Digital
Accuracy
Real-time
Tracking and Scoring
System

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 players 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. 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 used for statistical/performance purposes
- Scoring System
    - Capable of updating displays after each dart is thrown and identified
    - Capable of pulling statistics remotely
        - Win %
        - Hit % by number
        - Hit % by double ring, triple ring, bull, bullseye

Limitations

System capabilities were 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 only
    - Limited to two games
        - Framework available for future games
    - Limited to two-players per game

Functional Description

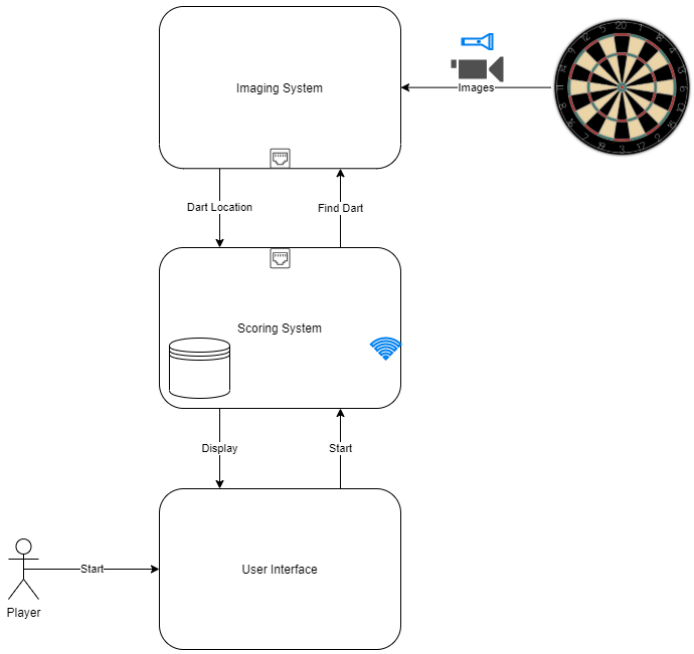

Figure 2: Functional Block Diagram

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 the dart, and report the 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] project and tools from Nvidia.

Configuration

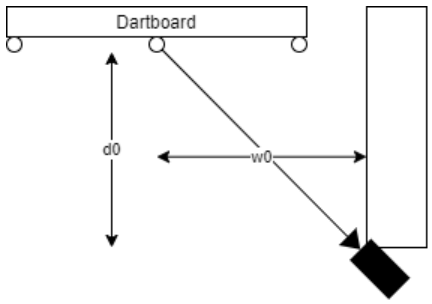

Figure 3: Dartboard Setup

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 $d0$ of 18 inches and $w0$ of 14.5 inches from the center of the dartboard.

Figure 4: 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] project provides tools to properly capture, train, and detect custom objects.

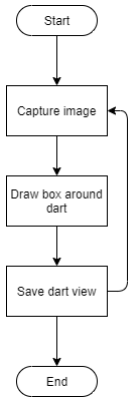

Figure 5: Training Logic

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] machine learning architecture 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.

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 frame.

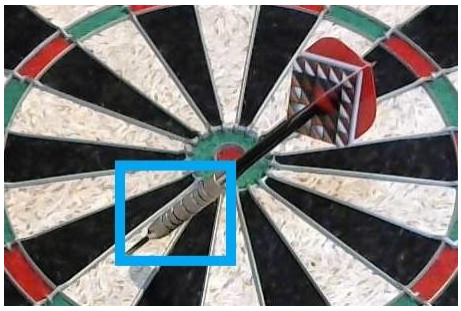

Figure 6: Object Detection Box

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 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 7: Dartboard Source Features

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

$$top_{src} = x, y - \left(\frac{a}{2}\right)$$

Equation 1: Top Source Equation

$$center_{src} = x, y - factor$$

Equation 2: Center Source Equation

$$bottom_{src} = x, y + \left(\frac{a}{2}\right)$$

Equation 3: Bottom Source Equation

$$left_{src} = x - \left(\frac{b}{2}\right), y - factor$$

Equation 4: Left Source Equation

$$right_{src} = x + \left(\frac{b}{2}\right), y - factor$$

Equation 5: 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 8: Dartboard Destination Features

Figure 8 shows the creation of features on the destination image. Fitting a circle yields the origin (x, y) and radius (r) in the pixel frame.

$$top_{dest} = x, y - r$$

Equation 6: Top Destination Equation

$$center_{dest} = x, y$$

Equation 7: Center Destination Equation

$$bottom_{dest} = x, y + r$$

Equation 8: Bottom Destination Equation

$$left_{dest} = x - r, y$$

Equation 9: Left Destination Equation

$$right_{dest} = x + r, y$$

Equation 10: Right Destination Equation

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

OpenCV – Homography is used to compute the homographic translation matrix. The source and destination points are used to form input arrays for the matrix computation.

$$src = np.array([top_{src}, center_{src}, bottom_{src}, left_{src}, right_{src}])$$

Equation 11: Source Array Equation

$$dest = np.array([top_{dest}, center_{dest}, bottom_{dest}, left_{dest}, right_{dest}])$$

Equation 12: Destination Array Equation

$$H = cv2.findHomography(src, dest)$$

Equation 13: Homographic Matrix Equation

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

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = H \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Equation 14: Translation Equation

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

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.

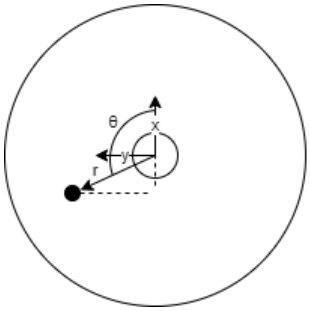

Figure 9: Dartboard Polar Coordinates

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.

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

Equation 15: Radius Equation

Equation 15 shows how the radius is computed from x and y distances. Pythagoras Theorem 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):

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

Equation 16: Quadrant 1 Angle Equation

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

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

Equation 17: Quadrant 2 Angle Equation

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

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

Equation 18: Quadrant 3 Angle Equation

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

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

Equation 19: 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 the quadrant hit.

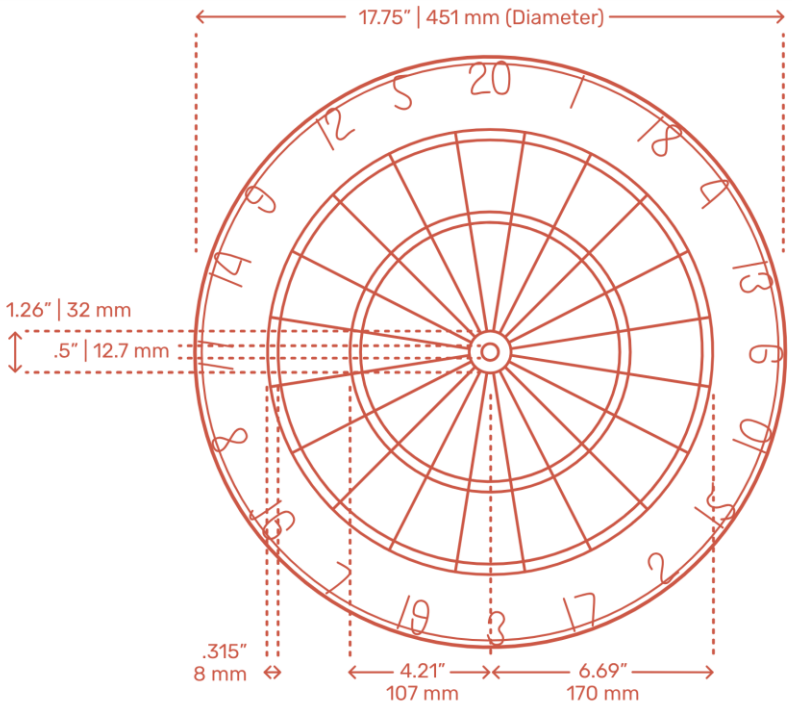

Figure 10: Standard Dartboard Distances

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 the ring 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 1: Dartboard Ring Mapping

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

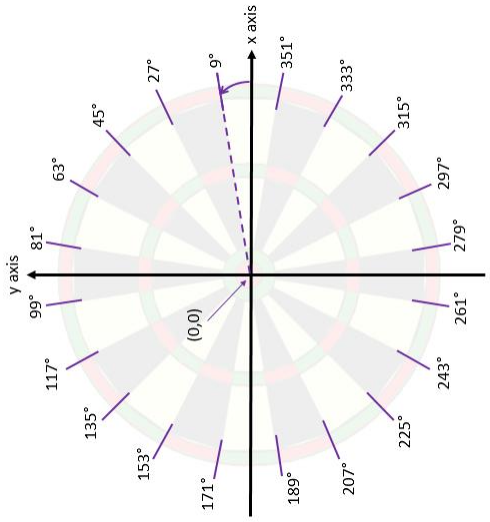

Figure 11: Standard Dartboard Angles

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. The camera is mounted 90 degrees from the top of the dartboard meaning number 11 is in line the camera angle.

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 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

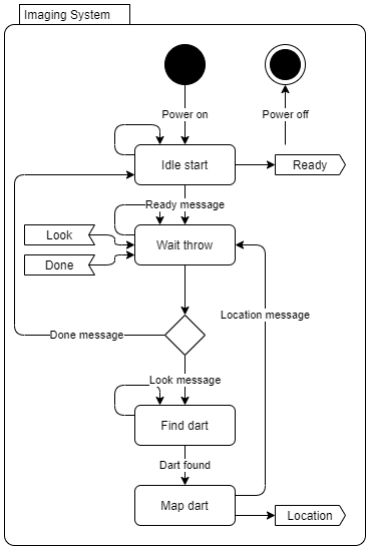

Figure 12: Imaging System State Machine Diagram

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

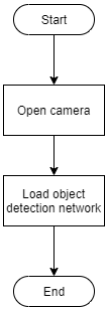

Figure 13: Imaging System Startup Logic

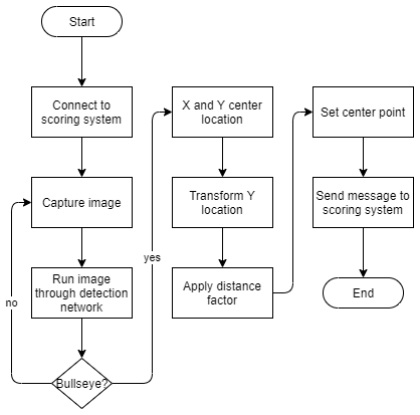

Figure 14: Imaging System IDLE START Logic

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

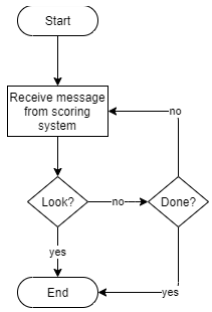

Figure 15: Imaging System WAIT THROW Logic

2. 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 the system receives a DONE message is received, it will transition to the IDLE START state.

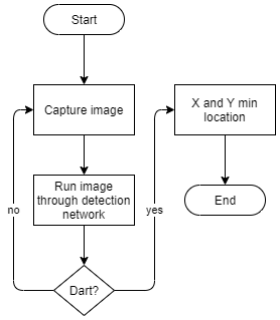

Figure 16: Imaging System FIND DART Logic

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.

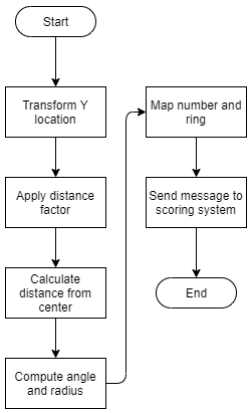

Figure 17: Imaging System MAP DART Logic

4. In the MAP DART state, the system will map the image location of the dart to a hit point on the board. The system will then send a LOCATION 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.

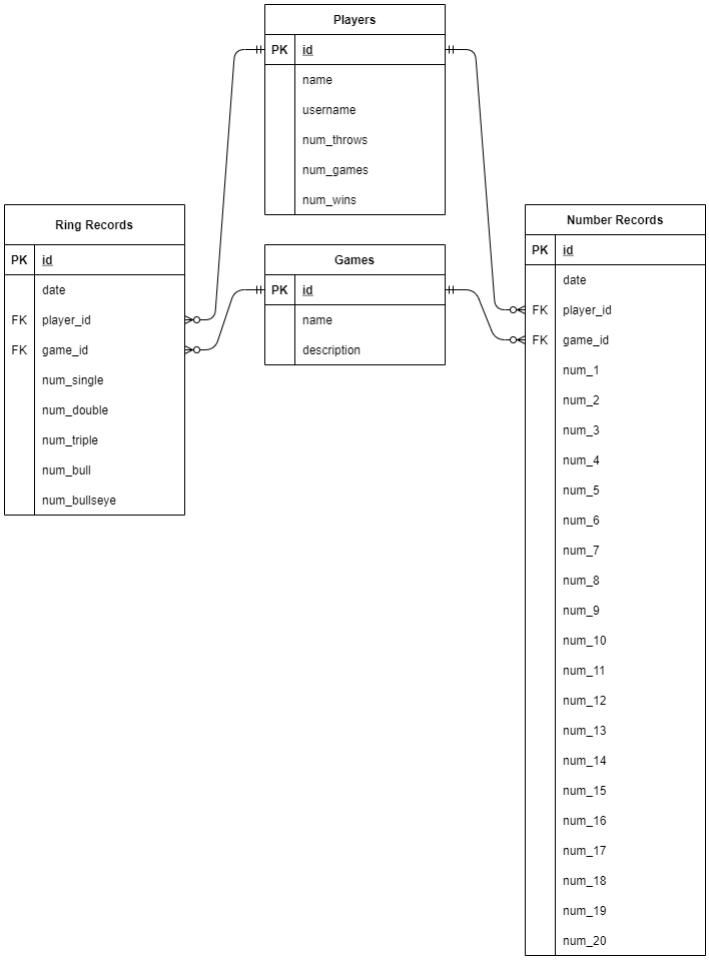

Figure 18: Scoring System Database Entity Relationship

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.

Gameplay

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

Table 3: Hosted Games

The system is capable of hosting scoring and knockout games. Table 3 shows what dart games are available. The system provides a framework for implementing additional dart games.

"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 number. 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.

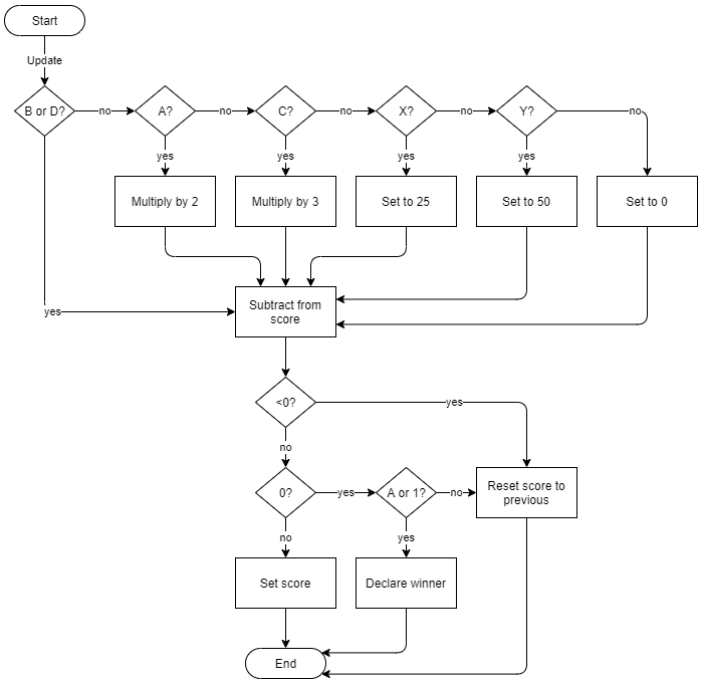

Figure 19: "501" Game Logic

The system uses mappings as seen in Table 1 to determine multipliers and scores for tallying player scores throughout the game. Figure 19 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 4: "501" Unit Tests

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

"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

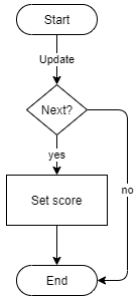

Figure 20: "Around the World" Game Logic

The system uses a mapping technique to check the current and next number to determine sequential knockout of the dartboard. Figure 20 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 6: "Around the World" Unit Tests

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

State Machine

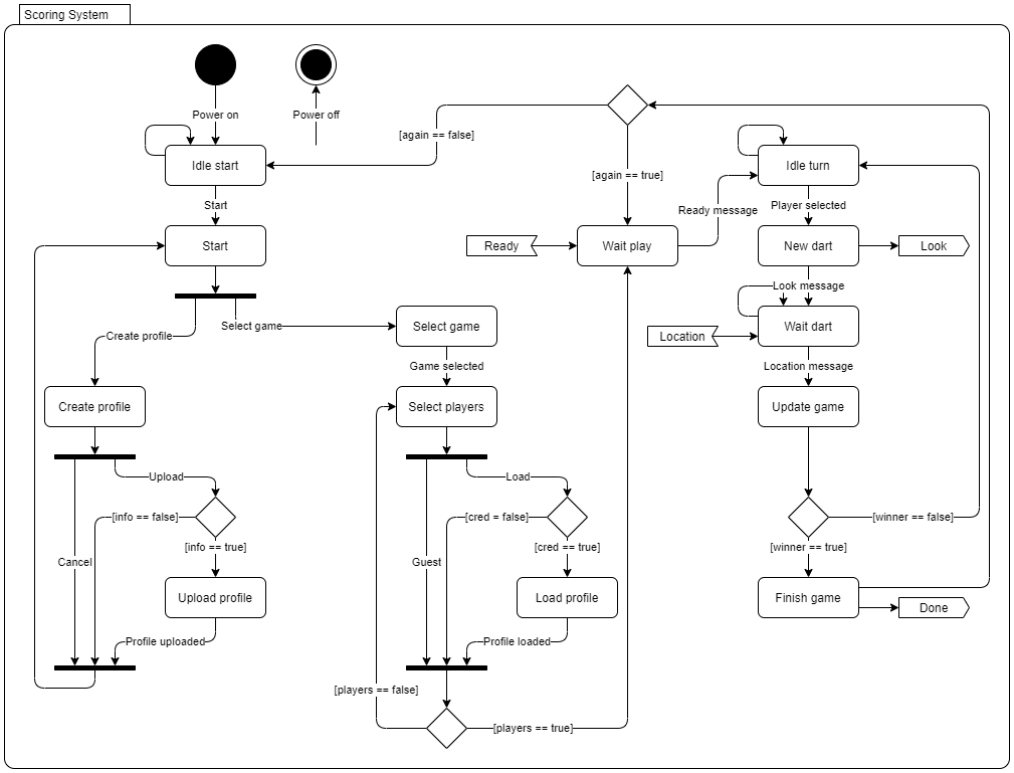

Figure 21: Scoring System State Machine Diagram

Figure 21 shows how states flow within the Scoring System. This system is comprised of game and user interface logic. The state machine functions as follows:

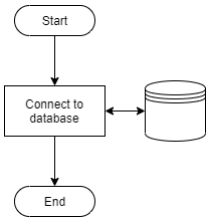

Figure 22: Scoring System Startup Logic

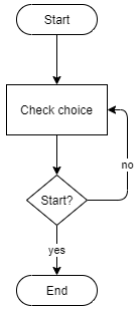

Figure 23: Scoring System IDLE START Logic

1. After power-up, the application will open, connect to the database, and enter the IDLE START state. The system will remain in this state until the player starts the application. Once started, the system will transition to the START state.

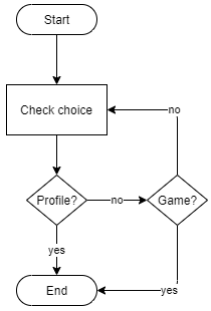

Figure 24: Scoring System START Logic

2. In the START state, the system will wait for the player to choose to create a profile or select a game. Once chosen, the system will transition to the appropriate state.

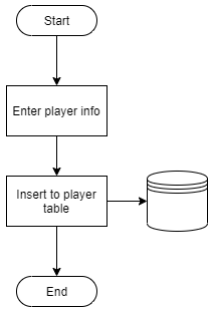

Figure 25: Scoring System CREATE PROFILE Logic

3. If the player chooses to create a profile, the system will enter the CREATE PROFILE state. Players will enter information to upload a profile or cancel the request. Upon uploading a profile, the system will perform an insert query to the database. The system will then transition to the START state.

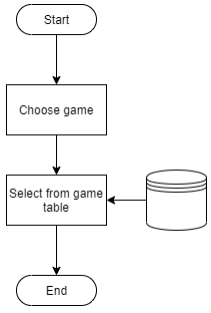

Figure 26: Scoring System SELECT GAME Logic

4. If the player chooses to select a game, the system will enter the SELECT GAME state. Players will choose a game from a pre-defined list or cancel the game. Upon choosing a game, the system will perform a select query from the database. The system will then transition to the SELECT PLAYERS state.

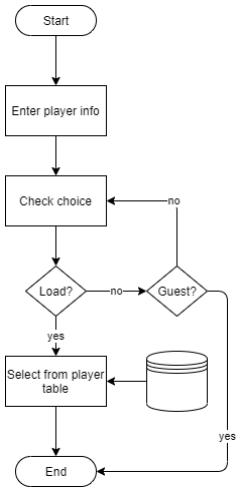

Figure 27: Scoring System SELECT PLAYERS Logic

5. The system will remain in the SELECT PLAYERS state until the desired number of players are ready for the game specified. Players can play as guests or load existing profiles. Upon loading a profile, the system will perform a select query from the database. Once players are specified, the system will transition to the WAIT PLAY state.

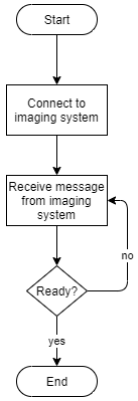

Figure 28: Scoring System WAIT PLAY Logic

6. In the WAIT PLAY state, the system will connect to the Imaging System. The system will remain in this state until a message is received from the Imaging System. Once a READY message is received, the system will transition to the IDLE TURN state.

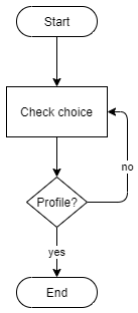

Figure 29: Scoring System IDLE TURN Logic

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

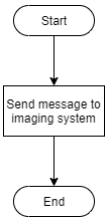

Figure 30: Scoring System NEW DART Logic

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

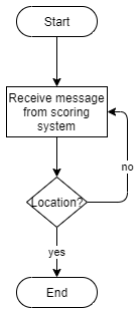

Figure 31: Scoring System WAIT DART Logic

9. 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. Once a LOCATION message is received, the system will transition to the UPDATE GAME state.

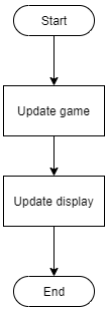

Figure 32: Scoring System UPDATE GAME Logic

10. In the UPDATE GAME state, the system will update player scores based on the game selected. If the system determines that a player has won, the system will transition to the FINISH GAME state otherwise the system will transition to the IDLE TURN state.

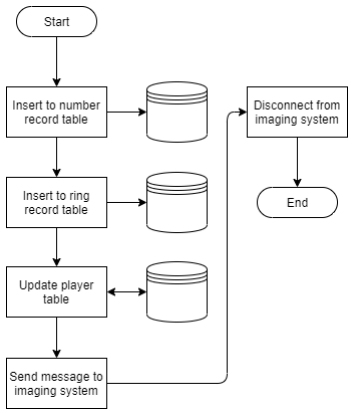

Figure 33: Scoring System FINISH GAME Logic

11. In the FINISH GAME state, the system will perform an insert and update query to the database. The system will then send a DONE message to the Imaging System indicating the game is finished and disconnect. After disconnection, the system will check if the player(s) want to play again. If yes, the system will transition to the WAIT PLAY state otherwise 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 21.

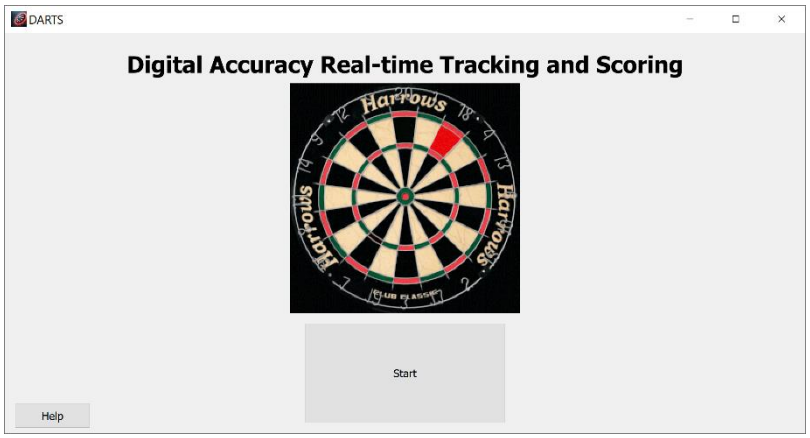

Figure 34: Idle Start Display

Figure 34 shows what displays when the application is started and in the IDLE START state. ‘Help’ shows directions on how to start. ‘Start’ transitions the system to the START state and opens a new window.

Figure 35: Start Display

Figure 35 shows what displays after when in the START state. ‘Help’ shows directions on what each button option does. ‘Create Profile’ transitions the system to the CREATE PROFILE state and opens a new window. ‘Select Game’ transitions the system to the SELECT GAME state and opens a new window. ‘Cancel’ transitions the system to the IDLE START state and returns to the display as seen in Figure 34.

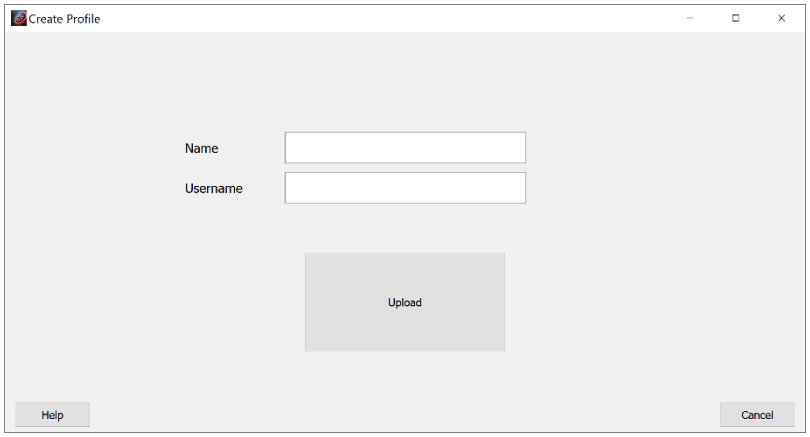

Figure 36: Create Profile Display

Figure 36 shows what displays when in the CREATE PROFILE state. ‘Help’ shows directions on credential input. ‘Upload’ follows logic as seen in Figure 25, transitions the system to the START state, and returns to the display as seen in Figure 35. ‘Cancel’ transitions the system to the START state and returns to the display as seen in Figure 35.

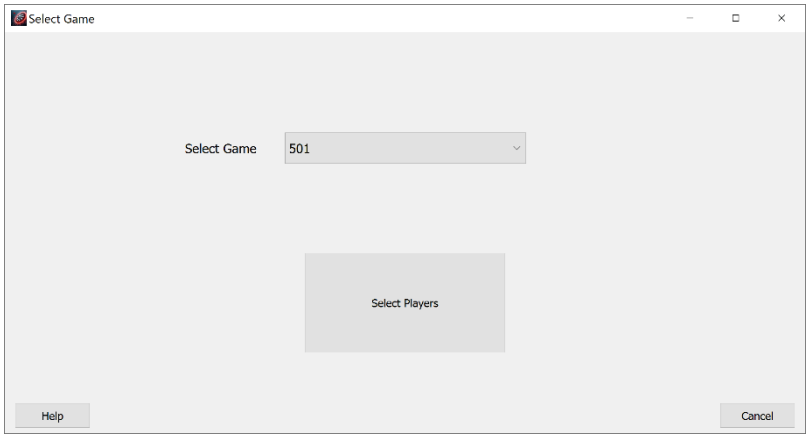

Figure 37: Select Game Display

Figure 37 shows what displays when in the SELECT GAME state. ‘Help’ shows summaries of each game. ‘Select Players’ follows logic as seen in Figure 26, transitions the system to the SELECT PLAYERS state, and opens a new window. ‘Cancel’ transitions the system to the START state and returns to the display as seen in Figure 35.

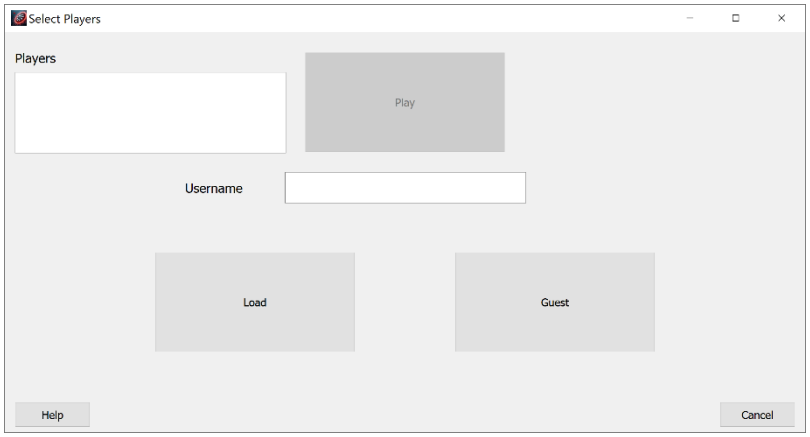

Figure 38: Select Players Display

Figure 38 shows what displays when in the SELECT PLAYERS state. ‘Help’ shows directions on what each button option does. ‘Load’ and ‘Guest’ follow logic as seen in Figure 27. ‘Play’ is only available when the minimum number of players for the specified game has been met. ‘Play’ follows logic as seen in Figure 28, transitions the system to the WAIT PLAY state, and opens a new window. ‘Cancel’ transitions the system to the SELECT GAME state and returns to the display as seen in Figure 37.

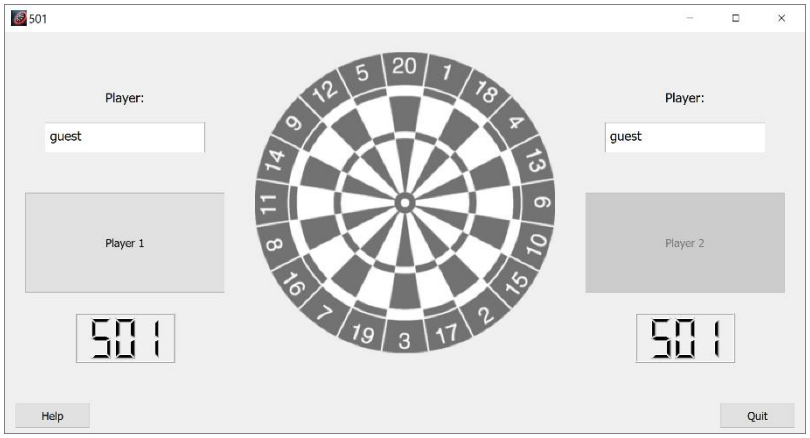

Figure 39: “501” Scoreboard Display

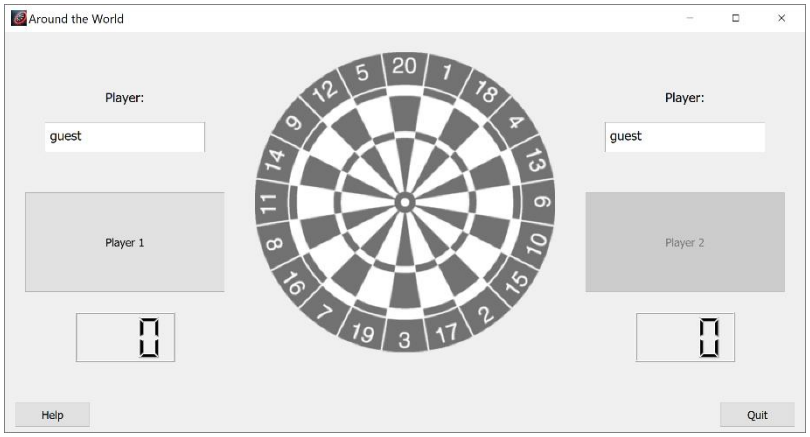

Figure 40: "Around the World" Scoreboard Display

Figure 39 and Figure 40 show the scoreboards for the games available when in the IDLE TURN and subsequent states. The dartboard appears in grayscale to see hit markers. Hit markers display analogous to the player throwing (red or black). Usernames appear above the player button for the game. Scores appear below the button.

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.

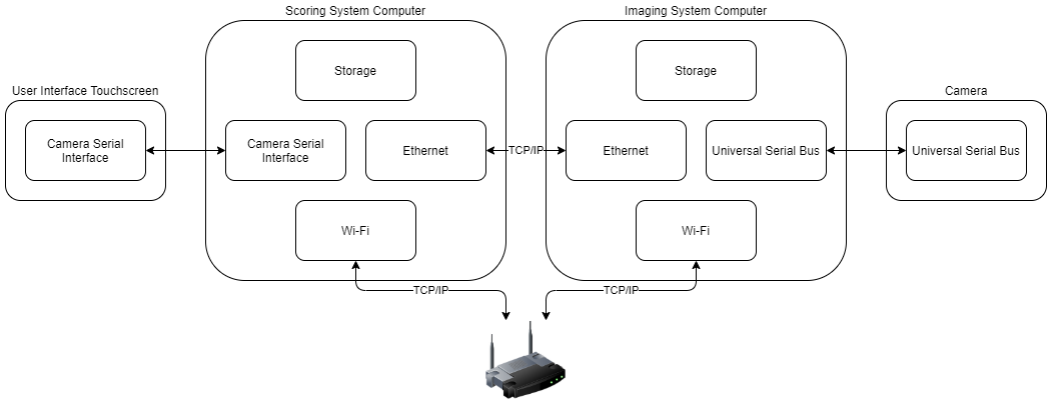

Figure 41: Interface Diagram

Figure 41 shows how subsystems communicate across the 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.

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 the home server.

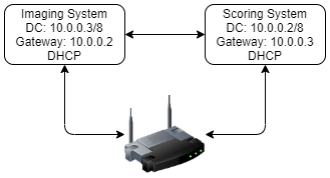

Figure 42: Network Diagram

Figure 42 shows the DARTS network. The Scoring System is directly connected to the Imaging System via Ethernet. Both the Imaging and Scoring Systems 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”
Scoring	Imaging	Game finished	“DONE”
Imaging	Scoring	Dart location	LOCATION message

Table 7: 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 8: 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.

Material Requirements

DARTS is comprised of two main computer 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 9: 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 were 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 10: VS Code Extensions

Table 10 shows a list of VS Code extensions that were used for faster system development. These extensions allowed 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	Web server	uwsgi	https://uwsgi-docs.readthedocs.io/en/latest/Install.html
Imaging	Object detection and training	*	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 11: 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 is required and installs several dependencies for image detection and training. Libraries not listed but used in the code are assumed to be available within the standard Python 3 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
Qt Designer	GUI development

Table 12: Third-Party Applications

Table 12 shows a list of third-party applications that were used during development for debugging purposes and system development.

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

The following projects/resources were referenced for GUI and database development:

- https://www.sqlitetutorial.net/sqlite-python/
- https://opensource.com/article/21/3/web-hosting-raspberry-pi

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 underwent CM in an online repository which can be accessed from the following link:

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

GitHub Projects was used to plan and track progress throughout development. Projects allowed 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

Milestones and Schedule

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

Planned Date	**Milestone**	**Realized Date**
10/2 – 10/23	Dart Recognition Development	10/2 – 11/4
10/23 – 11/6	Scoring and Game Development	11/4 – 11/10
11/6 – 11/13	Database Development	11/10 – 11/14
11/13 – 12/4	User Interface Development	11/14 – 12/4

Table 13: Milestone Schedule

Table 13 shows a comparison of the planned milestone timelines compared to the realized timelines. The Dart Recognition Development milestone took the longest to complete due to data collection efforts, training time for the image recognition model, and availability of camera hardware. The additional time needed for this milestone was made up over the next three milestones. Ultimately, all planned features of DARTS were completed by the completion date of 12/4.

Risk

	Low	Impact			High
Very Likely					
Likelihood					
				2	
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. Consistent lighting and angle viewpoints for Imaging System likely non-deterministic

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

The lighting for the Imaging System remains a risk for this project. The data trained for the dart recognition model does not have a variety of background lighting. This leads to inconsistent results when exposing the camera and dartboard to different brightness levels.

The system has two additional risks that were burned down during development:

- Loss of home internet connectivity

The internet connectivity risk no longer applies since the Scoring and Imaging Systems communicate via direct-connect Ethernet.

- Machine learning for image recognition likely to be difficult

The machine learning effort risk was burned down following completion of the Dart Recognition Development milestone.

Assembly

Getting Started With Jetson Nano Developer Kit should be referenced for introduction to the Nvidia Jetson Nano Developer Kit.

Jetson Nano Case should be referenced for construction of Jetson casing and subcomponents.

Getting started with your Raspberry Pi should be referenced for introduction to the Raspberry Pi.

Freenove Touchscreen Monitor for Raspberry Pi should be referenced for assembly of Raspberry Pi touchscreen.

Refer to https://github.com/kparlak/dart-scoring-system/blob/main/imaging/README.md for instructions on installing necessary software packages for the Imaging System.

Refer to https://github.com/kparlak/dart-scoring-system/blob/main/scoring/README.md for instructions on installing necessary software packages for the Scoring System.

Gameplay

1. Power on the Jetson Nano
2. Log in and navigate to the 'imaging' folder of the repository
3. Execute the setup script (*. setup.sh*)
4. Execute the main program (*./main_imaging.py*)
5. Power on the Raspberry Pi
6. Log in and navigate to the 'scoring' folder of the repository
7. Execute the main GUI (*./main_scoring_gui.py*)

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

Performance

Imaging System

Refer to https://github.com/kparlak/dart-scoring-system/tree/main/imaging/demo for video demonstrations of the Imaging System.

Figure 43: Dart and Bull Detection

Figure 43 shows a snapshot of the Imaging System detecting the dart and bull. The system draws a detection box around each item corresponding to coordinates in the pixel frame.

Scoring System

Refer to https://github.com/kparlak/dart-scoring-system/tree/main/scoring/demo for video demonstrations of the Scoring System.

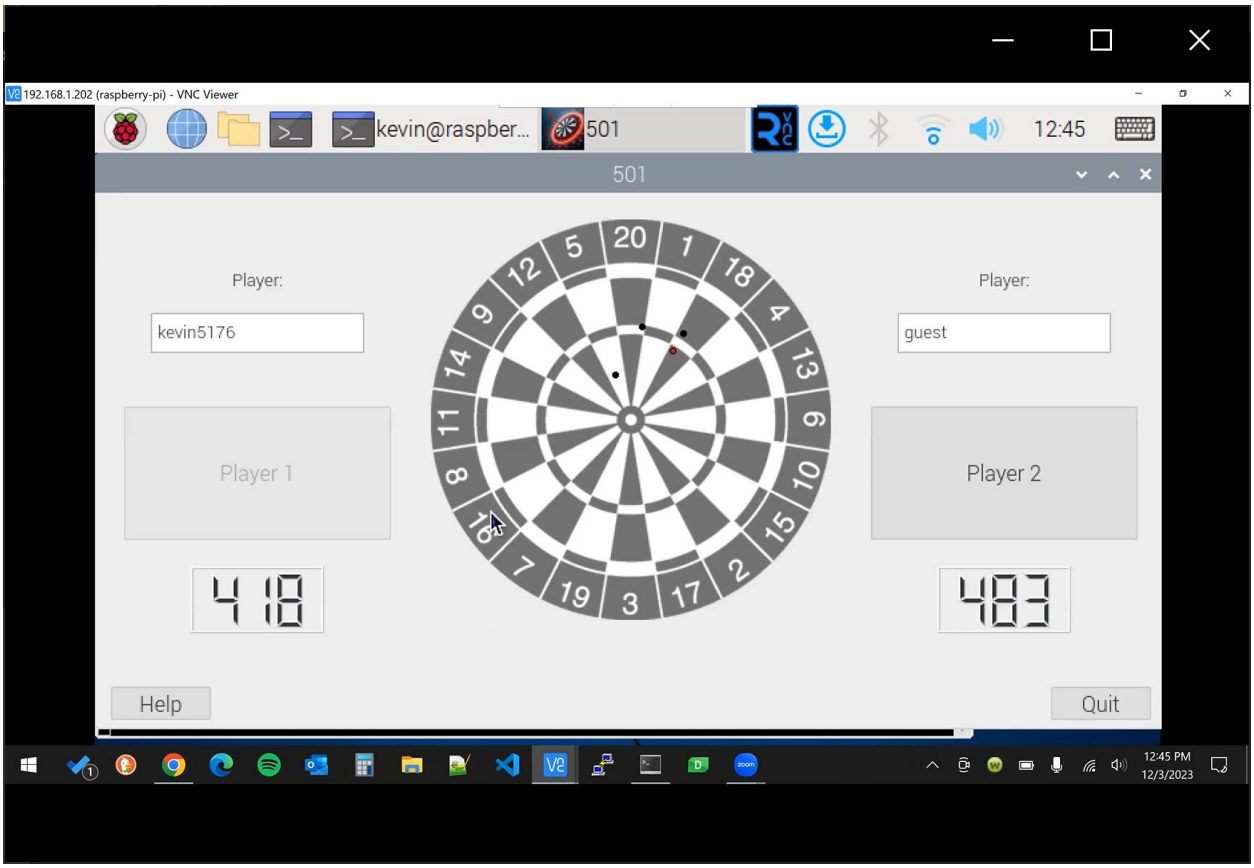

Figure 44: Scoreboard Update

Figure 44 shows a snapshot of the Scoring System during gameplay. The system updates the player buttons and scores based on the defined number of turns for the game being played. The hit map updates by drawing black or red dots of the hit location following an update from the Imaging System.

Database/Web Server

Players report

Name	Username	Win	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Single	Double	Triple	Bull	Bullseye
Kevin Parlak	kevin5176	0.0%	0.0%	0.0%	0.0%	0.0%	33.33%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	33.33%	0.0%	33.33%	66.67%	0.0%	33.33%	0.0%	0.0%

Figure 45: Player Statistics

Figure 45 shows player statistic results from queries of the hosted database. Hit percentages per number and ring update following the completion or ending of a game.

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://www.python.org/download/releases/3.0/

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

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

https://www.raspberrypi.com/documentation/computers/getting-started.html

https://github.com/Freenove/Freenove_Touchscreen_Monitor_for_Raspberry_Pi

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
DARTS	Digital Accuracy Real-time Tracking and Scoring System
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.