Dart Scoring System Preliminary Design Review

EN.525.743 – Embedded Systems Development Laboratory

Kevin Parlak

Agenda

- Project Description
- Capabilities & Limitations
- Functional Description
- Interface Description
- Material & Resource Requirements
- Development Plan
- Risks
- References

Project Description

Problem: Currently own steel-tipped dart board with no means of keeping score

Solution: Utilize computer vision techniques to track where dart lands and automatically update game score/statistics

- Computer vision will find, map, and report dart location
  - Based on known dataset with training as game is played
- User interface will update game scores as dart location is identified
- Database will hold score information and be accessible via touchscreen display or mobile device

Bigger Picture:

- System will be capable of two games at first – expansion of game selection easy once concept is complete
- Machine learning possible based on user data over time – help user get better at playing

Capabilities

- Database shall be capable of holding up to 10 players information
- User interface shall be capable of displaying dart locations live on a dartboard hit map
- Imaging system shall be capable of re-calibrating upon power-up
- User interface/mobile app shall be capable of producing statistics from database entries
  - Hit % by number, win %, double & triple ring, outer & inner bullseye hit %

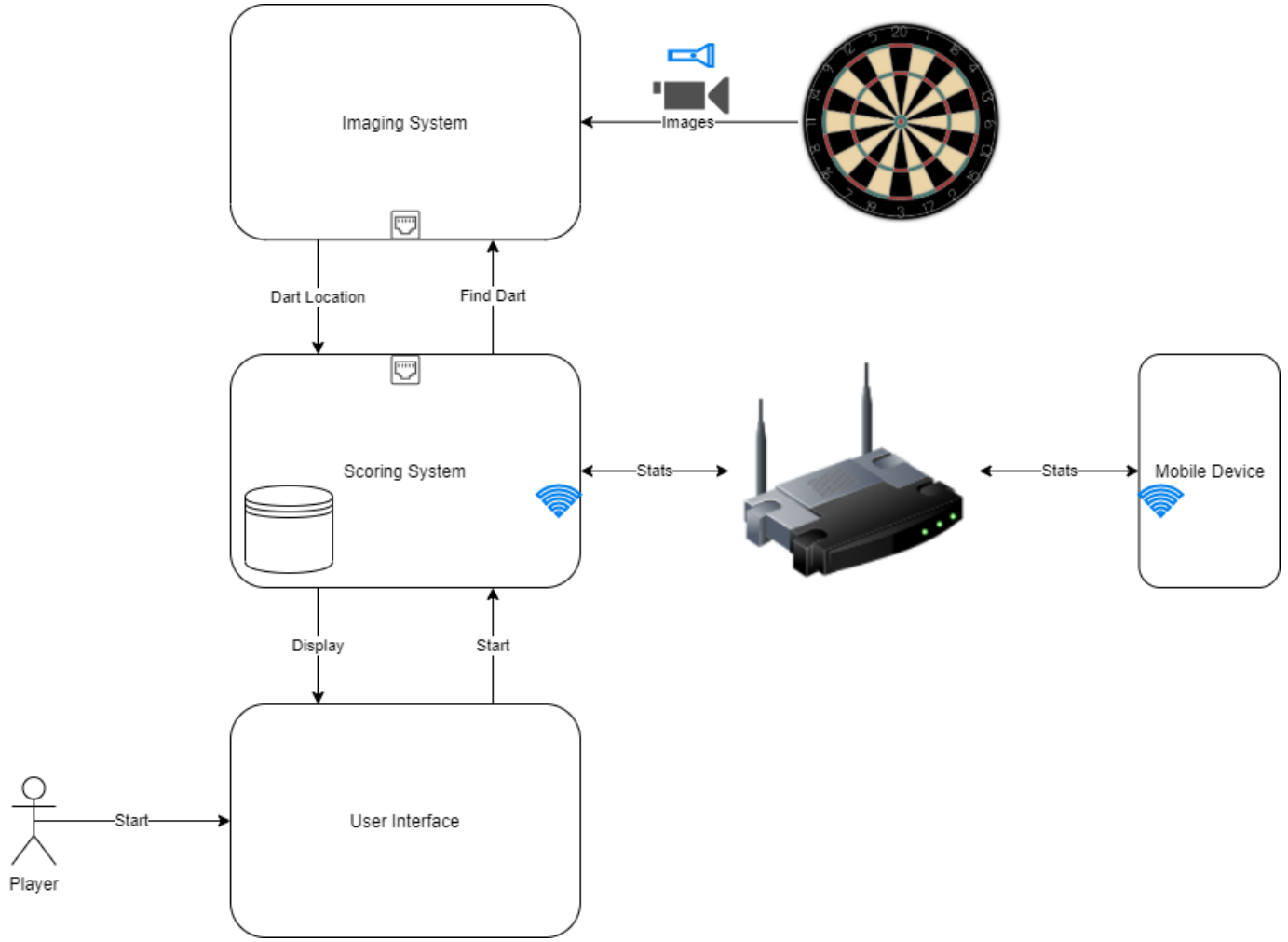
Limitations

- Database shall only be accessible when connected to home network or WAP
- Imaging system shall only be capable of tracking a single set of darts at a time
  - Player to select profile on physical user interface before throwing
- System shall only host two games
  - “501” (score-based), “Around the World” (knockout-based)
- System shall only be capable of starting games from the physical user interface
- System shall have a two-player limit per game
- Database shall only be capable of adding new profiles from physical user interface
- Mobile app shall only be usable on Android devices

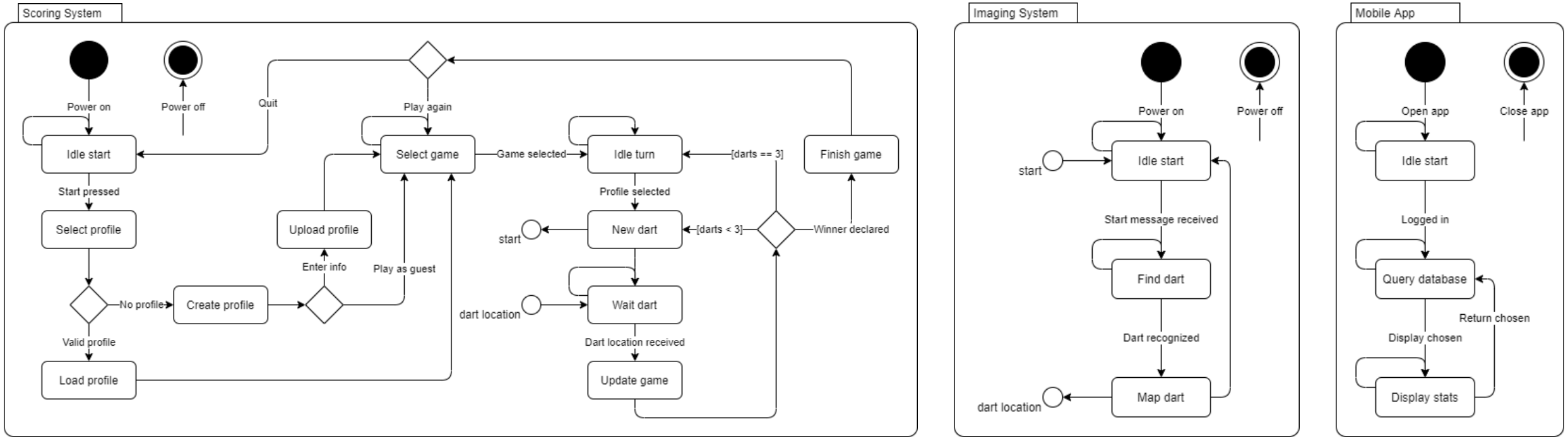
Functional Description

- User will start game, select profile, and choose game to play via touchscreen user interface
- Imaging system will begin looking for dart once user profile is selected
  - Will use identify dart tip location, map to dartboard, and send to scoring system
- Scoring system will receive update and calculate statistics and score
  - Will upload to database and update live hit map for viewing on user interface and mobile device
- System will repeat until winner is declared or game is ended
- Mobile app will show user statistics and scores

Functional Description – Block Diagram

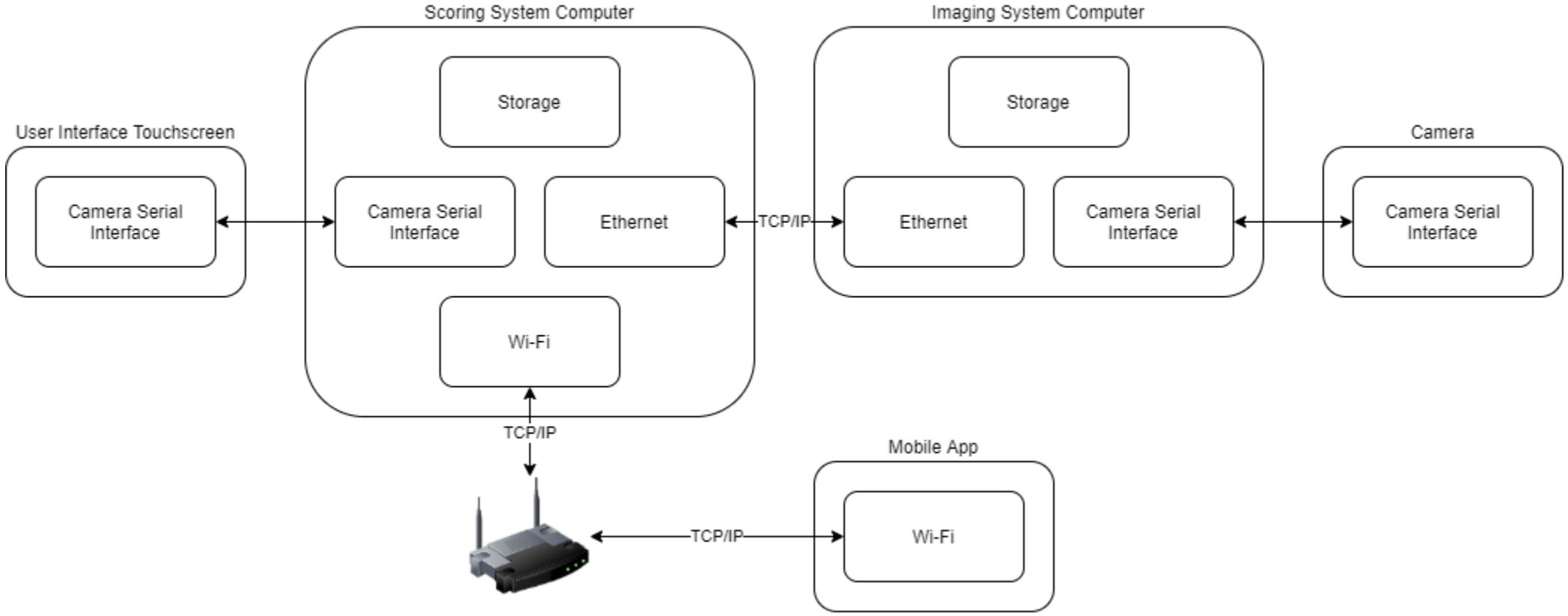


Functional Description – State Machines



Scoring and imaging state machines dependent on each other

Interface Description – Block Diagram

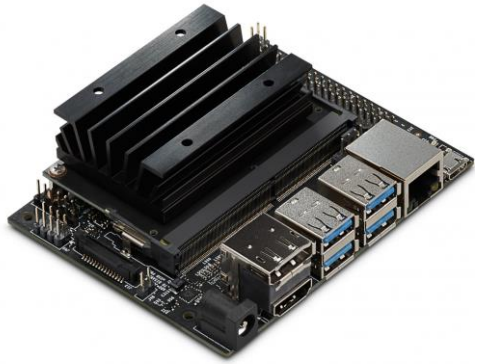


Interface Description – Communication

- Camera Serial Interface
  - Touchscreen interaction – driver-free
  - Image acquisition – Python libraries for video streaming
- TCP/IP Protocol
  - Python libraries for socket programming
  - Scoring system to imaging system communication
  - Database querying and communication with mobile app

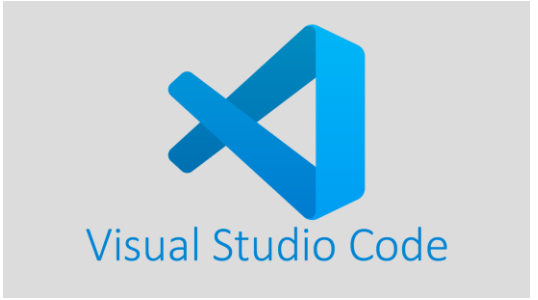
Material Requirements

- Scoring System
  - Raspberry Pi 3B+ with Wi-Fi, Ethernet, and Camera Serial Interface (CSI) connectivity
- User Interface
  - FREENOVE touchscreen monitor for interaction with Pi via CSI
- Imaging System
  - Nvidia Jetson Nano with Ethernet and CSI connectivity
- Camera
  - SainSmart IMX219 8MP module for interaction with Jetson via CSI



Materials readily available and already acquired

Resource Requirements



- Development will occur in Python language
  - Main IDE – VS Code with Python extensions (Python, Pylance, IntelliCode)
- Scoring system
  - Database development – sqllite3 libraries
  - Wi-Fi communication – SSID connection to home network upon imaging of OS
  - User interface – driver-free display via CSI; turtle, pyqt5 libraries
- Imaging system
  - Computer vision – opencv-python libraries
  - Machine learning – tensorflow libraries
- Mobile app
  - Main IDE – Android Studio
- Additional tools
  - Wireshark – packet observation
  - VNC Viewer – remote Pi access
- Existing computer vision dart recognition solutions will be heavily relied on

FOSS software and libraries will be used for core development

Development Plan – Approach

Utilize online repository for issue and progress tracking

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

1. Get imaging system to produce valid dart positions
   - Little experience with computer vision/machine learning – flush out early
2. Get scoring system to run both dart games and communicate with imaging system with basic user interaction
   - Minimalist approach to produce core of dart scoring system
3. Create user interface for displaying scores
   - Minimal experience with GUI development – flush out after core system is functional
4. Create mobile application
   - Little experience with mobile apps – flush out after full-cycle system is functional

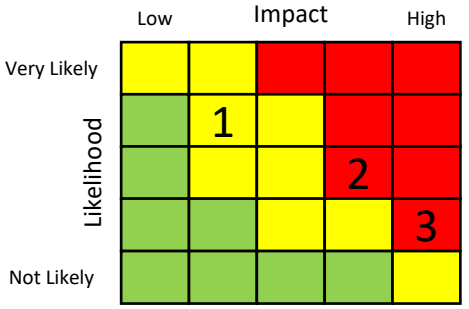
Tiered approach to alleviate possible blockers

Development Plan – Schedule

Date	Milestone	Progress
9/18 – 10/9	Dart Recognition Development	• Dart recognition • Dart location mapping • TCP/IP communication
10/9 – 10/23	Scoring and Game Development	• TCP/IP communication • ‘501’ and ‘Around the World’ game implementation
10/23 – 10/30	Database Development	• Database creation • Database querying
10/30 – 11/13	User Interface Development	• Profile creation • Scoreboard implementation • Hit map implementation
11/13 – 11/20	Scoring and Imaging System Integration	• Functional system minus mobile app
11/20 – 12/4	Mobile App Development	• Beta Android app deployment • Database querying

Milestone based development with tracking in GitHub

Risks



Known Risks

1. Loss of home internet connectivity
2. Machine learning for image recognition likely to be difficult
3. Consistence lighting and angle viewpoints for imaging system

Mitigation

1. Switch Pi to act as wireless access point rather than Wi-Fi client
2. Three weeks built-in for imaging system; use known trained datasets from online repositories to feed solution
3. Implement calibration routines

References

- Nvidia Jetson Dart Score Implementation
- GitHub OpenCV Steel Darts
- Python Libaries