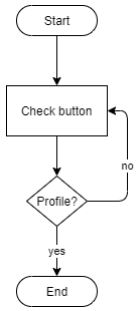

Figure 21: Scoring System IDLE TURN Logic

6. 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.

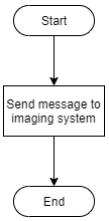

Figure 22: Scoring System NEW DART Logic

7. 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.

Figure 23: Scoring System WAIT DART Logic

8. 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.

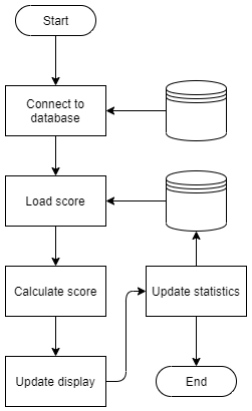

Figure 24: Scoring System UPDATE GAME Logic

9. In the UPDATE GAME state, the system will update player throw statistics and scores based on the game selected. If the system calculates that the player won, the system will enter the FINISH GAME state. If a winner is not declared, the system will check two other conditionals. If the player has thrown enough darts per turn the system will enter the IDLE TURN state, otherwise the system will enter the NEW DART state.

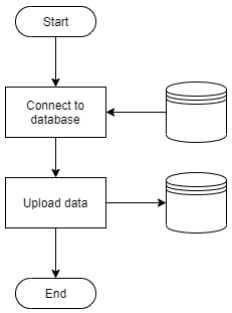

Figure 25: Scoring System FINISH GAME Logic

10. 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.

User Interface

The user interface for the scoring system consists of nested displays based on states defined in Figure 17.

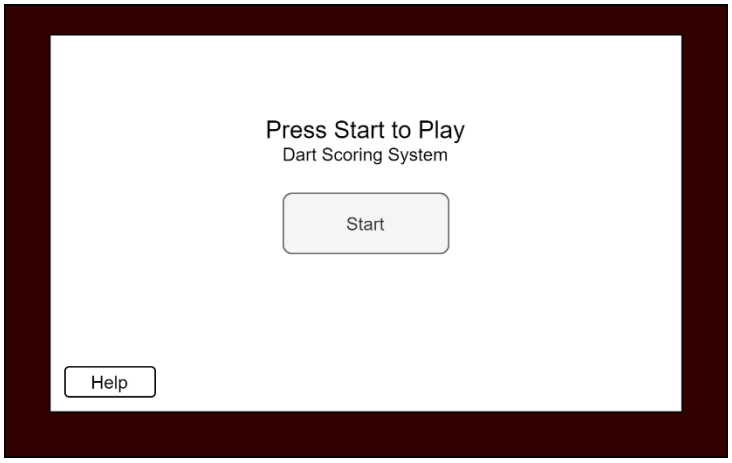

Figure 26: Startup Display

Figure 26 shows what will display on the user interface when in the IDLE START state. 'Help' will show directions on how to start.

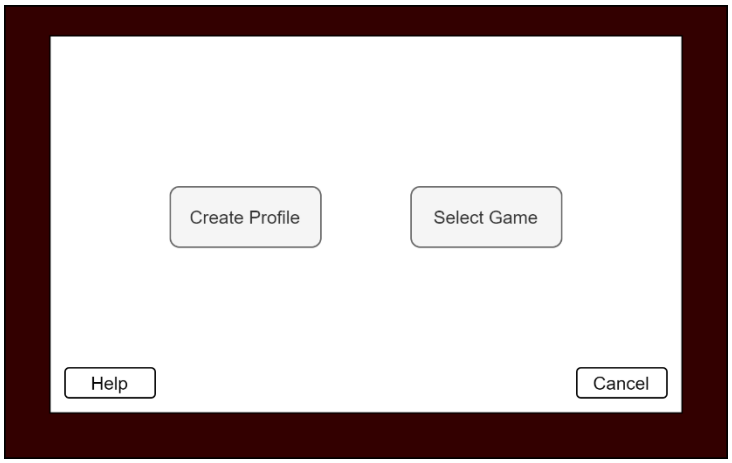

Figure 27: Profile Display

Figure 27 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 26.

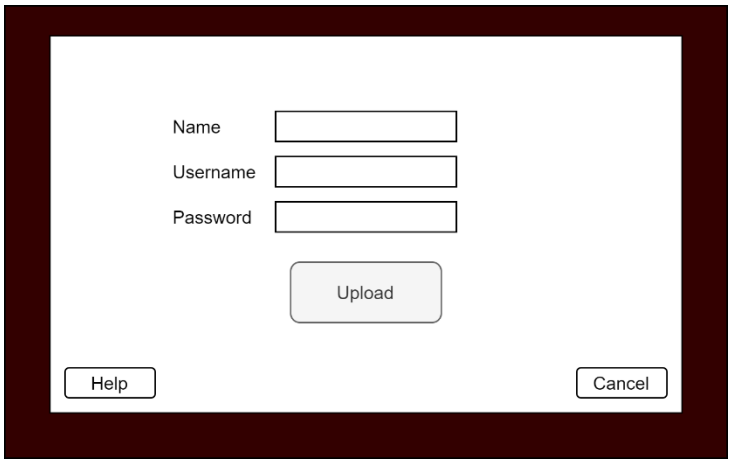

Figure 28: Create Profile Display

Figure 28 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 27.

Figure 29: Select Game Display

Figure 29 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 27.

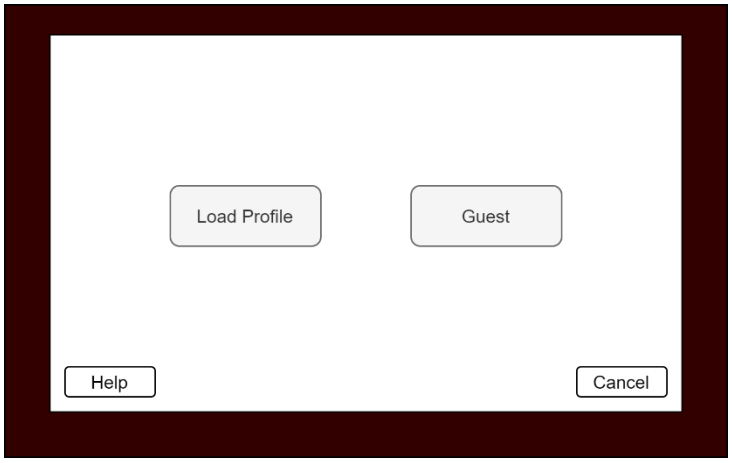

Figure 30: Select Players Display

Figure 30 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 29.

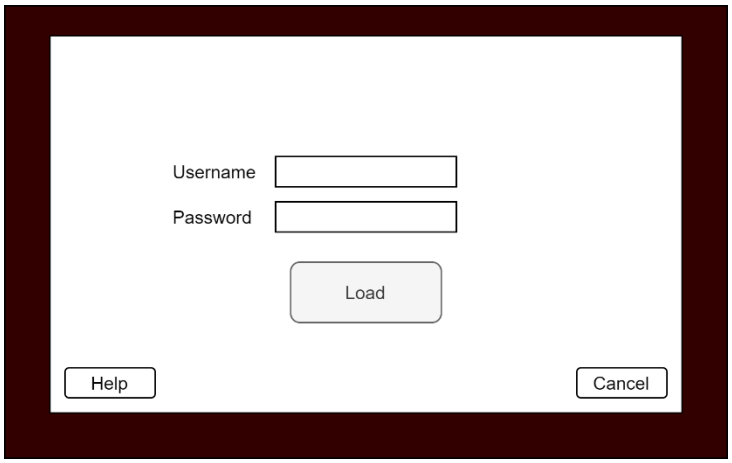

Figure 31: Load Profile Display

Figure 31 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 30.

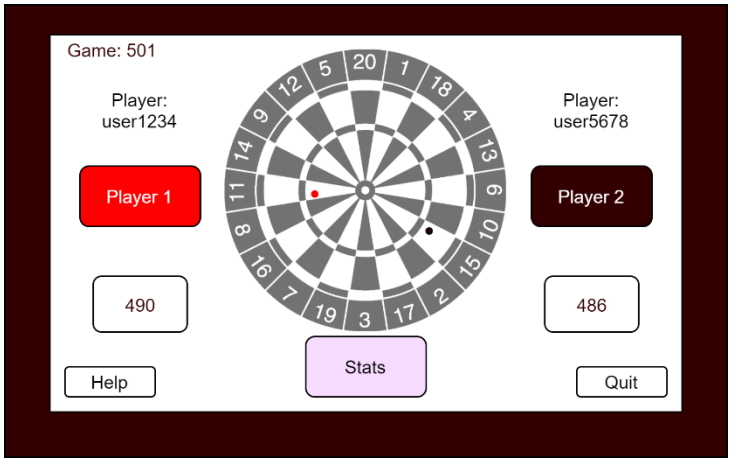

Figure 32: Scoreboard Display

Figure 32 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 3: Hosted Games

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

Figure 24 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.

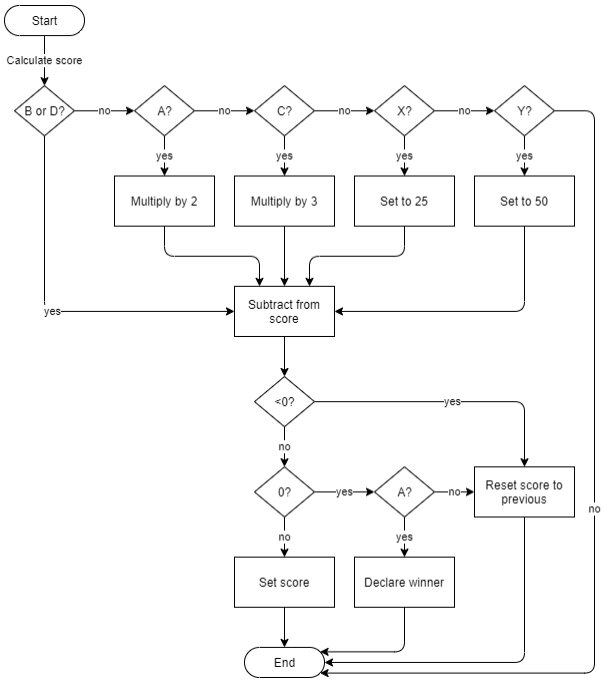

Figure 33: "501" Game Logic

The system will use mappings as seen in Table 1: Dartboard Ring MappingTable 1 to determine multipliers and scores to tally player scores throughout the game. Figure 33 shows how "501" will logically work. When calculating the score, the system will check what number was reported by the imaging system. The number determination will yield a multiplier, straight value, or nothing. The system will then subtract the modified value from the loaded score. If the score goes below 0, the system will reset the score to the loaded value. If the score equates to 0, the system will check if a double ring was hit. If a double ring was hit, the system will declare a winner. Otherwise, the system will reset the score to the loaded value. If the score is above 0, the system will set the computed score.

ASSERT			**EXPECT**
Number	**Ring**	**Current Score**	
20	A	501	461
20	B	501	481
20	C	501	441
20	D	501	461
20	X	501	476

20	Y	501	451
20	Z	501	501
20	A	20	20
20	B	20	20
10	A	20	0 & WINNER

Table 4: "501" Unit Tests

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

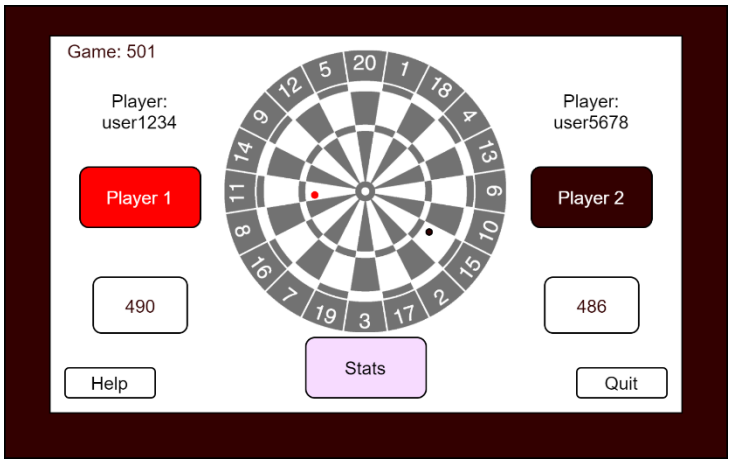

Figure 34: "501" Scoreboard

Figure 34 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 5: “Around the World” Mapping

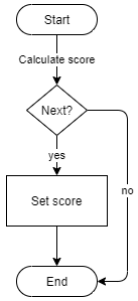

Figure 35: “Around the World” Game Logic

The system will use a mapping technique to check the current and next number to determine sequential knockout of the dartboard. Figure 35 shows how “Around the World” will logically work. When calculating the score, the system will check the number reported by the imaging system. This value will be graded against the next number associated with the current number as seen in Table 5. If the values equate, the system will set the current number.

ASSERT		**EXPECT**
Number	**Current Number**	
1	0	1
20	0	0
3	17	3
20	5	20 & WINNER

Table 6: "Around the World" Unit Tests

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

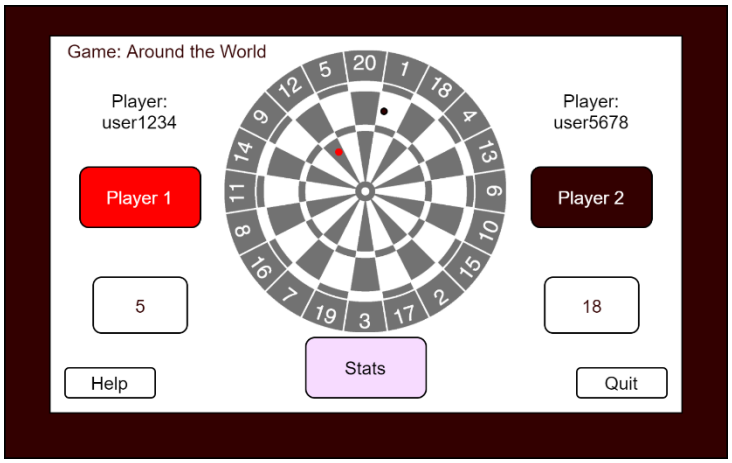

Figure 36: "Around the World" Scoreboard

Figure 36 shows what the “Around the World” scoreboard will look. Scores will update after each throw, displaying the next number the player needs to hit.

Interface Description

The dart scoring system will use three main interface standards to communicate across subsystems. The first standard will be CSI which is a specification of MIPI. CSI defines an interface between a processor and camera. The second will be TCP which is an IP standard for communication within a network. The third will be USB which is a serial standard for peripheral communication.

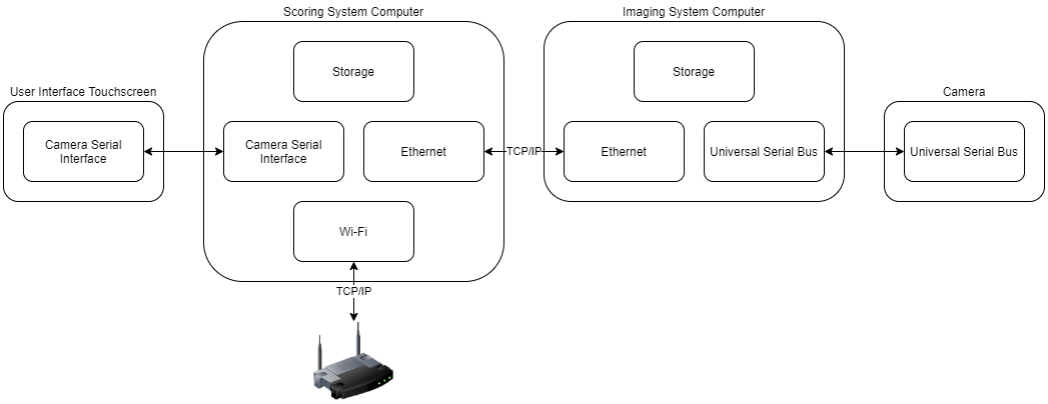

Figure 37: Interface Diagram

Figure 37 shows how subsystems will communicate across the system. The system will use a USB camera with the imaging system for easier manipulation of picture and physical mount location.

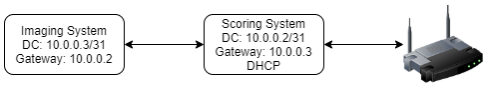

Figure 38: Network Diagram

Figure 38 shows the network for the dart scoring system. The scoring system will be directly connected to the imaging system rather than going through a switch. The scoring system will also be on the ISP subnet of the household. DHCP will be used to assign an IP address to the scoring system, allowing remote access from any household device.

Data Type	**Variable Name**
float	radius
float	angle
int	number
str	ring

Table 7: LOCATION Message

Table 7 shows the contents of the dart location structure the imaging system will populate once a dart is identified. This data will be sent to the scoring system for game progression.

Sender	**Receiver**	**Purpose**	**Content**
Scoring	Imaging	Image processing	“LOOK”
Imaging	Scoring	Dart location	LOCATION message

Table 8: Communication Matrix

Table 8 shows how message content will flow throughout the system. The imaging system will act as the server while the scoring system acts as the client in the TCP communication scheme. Port 1000 will be used for traffic between the two systems. The system will use SCPI-like syntax to initiate state triggers.

Imaging System

The imaging system will use USB to communicate with the offboard camera. This will be the main interface for acquiring images of the dartboard. The system will use TCP/IP over Ethernet to communicate with the scoring system. This will allow the system to receive messages to look for an incoming dart. The system will transmit dart location.

Scoring System

The scoring system will use CSI to communicate with the user interface touchscreen. This will be the main interface for user interaction with the application. The system will use TCP/IP over Ethernet to communicate with the imaging system and TCP/IP over Wi-Fi to communicate with home server.

Material Requirements

The dart scoring system will be comprised of two main computing systems.

The scoring system will run 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 will be used to communicate with a touchscreen user interface.

The imaging system will run 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
Jetson Nano Power Supply	$13.68	https://www.amazon.com/5V-Power-Supply-Adapter-Universal/dp/B07RTWD725?th=1
TOTAL	$311.05	

Table 9: Bill of Materials

Resource Requirements

Development for the dart scoring system will occur in the Python language. Python was chosen due to the amount of FOSS libraries and examples that will be referenced during project development.

VS Code will be 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

Table 10: VS Code Extensions

Table 10 shows a list of VS Code extensions that will be 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	Database development	sqllite3	https://docs.python.org/3/library/sqlite3.html
Scoring	GUI development	pyqt5	https://pypi.org/project/PyQt5/
Scoring	Graphics	turtle	https://docs.python.org/3/library/turtle.html
Scoring	TCP/IP communication	socket	https://docs.python.org/3/library/socket.html
Imaging	Computer vision	opencv-python	https://pypi.org/project/opencv-python/
Imaging	Machine learning	tensorflow	https://pypi.org/project/tensorflow/
Imaging	TCP/IP communication	socket	https://docs.python.org/3/library/socket.html

Table 11: Python Libraries

Table 11 shows a list of Python libraries that will be used to implement the dart scoring system. These libraries will accelerate design implementations.

Application	Reason
VNC Viewer	Remote Raspberry Pi/Jetson Nano access
Wireshark	TCP/IP packet observation

Table 12: 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 will be referenced to speed up development and integration of this project.

The following projects will be 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 will be 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 will take place 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 will be used to plan and track progress throughout development. Projects allow milestones to be tracked with related issues. Issues within Projects will be 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

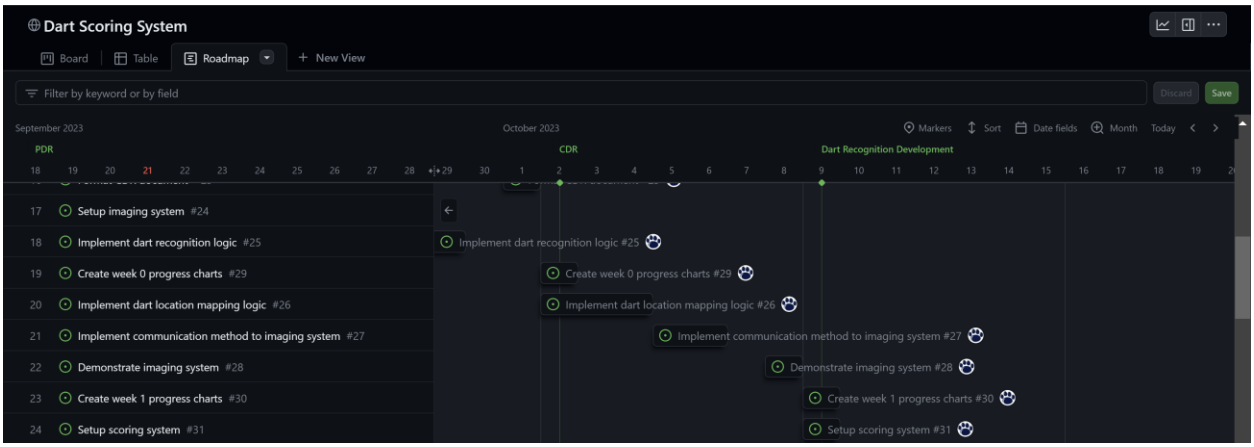

Figure 39: Roadmap Example

Figure 39 shows a snapshot of the project roadmap in GitHub. Milestones are represented as vertical lines traversing the plan. Issues will be 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 13: Milestone Schedule

Risk

	Low	Impact		High
Very Likely				
Likelihood		**1**		
			2	
				3
Not Likely				

Table 14: 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

2. Machine learning for image recognition likely to be difficult

Mitigation: Three weeks built-in for image recognition milestone; use known trained datasets from online repositories to feed solution

3. Consistent lighting and angle viewpoints for imaging system likely non-deterministic

Mitigation: Implement calibration routines that will run when imaging system is powered on

References

https://developer.nvidia.com/embedded/jetson-nano-developer-kit

https://www.raspberrypi.com/news/coding-on-raspberry-pi-remotely-with-visual-studio-code/

https://www.dimensions.com/element/dartboard

https://www.mathsisfun.com/polar-cartesian-coordinates.html

https://www.lucidchart.com/pages/database-diagram/database-design

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