UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN

Automatic Trading Card Sorter

Team #32

Andrejun Agsalud, Steve Guzman, David Medina

Meet the Team

Andrejun Agsalud
Computer Engineering

Steve Guzman
Computer Engineering

David Medina
Computer Engineering

Objective

Objective

The Problem

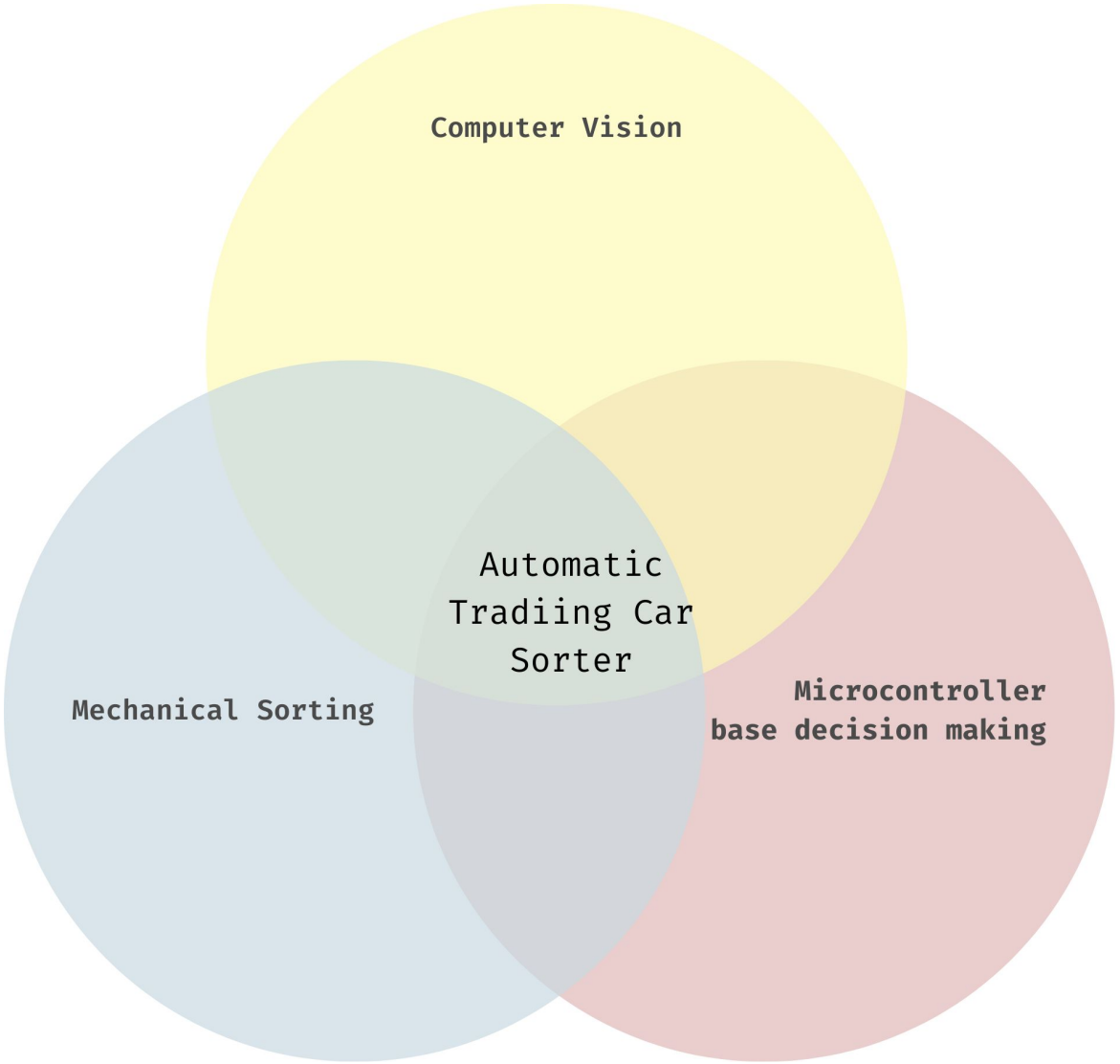
The trading card market is expanding rapidly, with the market valued at $956 million in 2021 and projected to grow annually.

Now vendors and collectors alike need to sort through thousands of cards.

Organizing cards by hand is:

- tedious
- time-consuming
- prone to error
- potentially damaging, affecting value

Objective



Our Solution

The Automatic Trading Card Sorter (ATCS) is designed to use:

- openCV
- Mechanical Sorting
- Microcontroller Processing

in order to automate the process of sorting trading cards in a quick, efficient, and accurate manner.

High Level Requirements

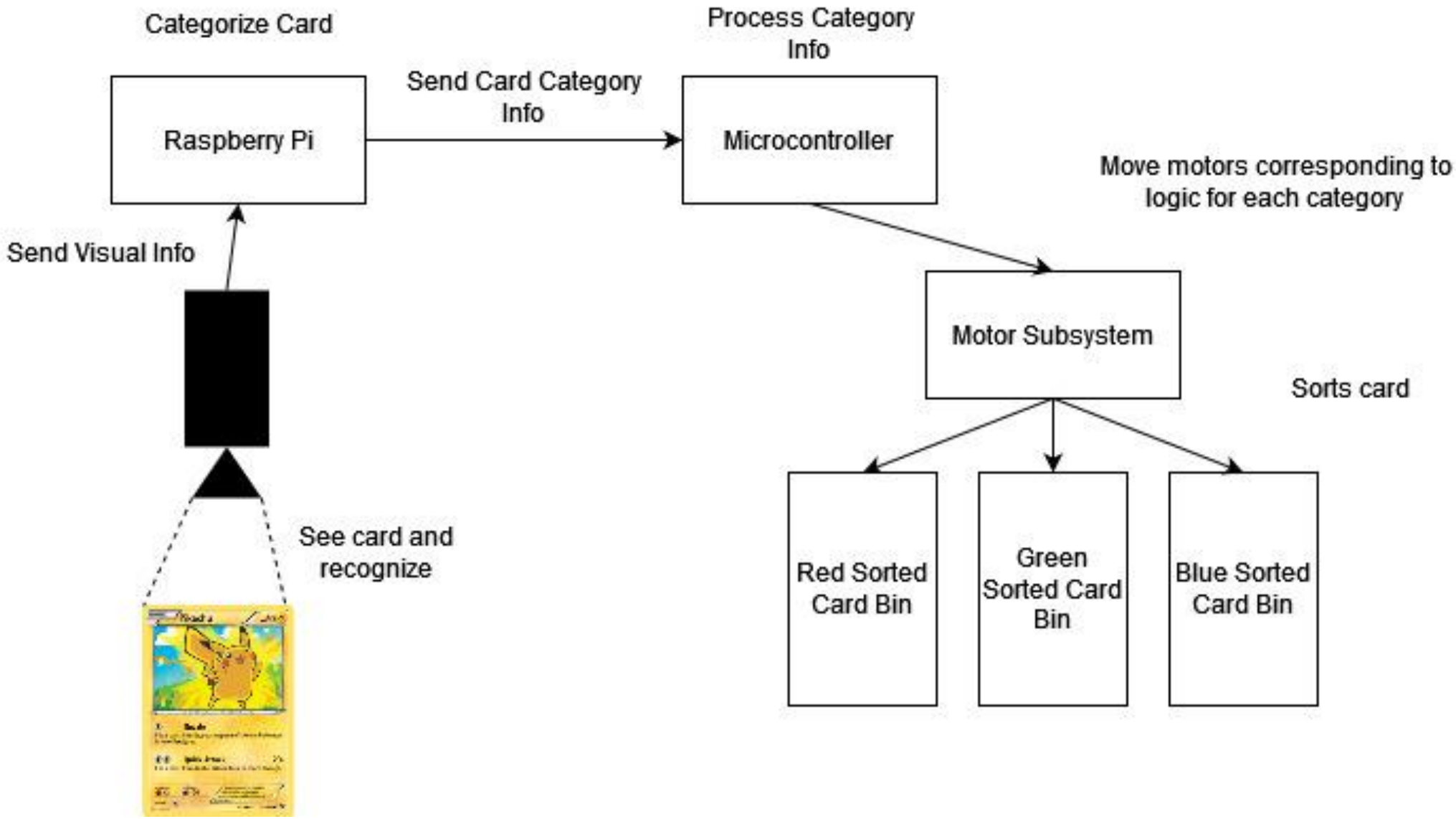
- Two packs of Pokemon cards (22 cards) will be sorted by primary color with one extra bin for non-RGB cards. Maximum of 2 mis-colored cards with the goal of achieving at least **80% accuracy** in sorting. It is a **success** if 16 out of the 20 actual pokemon cards are sorted in the correct spaces.

- A single pack (12 cards) should be able to be sorted in **30 ± 2 seconds**.

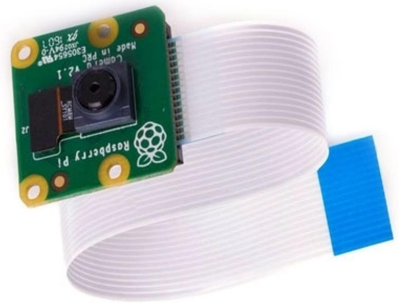
- Will **identify RGB** (primary colors) with **90% accuracy**. Computer vision will recognize if color doesn't match.

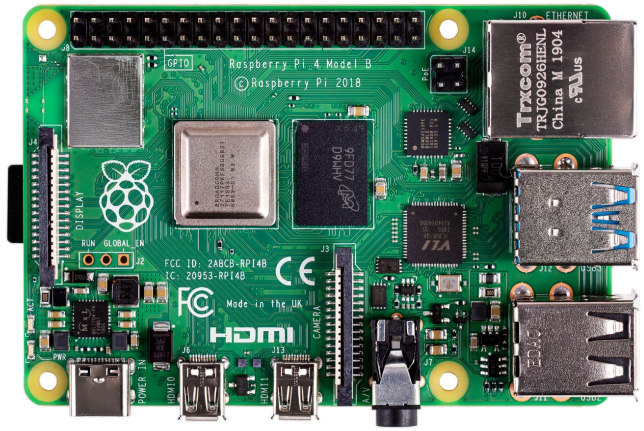
Design

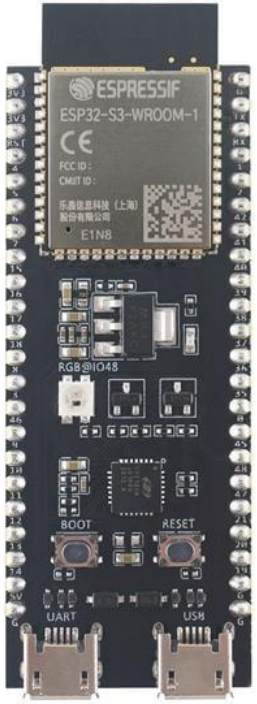
High Level Diagram

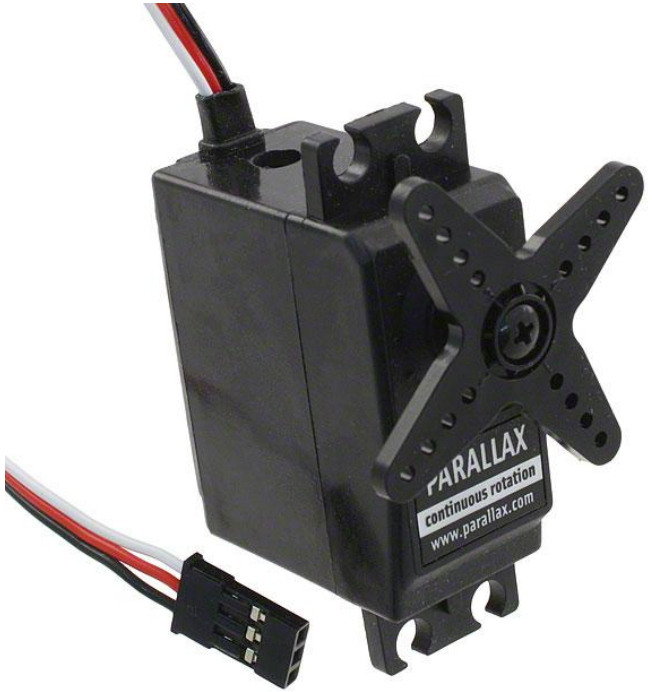


Components

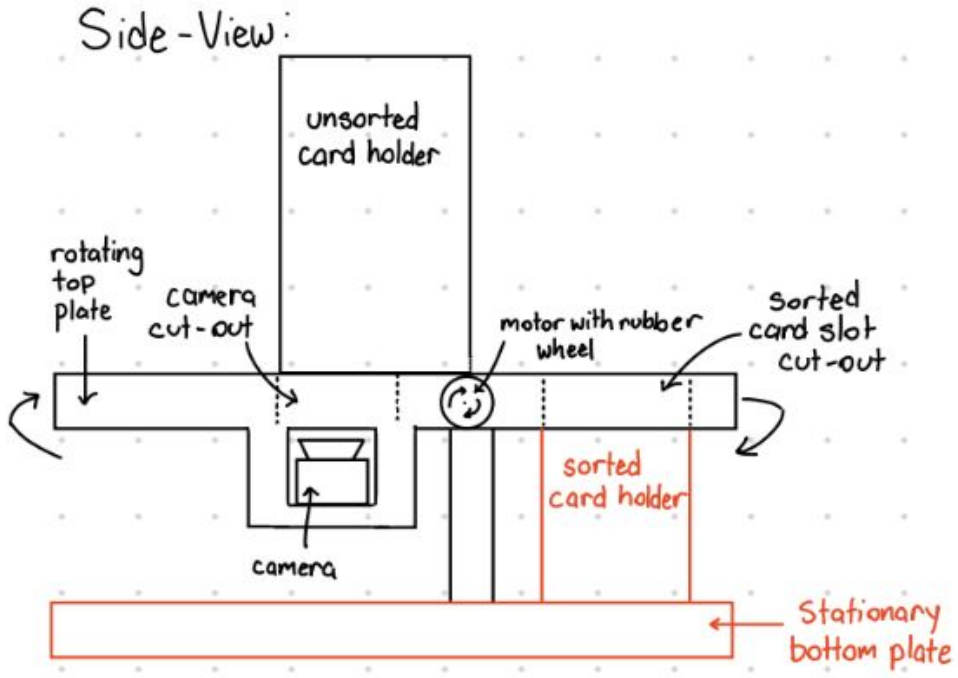


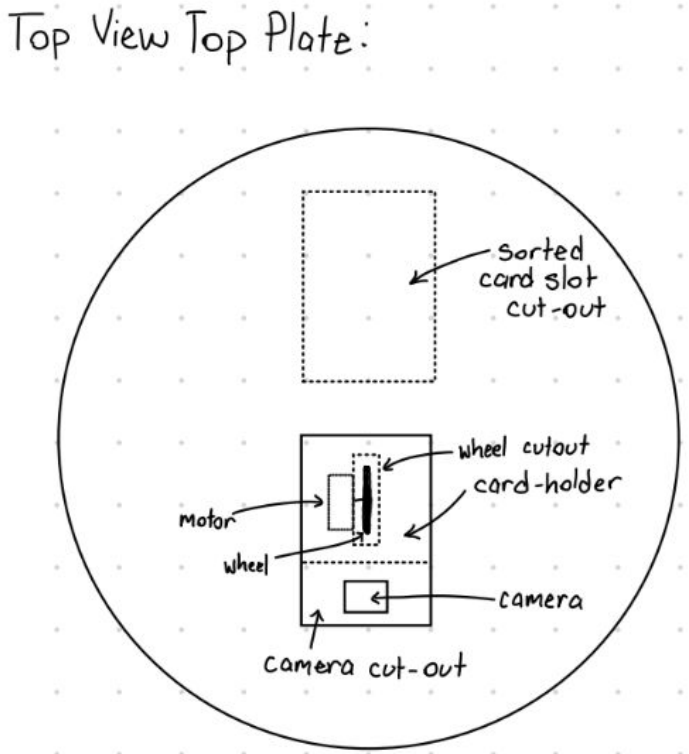

Raspberry Pi 4 Model B
©Raspberry Pi 2018
Trxcom
TRJG0926HENL
China M 1904
FCC ID: 2ABCB-RPI4B
IC: 20953-RPI4B
Made in the UK
HDMI

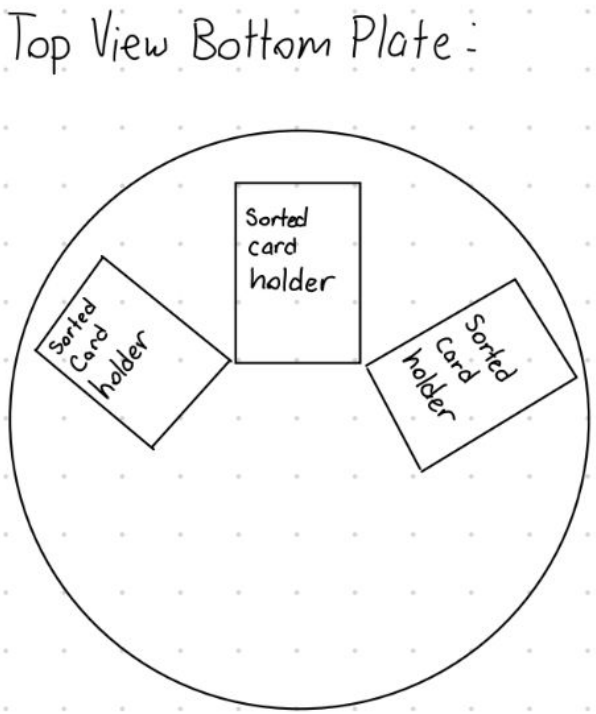

ESPRESSIF
ESP32-S3-WROOM-1
FCC ID:
E1N8
RGB@IO48
BOOT
RESET
UART
USB


PARALLAX
continuous rotation
www.parallax.com

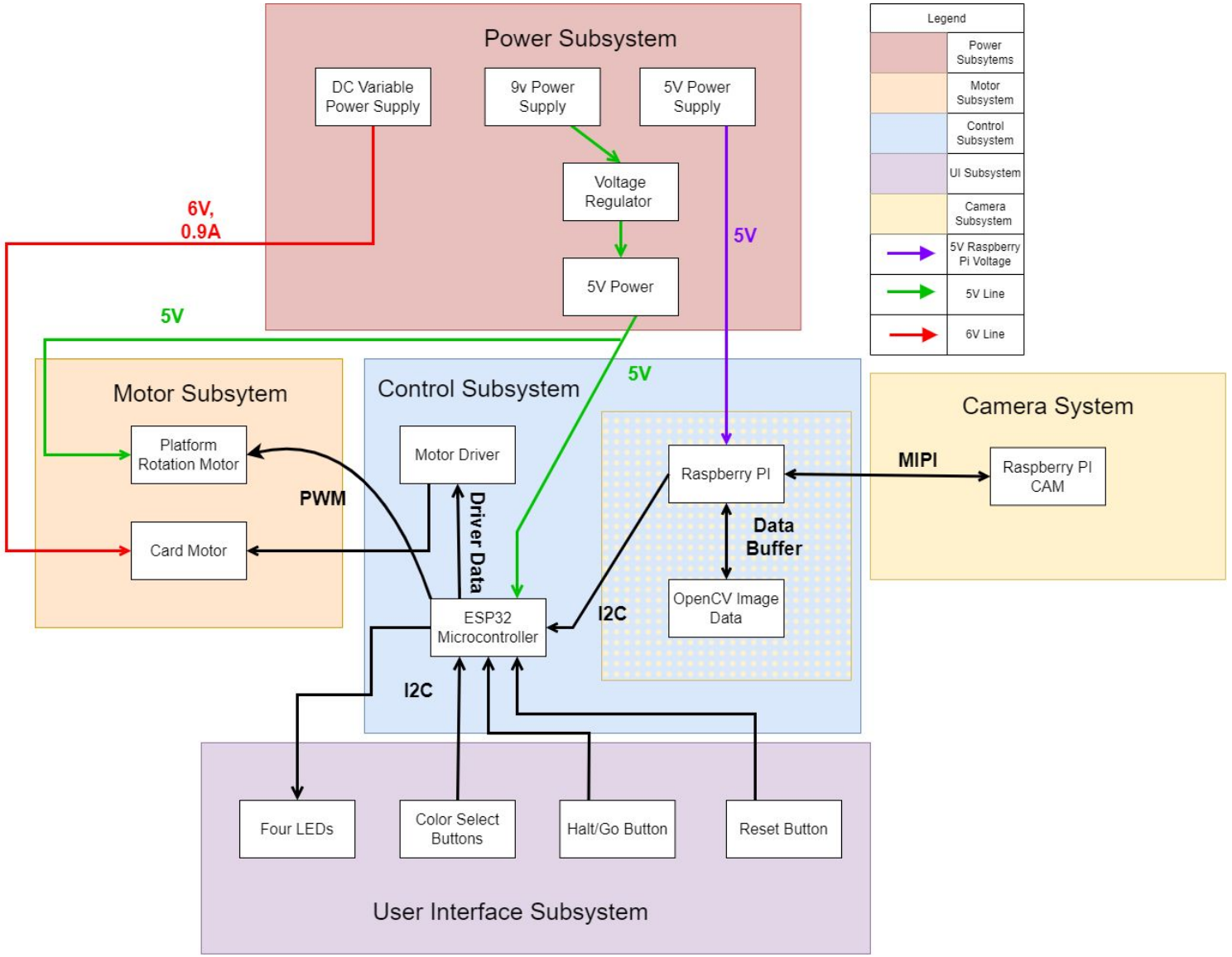
Initial Physical Design

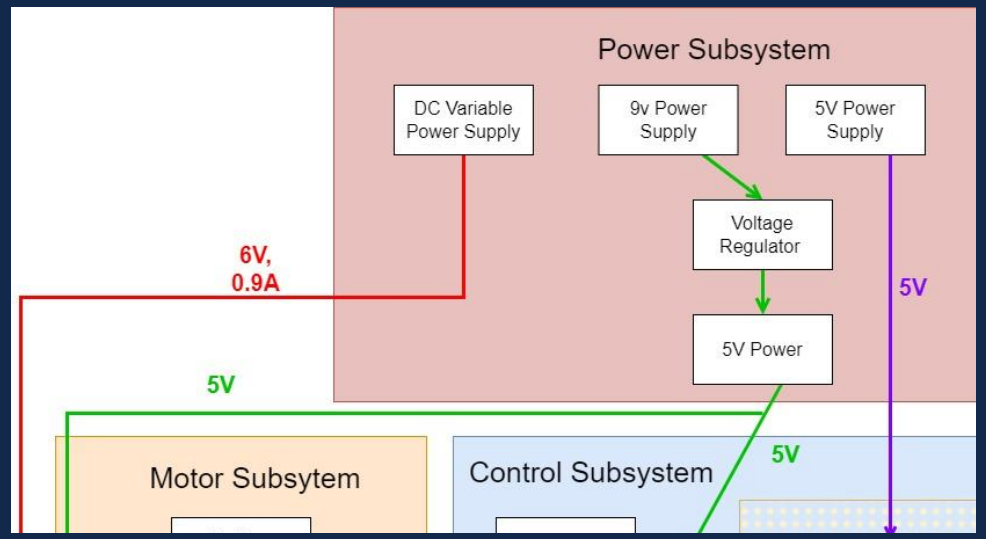






Block Diagram





Power Subsystem

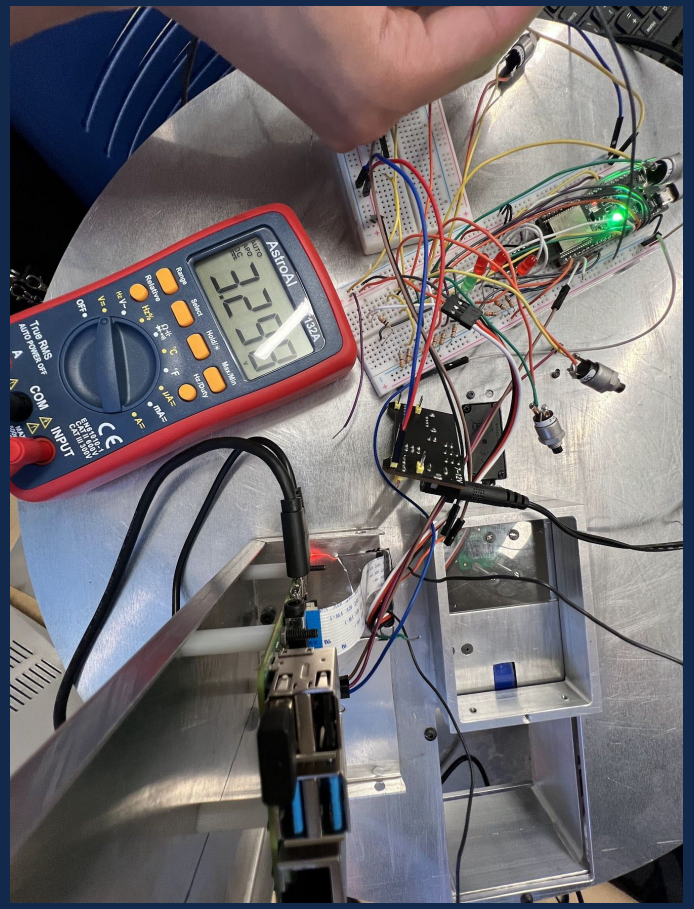
How the power is separated

- 5V independent supply for the Raspberry Pi
- Stepped down 5V to the ESP32
- DC Versatile power supply for the DC motor

Power Subsystem

Requirements	Verification
• Maintain the ESP32's 2.2V to 3.6V operational voltage with external power supply	• Use and monitor the ground and 3.3V pins on the ESP to check for steady voltage • ESP should function independently without needing connection from USB
• Maintain the ESP32's less than 500mA current draw during normal operation(NO WiFi)	• Be aware of flash pins that spike during boot that may cause damage to components • Use multimeter to check and verify nominal current flow • Use DC supply to check what the optimal operating current for the DC and Servo are
• Circuit must protect the ESP32 from current spikes from an inductor(motor) supplied with 6v, 0.9A.	• Do not share grounds with motor power supply and ESP32 • The Servo's digital must never exceed 5V. To ensure this, 10k resistor is used on the digital pin and monitored with multimeter • The servos H-bridge connection is guarded by a diode to protect the circuit from spikes and monitored with multimeter

Power Subsystem



ESP Voltage
3.3V

ESP Current
0.082 A (300mA with servo)

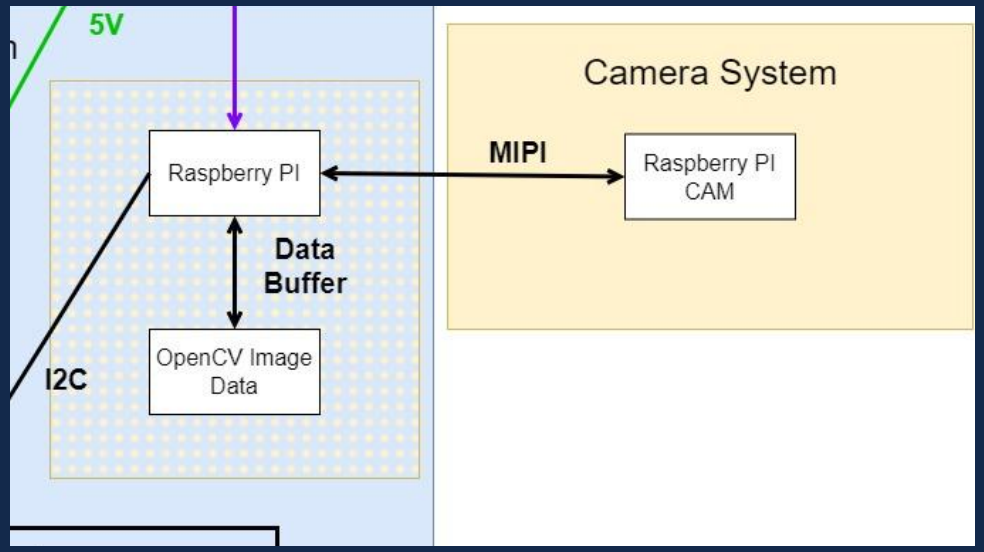
While fully operational with proper supply, ESP function well within our Requirements

- The current protection failed since DC motor would spike up to 0.9-1.1A

Camera Subsystem

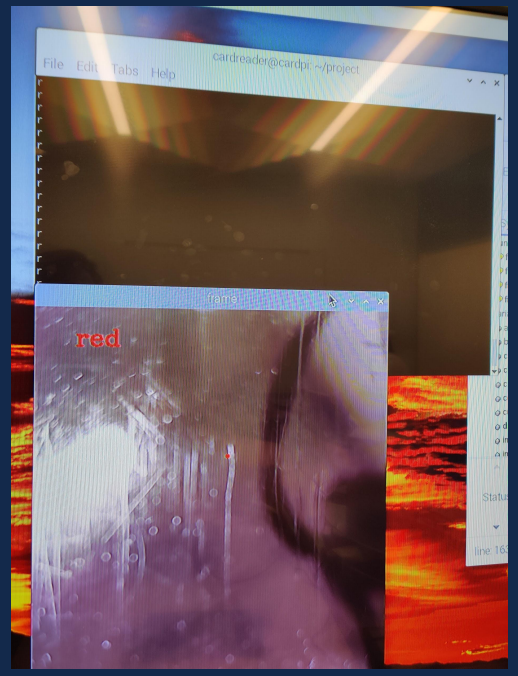
Requirements	Verification
• The pi should receive a consistent camera feed from the Pi Camera 2	• Pi camera 2 should be properly connected to the pi. • The video feed should appear on the screen and react to new objects in range of the camera at a reasonable frame rate and resolution.
• Pi should be able to process video feed from the camera in real time	• Video feed should appear after running the software. • Data parsed from the feed should appear, be correct, and updated frequently.
• Raspberry pi should be able to identify colors with a 90% accuracy	• Place cards into the slot to be identified using the camera subsystem and check to see if the color is correctly identified 90% of the time.
• Camera subsystem should be able to send parsed data from the camera feed to the ESP32 using i2c	• The SDA, SCL, and ground pins of the raspberry pi should be connected to their corresponding counterparts on the esp32. • Color data in the form of a character byte should be found on the serial output of the esp32. This byte should match the output of the raspberry pi.

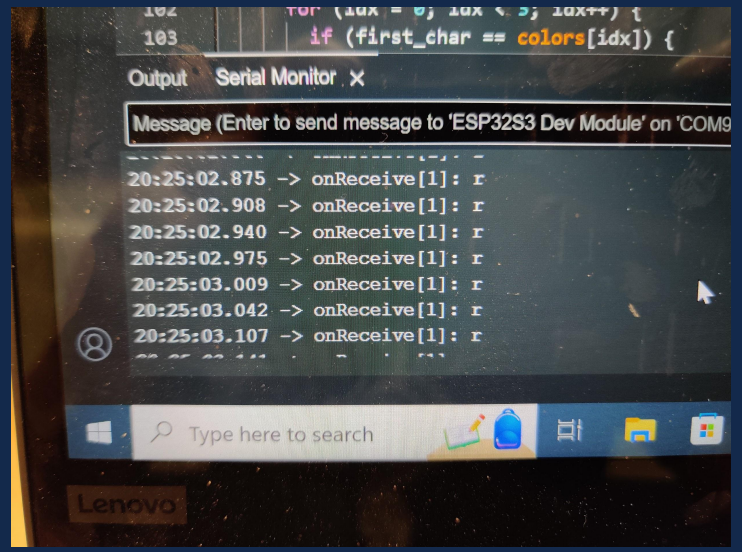
Camera Subsystem



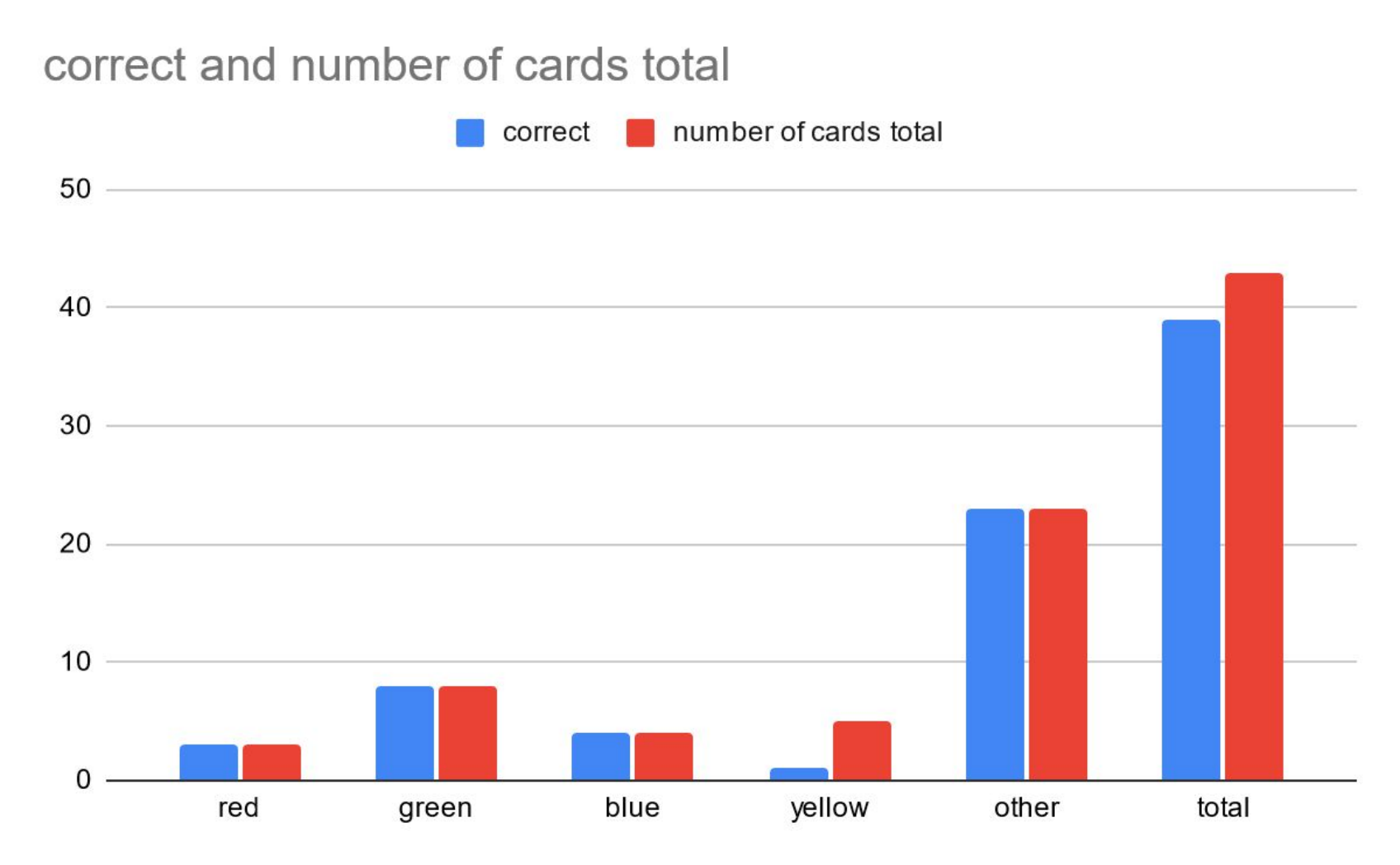
OpenCV

- Feature Detection using ORB (Oriented Fast and Rotated BRIEF)
- Image Pre-Processing
  - Gaussian Blur
  - Greyscale
  - Color-Space Conversion (HSV)
  - White-Balance Normalization





Camera Subsystem

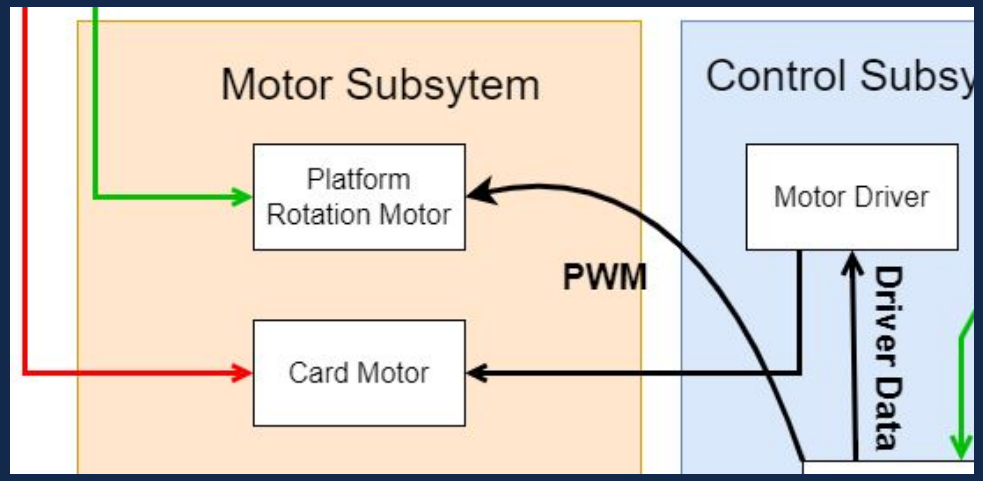


We had 100% success rate with:

- red
- green
- blue
- other

Overall we ended up with a 90.697% success rate

Motor Subsystem

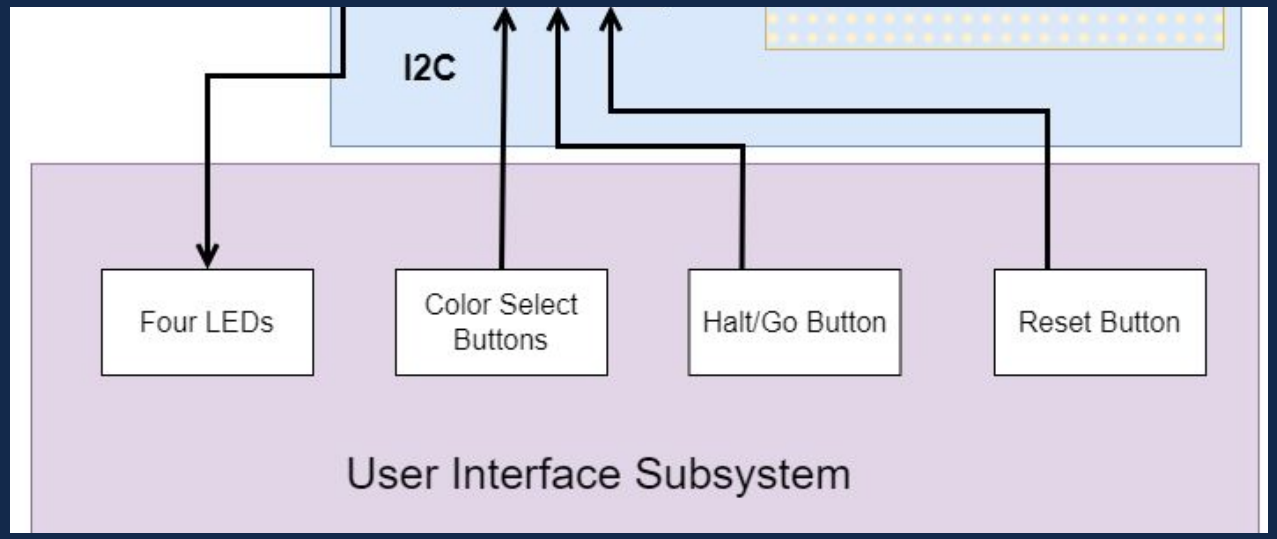


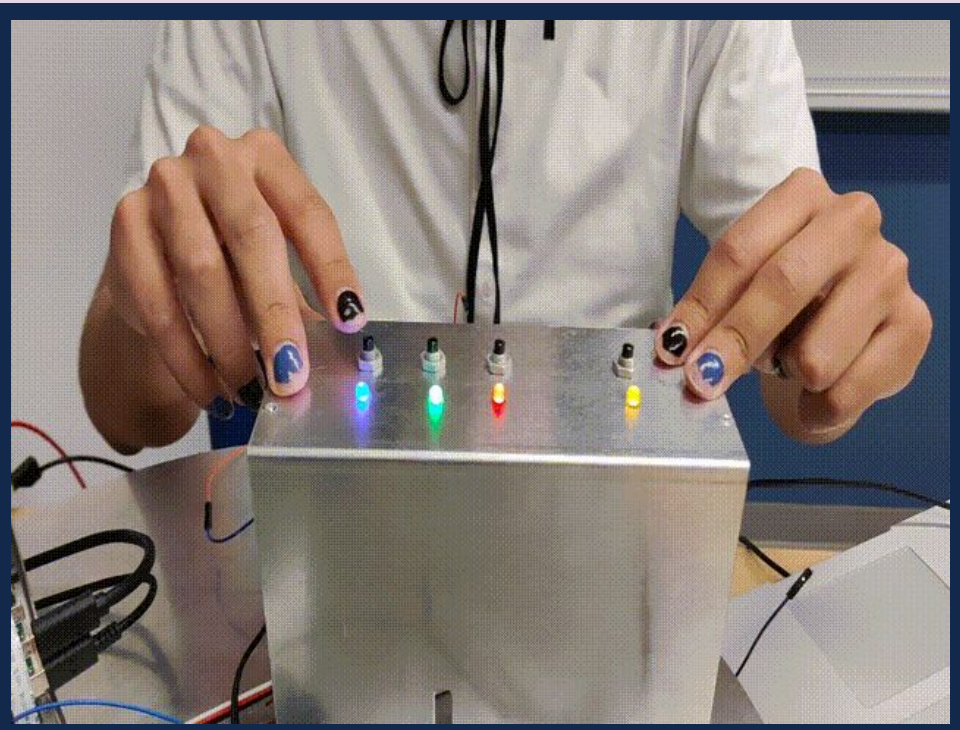
- Platform Servo motor turns the top level
- Card motor(DC) will move the cards out of their container onto a drop down
- ESP32 will control the both the Platform(servo) and Card mtor(DC) will be controlled through PWM signals

Motor Subsystem

Requirements	Verification
• The 5V servo should be able to reach all four card slots within it's operable range	• When running, the servo motor must be able to get the top plate in position to dispense a card in every position there is a card slot. • The reset button must stop operation and return servo to a default starting position
• The DC motor should be able to dispense one card	• When running, the DC must be able to dispense one card into one of the four card slot positions.

User Interface Subsystem

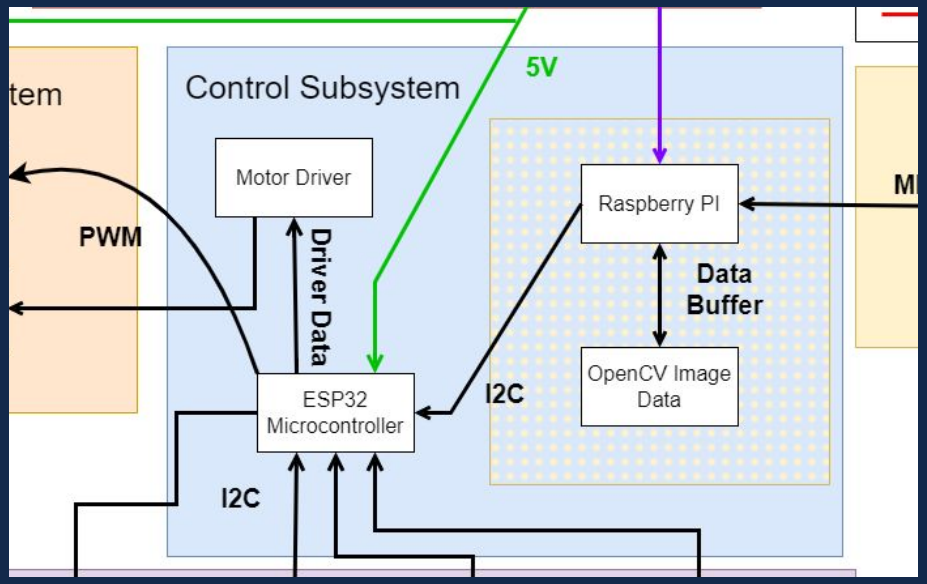




- Select which primary color to be sorted
- Halt/Go works as a start/pause for the user
- The reset button places the device back to
  default position for it to be restarted
- Four leds correspond to RGB-other and are used
  to display currently on sorting modes

User Interface Subsystem

Requirements	Verification
• Minimum response time of 0.5 seconds when switch is activated by user	• Onboard test circuit with led will verify if switches and the software can display the light up of the led within 0.5 seconds
• System must be able to halt within a 1 second	• When running, press switch and measure time to stop for all motors.



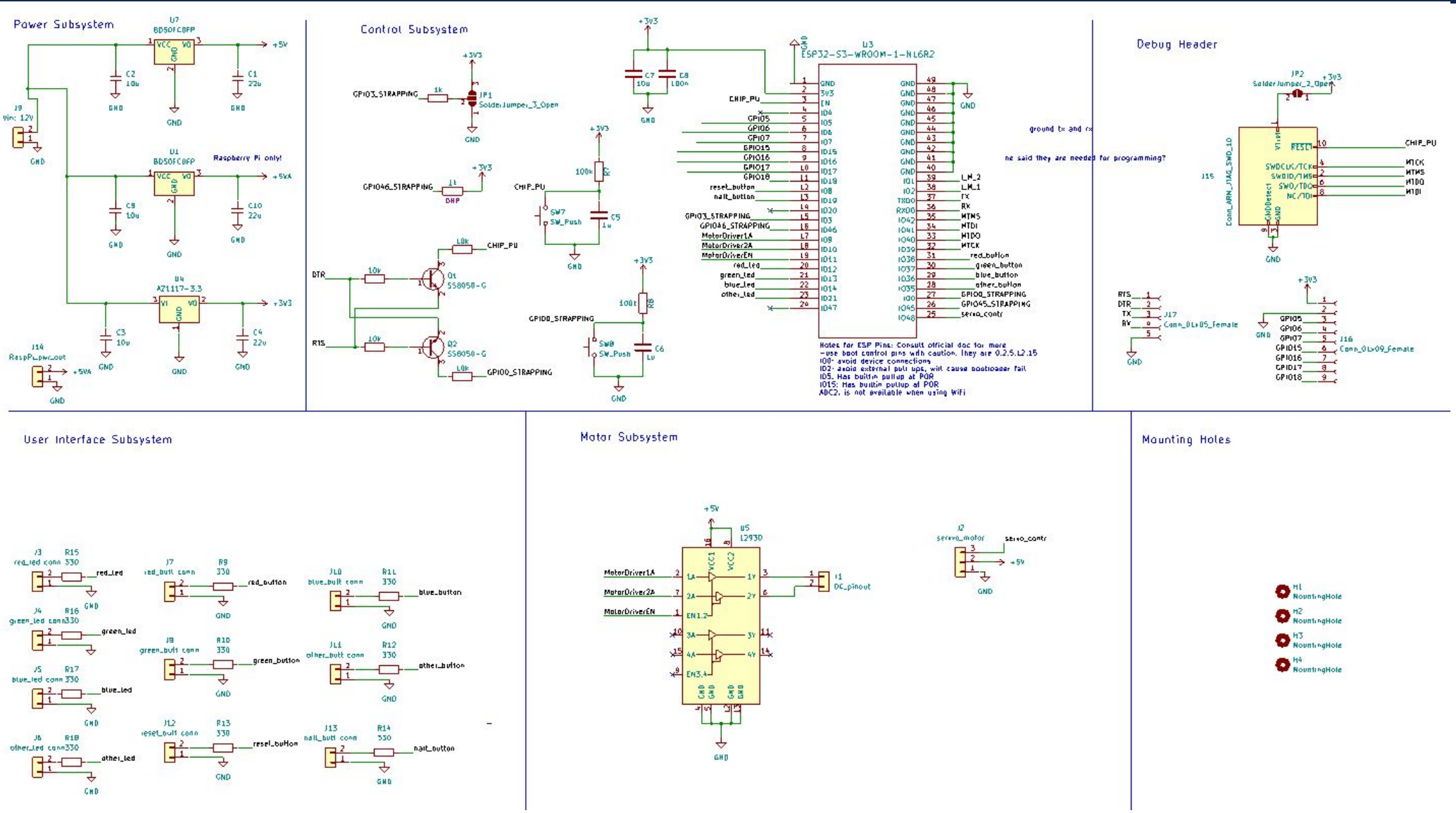
Control Subsystem

- Takes in Raspberry Pi I2c data stream
- Sends PWM signals to both motors
- Listens for user input from buttons and displays LEDs accordingly

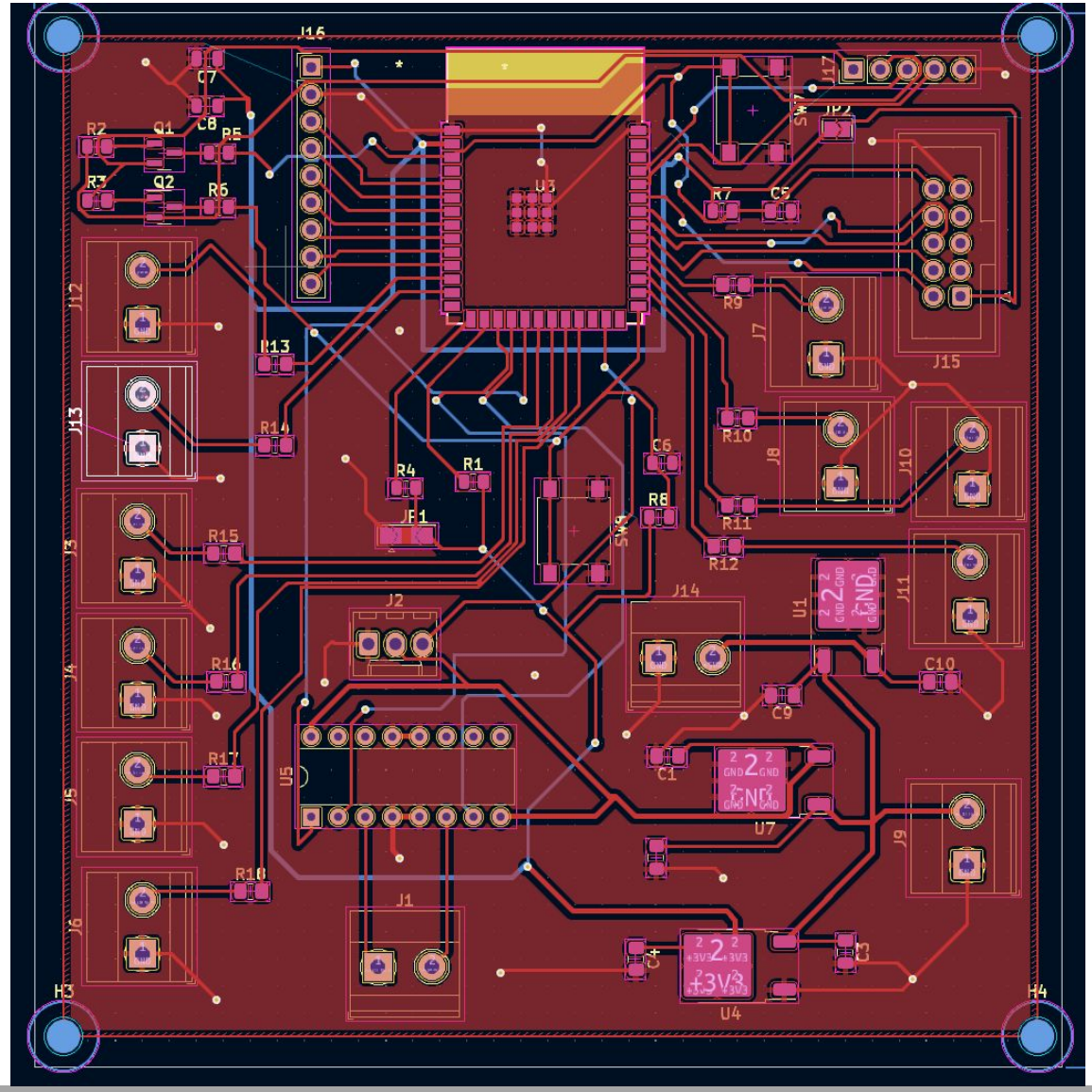
Control Subsystem

Requirements	Verification
• The ESP32 should be able to receive data from the pi through i2c.	• The SDA, SCL, and ground pins of the raspberry pi should be connected to their corresponding counterparts on the esp32. • Data should be able to read through the serial out and match what the camera subsystem sees
• The ESP32 should be able to use input from the User Interface to change the categories cards are going to be sorted by.	• After giving user input, the cards should be sorted by the machine into groups based on the input. If the user pressed buttons for only red cards, the cards will be sorted into a red card pile and an “other” card pile.

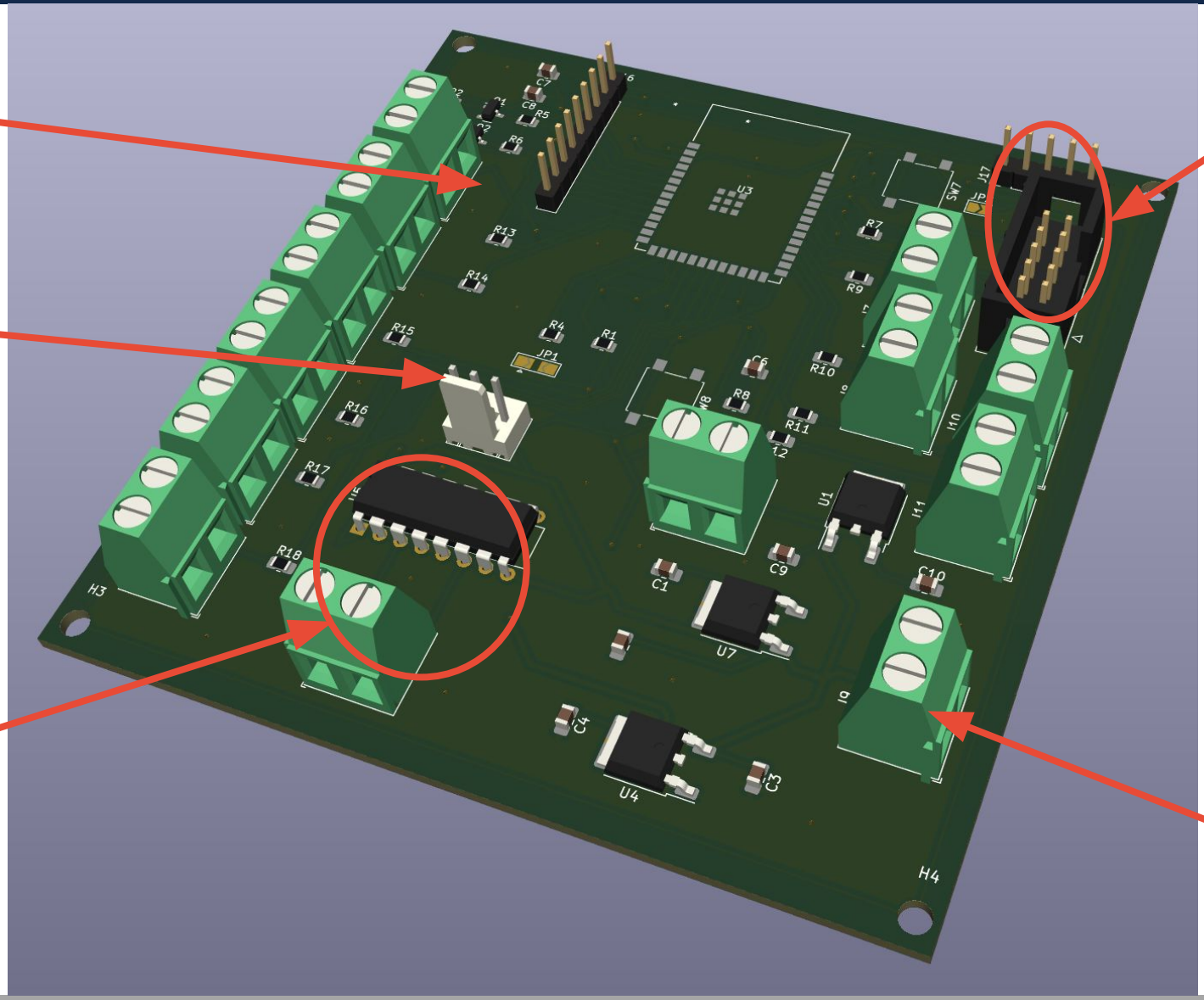
PCB Sketch


Power Subsystem
Control Subsystem
Debug Header
User Interface Subsystem
Motor Subsystem
Mounting Holes
ESP32-S3-WROOM-1-N16R2
L293D
AZ1117-3.3
BD50FC0FP
SS8050-G
Raspberry Pi only!
CHIP_PU
GPIO3_STRAPPING
GPIO46_STRAPPING
GPIO0_STRAPPING
GPIO45_STRAPPING
GPIO0_STRAPPING
reset_button
wait_button
red_button
green_button
blue_button
other_button
red_led
green_led
blue_led
other_led
servo_contr
MotorDriver1A
MotorDriver2A
MotorDriverEN
Conn_01x05_Female
Conn_01x09_Female
MountingHole

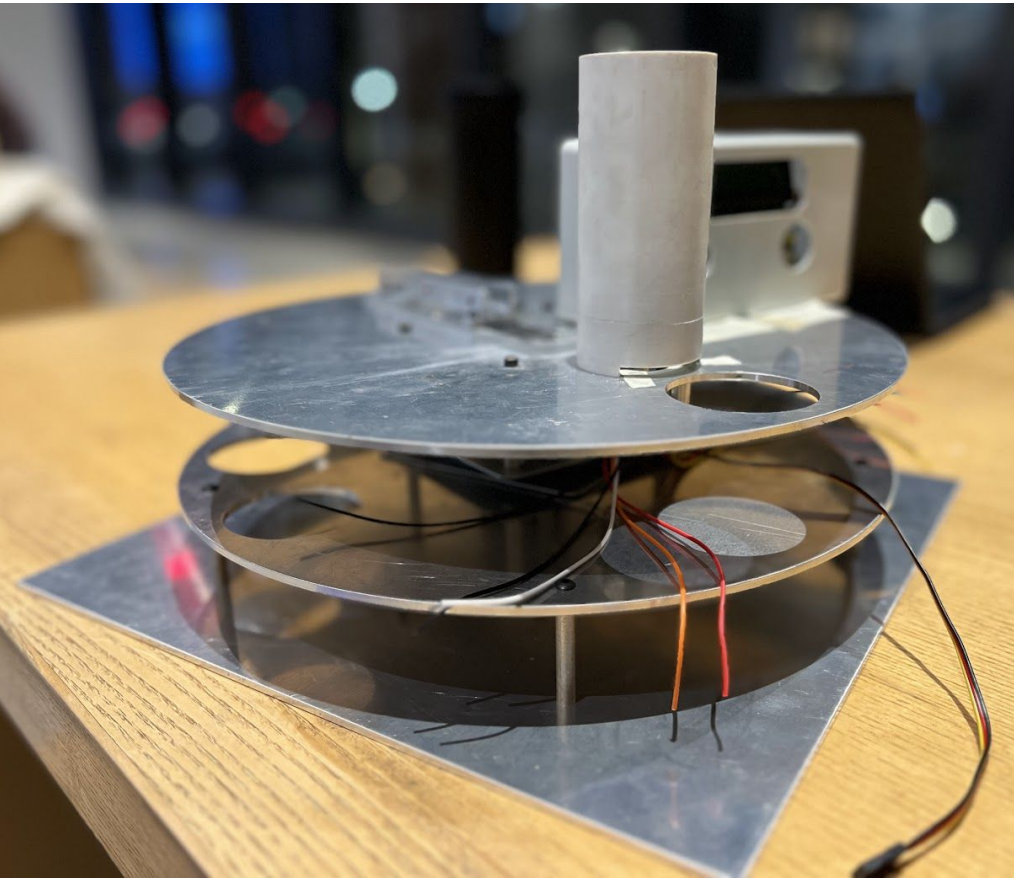
PCB Sketch


J16
C7
C8
R2
Q1
R5
R3
Q2
R6
U3
R7
C5
JP2
J17
R9
J7
J15
J12
J13
R14
R10
J8
J10
C6
R4
R1
R8
JF1
R11
R15
R12
J3
J14
J11
U1
J2
J4
R16
C10
C9
C1
R17
U5
U7
J5
J9
R18
J1
J6
C2
U4
H3
H4

PCB Sketch

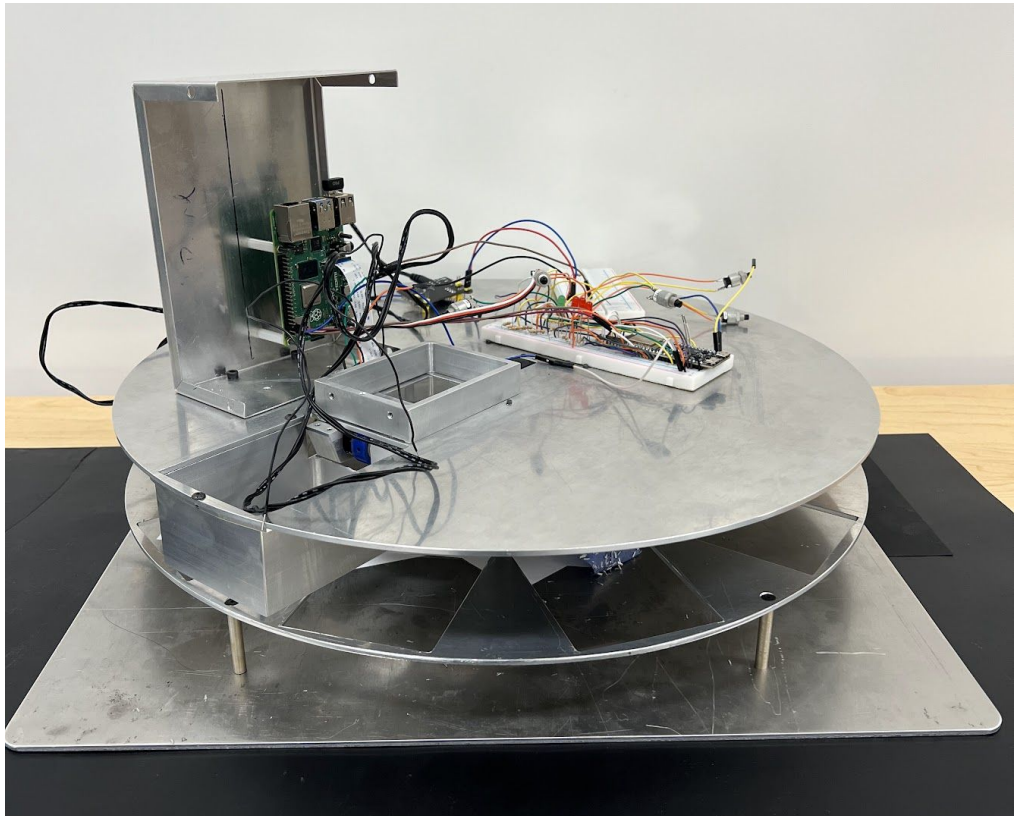


The Build and Functionality



Initial Design

- Based off previous project
- Wanted the device to be simple to ensure that mechanical kinks could be worked out
- Initial size too small
  - Could not fit card

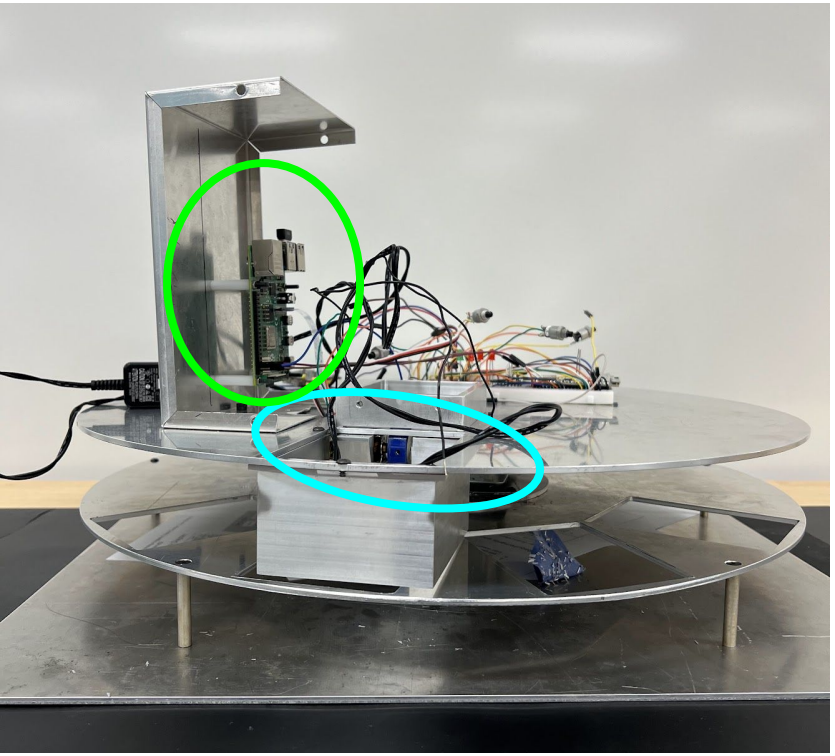


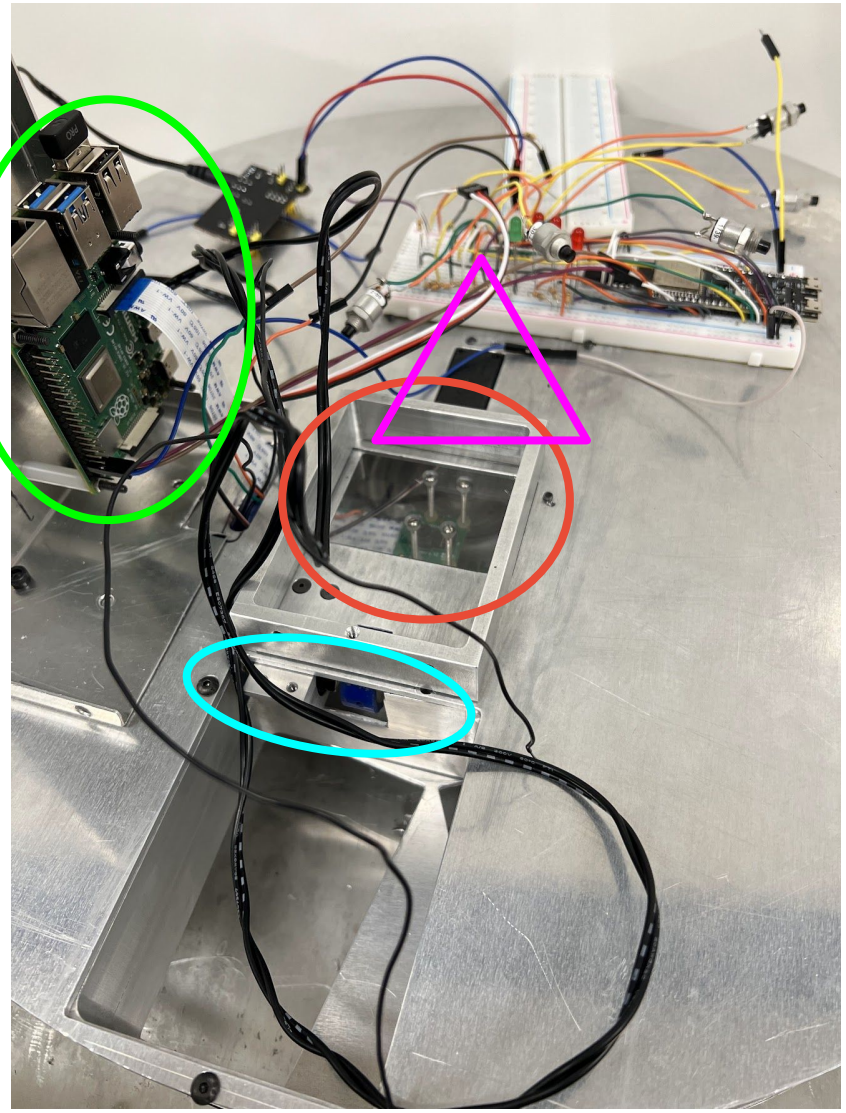
Mechanical Design

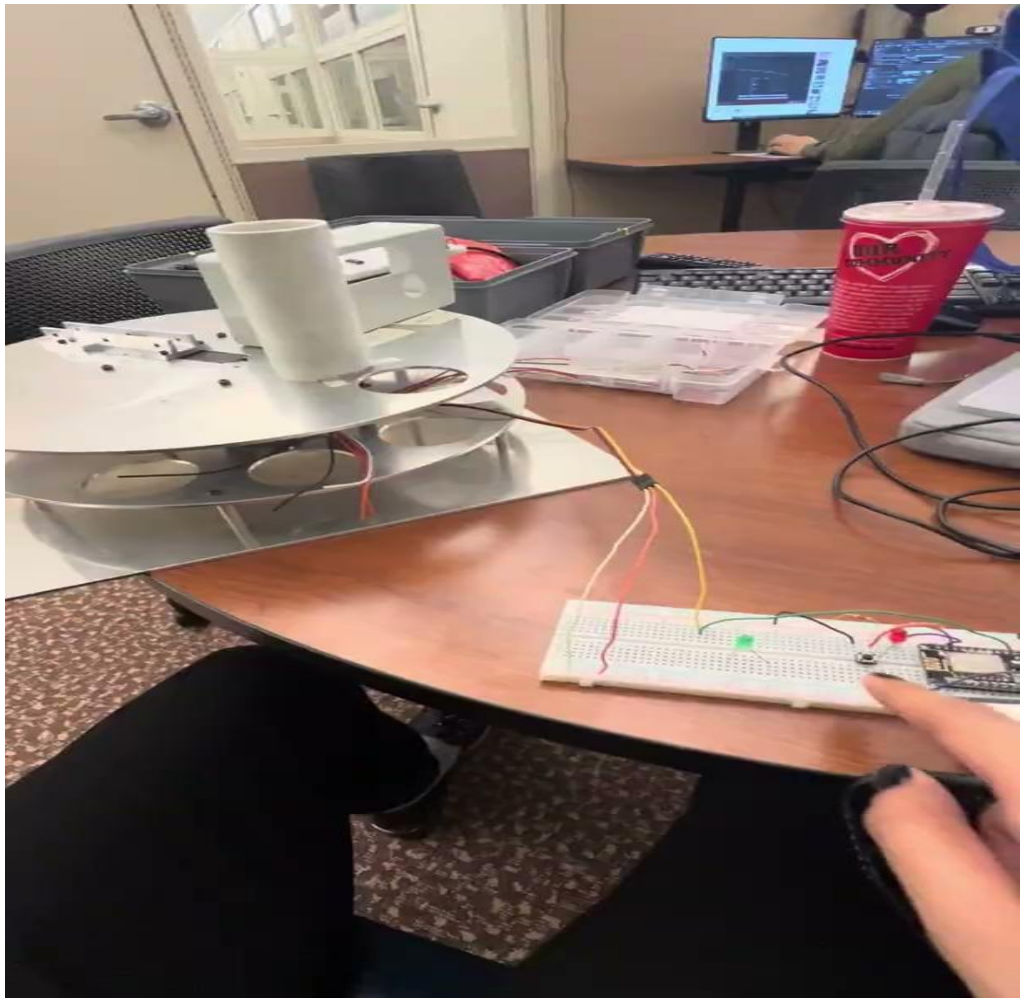
- Material: CNC-cut metal plates
  - Durability, precision, safety
- Base Design: Rotating base
  - smoother card sorting operations
- Card Drop Shoot: 2.5 inches by 3.5 inches

Device Build

Build Runthrough

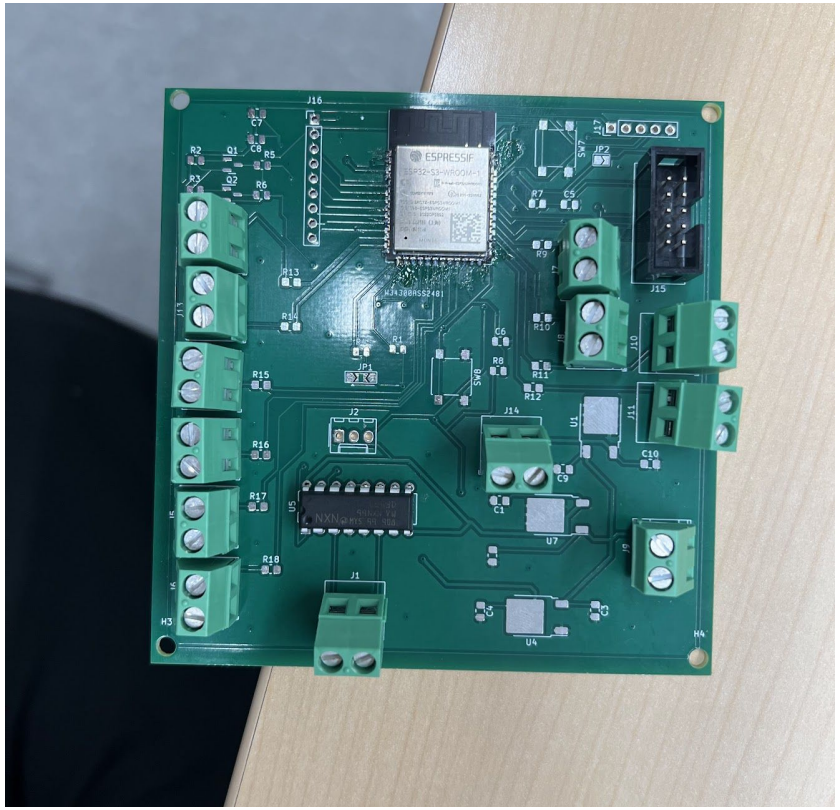






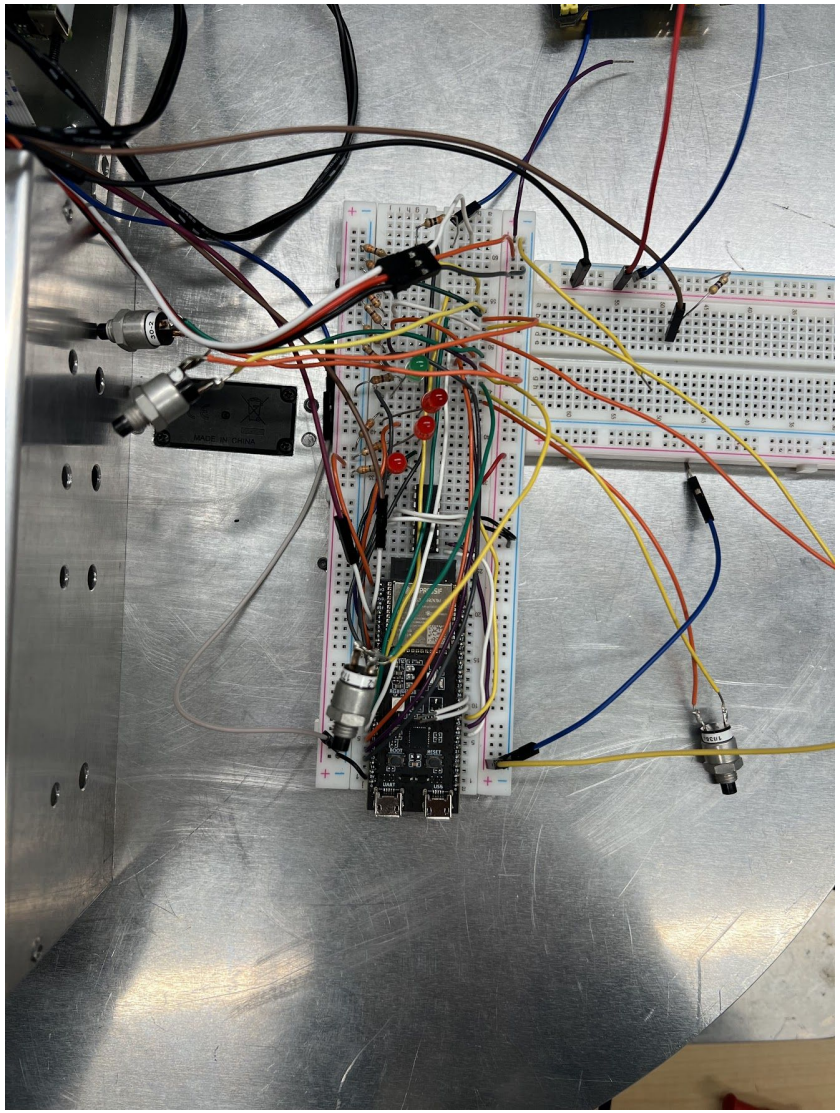
Electrical Prototyping

- Used ESP8266 to test out initial machine shop build
  - Ensure both servo & a DC motor work
- Used button and led circuit to test user control

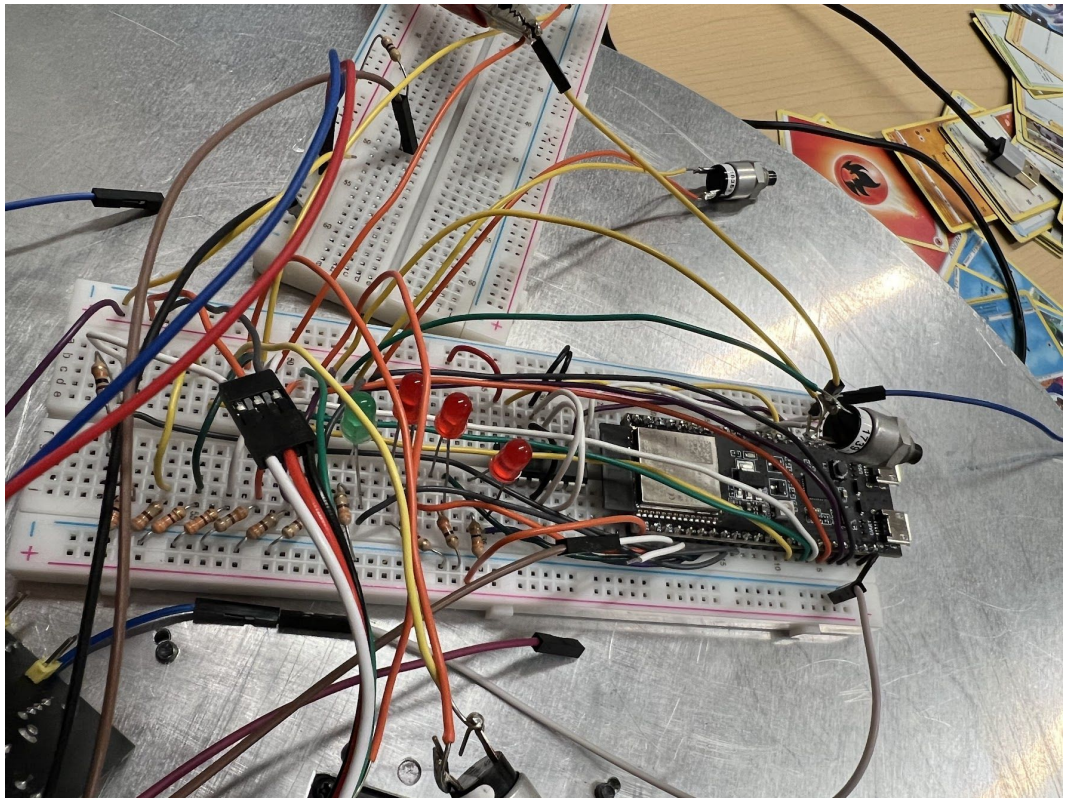


Initial Hardware Layout

- Initial PCBs had incorrect power line configuration
- Changes were redone and and new PCB was ordered in the 5th PCB wave
- With machine shop timings and falling breadboard circuitry, the PCB was set the side to ensure usable product

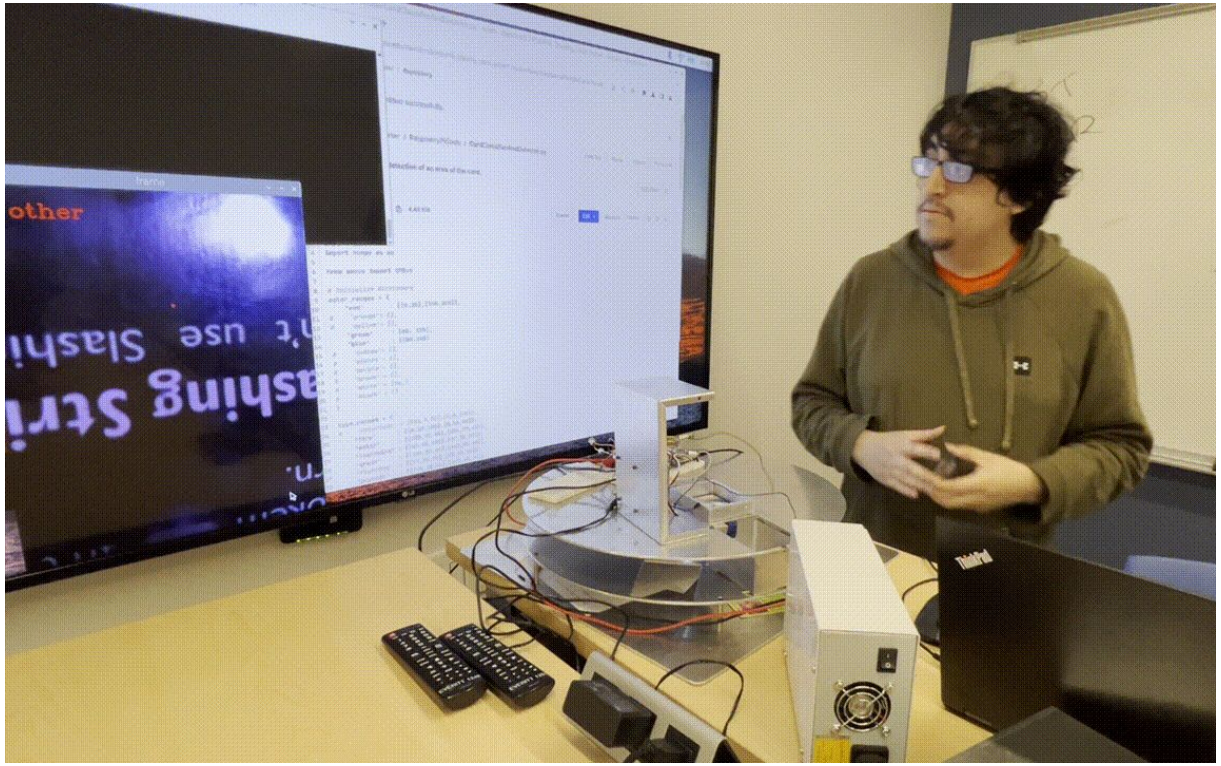


Final Hardware Layout



User Interface Mechanism

1. Select between 4 color options
   a. Red, Green, Blue, Other
2. Additional start/stop and reset buttons
3. LEDs display user choices and testing

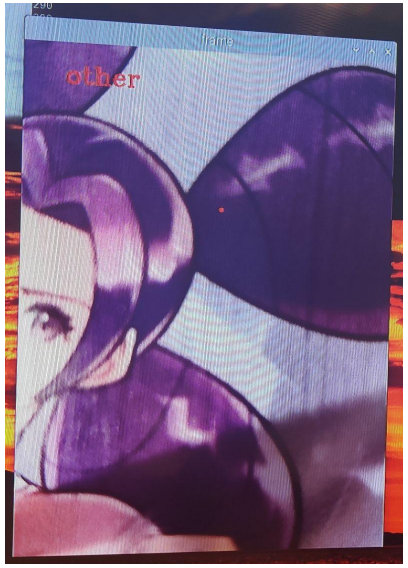


Sorting Mechanism

- Raspberry Pi
  - OpenCV checks hue color from video feed
    - Chooses between “R”, “G”, “B”, or “O”
  - Sends chosen color to ESP32
    - I2C communication
- ESP32
  - Compares color to user-selected colors
  - Moves to specific color bin
    - Else sort to final slot

Challenges and Progression

Camera Focal Length



Problem:

- Unreliable feature detection
  - Camera positioned further away from card than expected
  - Noise added to camera feed
    - Plexiglass scratches

Things learned:

- The image can be preprocessed with:
  - gaussian blur
  - white-balance corrections
  - grayscale color space
- Preprocessing can only go so far
  - “Garbage in, Garbage out”
- Test out distance and focal length of a camera before designing

Solution:

- Drop feature detection, focus on average hue value

Power Subsystem

Problem:

- DC motor pulled more power than anticipated
  - Shorted microcontroller
  - Power supply inefficient for the DC motor
  - Low current on all components

Things learned:

- Add more flux protection to components interacting with the higher power supply
- Test other lower power components to lower current draw

Solution:

- Current-limiting circuit
- Use a stepper motor and daughter board instead of DC motor
- Higher current power supply to properly power all components

Shorted ESP →

Geometric Design

Problem:

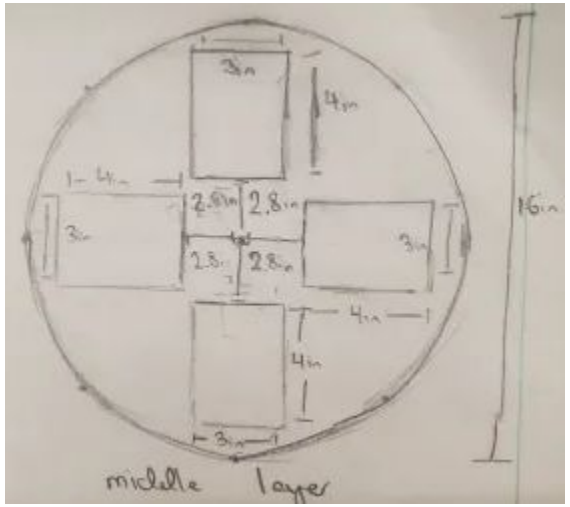
- Initial design could not fit all four slots
  - Physical size too small to fit card slots and holder
  - Servo motor scope of rotation
    - Limited range to move eliminated cardinal slot placements

Things learned:

- More detailed blueprints results in quicker realizations

Solution:

- Increase disk radius
  - Allows card slot and holder to fit
  - Reduces range servo motor must move



First design of middle layer

Moving Forward

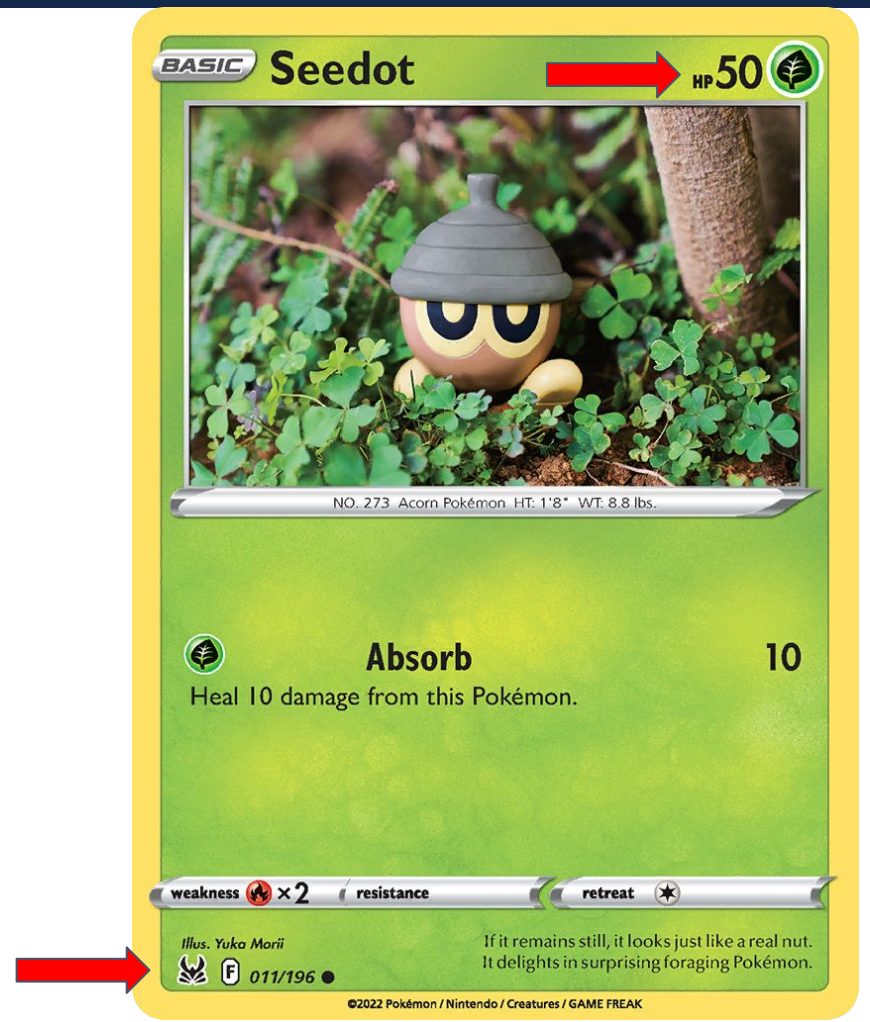
What needs to be done to be consumer friendly

- A simple, safe and tested power subsystem
- More Intuitive UI
- Redesign the motor subsystem to not damage the card

Current UI design

Redesigning for the future

- Pick a more robust card dispensing motor
- Further implementing feature detection to sort by:
  - Set
  - Type
  - Value
- Single power system for the whole machine

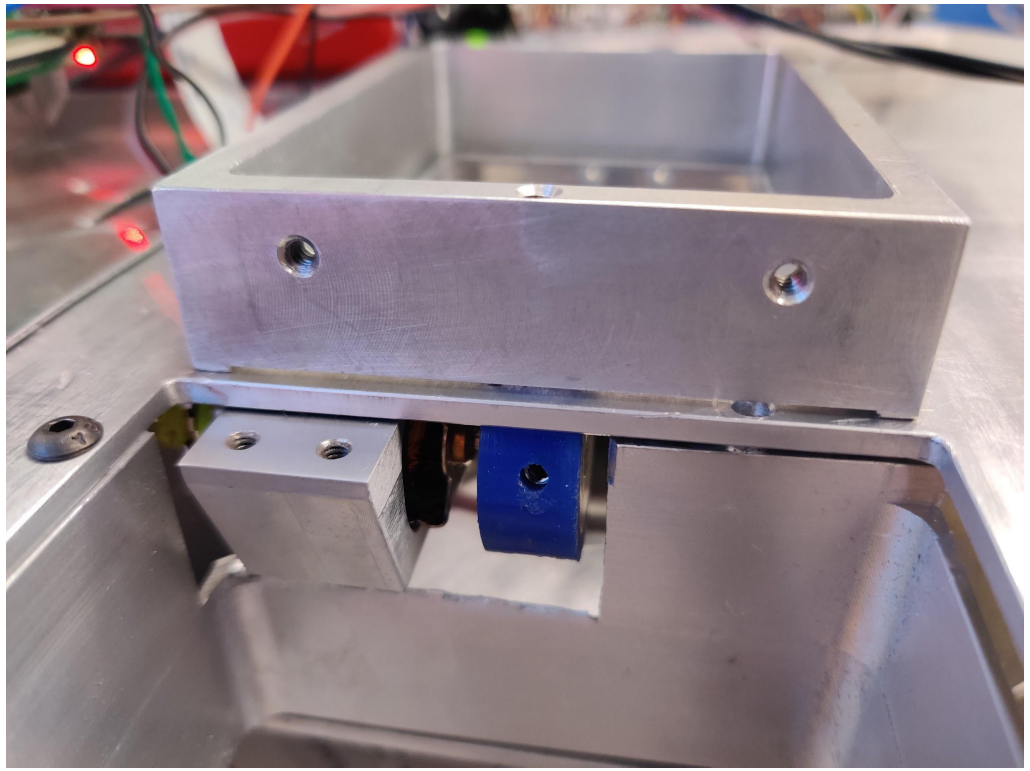


TCGPlayer Near Mint Price: $0.01

Areas of improvement

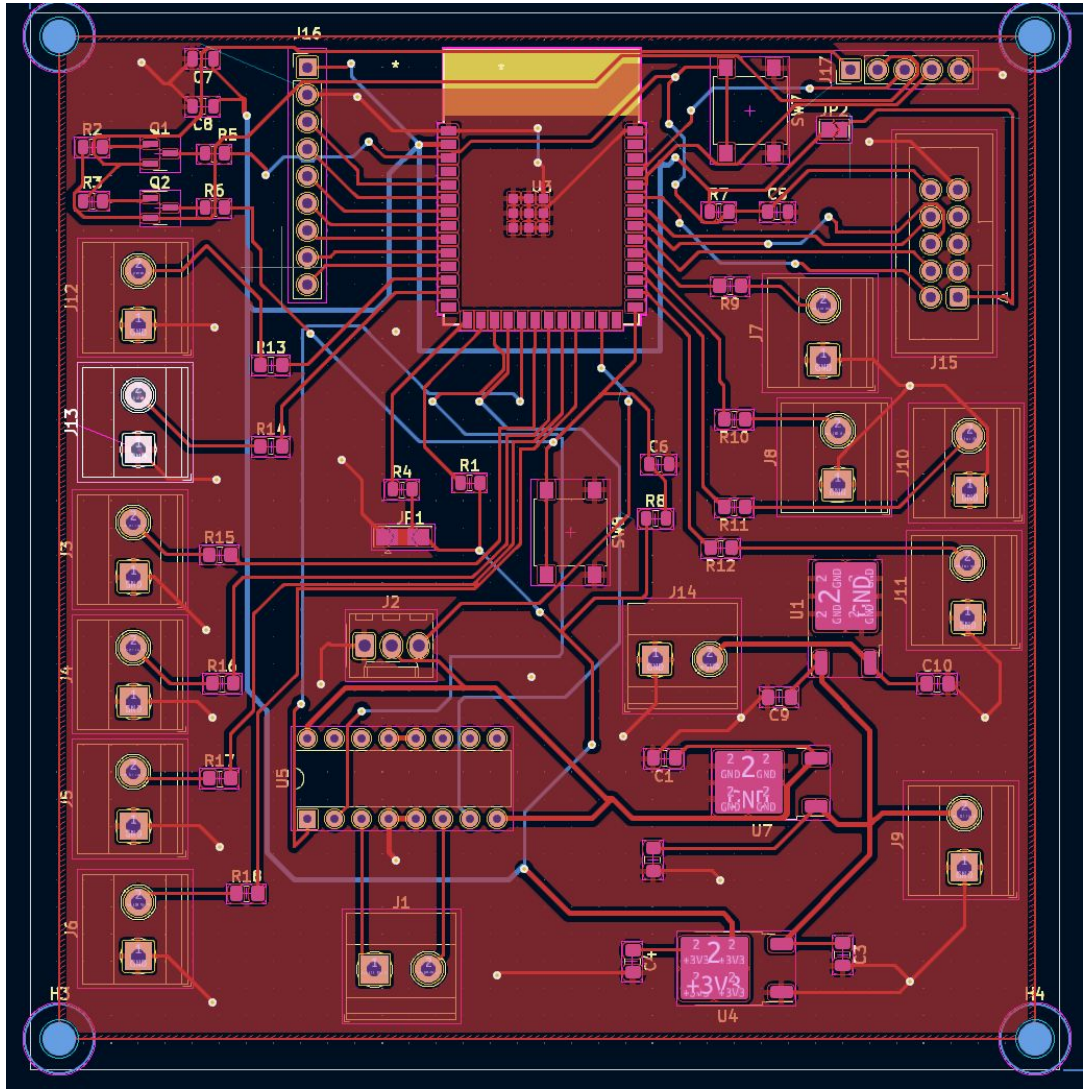
- Speed
  - Lack of torque on DC motor

- Accuracy
  - Size of DC motor wheel
  - Size of gap to shoot cards out of



Unified PCB

- Have one power source provide power to the Raspberry Pi, ESP, and motors
- Ensures faster prototyping and manufacturing
- Increased reliability



Conclusions

What we learned:

- Set aside more time for:
  - learning
  - mistakes
  - troubleshooting
- Test parts for intended use even after calculations
  - Datasheets don’t give the full story
- Building modularly makes testing easier
- Integrate separate systems together sooner

What we would have done differently

- Spend less time on the PCB design and more time communicating with the machine shop and working a physical prototype

- Initially design PCB with less features
  - Ensure the MCU could be programmed

- Start small with ESP32 programming
  - Build small functions to build upon

Favorite parts for each member:

- Andrejun
  - I enjoyed learning about openCV and i2c since they feel like things I can implement on personal projects going forward
- David
  - Loved being able to wire and test with something physical
- Steve
  - Programming the ESP32 and seeing it work with the circuit

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN