IM(s) Keeping Score

Group 18: Luciano Almaraz, Sierra Andrade, Javier Vargas Carpio, Erik Vento

Project Description

- Sets out to design a way to automate the task of stat keeping in basketball. At the very least at the recreational/amateur level.
- We want to increase the accuracy of stat keeping, narrowing down on the human error aspect.
- In hopes to promote sportsmanship and recreation. By directing frustration and complaints away from the bookkeepers, gym staff, and players themselves. Especially in close game or playoff situations.
- We also want to be able to keep individual player stats as well. Because, yes, every game has someone at the scorer's table. But outside of school organized games, usually, only the total game score is kept. Or if individual stats are kept, they aren't accurate.

Project Motivation/Objective

One of our members plays Intramural Basketball here every semester. And knows from experience how intense players/teams can be when scores and the clock are even slightly off, and that individual stats are not kept. Especially during the playoff portion of the seasons. Even going so far as leading to physical altercations, ejections and league suspensions.

So this idea came from wanting to improve student and IM staff experience during IM basketball. Something that seemingly has no downside for all parties involved.

- Giving players their personal stats.
- Keeping accurate scores by eliminating mistakes like:
  - forgetting to add the score
  - adding points to the wrong team
  - adding too many/too little points
- Protecting students by avoiding heated interactions between players v. players & players v. staff.

Some motivation is also inspired by IMs itself. As they provide Recreational as well as Competitive leagues. Where clearly players and referees have a place where they can take things a bit more serious. And this could go hand in hand.

Project Objective

Current Goals	Stretch Goals
Shot Detection and update total running score automatically	Distinguish 1pt vs. 2pt vs. 3pt Shots: ideal of course for accurate stat keeping. But shot detection in general came first in priority.
Track individual players: Points, Field Goal Attempts, and Field Goals Made Determined by timing events between which player had the ball most recently before the most recent basket.	Face Recognition per Player: Could help increase points tracking in cases where jerseys can't be read or are obscured.
Camera Detection at significant distance, 47ft to cover a half court. And reliably in the corners as well.	Ball Movement Tracking for more advanced stats. Such as rebounds, turnovers, steals.

Project Specifications

- Shot Detection : Want to detect shots passing through the hoop with 85-90% accuracy. We think the ToF sensors will be the most stable components. With the ball being large relative to the sensors and both sensors working to ensure no false positives count.
  - Total Running Points: running score will be tied directly to the shot detection. Because it can be updated regardless of individual points. Kept at
- Camera Reception Distance: The camera should be able to reach and detect at 47ft away from basket. To cover up to half court. One half per side.

Project Specifications

- Correct Player Points Assignment: This will be one of the lower specification ratings at 60% accuracy, with room for improvement. Because with defense in actual games there are high chances jersey numbers will be obscured. But will still be able to track most layups, and free throws.
  - Free throws specifically, we want to achieve a 90% accuracy. Unobscured view, players are still, reliable distance

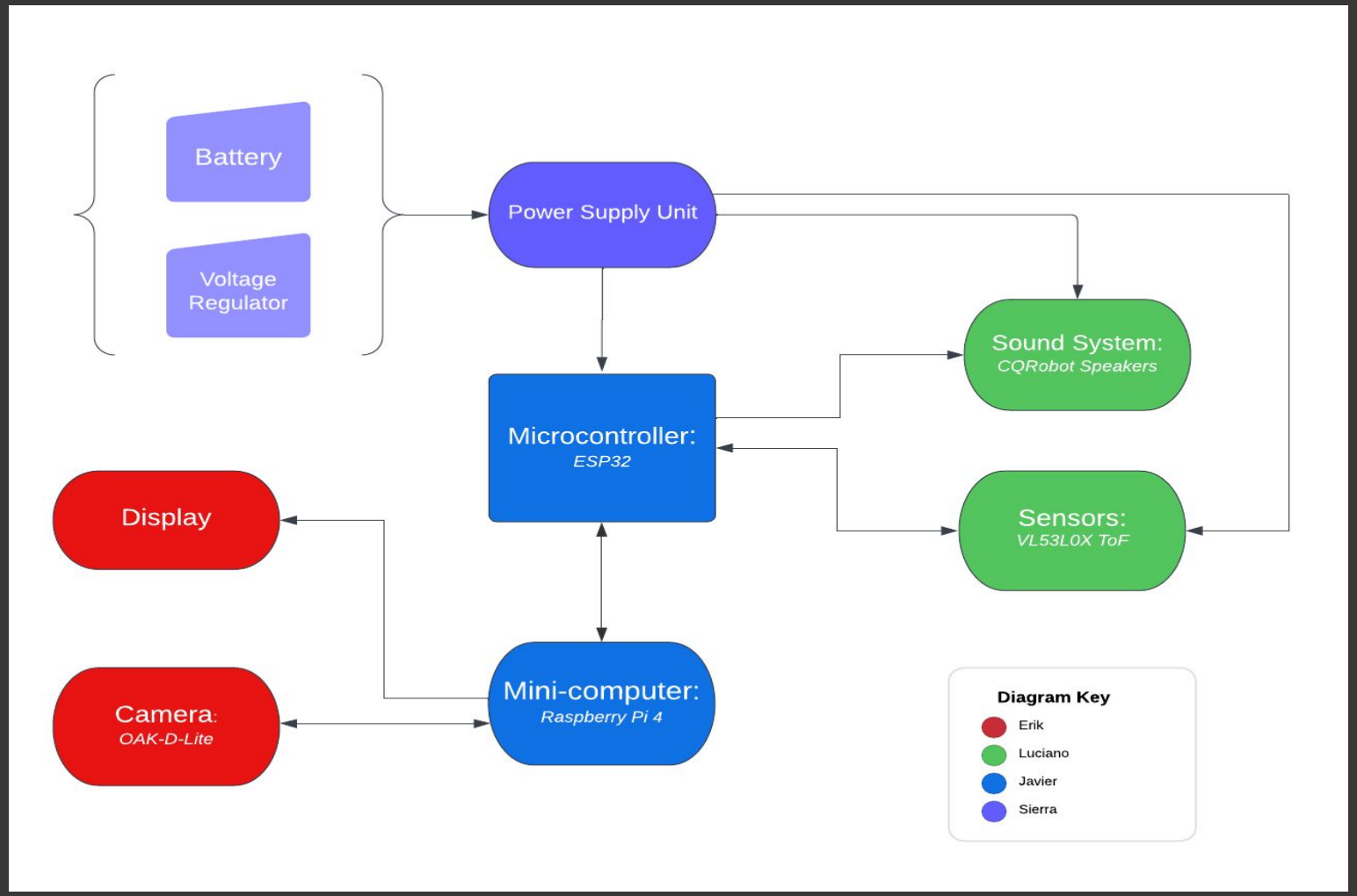
System Hardware

Hardware Overview

For hardware, the main components we need for our project to function are:

- Primary micro-controller
- Secondary micro-computer
- Sensors (2)
- Camera
- Display
- Speakers (2)
- Power Supply

Hardware Block Diagram



Microcontroller Comparison

	ESP32	Raspberry Pi Pico W
GPIO Pins	34	26
Architecture	32-bit, RISC-V	32-bit, ARM
Comm. Protocol	I2C, SPI, UART	I2C, SPI, UART
Operating Volt	2.2V – 3.6V	1.8V - 5.5V
Programming Difficulty	Simple	Simple

Microcontroller

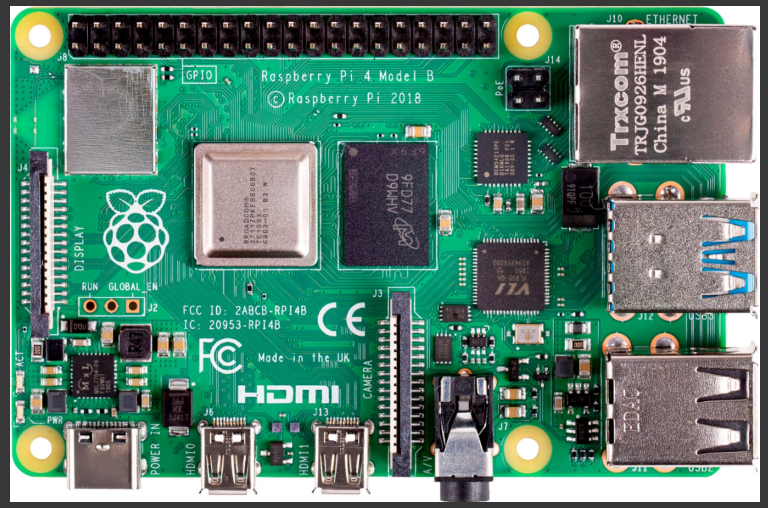
- ESP32-WROOM-32U
- Capability to function as an MCU and Bluetooth module (100m range)
- Responsibilities:
  - Speaker operation
  - Sensor function
  - Communicate with Mini-Computer

ESP32

Mini-Computer

- Allows for control of display and camera since they will be set up at a distance from the hoop
- Responsibilities:
  - Display scoring
  - Camera function
  - Communicate with MCU

Raspberry Pi 4



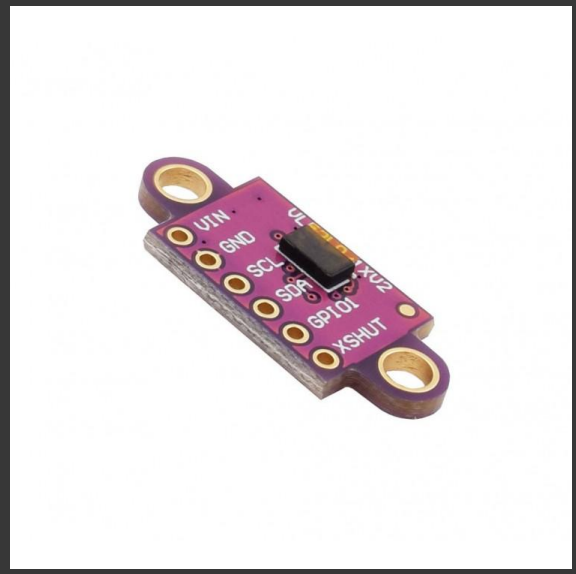
Sensor Comparison

	HC-SR04	VL53L0X ToF	RCWL-0516
Type	Ultrasonic	Active IR	Radar
Range	2cm – 400 cm	200cm	500cm – 800cm
Response Time	Slow	Fast	Fast
Operating Volt.	5V	3V	5V

Sensors

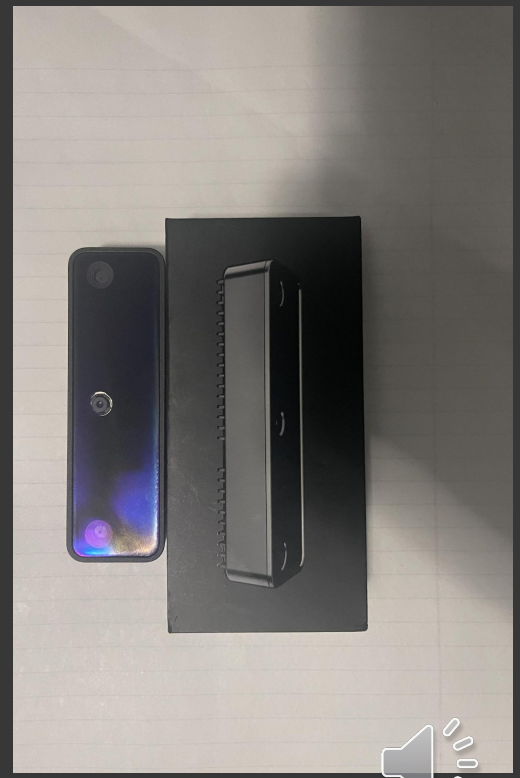
- Best results upon testing, compared to other sensors
  - Quick response time
  - Power efficient
  - Easily implemented
- Will be using two in order to check that a ball has fully gone into and exited the hoop

VL53L0X ToF



Camera: OAK-D Lite

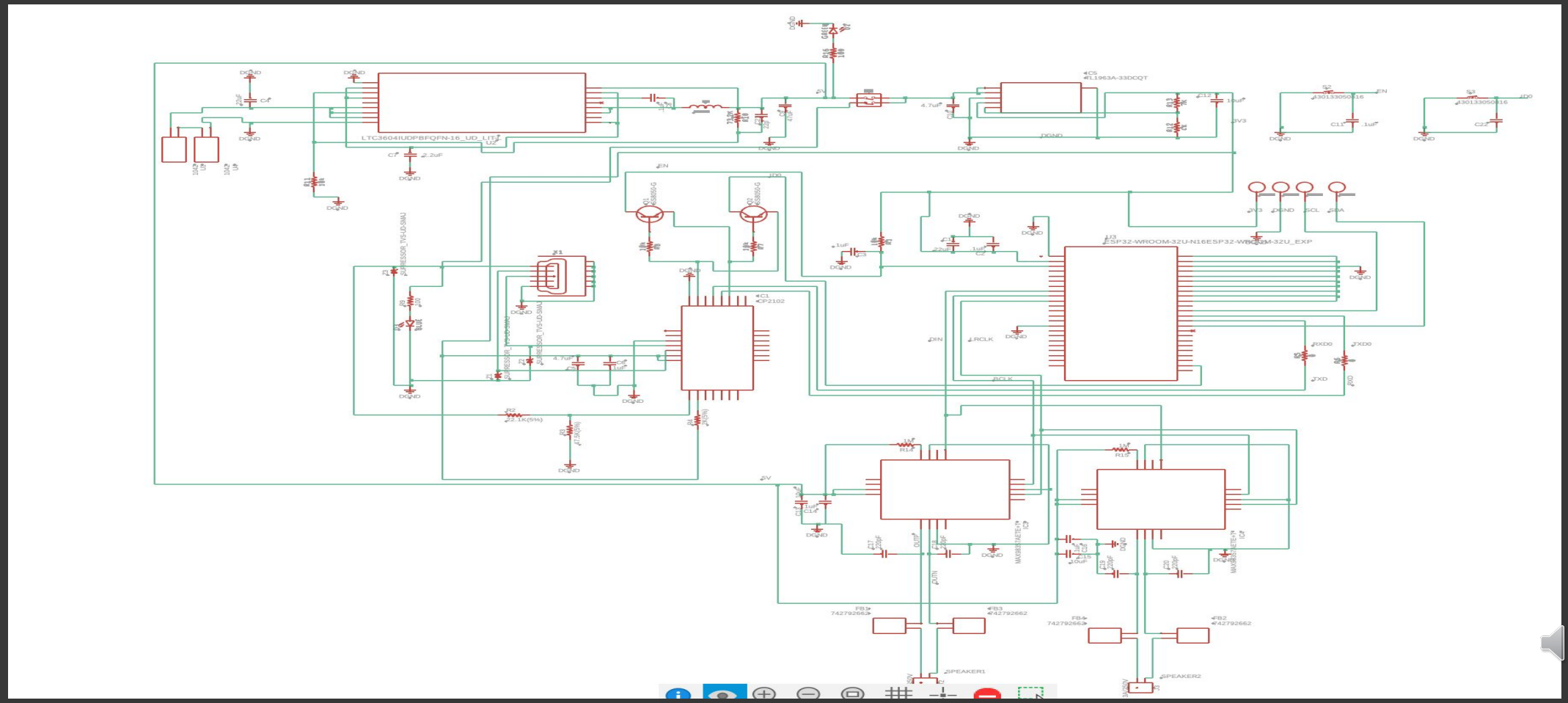
- Powerful Depth Sensing Camera
  - Accurate & Real Time Depth Perception:
- High resolution RGB, 13MP (4208x3120), clear visibility for Jerseys
- Max Frame Rate – 35FPS. Sufficient for our use.
- Compact and Portable: compact enough for easy mobility and placement on/near backboard
- More affordable than it's other versions at $149
- Versatile Development framework: Open AI has a large SDK and large API set for the OAK-D Lite
  - SDK offers smooth integration with many deep learning networks we could use for our player detection



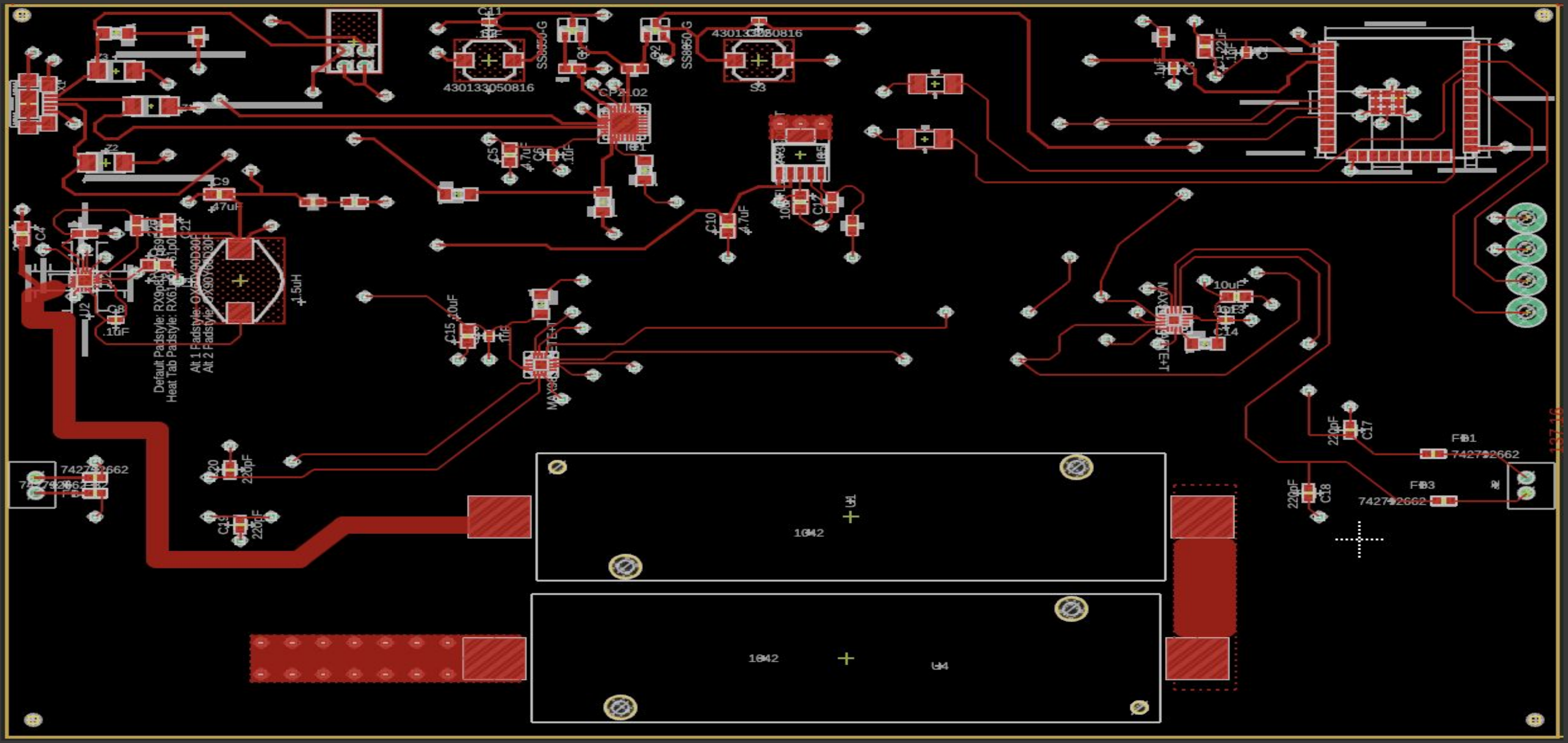
Sound System

- CQRobot Mini Loudspeaker
- 5-Watt, 8 Ohm
- Two will be connected separately to the PCB
- Compatible with DIY projects and Raspberry Pi
- Compact and Loud

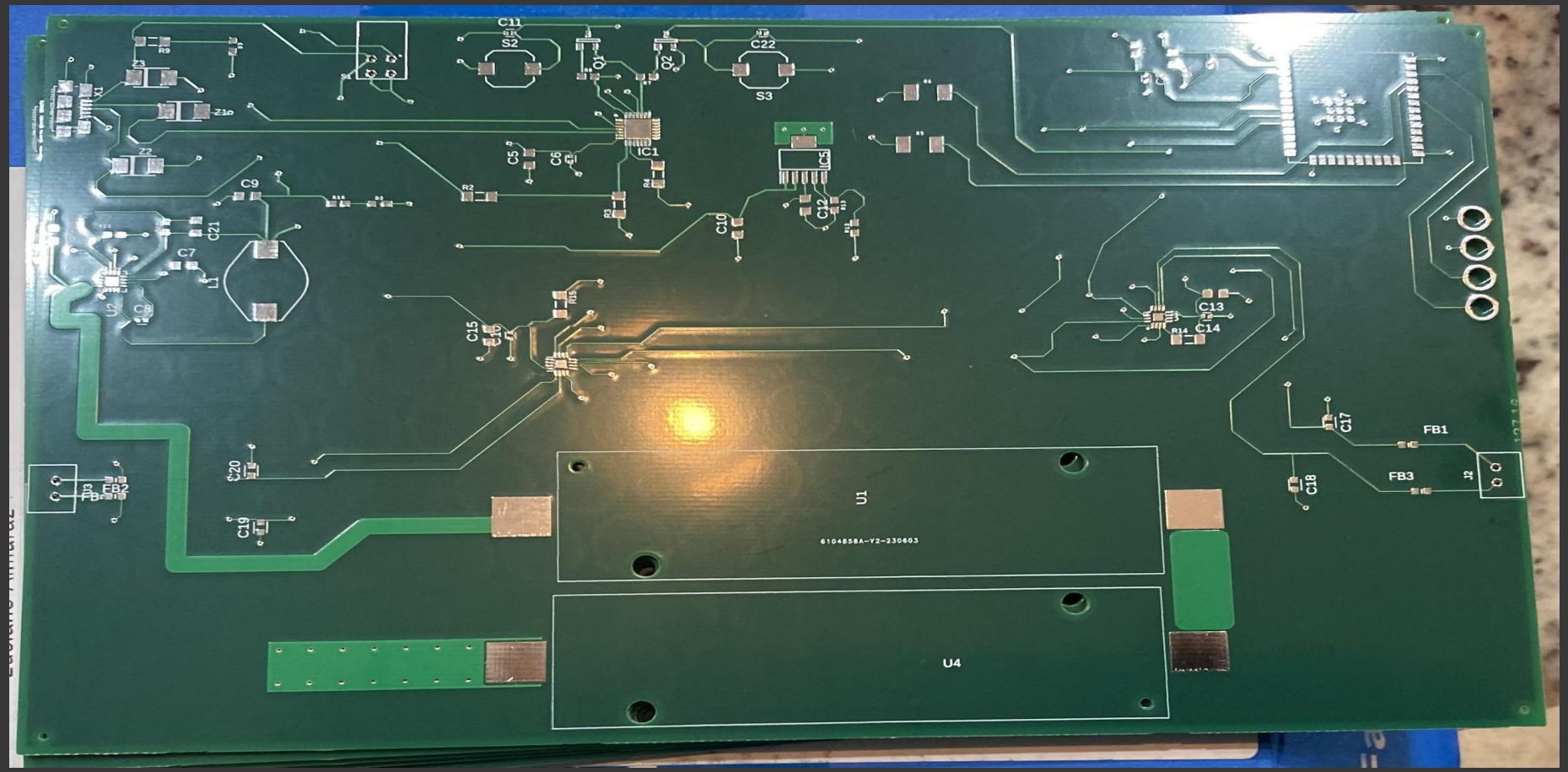
Schematic Design



PCB Layout



PCB Layout Cont.



PCB Design

- Requirements for the PCB included that the ESP32 module be placed in either the top right corner or bottom right corner of the board.
- They recommend the use of a 4-layer PCB
- 2nd layer should be a ground plane as per recommendation with no traces routed in this plane
- Third layer should be power plane traces can be routed in this plane
- For our design we choose to implemented a 4-layer design

Electrical System

Power System

- Since we want to mount our PCB board behind the basketball backboard we choose to go with a portable battery.
- Another requirement we want was to have a secondary battery (a rechargeable battery).
- We also want to be able to mount the battery onto the PCB Board.
- Last requirement was to choose a battery with a high amp hours to ensure that our product would last for at least an hour.

Power Source

- Battery Selection
- Came down to price and being able to place the component on the PCB board as well

Battery Name	MLP806696	ASR00050
Voltage rating	3.7 V	3.7 V
Amp Hours	6000 mAh	2500 mAh
Type	Li-Poly	Li-Ion
Size	-----	18650
Price	$21.90	$5.21

Voltage Regulators

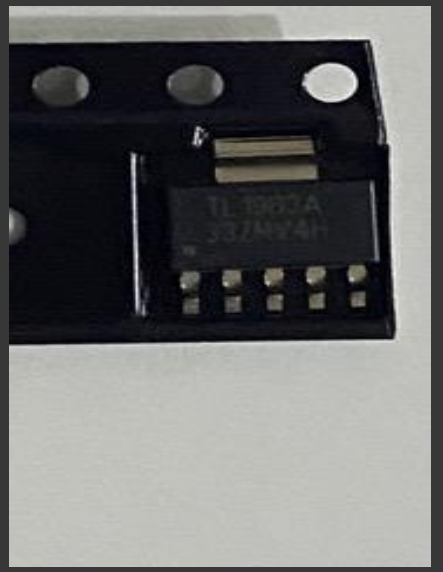
- We are using the LTC3604 for our 5-volt regulator, this is a switching regulator that will convert our input of 7.4 volts from the batteries. This provide power to our Max98357a I2S IC, as well our USB to UART IC, and last it will provide the input to our 3v3 LDO.

Brand	LTC3604
Input Voltage	3V to 15V
Operating Switching Frequency	Multiple/Selectable
Operating Current	2.5 Amps

Voltage Regulators

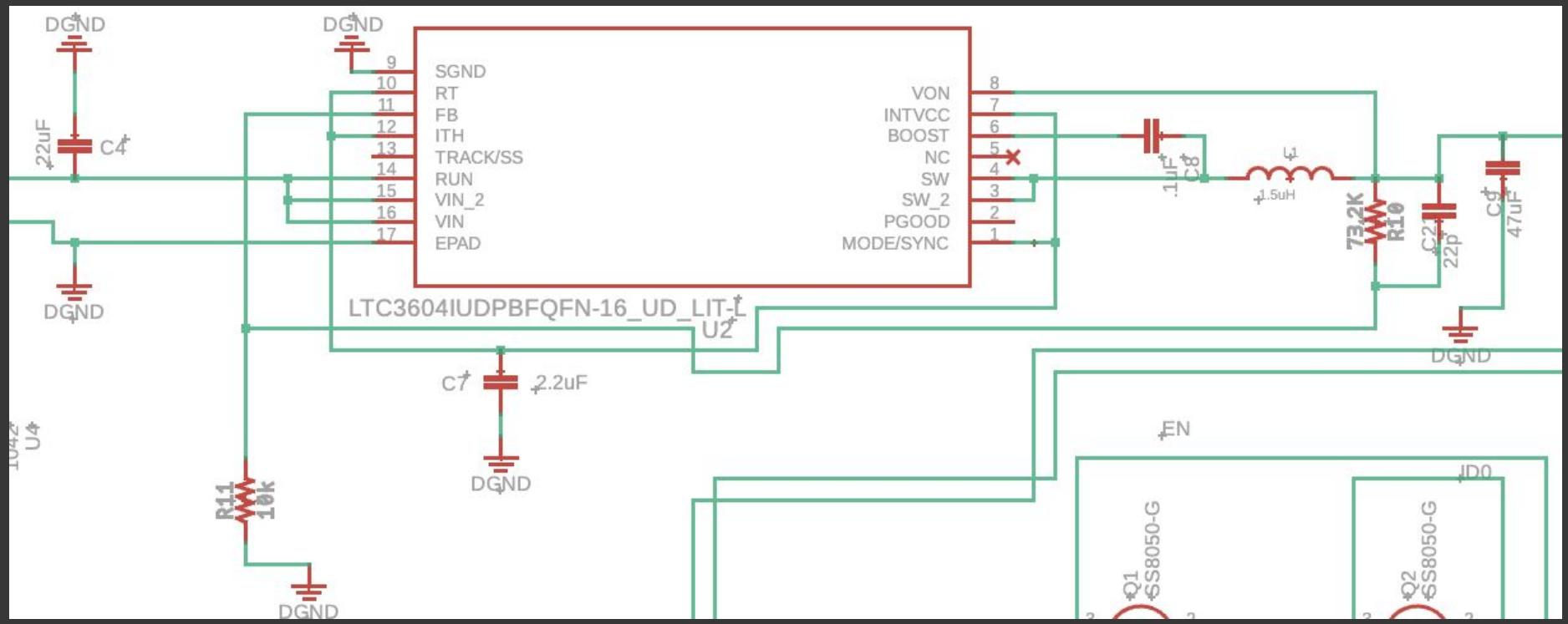
- TL1936A is what we used for our 3v3 voltage regulator, this is our LDO that is meant to power up our Esp32 Microcontroller, the VL53L0X Proximity sensor, and also provides some power to our USB to UART IC.

Brand	Texas Instruments
Input Voltage	-20 V to 20 V
Output Current	1.5 Amps
Dropout Voltage	350 mVolts



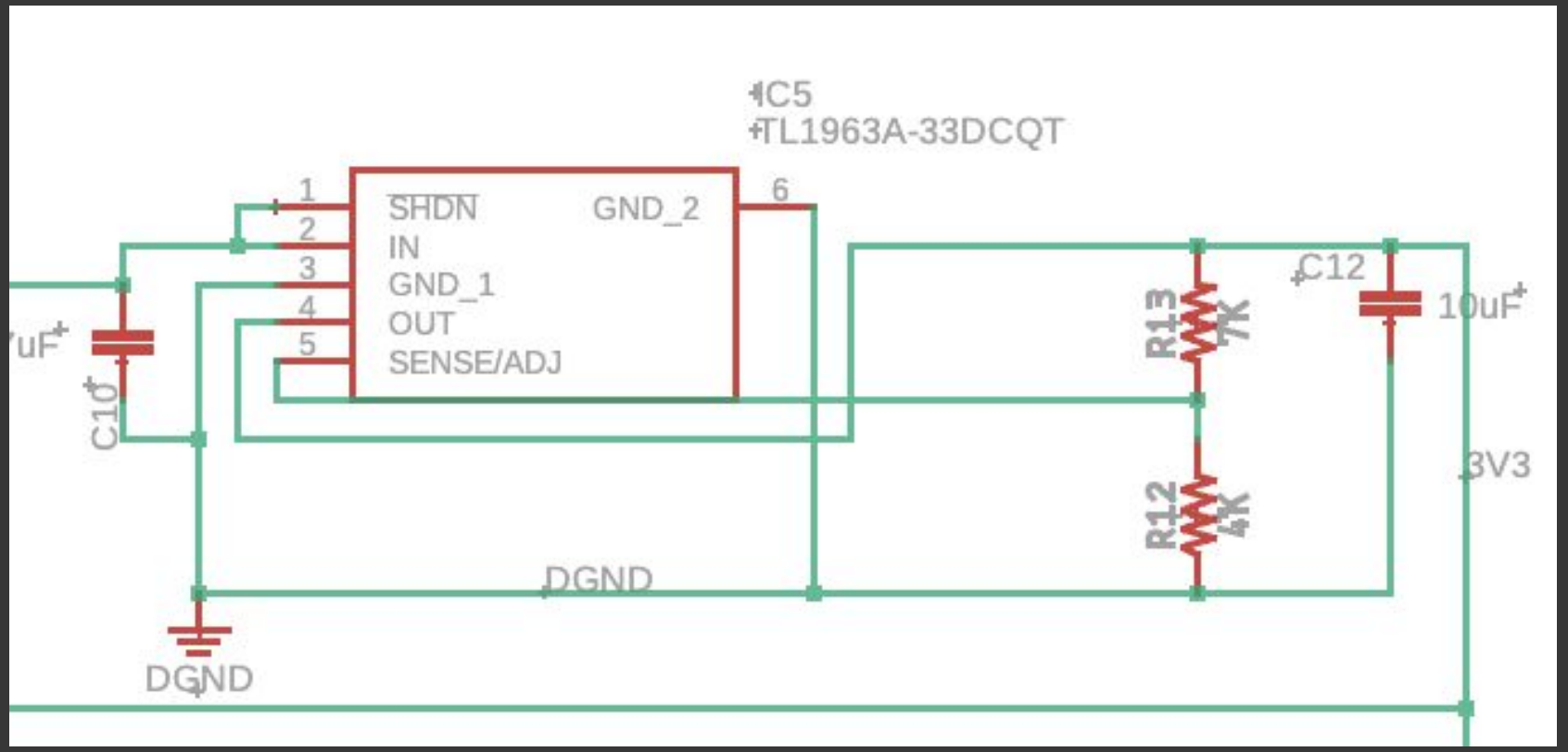
Power Supply Schematic

- LTC3604



Power Supply Schematic Cont.

- TL1936A



System Software

Design Overview

- ESP32 MCU: Controls ToF sensors for successful shot detection
- ToF Sensors: At least two work together. Gauge distance btwn ball and hoop. Must be triggered in correct order to count.
  - Having 2 (one to capture entering the rim and exiting the net in that order) eliminates larger % of illegitimate 'made shots'
  - If triggered in order: speaker sounds and score and player stats update
- RaspberryPi: Used externally for its more powerful image processing power.
  - Uses the OpenCV & PytesseractOCR Libraries for Camera Vision, Object Detection, and Text Recognition

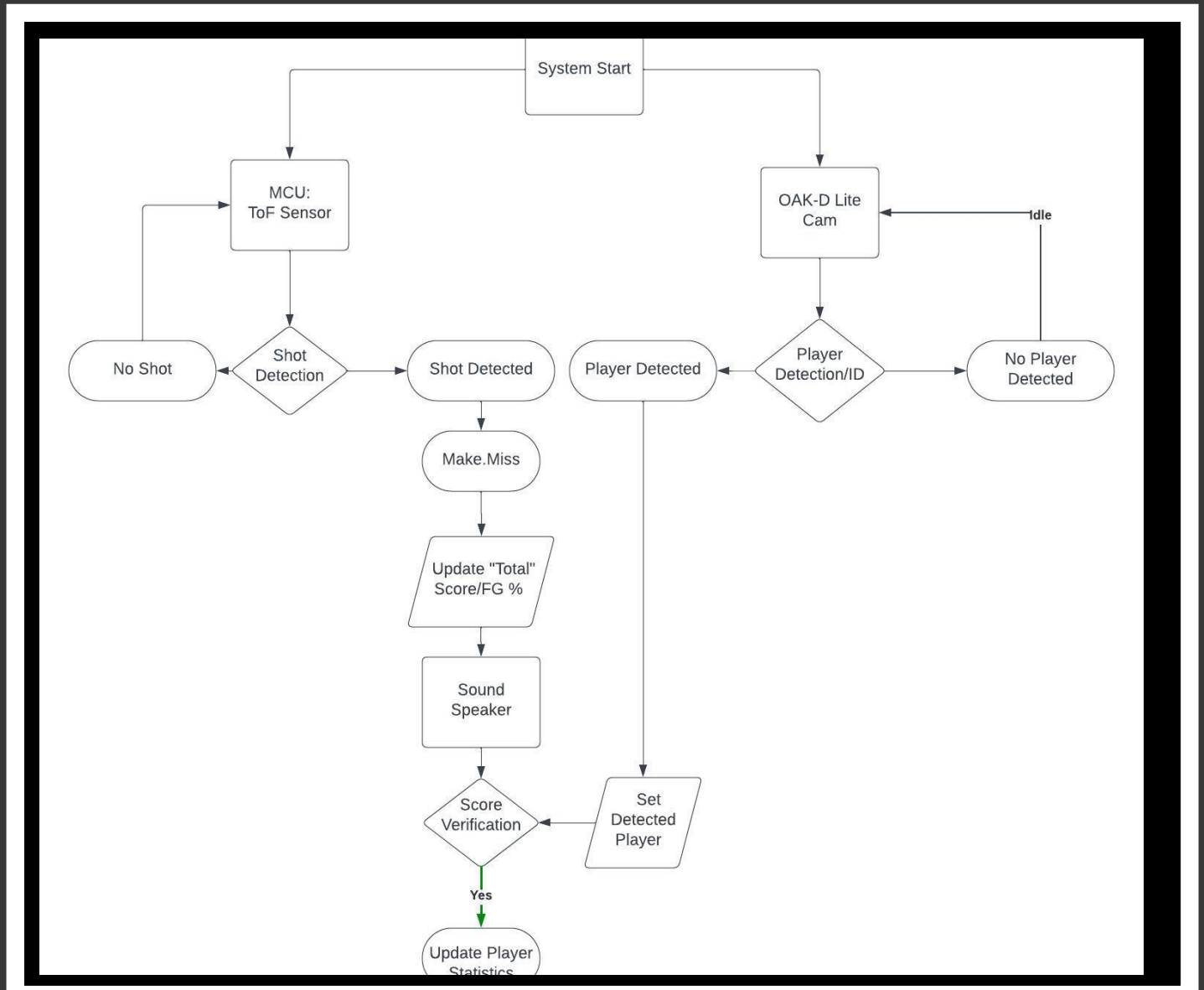
Design Overview

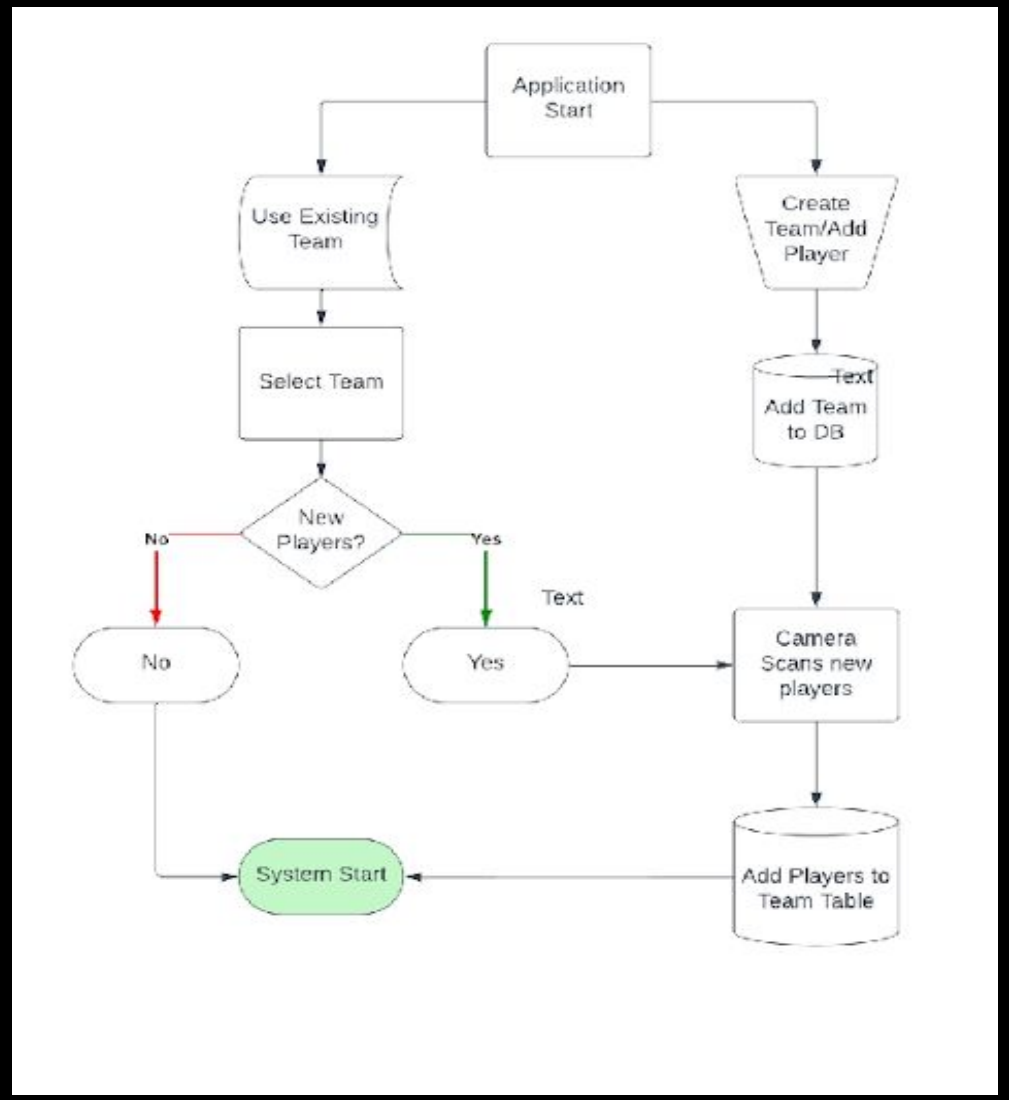
- To pair along side the system we want to develop an app that holds users teams and roster of players.
- The app will be a MERN stack app using React Native. A technology that was familiar to the CPE member.
  - Which uses: MongoDB, Express.js, React, and Node.js
  - And CSS and JavaScript for frontend development.

Number Identification

- Data Acquisition: Live feed from OAK-D Lite cam using RaspberryPi
- Image Processing: The feed will be processed using the OpenCV libray.
  - Contains preprocessing techniques: Resize, color normalize, and noise reduction. To regulate images and prime for text recognition
- OpenCV: will use Region of Interest to extract & narrow down areas to look for text. Increasing accuracy. Through OpenCV's object detection
- Pytesseract OCR (Optical Character Recognition): interprets text within the ROIs & post processes results
- Final Display & Association to Database

Software Block Diagram





Software: UI Block Diagram

Software Comparison & Selection

- Image Processing Libraries

TensorFlow	OpenCV/PyTesseractOCR
At first wanted to use TF to build and train a neural network. But was signigicantly more difficult, time consuming, and required more processing power for the Pi	Made precproccessing images quick and easy. And still allowed for object detection: mainly people and the ball.
Original idea was for face recognition to assign players stats. But gathering data from a user standpoint was a lengthy process.	Was easier and quicker to integrate with the Pi and not as GPU and resource heavy for the Pi
Although pairing with OpenCV did allow for loading pre-trained models. The challenge was enticing.	Eliminated having to create custom datasets, gather data, and saved time.

Admin

Milestone Chart

Task	Duration	Status
Order Parts and PCB	April 20th 2023	Complete
Connect all parts together and try and get them to work together	May 20th 2023	Complete
Finish code	June 2nd 2023	Complete
Work on code and finalize integrating parts	June 9th 2023	Complete
CDR presentation	June 15th 2023	Complete
CDR presentation review	June 16th 2023	In Progress
Midterm demonstration	June 28th 2023	In Progress
8-page conference paper	July 7th 2023	In Progress
Final presentation	July 16th 2023	In Progress
Final documentation	July 25th 2023	In Progress
Done end to end with entire project	August 1st 2023	In Progress

Budget

Items required for Smart Hoop	Cost
RaspberryPi CameraV2	$35
OAK-D Lite Camera	$159
Ultrasonic range sensor	$12
Light Detection sensor	$9 - $25
ToF Sensors	$10-25
Arduino/raspberryPi display	$25 - $79
Mini hoop	$30
Motion detection sensor	$3 - $25
Misc.	$100
Raspberry Pi Model 4B	$145
Raspberry Pi Pico W	$6
Sport Jerseys	$45
CQRRobot Speakers	$11
ESP32 MCU x2	$15
Total	$673

Work Distribution

- Luciano Almarez (EE) :Hardware building and testing, PCB design, hardware selection, microcontroller hardware, testing sensors selection research, testing sensors, overall hardware assembly and testing.
- Sierra Andrade (EE) : Hardware building and testing, PCB design, hardware selection, microcontroller hardware, testing sensors selection research, testing sensors, speaker testing.
- Erik Vento: Hardware building and testing, hardware selection research, testing software coding software, testing user interface design, testing camera code .
- Javier Vargas (CpE): Software design, coding software, web application design, server configuration testing and implementation, camera testing software coding software, testing user interface design

Challenges

- One of our biggest challenges was trying to figure out how to integrate the AI program that we were using to identify the jerseys with the other micro controller that we were using for the sensors on the basketball hoop.
- Another problem we were having was that the camera that we were using was not high enough quality to be able to read jersey numbers from a distance.
- Our last problem was with the speakers, the speaker we were using to test our initial design was not working with the bread board that we had built for the project.