Dart Scoring System

Critical Design Review

EN.525.743 – Embedded Systems Development Laboratory

Kevin Parlak

Contents

[Figures 3](#_Toc146184323)

[Tables 3](#_Toc146184324)

[Project Description 4](#_Toc146184325)

[Problem 4](#_Toc146184326)

[Solution 4](#_Toc146184327)

[Capabilities 4](#_Toc146184328)

[Limitations 5](#_Toc146184329)

[Functional Description 6](#_Toc146184330)

[Scoring System 6](#_Toc146184331)

[“501” Game Logic 7](#_Toc146184332)

[“Around the World” Game Logic 7](#_Toc146184333)

[Dartboard Mapping Logic 7](#_Toc146184334)

[State Machines 8](#_Toc146184335)

[Imaging System 10](#_Toc146184336)

[State Machine 10](#_Toc146184337)

[Mobile App 11](#_Toc146184338)

[Interface Description 12](#_Toc146184339)

[Material Requirements 13](#_Toc146184340)

[Resource Requirements 14](#_Toc146184341)

[Development Plan and Schedule 15](#_Toc146184342)

[Approach 15](#_Toc146184343)

[Milestones and Schedule 15](#_Toc146184344)

[Risk 15](#_Toc146184345)

[References 16](#_Toc146184346)

[Appendix 17](#_Toc146184347)

[Acronyms 17](#_Toc146184348)

# Figures

[Figure 1: Home Dartboard 4](#_Toc146184309)

[Figure 2: Functional Block Diagram 6](#_Toc146184310)

[Figure 3: Dartboard Scoring 7](#_Toc146184311)

[Figure 4: Scoring System State Machine Diagram 8](#_Toc146184312)

[Figure 5: Imaging System State Machine Diagram 10](#_Toc146184313)

# Tables

[Table 1: Bill of Materials 13](#_Toc146184266)

[Table 2: VS Code Extensions 14](#_Toc146184267)

[Table 3: Python Libraries 14](#_Toc146184268)

[Table 4: Third-Party Applications 14](#_Toc146184269)

# 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/mobile application 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
* Imaging system shall be capable of re-calibration after each game
  + Allows dartboard imaging corrections to occur each time a game is played

## 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
* Mobile application shall only be usable on Android devices

# Functional Description

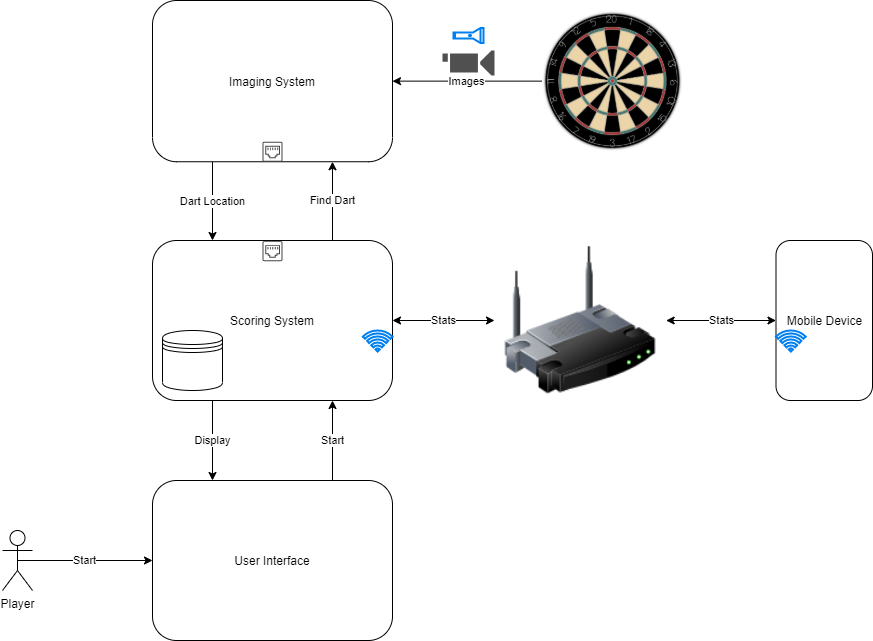


Figure 2: Functional Block Diagram

Figure 2 illustrates how the dart scoring system will functionally 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 scoring system
6. Scoring system will update game and player statistics from throw
7. Player(s) may query statistics from mobile app

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

## 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 is part of the scoring subsystem.

The scoring system will be capable of hosting two dart games. The first is a score-based game called “501”. The second is a knockout-based game called “Around the World”.

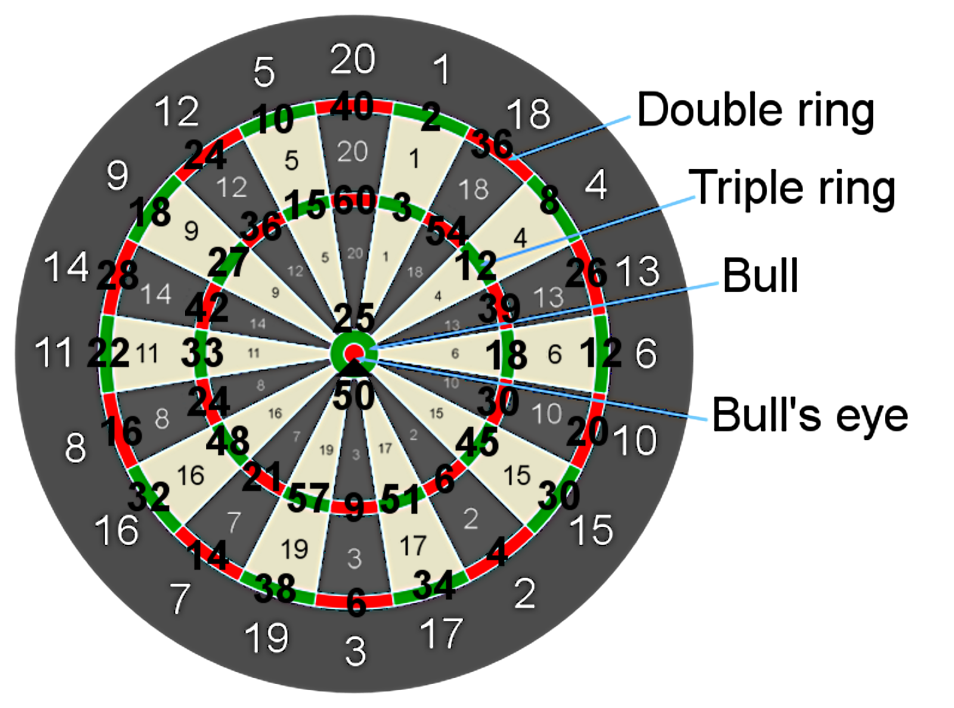


Figure 3: Dartboard Scoring

### “501” Game Logic

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 all three darts, sum the total, and subtract from 501. To win, a player must hit a double to reach 0 even. Referencing Figure 3, if a player has 20 remaining, they need to hit a double 10 to win.

TODO : “501” flow diagram

### “Around the World” Game Logic

The goal of “Around the World” is to hit each section of the dartboard except for the outer and inner bullseye. Players throw all three darts and start at position 1, moving clockwise around the board. Players must hit sections in sequential order to move on. Referencing Figure 3, if a player hits 1, 4, 18 then the player is only on 4 because 18 was the last section hit. Inner and outer bullseye yield nothing. Double and triple rings only count as singles toward the section hit.

TODO : “Around the World” flow diagram

### Dartboard Mapping Logic

TODO : Flow diagram

### State Machines

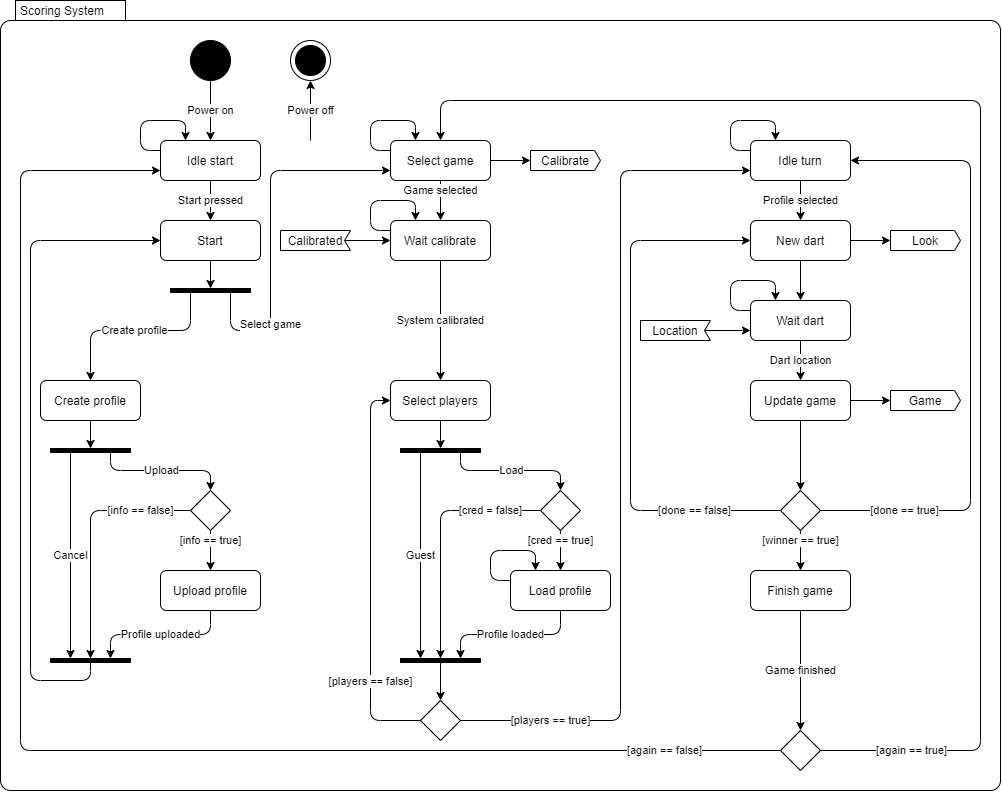


Figure 4: Scoring System State Machine Diagram

Figure 4 shows how states will flow within the scoring system. This subsystem will be comprised of game and user interface logic. The scoring system state machine will function as follows:

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.

TODO : IDLE START flow diagram

1. The system will remain in the START state until the player creates a profile or selects a game.

TODO : START flow diagram

1. 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.

TODO : CREATE PROFILE flow diagram

* 1. Upon upload, the system will check to ensure information is filled out and enter the UPLOAD PROFILE state if valid. The system will remain in this state until data is uploaded to the database.

TODO : UPLOAD PROFILE flow diagram

1. If the player selects a game, the system will enter the SELECT GAME state. In this state, the system will send a request to the imaging system to calibrate itself and enter the WAIT CALIBRATE state.

TODO : SELECT GAME flow diagram

1. The system will remain in the WAIT CALIBRATE state until a message is received from the imaging system indicating that it is calibrated. Upon receipt of this message, the system will enter the SELECT PLAYERS state.

TODO : WAIT CALIBRATE flow diagram

1. 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. After the appropriate number of players are selected, the system will enter the IDLE TURN state.

TODO : SELECT PLAYERS flow diagram

* 1. Upon upload, the system will check to ensure credentials are filled out and enter the LOAD PROFILE state if valid. The system will remain in this state until data is loaded from the database.

TODO : LOAD PROFILE flow diagram

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

TODO : IDLE TURN flow diagram

1. In the NEW DART state, the system will send a request to the imaging system indicating a dart is incoming and enter the WAIT DART state.

TODO : NEW DART flow diagram

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

TODO : WAIT DART flow diagram

1. In the UPDATE GAME state, the system will update player throw statistics and scores based on the game selected. The system will also send a message to the imaging system of the state of the game. 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 the appropriate number of darts per turn the system will enter the IDLE TURN state, otherwise the system will enter the NEW DART state.

TODO : UPDATE GAME flow

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

TODO : FINISH GAME flow

## Imaging System

The imaging system is the subsystem responsible for identifying and locating darts on a dartboard.

### State Machine

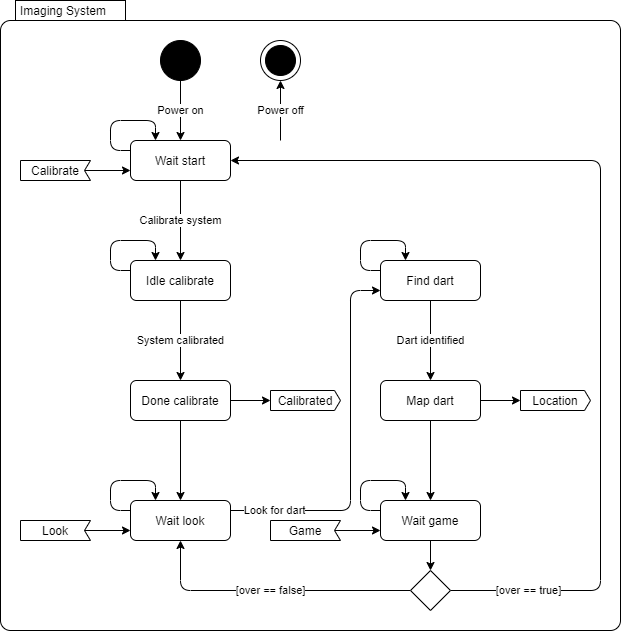


Figure 5: Imaging System State Machine Diagram

Figure 5 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:

1. After power-up the application will open and enter the WAIT START state. The system will remain in this state until the system receives a request to calibrate. After receiving the calibration request the system will enter the IDLE CALIBRATE state.

TODO : WAIT START flow diagram

1. The system will remain in the IDLE CALIBRATE state until proper calibration of the imaging system is complete.

TODO : IDLE CALIBRATE flow diagram

1. Once calibration is completed the system will enter the DONE CALIBRATE state. In this state the system will send a message to the scoring system that calibration is complete. The system will then enter the WAIT LOOK state.

TODO : DONE CALIBRATE flow diagram

1. The system will remain in the WAIT LOOK state until a message from the scoring system is received indicating a dart is incoming. After the message is received the system will enter the FIND DART state.

TODO : WAIT LOOK flow diagram

1. The system will remain in the FIND DART state until the system recognizes a dart has hit the dartboard.

TODO : FIND DART flow diagram

1. Once a dart is recognized the system will enter the MAP DART state. In this state the system will send a message to the scoring system with the coordinates of the dart that was recognized. The system will then enter the WAIT GAME state.

TODO : MAP DART flow diagram

1. The system will remain in the WAIT GAME state until a message from the scoring system is received indicating the status of the game being played. If the game is declared over, the system will enter the WAIT START state. If the game is still active, the system will enter the WAIT LOOK state.

TODO : WAIT GAME flow diagram

## Mobile App

# Interface Description

# 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 CSI connectivity which will be used to communicate with an external camera.

|  |  |  |
| --- | --- | --- |
| **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 |
| SainSmart IMX219 8MP Camera Module | $21.99 | https://www.amazon.com/SainSmart-IMX219-Camera-Module-Raspberry/dp/B07VFFRX4C |
| Jetson Nano Power Supply | $13.68 | https://www.amazon.com/5V-Power-Supply-Adapter-Universal/dp/B07RTWD725?th=1 |
| **TOTAL** | $313.14 |  |

Table 1: 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 2: VS Code Extensions

Table 2 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 3: Python Libraries

Table 3 shows a list of Python libraries that will be used to implement the dart scoring system. These libraries will accelerate design implementations.

Android Studio will be used to develop the mobile application for the dart scoring system. This IDE was chosen because it streamlines GUI development and testing for Android applications all in one software package.

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

Table 4: Third-Party Applications

Table 4 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

## Approach

## Milestones and Schedule

## Risk

# References

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

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

# Appendix

## Acronyms

|  |  |
| --- | --- |
| CSI | Camera Serial Interface |
| FOSS | Free and Open Source Software |
| IDE | Integrated Development Environment |
| SSH | Secure Shell |
| VS | Visual Studio |
| WAP | Wireless Access Point |