THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

Common Fence Media

Course: EE 494

Professor: Dennis Hite

Date: April 13, 2023

Team: Fencing Engineers

Team Members:

Abram Coleman, Erik Fong, Nathan Wight, Alex Rankin

EE 494

Professor Dennis Hite

April 13, 2023

Table of Contents

EE 494
Professor Dennis Hite
April 13, 2023

1. Abstract

The purpose of Common Fence Media is to provide a product that can automatically track and record a fencing bout. The product will automatically record fencers fencing, and the product has an integrated replay system with audio, a web based user interface, scoring integration, and a streaming application to YouTube. Since the system is automated, the need for human control is at a minimum, allowing for less resources to be needed during fencing bouts. The product is intended to be used during fencing tournaments or competitions and meets the requirements for the International Fencing Federation.

2. Problem Statement

Fencing is a sport where two athletes use their fencing weapons to achieve a "touch". The bout takes place on a fencing piste or fencing strip, and touches are registered by an electronic system hooked to the athlete's suit. A referee will award a point based on blade actions, athlete footwork, and touches registered by the scoring machine. The sport requires quick reflexes and excellent timing from the athletes, and the referees need keen eyes to judge the bout. While the judgment of awarding a point follows detailed rules, the justification for awarding a point can still be subjective. At higher level tournaments, video replay systems are used if the athletes disagree with the referee's decision on who has earned the point. The captured video footage is also used to train both athletes and referees. However, the footage and recordings available are often hard to find, and when available, the quality is poor with jerky camera movements and unintelligible referee audio as prime examples. The recordings are usually only made during higher level bouts at national tournaments or at international tournaments.

Currently, most fencing video replay systems are manually controlled and while sound is required to be captured from the athletes, the audio from the referee is often garbled. The goal of the project is to create a camera that can automatically pan, tilt, and zoom to keep the athletes in frame in a smooth fashion. The system will also incorporate a separate audio stream from the referee in the media stream. The project will also be open source with application programming interfaces (APIs) for other items and equipment to interface with it (i.e. scoring machine data or other metadata for viewers). The automated system will encourage users to live stream and archive the videos for later review. These features will allow for higher quality recordings for broadcasting and training.

3. Patent and Market Research

There are systems that have specifically been designed to track people or sports play autonomously, but there are only a couple designed specifically for fencing. The patents for the systems designed for fencing can be found below. The patents are for tracking a geometric shape, and for using more than one network camera streamed to a base application where paid users can watch the data stream. Our product does not infringe on these patents because our method of tracking is different, we do not use more than one network camera, and we do not stream to a base application for a charge.

- United States Patent #US-11283983-B2 *"System and method for providing virtual pan-tilt-zoom, PTZ, video functionality to a plurality of users over a data network"*

  A system for providing virtual pan-tilt-zoom, PTZ, video functionality video to a plurality of users (60a-60c) over a data network (30) is provided. The system (100) comprises one or more network video cameras (10a-10e) positioned to make a video recording of a real-world target field (20), or a respective subarea thereof, and output a respective raw video segment onto the data network (30), a back-end camera calibration unit (42) which is adapted to obtain a relative position and orientation of the respective network video camera (10a-10e) with respect to a three dimensional world space for the target field (30), and to save the obtained relative position and orientation as metadata for the respectable network video camera (10a-10e), a back-end video processing unit (44), connected to the data network (30) and adapted to receive the respective raw video segments recorded by the network video cameras (10a-10e) and to store or buffer video contents of the respectable raw video segments in a backend video storage (41), a back-end video streaming unit (46) adapted to stream the stored or buffered video contents and the saved metadata for the respective network video camera (10a-10e) onto the data network (30), a plurality of users (60a-60c). Each client device (50a-50c) is adapted to receive a streamed video contents and the metadata, determine a chosen virtual pan-tilt-zoom value and present, on a display (52a-52c) of the client device, the video contents by projecting directions, in the three dimensional world space, onto the display (52a-52c) based on the chosen virtual pan-tilt-zoom value and the metadata.

- Worldwide Patent #WO-2021084165-A1 *"METHOD AND SYSTEM FOR CHARACTERISING A MOVEMENT OF A MOVING ENTITY"*

  The invention relates to a method (1) for characterizing a movement of a moving entity carried out by a system (2) comprising a computing device (10) including a processor (11) and a data memory (12) configured to store a reference action repository (12a) and a reference movement repository (12b), the method

> comprising the following steps: - receiving (300) a video; - pre-selecting (400) one or more images (431); - generating (500) a plurality of characteristic points (501) of the moving entity (9); - calculating (600) a plurality of characteristic point descriptors (501); - comparing (700) the plurality of characteristic point descriptors with reference characteristic point descriptors; and - generating (800) an index value for characterizing a movement of the moving entity from the comparison (700) of the plurality of characteristic point descriptors.

4. Design Specifications

Our design specifications are based on the requirements put forth by the International Fencing Federation. We want our product to be marketable and usable for official tournament use, therefore at a minimum we have designed our system to meet the requirements of the International Fencing Federation. Although some of our specifications will exceed these requirements, our specific system specifications can be found below. The Fédération Internationale d'Escrime (FIE), also known as the International Fencing Federation, has specifications for video replay systems that need to be followed. They require a minimum of 720p video or higher with a framerate of at least 24 frames per second (fps). The video replay speed must be adjustable at least between 10% and 100%. They also specify some optional features like incorporating metadata and configuring video streams to external services [1]. Camera systems are typically at least 3 meters away from the fencing piste, and they are usually centered with the strip. The strip is linear and fencers are not permitted to pass each other. The length of the strip is 14 meters and the width of the strip is between 1.5 meters and 2 meters.

Our system is able to record at 720p at 30 fps. The recording produced also contains an audio stream. The Raspberry Pi 4 hosts a web based replay interface, manual control UI, scoring UI and API, and both a web and RTSP video stream. The Pi 4 also exposes an API for automatic control, allowing for automatic control to be easily swapped out for another type of system. Our completed system is approximately 900 millimeters tall, but the system can be lowered to 142 millimeters, since it is attached to a tripod. The system is approximately 720 millimeters wide when the tripod is fully extended, but can be reduced to 127 millimeters. All of the components are attached to the tripod except for the scoring integration components, making the system easily transportable.

5. Project Constraints

Constraint	Description of Constraint
Economic	The camera and the Raspberry Pi are the primary costs and are expensive components. The system budget was $800 and was almost exceeded due to price increases on electrical components. Better components can be purchased to create a higher resolution video and faster processing, but would greatly increase cost. Although the system is expensive, it can pay itself off over time by not paying for a manual camera operator at each event.
Environmental	With many electrical components, the product should only be used indoors or where moisture is not an issue. The product is not waterproof and exposure to water could create safety hazards as well as damage to the product. The product should also be used indoors with proper lighting to ensure maximum performance with the tracking software.
Ethical	The product is to be used as a video stream and replay system in which fairness to each athlete is necessary. The product should only be used as assistance to referees and not to solely be relied on as final judgment. The product does not show favoritism to one athlete or another as tracking may detect one athlete better than another.
Manufacturability	The components used for the project are commercially available and could be used to mass produce the product. The software used for each interface is open sourced and can also be downloaded onto each electrical component at the factory or by the customer.
Health and Safety	The potential safety hazards are electrical

	shock if a person touches a connection or the processor, possible physical harm from the moving camera, and/or a possible tripping or choking hazard if the power supply cord is not properly secured. However, the possibility of such injury occurring is very minimal with proper use.
Legal	Possible legal concerns stem from the potential safety hazards and not having the consent to film the bout from match affiliates, participants, referees, and/or members of the public in the background or field of view.
Political	Our complete system does not have any political gain, but an individual component does. Tracking and recording people autonomously could be used in military applications, government surveillance situations, and in machine learning.
Social	Our system primarily has a positive social perception, such as promotion of a team, individual, or organization through live stream or in the use of replay, the opposite can also be said with people not agreeing to have their performance recorded for future use. An individual may not want their technique to be displayed as it may have a competitive advantage, or a dirty move may bring negative attention to an individual. Automated filming may make people feel as if they are being spied on, but our system only tracks persons in the playing area, i.e in range, but is a noted concern.
Sustainability	The project uses components that are multi-use components and can be used for years until the electrical components stop operating. New and better components may be produced and can replace the outdated components for a better operating system.

6. Technical Approach

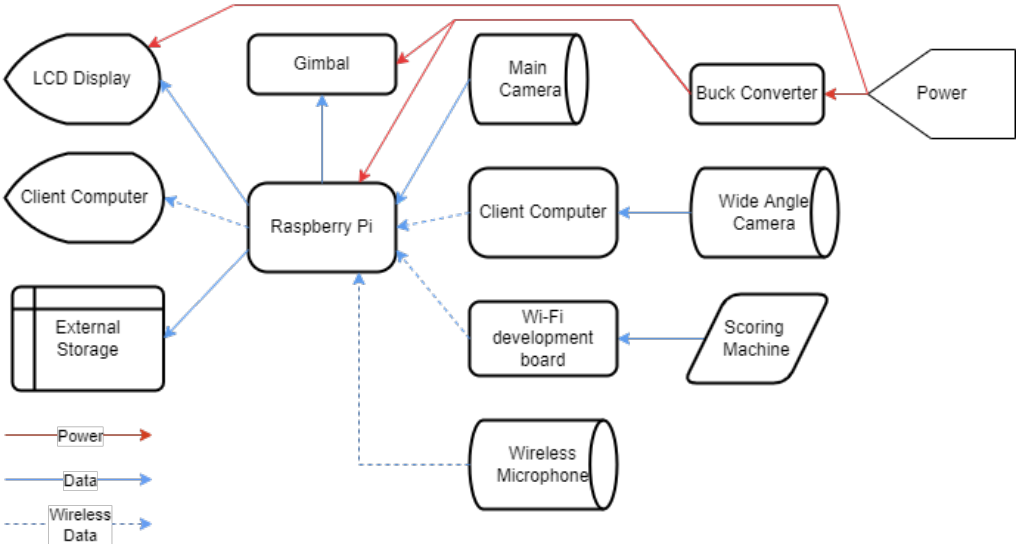
Theory of Operation

The Common Fence Media product consists of three major components: camera system, tracking software, and web UI software. First, the recording camera, the Arducam, will record the fencers during the bout. The recording camera provides a high resolution image to the Raspberry Pi to stream. The tracking camera, a USB webcam, will be used to track fencers. The wide angle provides a larger FOV so that even if the fencers leave the frame of the recording camera the fencers are still tracked in the tracking camera frame. The resolution of the tracking camera is lowered so that the tracking software can compute faster.

The second major component is the tracking software that provides the position data of the two fencers. Both cameras are mounted to a gimbal that is controlled by the tracking software. The software tracks the fencers and controls recording camera movements. The fencers can be detected manually or using object detection software. Movement commands generated by the tracking software are then sent to the web UI software. The commands lead to the next major component.

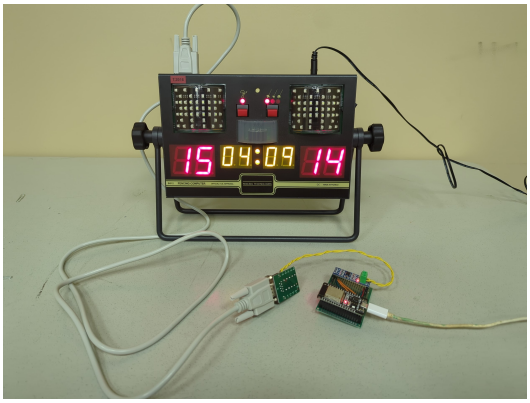
The final major component of the project is the web UI software. The web UI software provides the video and audio stream as well as the video replay capability of the project. The web UI streams and caches video in a user's browser for a live preview and allows video skipping for replay. The playback speed can be adjusted from 10%-100% speed using both a slider and numerical input. The web UI runs on the Raspberry Pi, but the tracking software runs on the client's computer.

The bulk of the processing is done using a Raspberry Pi 4, but the tracking software is too computationally heavy for the Raspberry Pi and will be done on a laptop. While tracking both fencers, the tracking software implemented in Python is too slow on the Raspberry Pi. The Raspberry Pi will be set up on the local network and start streaming the video and audio of the main camera and microphone to a web browser. The client will be asked to download and run the tracking software as a Python script. After the setup, the client can stream the resulting footage using free software such as Open Broadcasting Software (OBS) A basic diagram of the system's functionality can be seen below.



Hardware

On the hardware side, we will have a Raspberry Pi connected to the Arducam camera through a MIPI CSI-2 connection. The wireless lapel microphone will also be connected to Raspberry Pi using one of the USB ports found on the pi. The USB camera is connected to a user's computer that has the tracking software implemented. For scoring integration, we will have a fencing scoring machine connected to the MAX 485 module. The MAX485 module is then connected to an ESP32 board connected through a UART receive pin. This module is responsible for translating the differential RS-422 signal into a logic level signal for the ESP32. The ESP32 will then decode the data and send the information to a web user interface over WiFi. Once the ESP32 has been updated with the correct IP address for the Raspberry Pi, the ESP32 can then be powered by any micro USB cable supplying five volts. A picture of the connections for scoring integration can be seen above.



Software

A video stream from the primary is captured and distributed by a software called mediamtx [2]. This software serves two different streams using two different protocols. One of the video streams is just the video without audio, while another video stream has additional delay because it includes audio. Both of these streams are served using the HLS protocol and RTSP protocol.

The web video stream uses low latency HLS to display a preview with a delay of around 1 or 2 seconds, but this low latency stream does not contain audio. A separate video stream is provided that also includes audio for viewing the actual replay with the trade off of higher latency at 10 seconds: not good for a live preview, but more than adequate for when a replay is needed. Any laptop used today would be able to handle playing the video stream for replay.

The RTSP video stream is provided so the stream can be sent to another computer for live streaming purposes. The computer doing this streaming would need to have higher specifications to do the necessary video encoding: we were using the hardware encoding of a mobile NVIDIA GeForce GTX 1650, but CPU encoding could also be used. Nothing explicit can be explicitly stated as "minimum requirements" at the moment.

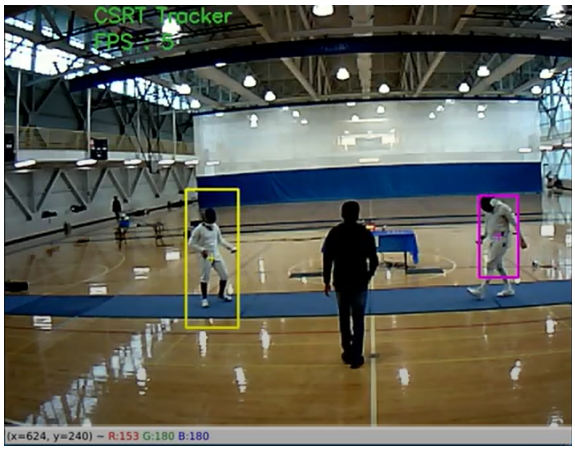
The Raspberry Pi 4 runs a flask based web server to serve web pages that are used for the video replay, scoring interface, and control interface.

The scoring interface uses a REST API to listen for updates from the ESP32 and reflects the updates using SocketIO on top of the websockets protocol to any client devices that have the score page pulled up in near real time: a latency of less than a second. This webpage can also be overlaid onto a video stream.

A control user interface provides manual control for debugging.The servo's of the gimbal are controlled using the gpiozero library in conjunction with the pigpio daemon [3][4]. The pigpio daemon was used instead of the default pin factory because of the more precise timing [5]. The focus and zoom were controlled over i2c using a python class provided by Arducam.

The websockets control UI receives commands from a separate tracking program. For the tracking software, we use Open Source Computer Vision Library (OpenCV). The library object tracking and object detection capability. The software implements tracking in two ways, object detection and object tracking. The object detection uses OpenCV's Haar feature-based cascade classifier implementation. The purpose of this is to detect the two fencers on the strip at the beginning of each point. A failsafe can be used in software to have the user define the bounding box around the fencers. Once the initial bounding boxes have been placed around the two fencers in frame, the software switches to using OpenCV's object tracking software. The CSRT (Discriminative Correlation Filter

with Channel and Spatial Reliability) tracker provides reliable but computationally slow tracking. The object tracking provides coordinates for a box that surrounds the fencer in the camera frame. The software runs through some logic to determine if the fencers are too close to the edges of the frame. If the fencers are too close to one side, the software will send move commands to the Raspberry Pi moving the camera gimbal.



7. Component Specification

Raspberry Pi 4

- Dimensions (in cm) 10.01 W x 7.01 H x 3.00 D
- Memory Storage Capacity: 4 GB
- Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1. 5GHz
- Power Supply: 5 VDC

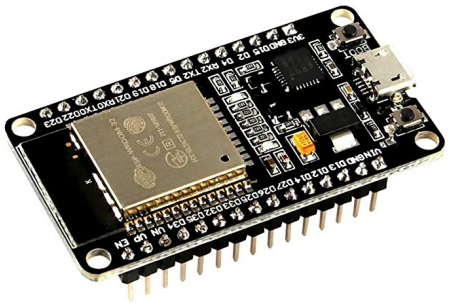
The Raspberry Pi offers a package that can handle hosting a web server and has enough I/O ports for each of the USB sensors that the project needs. The Raspberry Pi was chosen over other processors because a team member possessed one.

Aisizon Wireless Lapel Microphone

- Connection Type: USB
- Battery: 65 MAH Lithium Polymer
- Runtime: 10 hours per charge
- Connectivity Technology: Bluetooth
- Wireless Range: 20 meters

This lapel mic was necessary to get the referee calls during a bout. The Aisizon Wireless Clip Mic was selected for as it was inexpensive and had a USB interface. The microphone was also chosen as it had minimal wires allowing the referee full freedom to walk anywhere within the play area. The microphone was lavalier style making it a minimal bother for the referee to wear.

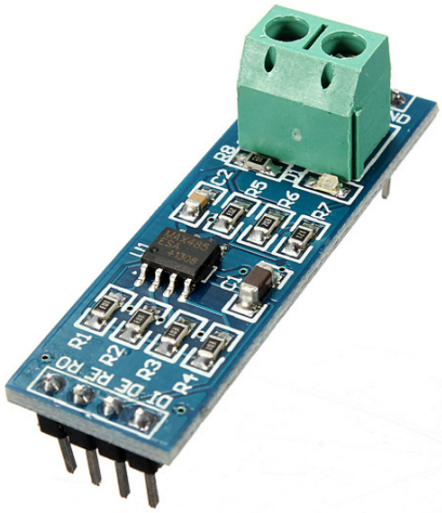
ESP32 DEVKIT Board



- Dimensions (in cm) 11.709 x 7.493 x 2.007
- Frequency: 2.4Ghz
- Connectivity Technology: WiFi
- Power Supply: 5 VDC

The ESP32 board has UART connection pins used to interpret the scoring machine data converted from the MAX485 board to send scoring data to the web based user interface via WiFi to display the score. The ESP32 board was selected for the UART and WiFi capabilities.

MAX485 Instrument Interface Module



- Dimensions (in mm) 44 x 14
- Operating Voltage: 5 VDC
- Communication Type: RS-485/RS-422

The interface module is used to convert the data stream being exported for the scoring machine into a data stream the ESP32 board can use. The interface module allows the data stream to be properly kept without bits of data flipping into unreadable data.

Arducam Camera

- Frame Rate: 720p at 60fps
- Active Pixels: 5MP
- Focal Length: 3.2-11.1mm ±5%
- Dimension: 36mm x 36mm
- Pan and Tilt Range: 0-180°
- Zoom Range (Wide => Tele): 2317 steps 0.000396mm/step
- Connection Type: 15-Pin FPC
- Power Supply: 5 VDC

The Arducam offers a package that contains both a camera and gimbal with easy connection to the Raspberry Pi. The camera outputs video at a better frame rate than required by the FIE. The Arducam provides a camera that has pan, tilt, and zoom capabilities within a reasonable price range.

Pyle LCD Display

- Dimensions (in cm) 19.304 W x 13.310 H x 2.489 D
- 17.78 cm TFT/ LCD Mirror Monitor Screen Display
- 16:9 Wide Screen
- Dual (RCA) Video Input Connector Jacks
- Power: 12 VDC

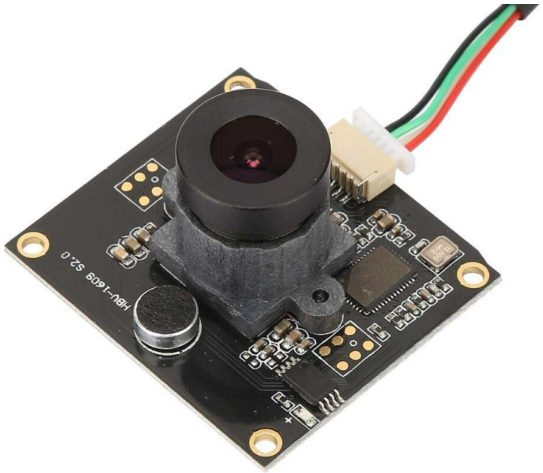
The LCD screen is necessary to see the video output of the Raspberry Pi's during setup. The selected LCD screen was chosen as a team member owned the model.

National Geographic Tripod

- Dimensions (in cm) 40 W x 152 H x 40 D when fully extended
- Weight ~0.89 kg

The tripod is used to mount the camera and other components. The adjustability of the tripod allows for the best video when obstacles or people could interfere. The tripod was selected as a team member owned the model.

<u>HBV-1609 USB Camera</u>



- Dimensions (in mm) 35 W x 31.5 H x 19.3 D
- Highest Resolution: 1600x1200
- Field of view: 120°
- Power Supply: 5 VDC

This camera is used as the tracking camera in the design. It was chosen due to the higher field of view over the primary camera. The FOV was necessary to be able to track the fencers even if they went out of frame in the main camera.

8. Work Breakdown

Team Roles

Abram Coleman – Project Lead

As Project Lead, Mr. Coleman will oversee all aspects of the project and the deliverables associated with it. These will include running and scheduling team meetings, creating and managing individual task deadlines, organization and submission of deliverables, delegating and overseeing individual tasks to team members, and creation of a detailed project budget.

Erik Fong – Software Lead

As Software Lead, Mr. Fong will manage all software related operations of the project. He will develop a Web Server for live streaming video and audio. Create systems to control the camera movement including pan, tilt, and zoom. He will also be responsible for overseeing all software testing and debugging. He will work with Mr. Wight to integrate the data from the sensors in use of tracking the athletes. He will also coordinate with Mr. Coleman to delegate any software related tasks to other team members.

Nathan Wight – Hardware Lead

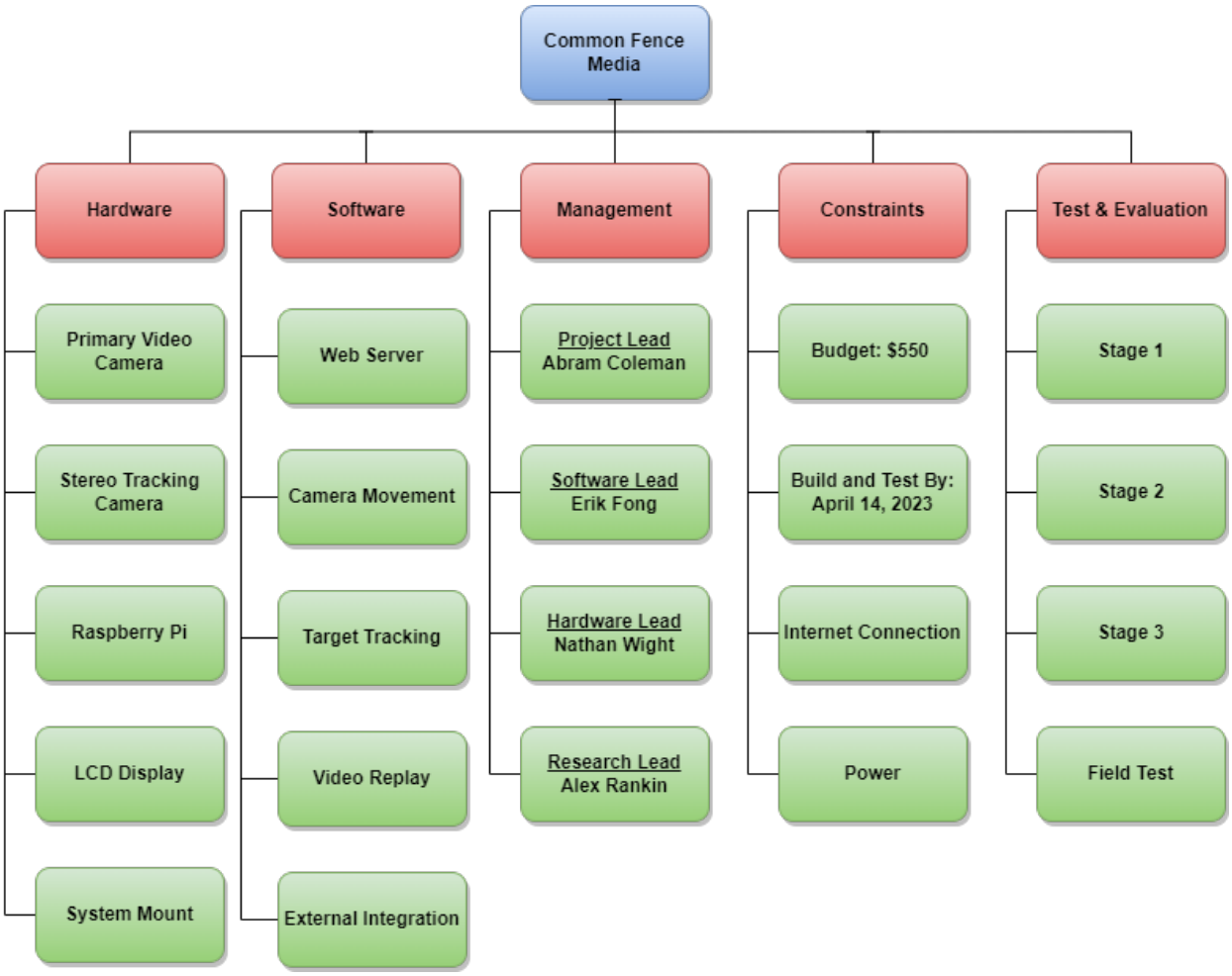
As Hardware Lead, Mr. Wight will supervise all hardware components involved in the project. This will include the design and implementation of all hardware related aspects of the project. He will be responsible for ensuring that the system can operate in the necessary conditions. He will coordinate with the Software Lead on assimilation of the hardware and software in the final design. Mr. Wight will also coordinate with Mr. Coleman to delegate hardware tasks to other team members.

Alex Rankin – Research Lead

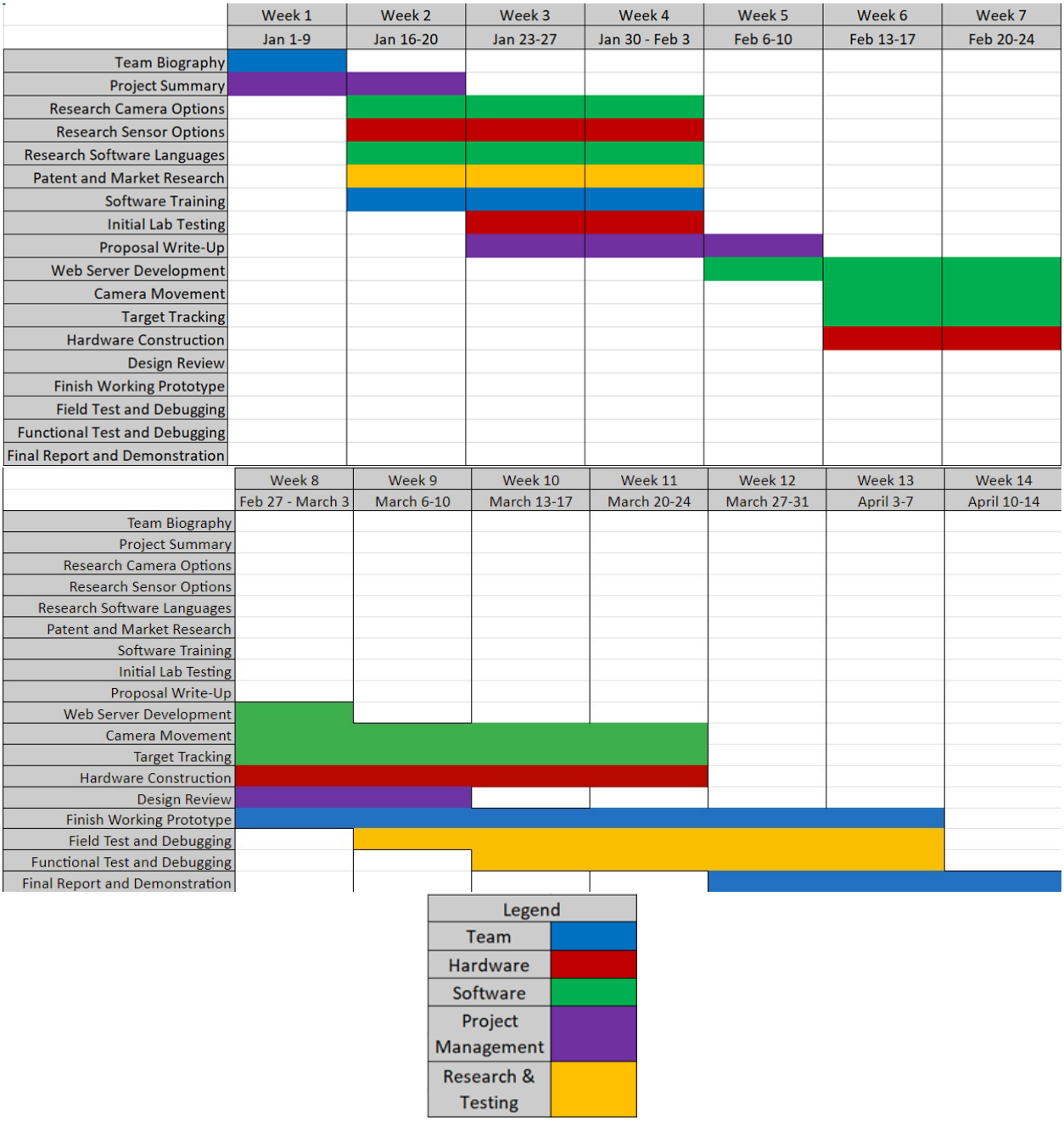
As Research Lead, Mr. Rankin will research and organize all information related to any hardware and software components used in the project. This information will be used to create product tests for the system. He will research and patent information, engineering ethics, or legal documents required for the project. He will also create and organize project reports and other general information related to the project. Mr. Rankin will

coordinate with hardware and software leads to appropriate tests for the project and will work with Mr. Coleman on project documents and reports.

Work Breakdown Structure



9. Schedule


Week 1
Jan 1-9
Week 2
Jan 16-20
Week 3
Jan 23-27
Week 4
Jan 30 - Feb 3
Week 5
Feb 6-10
Week 6
Feb 13-17
Week 7
Feb 20-24
Team Biography
Project Summary
Research Camera Options
Research Sensor Options
Research Software Languages
Patent and Market Research
Software Training
Initial Lab Testing
Proposal Write-Up
Web Server Development
Camera Movement
Target Tracking
Hardware Construction
Design Review
Finish Working Prototype
Field Test and Debugging
Functional Test and Debugging
Final Report and Demonstration
Week 8
Feb 27 - March 3
Week 9
March 6-10
Week 10
March 13-17
Week 11
March 20-24
Week 12
March 27-31
Week 13
April 3-7
Week 14
April 10-14
Legend
Team
Hardware
Software
Project Management
Research & Testing

10. Budget

The project was funded by the team members. Since the project was self funded, one way to keep overhead low was to use parts the team had access to. Using parts on hand reduced the overall cost for the project, but the project used expensive parts. A detailed parts list and prices can be found in the table below.

Part Number/ASIN	Part Description	Cost	Running Total
B0167B5	Arducam 5MP 1080p Pan Tilt Zoom PTZ Camera with Base for Raspberry Pi 4/3B+/3	$ 187.99	
B07TC2BK1X	Raspberry Pi 4	$ 165.00	
B009RIK3EO	LCD Screen	$ 47.99	
B09JNLWYSN	Lapel microphone	$ 32.08	
N/A	HBV-1609 USB Camera	$ 20.29	
N/A	3D-printed Case	$ 20.00	
B09GK74F7N	ESP32 DEVKIT WiFi Board	$ 18.99	
B07ZX5NKHQ	Tripod	$ 18.03	
B085T73CSD	DC-DC Buck Converter	$ 13.99	
B01N8WLEV0	MAX485 Chip Module TTL to RS-485 Instrument Interface Module	$ 8.99	$ 533.35

11. Test Plan

The system was tested in four stages with the last stage being a field test. The stage 1 test tests the camera system and the webserver. The camera system tests for tracking one fencer and sending movement data to the Raspberry Pi, manual control for the main recording camera, and the stability of the video. The webserver was tested to ensure video streaming and recording. The conditions for a successful stage 1 test is a video stream and recording of 720p 30fps and the tracking camera issues movement data. For the test, we had one person start in the center of the camera and set the tracking box onto the person. Once the tracking was acquired, the person walked left and right to the bounds of the frame. As the tracking software tracks the person from

left to right without dropping tracking, the main viewing camera should follow the person. The tracking software gave position data on the person, but failed to issue movement commands to the Raspberry Pi as the person neared the edges of the frame. The video stream was tested by streaming the video to a web browser with the camera aimed at a watch. It was found that the method used to stream video, while close to real time, was too jittery with regards to frames playing at the right time, and too high bandwidth. This led to a redesign of how the video was streamed to a client computer.

The stage 2 test tests the camera system's ability to track two fencers, and for the main recording camera to automatically move. The stage 2 test also tests the webserver's video replay, the inclusion of audio, and integration of the scoring machine to the webserver. The conditions for a successful test are video replay can be viewed, audio can be understood clearly, scoring machine data can be viewed in the video stream, and fencer's only leave the frame once per point. The web server was tested by starting the software, accessing the WebUI and skipping around with the video scrub bar to examine its limitations: the camera was also aimed at a watch to examine if frames were on time, and to more easily notice when skipping backwards in the video. The audio was tested by adding the configuration that combines the video stream and tweaking the audio delay to synchronize it till one could not notice desync between the way one is used to seeing sounds being formed and the sounds heard. The tracking software was tested with two people walking to each edge of the frame. The people had to cross in front of each other, and a separate person walked in front of them. The tracking was not able to meet the standard of keeping fencers in frame and lost tracking more than one time per point. The scoring integration was tested by first outputting the data stream from the scoring machine into the serial input of the ESP32. Once data was read to the serial input, there the determination of scores could be made and code could be written on a serial output to display the scores in a serial console. The scoring was then tested by increasing the left and right scores and triggering the scoring lights. If the score on the ESP32 serial output was the same as the fencing scoring machine, then the scoring integration passed the test. When tested, the score was consistent and was deemed successful.

Stage 3 tested the integration of all components for the camera system. The webserver was tested for the user interface, and user functionality was tested to ensure a lightly trained user could operate the system. The conditions for a successful test were the system recorded, replayed, and displayed scores, the system operated independently after startup, and the fencers only leave frame 0.6 times per point. The movement of the camera controlled by the tracking program was tested by creating a python class that can handle the background connections so a single function could be called to move the camera; example data was sent and the camera moved. The tracking software was tested by starting the software on two fencers in a fencing practice. The tracking lost positions on the fencers at each point, resulting in fencers leaving the frame more than 0.6 times per point, and was not able to issue movement commands to the Raspberry Pi. The system was not able to operate independently after initial startup due to tracking errors. The scoring machine was also tested to be connected to the webserver. In this

test we looked to see if the score shown in the web UI matched what was on the scoring machine. At first data was not being relayed using websocket, so data had to instead be sent using HTTP POST requests: this method did work. This stage also included a test video stream to YouTube to ensure it would work, and this successfully passed by sending a video stream to a client device that could stream to YouTube. We found that because of a lack of UI design experience, we forgoed testing the useability of our UI with an inexperienced user.

The final test was a field test where the system was tested at a UAH hosted fencing tournament. The system was set up on the finals strip and ready for autonomous operation after proper initialization. The system was tested for tracking two fencers, recording video, streaming video to YouTube, video replay, audio incorporation, web user interface, and scoring integration. A successful test was defined as the system operating autonomously, fencers only leave frame 0.6 times per point, camera jerk below 5 movements per point (camera jerk is defined as unnecessary movement of the camera when both fencers are in center frame already), correct score displayed before next point begins, distinguishable audio, and the video stream is live. To test the tracking the system was used during a bout. We had fencers leaving the frame at a rate of 1.28 times per point. Due to tracking errors and interruption with the referee this specification was not met. The system did exceed expectations with only 3 camera jerks per point which is below the 5 camera jerks for current standards. The tracking did not meet the automatic detection standard. The user had to manually find the fencers after each point. The tracking did follow and issue movement commands to the Raspberry Pi which adequately moved the main camera back and forth. Scoring integration was successful during the field test, and accurate scoring info, including the lights to indicate which fencer scored the point, was displayed in the media stream. The referee audio failed to work during our field test and could not be debugged during the test. After the field test, the audio was reconfigured and operated properly thereafter. As in stage 3, we did not test out an inexperienced user with our system.

12. Summary and Conclusion

The project, Common Fence Media, is a system that tracks fencers, streams video, and replay points during a fencing bout. There are several different hardware and software components involved in the design of the system. The components used to provide the system with its tracking, recording, and streaming capabilities include: cameras, gimbal, tracking software, and web UI software. The recording functionality of the system can be used for training purposes for both athletes and referees. The system uses a tracking algorithm to track the athletes during their match. The recording is then able to be streamed using a web server which also incorporates scoring data and referee audio into the media stream. Each component of the system is combined to create a system that can be used for recording and streaming of fencing competitions.

The Common Fence Media system is a functioning project that tracks, records, and streams audio and video of fencing matches. The system went through several stages of testing to

ensure the functionality of each component of the system. At the end of the design process and after several stages of testing the system is functional with room for improvement in the future. The system did not meet our desired requirements, but was not too far off. We had fencers leaving the frame at a rate of 1.28 times per point. We had aimed for below 0.6 times per point but due to tracking errors and interruption with the referee this specification was not met. The system did exceed expectations with only 3 camera jerks per point, defined as unnecessary movement of the camera when fencers are already in frame, which is below the 5 camera jerks required. Audio and scoring integration is incorporated and meets the desired requirements set forth. There could be improvements across the system including software and hardware improvements for future versions of the project. The group's recommendations for further improvements are expanded upon in the section below.

13. Recommendations

The project works, but there are many places where improvements can be made. For each major system, we list a few improvements that impact the performance. First, the tracking software can be greatly improved upon. The first recommendation is that the software be rewritten in C++ and/or Rust. These languages perform much faster than Python. The software loses the fencers during the bout if they are moving too quickly. If the tracking software can run faster, the fencer's movements are less likely to confuse the tracking software. Also in the tracking software, the ability to automatically detect fencers at the start of each bout is currently more reliant on the user to find the fencers. The detection algorithm searches for people in the frame, and if there are people in the background or the referee walks in frame, the algorithm might lock onto them instead of the fencers. OpenCV provides functionality to train their detection algorithm. To further the project, we would train a detection algorithm on pictures of fencers. The improved specificity of the algorithm would decrease the burden on the user to initially find the fencers. Other improvements on the tracking software would be implemented in the UI. If the user can define regions where the computer should look in the frame for the fencers at the beginning of the bout, the software can ignore fencers in the background.

Another recommendation would be purchasing a higher frame rate camera that would allow for a better replay experience. The camera used performs properly and provides a decent picture, but the picture is a bit grainy and not as high of a frame rate as desired. Although a higher resolution camera would increase the budget, the quality of replay and live stream would be improved. A recommendation would be either the Raspberry Pi Global Shutter Camera or HQ Camera at $50 or even the Camera Module 3 at $25 [7]. Some sort of adapter should be able to connect our current zoom lens to one of these cameras. An alternative idea to the camera replacement would be a HDMI to CSI adapter [8]. This would allow for a more general purpose camera to be used and many cameras have an HDMI output, but the tradeoff is a new mounting system would need to be constructed with stronger servos to support the larger cameras.

Another factor impacting the quality of the camera was the recording bitrate. During some adjustments, the bitrate defaulted to about 1Mbps and should have been higher. This fix has already been implemented, but was not used during the testing phases.

In our design, the tracking camera was moving with the primary camera. This is not ideal as it is easier for this camera to lose track of the fencers. It would be preferable to have a camera with a wider field of view and higher resolution for tracking. This way, we would have more data for the tracking algorithm and the camera would be stationary with a view of the whole strip.

The zoom functionality of the camera was not used during our testing. We could replace our camera module with another camera with some sort of auto focus instead. OpenCV could also implement an autofocus routine for the camera we used, but we never implemented it.

The replay interface performed the functionality needed, but there were a few bugs. One major bug was how the video scrubbing did not properly reflect the span of video you could scrub through. One method of fixing this would be to constantly update the scrubbing bar. Another would be to have a button where a video clip could be rendered: this method was tried, but not enough memory was available to clip and re-encode the video. It would also be useful to add marks on the scrub bar that indicate when a touch happened.

The webserver is also not secured in a fashion where one should run it on a publicly available network as anyone would be able to send data to the API endpoints to change how the system operates. This could be fixed by implementing Transport Layer Security (TLS) and access tokens, so only authorized devices could send data to control the camera movements or scoring data.

Citations

[1] https://static.fie.org/uploads/3/18851-Handbook_of_specification_Video%20Refereeing.pdf
[2] https://github.com/aler9/mediamtx
[3] https://gpiozero.readthedocs.io/en/stable/
[4] https://abyz.me.uk/rpi/pigpio/pigpiod.html
[5] https://abyz.me.uk/rpi/pigpio/pigpiod.html
[6] https://gpiozero.readthedocs.io/en/stable/api_output.html#servo
[7] https://www.raspberrypi.com/documentation/accessories/camera.html#hardware-specification
[8] https://www.amazon.com/Waveshare-Adapter-Raspberry-Backward-Compatible/dp/B08TBD68MH
[9]

Appendix

Libraries Used:

- Python:

```
#Used for web server
Flask==1.1.2
Flask-SocketIO==5.3.3
gpiozero==1.6.2
#Installed using apt
#python3-smbus/stable,now 4.2-1+b1

#Used for Motion Tracking
python-socketio==5.8.0
opencv-contrib-python==4.7.0.72
numpy==1.19.5
```

- Arduino
  - ArduinoJson (https://arduinojson.org/)
  - https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html
    - HTTPClient
    - HardwareSerial

- WiFi

Source Code:

- Two mirrors if one doesn't work
  - https://github.com/UAHFencingClub/VideoReplaySystem
  - https://gitlab.kn4vhm.com/efong/fencing-video-tracker