UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Automatic Card Sorter

By Alfred Hofmann, Kyle Mahler, and Rocky Daehler

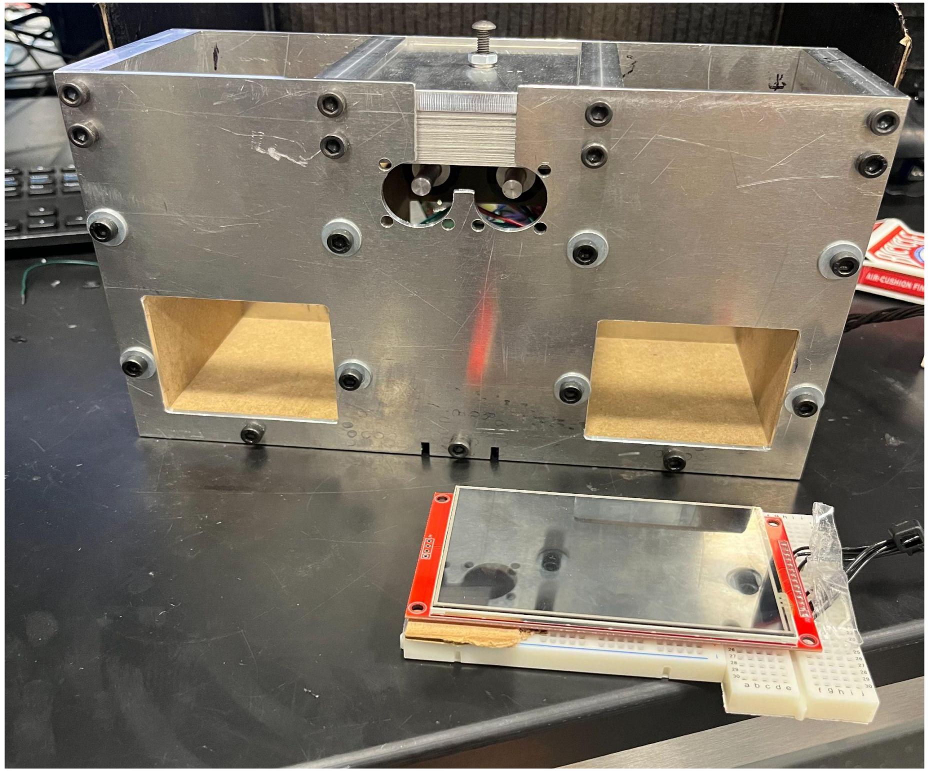
Group #37

5/6/2025

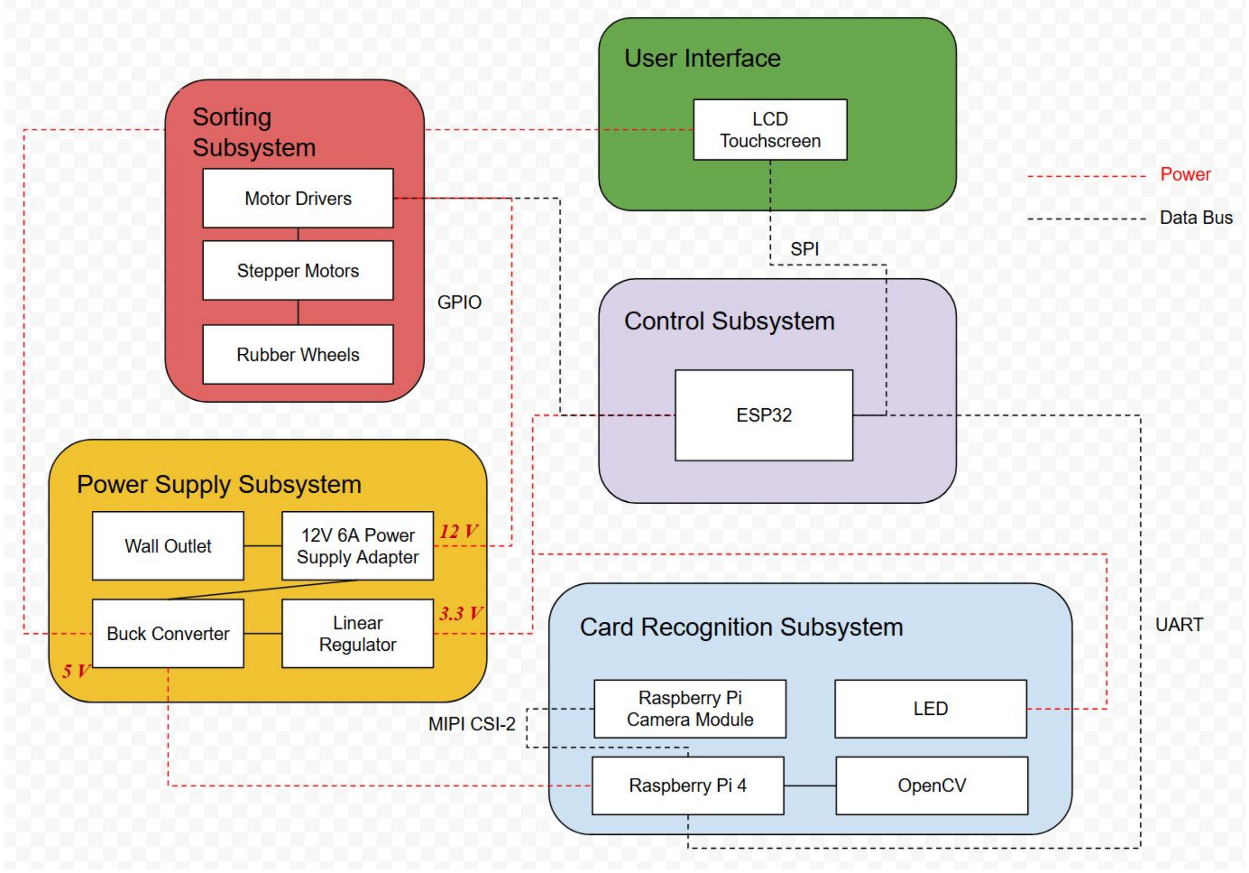
Objective

Several popular card games use only a fraction of the entire 52 card deck, while many other use all the cards. Going through and sorting all these cards can be a tedious process. This project helps to sort the cards automatically for these games to alleviate the tedious work of sorting the cards.

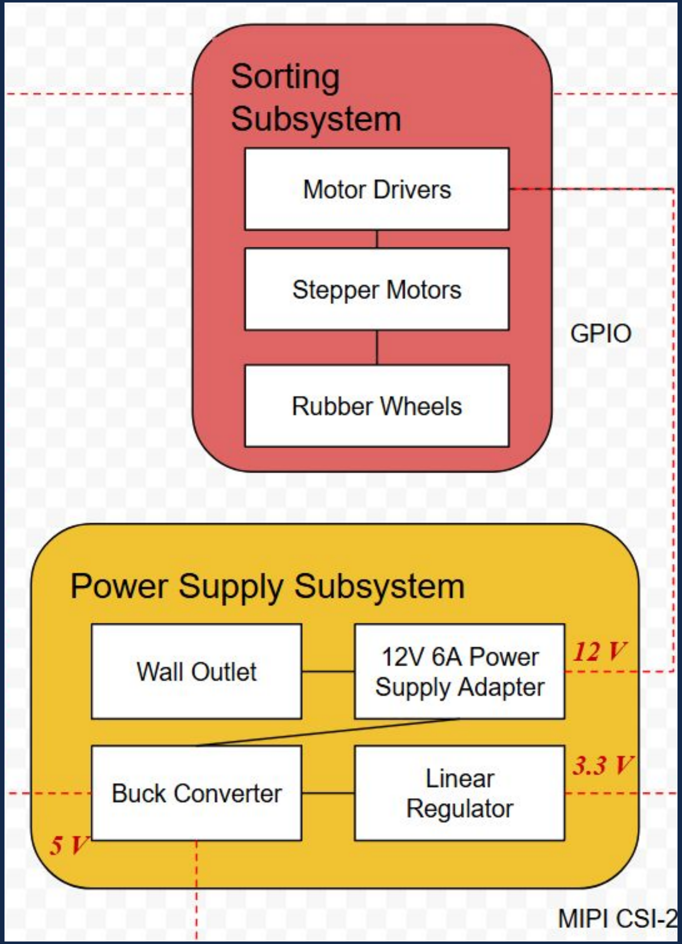
Design



Physical Design



Block Diagram



Sorting Subsystem Requirements

- Can move a card to either side based on an electronic input.
- Can move just the top card of a stack of cards without affecting the rest of the stack

Power Supply Subsystem Requirements

- Powers everything without overheating or slowing down the system.

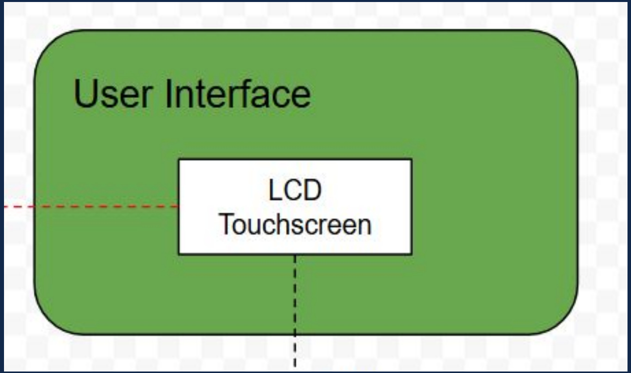
User Interface Requirements

- The user can select a game to sort the cards for.

- When ‘Start’ is pressed, a signal communicating what game has been selected will be sent to the Control System.

- Displays an error message when given a signal from the Control System.

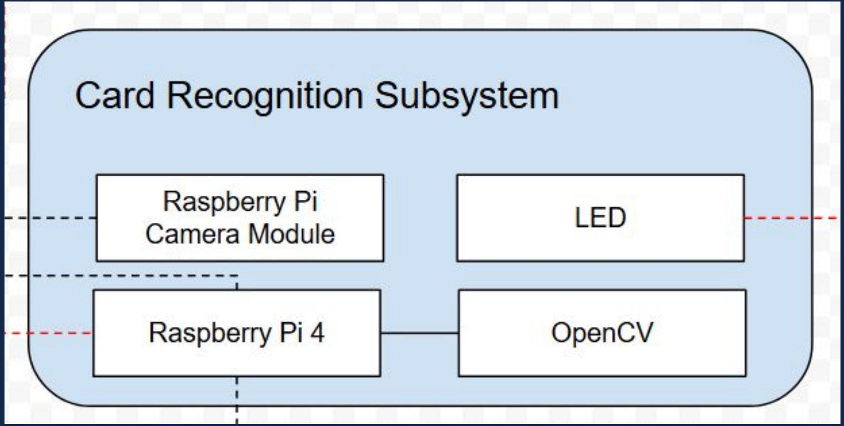
    - Displays multiple different error messages when given different signals. These include:
        - A jam
        - An unrecognized card



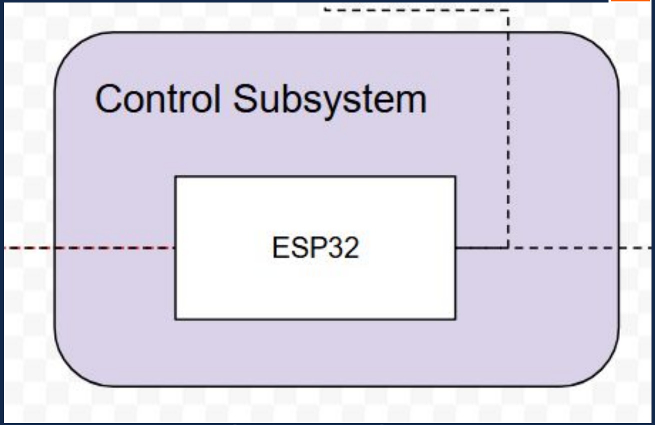
Card Recognition Subsystem Requirements

- Takes a clear picture of the card corner and sends the card data to the Raspberry Pi

- The Raspberry Pi can identify the rank and suit of a card and send that information to the control system upon request



Control System Requirements



- Prepares what cards to accept and start sorting when given a signal from the User Interface.
- Sends a signal to the Camera System to recognize a card after the Sorting System sorts a card.
- Analyzes the received card data and give one of two signals to the Sorting System (indicating which direction to move the card).
- Recognizes various errors and sends a signal to the User Interface when they occur. The detectable errors include:
  - A jam
  - An unrecognized card

Changes made to our original design

- Errors
  - Got rid of the ‘missing card’ error
- System no longer stops on its own
- Camera system
  - Light no longer flashes

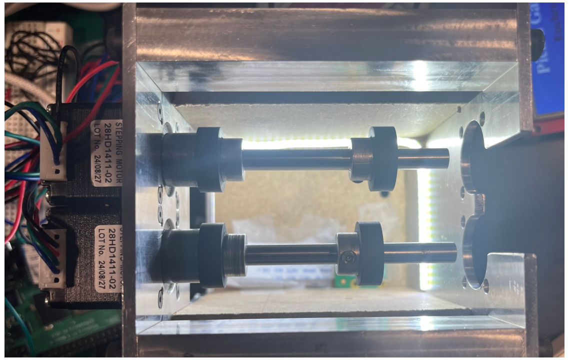
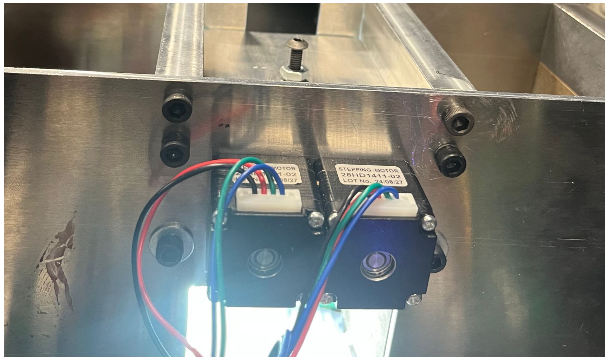
Project Build

Sorting System

Sorting

Build Overview

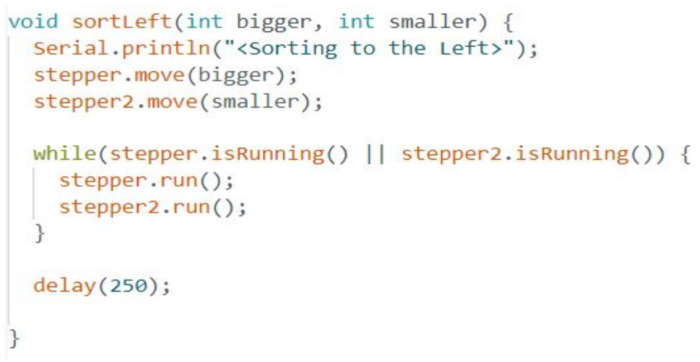
The sorting subsystem consists of three main components, one being two bipolar stepper motors (1528-1062-ND from Digikey), and the other hardware component being its drivers. For the drivers, we landed on using the DRV8825 from Pololu. Finally, the ECE machine shop provided axles and rubber wheels to connect to the stepper motors to complete the sorting subsystem.

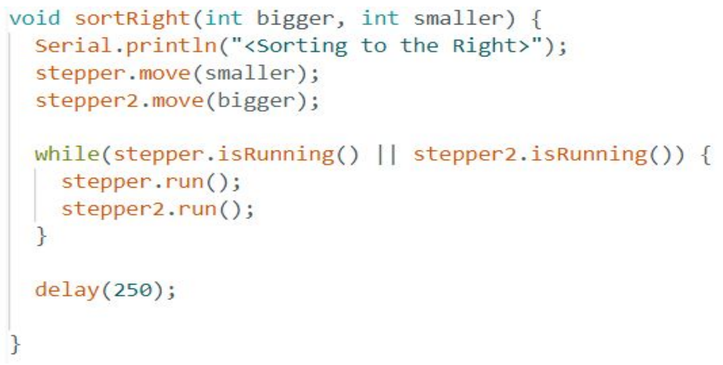
Motor Drivers

Software Overview

In order to utilize the motor drivers to accurately control the motors how you want to, you must utilize a small amount of software on the microcontroller. We used the AccelSteppers library on our ESP32 for this. This software was fairly simple and small compared to the other software components, however extremely important to the success of the project as a whole.



```
void sortLeft(int bigger, int smaller) {
  Serial.println("<Sorting to the Left>");
  stepper.move(bigger);
  stepper2.move(smaller);

  while(stepper.isRunning() || stepper2.isRunning()) {
    stepper.run();
    stepper2.run();
  }

  delay(250);

}
```



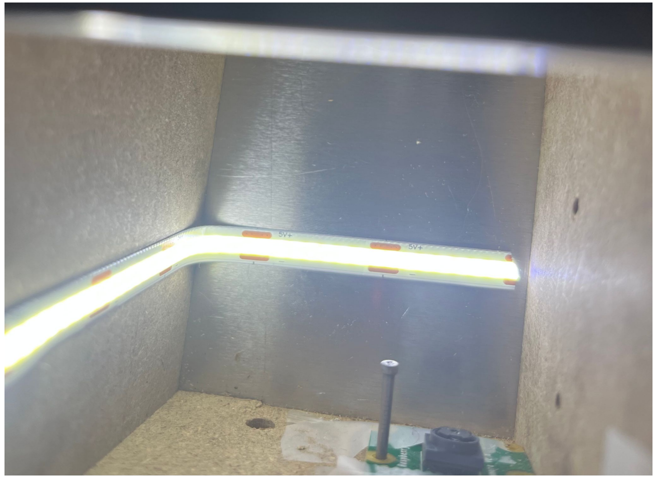
```
void sortRight(int bigger, int smaller) {
  Serial.println("<Sorting to the Right>");
  stepper.move(smaller);
  stepper2.move(bigger);

  while(stepper.isRunning() || stepper2.isRunning()) {
    stepper.run();
    stepper2.run();
  }

  delay(250);

}
```

Card Recognition System

Card Recognition

Build Overview

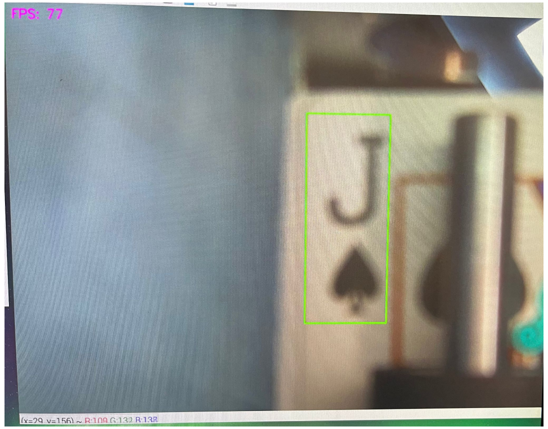
The Card Recognition subsystem includes a Raspberry Pi 4, a Camera Module, and an LED strip to illuminate the cards

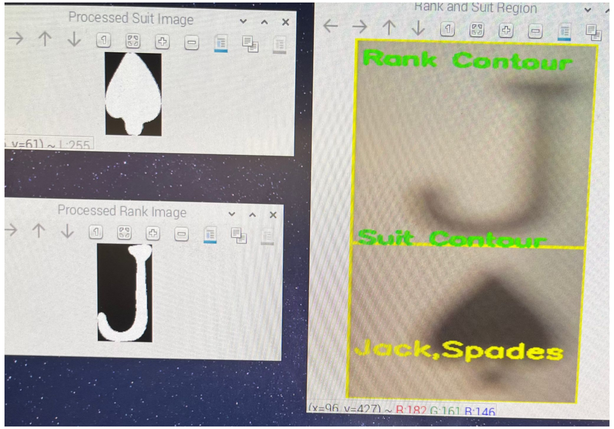


Card Recognition

Software Overview

The camera is placed inside the device, placed at the bottom pointing up at the corner of the playing cards where the suit and rank are located. The Region of Interest (ROI) is extracted where the suit and rank are located, then is preprocessed and split between the rank and suit region. Each is cropped to the largest contour, and matched against templated images to determine the correct classification.



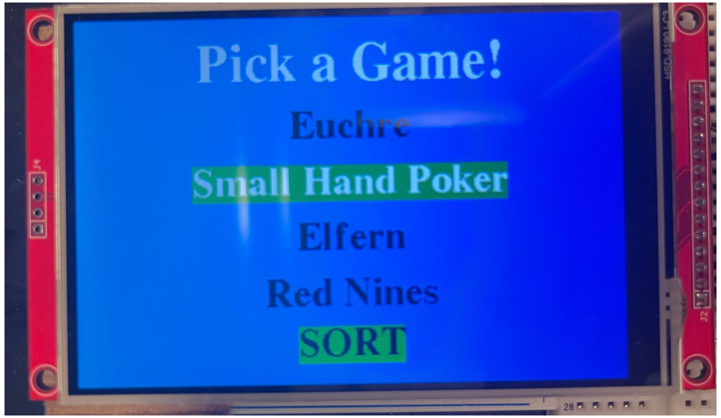


User Interface

Build overview

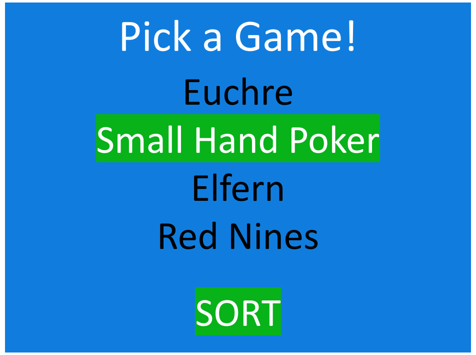
The User Interface consists of a touch screen connected to the ESP32 via an SPI connection.

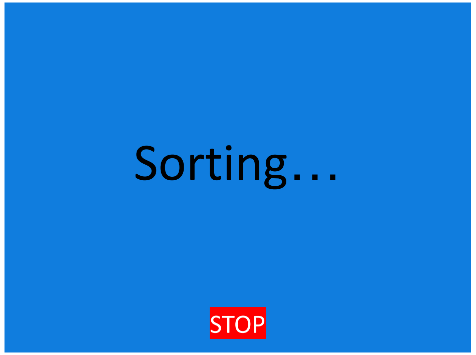
- Hosyond 4.0” Touchscreen
  - ST7796 SPI driver



- TFT_eSPI Library
  - Drawing functions
    - Shapes
    - Text
  - Touch screen functions
    - Reading touch inputs
    - Calibrating touch sensors
    - Getting x and y inputs

SORT button highlights when a game is selected

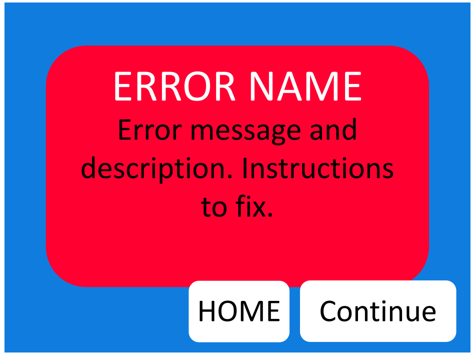




Sorting animation plays so user knows the device is still operating

Colors are communicated via 565 encoding:

- 5 bits of red
- 6 bits of green
- 5 bits of blue


ERROR NAME

Error message and description. Instructions to fix.

Error name is prominent and a description telling the user what to fix appears

Screen mockups

Control System

Control System

Connections

The control system is responsible for listening to and controlling the actions of the other subsystems.

Inputs

- Touch screen inputs
  - Game select
  - Start sorting
  - Stop sorting
  - Error handling
- Card reading
  - Rank and suit signal

Outputs

- Control motors
  - Sort left or right
- Request card
  - Signal to Raspberry Pi
- Raise Errors (internal)
  - Jam error
  - Unrecognized Card error

Control System

Sorting Mechanism

- One hot encoding used for different cards

X	X	X	A	K	Q	J	10	9	8	7	6	5	4	3	2

  - 0x1000 indicates an Ace
  - 0x0001 indicates a 2
  - Upper most 3 bits reserved for other uses
    - 0x2000 indicates unrecognized card
- Games encoded as bitmask of what cards are accepted
  - **Euchre** uses 9s through Aces

0	0	0	A	K	Q	J	10	9	8	7	6	5	4	3	2
0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0

    - 0x1F80 encoding

Sorting Mechanism

- To check card acceptance- Bitwise AND game bitmask and card rank encoding
  - Example for an **8** with the game **Euchre**

X	X	X	A	K	Q	J	10	9	8	7	6	5	4	3	2
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0

=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

=0x0000 → Not accepted

Control System

Software

- Pseudocode diagram

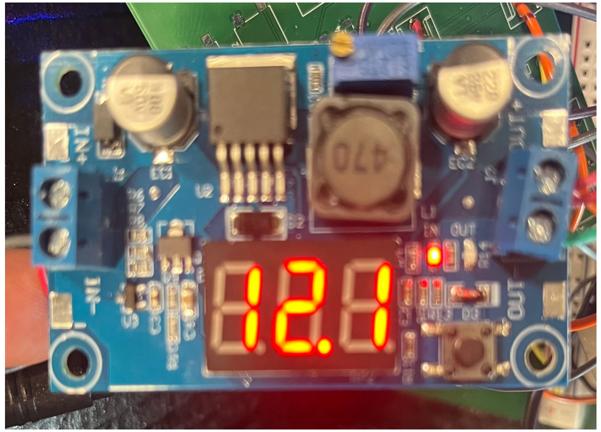
```
Setup()
    Initialize UART connection with
Raspberry Pi

    Set motor Max Speed
    Set motor Acceleration

    Initialize SPI connection
    Set screen rotation
```

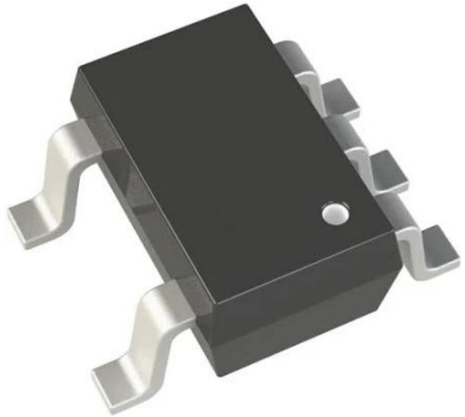
```
Loop()
    If in sorting state
        If message on UART:
            If unrecognized:
                Raise unrecognized error
            Else If same card as previous card
                Raise Jam error
            Else
                Sort card based on selected game

    If screen touched
        Get x and y
        Handle x and y based on current state/screen

    If should refresh
        Draw screen based on UI_State
```

Power System

Build Overview

12V is brought in from a wall outlet power adapter. A buck converter brings this down to 5V. Finally, a linear regulator is used to take the 5V down to 3.3V.





$$\begin{tikzcd} 12V \arrow[r] & 5V \arrow[r] & 3.3V \end{tikzcd}$$

Successes and Challenges

Successes and Challenges

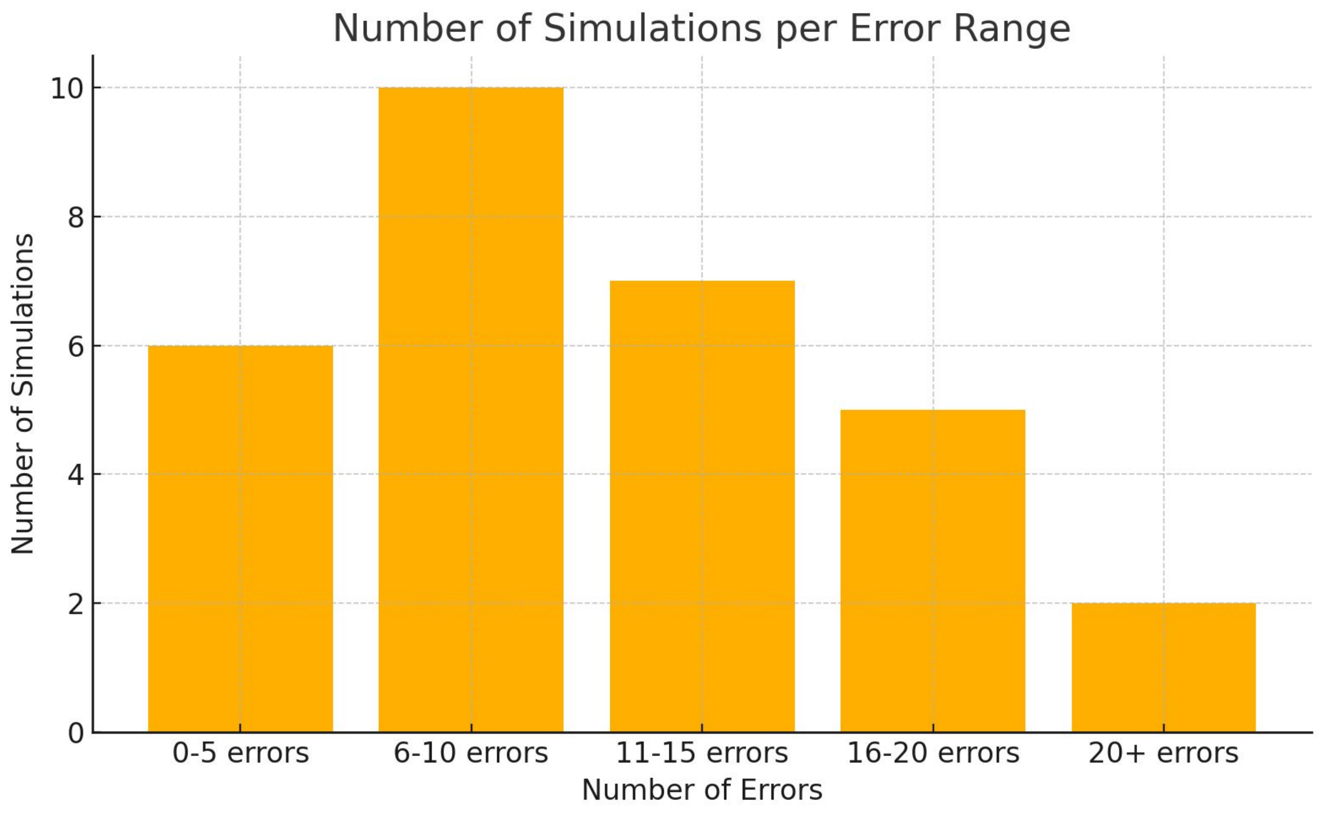
Successes

- All subsystems were powered successfully
- Communication among subsystems was completely functional
- Touchscreen and card detection features were functional and accurate
- Final result of the sorted and unsorted piles was correct for each game option

Challenges

- Trouble moving exactly one card at once
- Multiple jams occurred for each sort
- Not all subsystems were on the PCB, leading to a bulky design

Data - 30 Simulations



Conclusions

Conclusions

- Learned how to operate various tools and technologies (RasPi, Motor Drivers, Touchscreen, ESP32)
- Learned more about communication protocols such as UART and SPI
- Learned about PCB design and soldering
- No ethical concerns with our project

Possible Changes

- We would rethink our sorting mechanics if we redid our project
- Make motor/sorting settings adjustable
- Use USB connection for programming ESP32S3

Acknowledgements

A special thank you to our TA Sanjana Pingali, Skee Aldrich and Gregg Bennett from the Machine Shop, our mentor Jack Blevins, and all ECE 445 staff that have reviewed our project.

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Thank You!

Questions?