

AUTOMATIC MAIL SORTER

By

Angelo Santos (angelos4)

Lisa Pachikara (lisamp2)

Sahas Munamala (sahasrm2)

Final Report for ECE 445, Senior Design, Spring 2023

TA: Yixuan Wang

May 2023

Project No. 64

Abstract

In this project, we developed an automatic mail sorter capable of reading printed characters on envelopes and sorting them into designated mailboxes based on the recipient. The system utilizes advanced character recognition technology to accurately identify and process addresses. Additionally, mailbox owners can customize their preferences through a web server, enabling them to filter out specific senders and control the types of mail they receive. Our main findings indicate that the sorter effectively streamlines the mail sorting process and enhances user control, offering a practical and efficient solution for managing mail delivery.

Contents

| | |
|---|-----------|
| Contents..... | 2 |
| 1. Introduction..... | 2 |
| 2. Design..... | 3 |
| 2.1 Introduction..... | 3 |
| 2.2 Subsystem Design..... | 6 |
| 2.2.1 Control Subsystem..... | 6 |
| 2.2.2 Mail Recognition Subsystem..... | 6 |
| 2.2.3 Mail Sorting Subsystem..... | 7 |
| 2.2.4 Power Subsystem..... | 7 |
| 2.3 Design Verification..... | 7 |
| 2.3.1 Control Subsystem Verification..... | 7 |
| 2.3.2 Mail Recognition Subsystem Verification..... | 8 |
| 2.3.3 Mail Sorting Subsystem Verification..... | 8 |
| 2.3.4 Power Subsystem Verification..... | 8 |
| 2.4. Costs..... | 9 |
| 2.4.1 Parts..... | 9 |
| 2.4.2 Labor..... | 9 |
| 2.5. Conclusion..... | 10 |
| 2.6 References..... | 11 |
| Appendix A Requirement and Verification Table..... | 12 |
| Appendix B Mail Sensing Results..... | 15 |

1. Introduction

The automatic mail sorter addresses the common issue of misdelivered mail, including mail for prior tenants and unwanted mail from blacklisted senders or advertisers. These mail delivery errors can lead to security threats and legal implications. Additionally, mail delivery workers face the tedious task of managing multiple mailboxes in apartment complexes. To solve these problems, we developed an automatic mail sorter that organizes mail based on recipient names and sender restrictions, improving both security and efficiency.

This project comprises four subsystems: Control, Mail Sensing, Mail Sorting, and Power. The Control subsystem manages communication between the microcontroller and Raspberry Pi, which handles image analysis and web server hosting. The Mail Sensing subsystem employs cameras and OCR technology to read and process mail information. Finally, the Mail Sorting subsystem utilizes motors and sensors to guide mail into designated bins. The Power subsystem supplies the necessary voltage for the sorter.

The report is organized into chapters detailing each subsystem, along with the design, implementation, and testing of the overall system. The concluding chapter presents our main achievements: a 90% character accuracy in extracting names, a user interface for customizable mail filtering, and an 80% success rate in physically guiding mail. Our automatic mail sorter effectively addresses the issues of mail mismanagement, enhancing both security and efficiency in mail delivery.

2. Design

2.1 Introduction

The technical design of our automatic mail sorter project encompasses four subsystems: Control, Mail Sensing, Mail Sorting, and Power, as shown in Figure 1. The Control subsystem relies on a Raspberry Pi to host the webserver and manage communication with the OCR system. The ATmega328p microcontroller is responsible for motor control, sensor logic, and communication with the Raspberry Pi, ensuring seamless operation throughout the process.

A fully integrated PCB houses the microcontroller, motor pins, and laser pins, allowing for efficient coordination between the subsystems. This design enables the automatic mail sorter to accurately read and process mail information, as well as physically guide the mail into designated bins based on recipient names and sender restrictions. Figure 2 shows the physical design of the mail sorter.

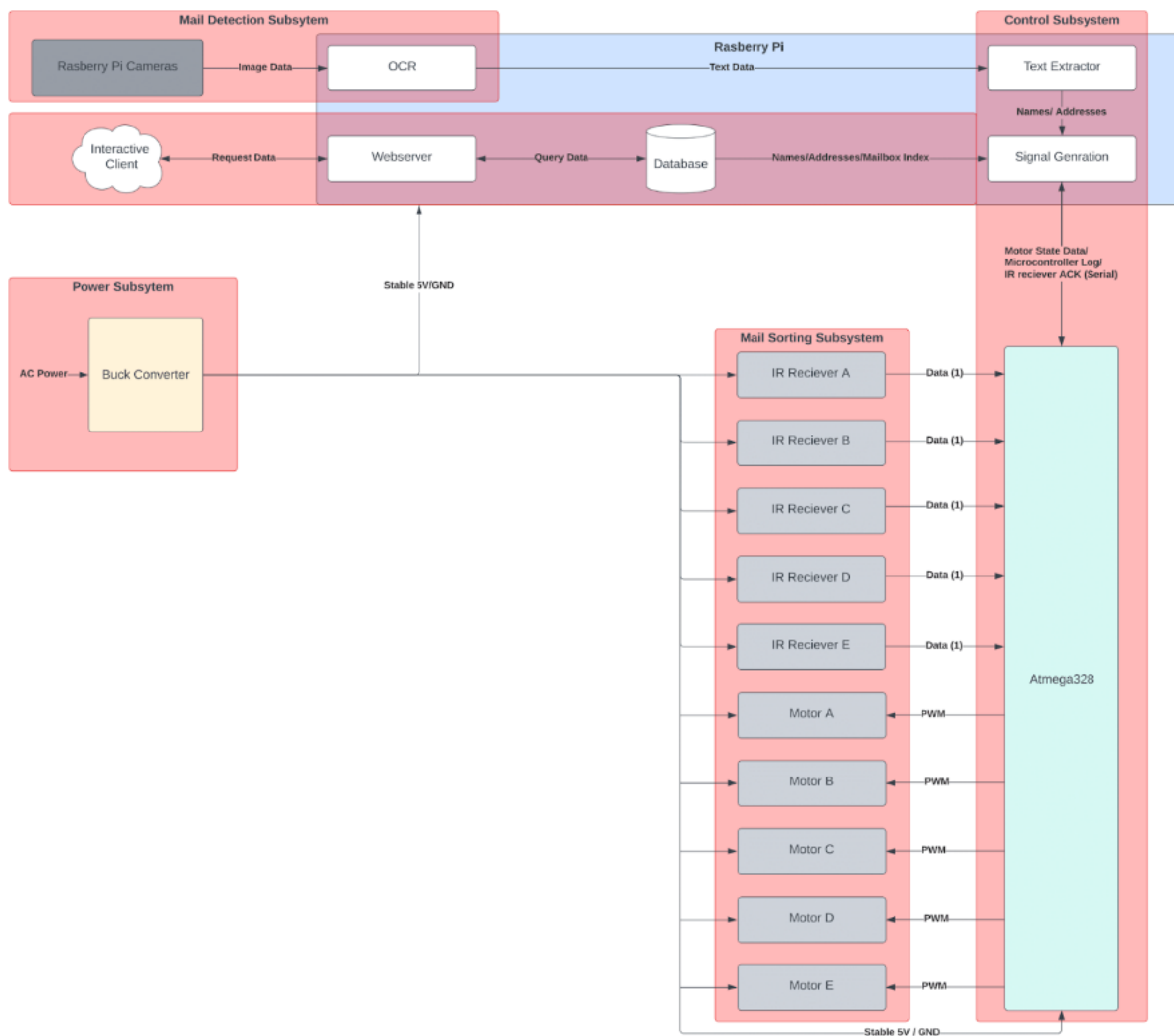


Figure 1 Block Diagram showcases the automatic mail sorter's subsystems and their interconnectivity, illustrating the data flow between components for efficient operation.

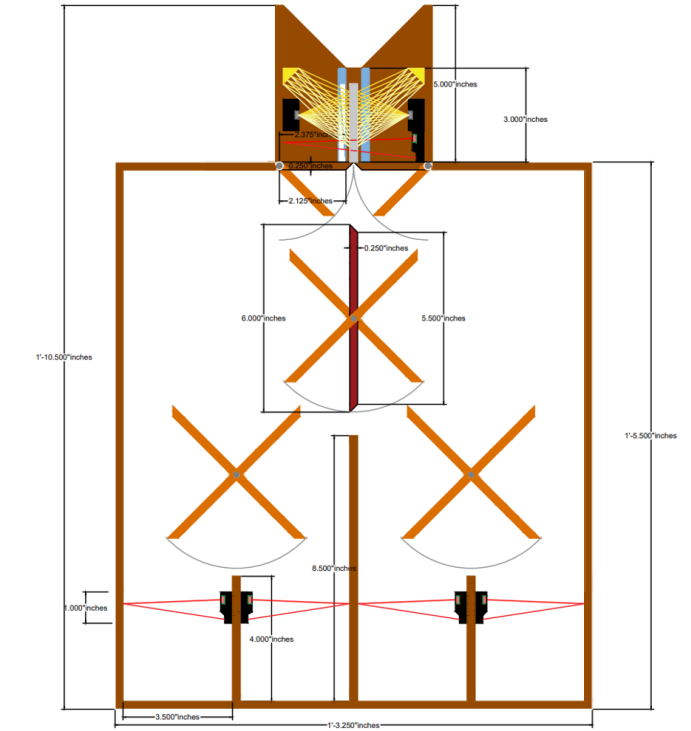


Figure 2 Physical Design of the automatic mail sorter features a wooden base measuring 22.5" x 15.25" and a transparent acrylic front panel for easy observation and debugging. A 0.5" entryway accommodates mail, ensuring proper alignment for image capture, while 1" gaps at the base of each mailbox house sensors. The design includes ample space for the power source input, Raspberry Pi, and wiring between components.



Figure 3 Final Device

2.2 Subsystem Design

2.2.1 Control Subsystem

Controls the image capturing of the camera based on the optical switch, and runs an OCR to determine the sender and receiver from the printed or handwritten text. It will then compare the data to names/aliases within a local database to determine the destination of the mail being processed. It will also host the web server that can allow the host to append or change the database externally. Further, it will send control signals to different electric motors in the organization system.

The control subsystem is made up of a microcontroller and a raspberry pi working together. The microcontroller is responsible for communicating with the hardware, moving servos, and reading light sensor values. The raspberry pi is responsible for making sorting decisions, storing the known sorting preferences in a database, and running the image processing pipeline.

2.2.2 Mail Recognition Subsystem

This component will consist of an optical switch connected to the main control unit that will determine if mail is placed properly in the scanner. It will also contain 2 cameras and light sources to capture images of both sides of the mail. On a signal from the control subsystem, the raspberry pi will take images with both cameras and feed them through an image processing pipeline. The processed images will be run through pytesseract OCR to extract the text for sorting.

The OCR pipeline consists of four steps. First is to grayscale the images, second is to crop the incoming images to a smaller size so pytesseract OCR can determine the bounds of the text easier. Third is to threshold the image so it only consists of entirely white or black pixels. And lastly, we deskew the images so that pytesseract OCR can easily extract and tag lines of text. Refer to Table [x] in the appendix to see before and after images of our image processing pipeline.

```
# grayscale
img_proc = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
# crop
img_proc = img_proc[150:350,150:450]
# threshold
ret, img_proc = cv2.threshold(img_proc,100,255,cv2.THRESH_BINARY)
# deskew
angle = deskew.determine_skew(img_proc)
print(angle)
if(angle and abs(angle) > 1 and abs(angle) < 15):
    img_proc = rotate_image(img_proc, angle)
```

Figure 4. Image processing Pipeline

2.2.3 Mail Sorting Subsystem

The Mail Sorting Subsystem utilizes the Arduino IDE to program motor controls and sensor logic. Upon inserting mail into the entry slot, the top laser receiver is activated, signaling the Raspberry Pi that it is "Ready", through Serial Bus communication. The Pi then communicates the mailbox number to the Atmega, which uses pre-assigned servo motor angles to direct the mail to the intended recipient. Once the mail reaches the mailbox, the laser receiver in the mailbox is activated, resetting the motors and preparing the system for the next piece of mail.

```
4  #define LL_CLOSED 100
5  #define LL_OPEN 160
6  #define RL_CLOSED 90
7  #define RL_OPEN 20
8
9  #define L_45 55
10 #define L_135 110
11 #define R_45 55
12 #define R_135 100
13 #define M_45 55
14 #define M_135 135
15 #define NEUTRAL 90
```

Figure 5. Angle Settings for Servos

2.2.4 Power Subsystem

This is the subsystem that will provide the power for the entire project. Originally, there were plans to add buck converters and voltage regulators on top of a power supply to obtain the proper 5 Volt and 8 Amp maximum requirements of the system. However, we found a power supply that directly met these requirements.

2.3 Design Verification

2.3.1 Control Subsystem Verification

The main requirements of this subsystem were to execute the main control flow of the program from obtaining an image of the mail, to changing the orientation of the motors for the mail to fall. The main things we had to account for were the proper orientation of the motor angles, the halting of the main flow when mail has not yet entered the chute, and the sensing of mail in the scanning area. This was done through the analog reading of many laser sensors on the system and the communication between both the Arduino and the Raspberry Pi through the serial data channels. The solution for this communication handshake was inspired by modern TCP connection protocols. For the webserver implementation, we verified the functionality by making changes to the database for new tenants and blacklists, and immediately placing mail into the chutes where the changes should be reflected.

2.3.2 Mail Recognition Subsystem Verification

The main requirements of the mail sensing subsystem was accuracy. In order to test the system, we took 32 images out of our camera setup for processing. Out of these, 8 of them had text appear in the image. To make sure the system works, we ran all the images through the pipeline and tagged them as empty or with text, and tried to extract the text. Appendix B shows our results. We then manually counted the correct and incorrect characters to determine an estimate for the letter accuracy of our system.

2.3.3 Mail Sorting Subsystem Verification

To ensure mail release only occurs after a successful scan, we tested the system by processing a piece of mail and checking the Raspberry Pi's system log for image analysis and mailbox determination before motor activation. To confirm correct paddle positioning, we visually inspected the paddles' alignment with the designated mailbox as the mail passed through the system (Figure 7). This came out to be 100% accurate when given the box number from the Raspberry Pi. Lastly, to verify the system resets only after mail delivery, we introduced another piece of mail while the previous mail was still in transit, observing if the system waited until the mail reached the mailbox before resetting.

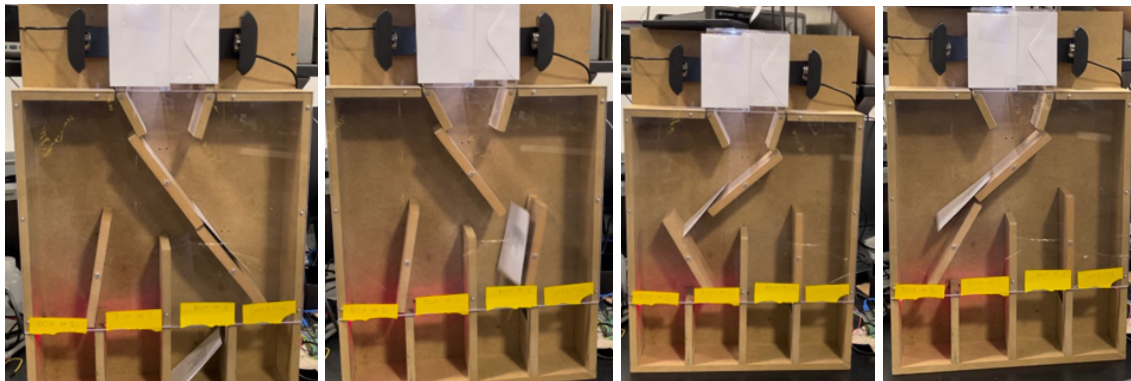


Figure 6 Visual Verification that the paddles placed the mail in the correct mailbox given the Box number from the Raspberry Pi. The orientations are shown from box 4 to box 1 when seen from left to right.

2.3.4 Power Subsystem Verification

We recognized through studying the various datasheets of the components that we were using, that we would have a maximum power draw from the system when everything draws power simultaneously. We saw through the datasheet for the HS-311 servo motor that the maximum current draw for each motor at a stall is 800mA [4]. This was an unlikely current draw given that the weight and torque of the motors should never reach this power consumption rate given that the motors don't attempt to push against the main body. We ran tests on a sample set of motors while reading the amperage value on the ammeter on the lab to find that a power draw of less than a quarter of an Amp was used by 3 motors with no weights attached. The raspberry pi used was a model 3 B+. So we would expect a current draw of about 1 Amp [9] as a worst case scenario benchmark. The current requirements for the ATmega328P were about 40mA

according to the datasheet [10]. The other components had a negligible power requirement and were not considered. In total, we found that we would need about 6 Amps at a worst case scenario for the components. We were considering using a buck converter and a voltage regulator to get these requirements from a standard voltage source. We then realized that this system becomes redundant if we purchase a power supply rated at 5V 8Amps. In lieu of time and PCB order delays, we decided to just use this power supply for the project. We also programmed the motors to work in sequence which minimized the maximum power requirements as well.

2.4. Costs

2.4.1 Parts

Table X Parts Costs

| Part | Manufacturer | Retail Cost (\$) | Bulk Purchase Cost (\$) | Actual Cost (\$) |
|--------------------|---------------------|-------------------------|--------------------------------|-------------------------|
| Camera | Angetube 1080p | 39.85 | 32.00 | 34.00 |
| ATMega328p | Atmel | 3.00 | 2.02 | 3.00 |
| Servos | HiTec | 15.00 | 15.00 | 75.00 |
| Crimping Tool | Qibaok | 40.00 | 40.00 | 40.00 |
| Lasers | HiLetgo | 7.00 | 7.00 | 14.00 |
| Laser Receiver | GeekStory | 11.00 | 11.00 | 11.00 |
| Crystal Oscillator | uxcell | 7.18 | 7.18 | 7.18 |
| Raspberry Pi | RaspberryPi | 62.00 | 62.00 | 62.00 |
| | | | | |
| Total | | | | 246.18 |

2.4.2 Labor

Hourly Rate: \$50/hr

Estimated Hours to complete: 100 hrs

Labor Cost: $\$50 \times 2.5 \times 100 = \$12,500$

2.5. Conclusion

In conclusion, our automatic mail sorter project achieved its objectives and can be considered a success. With OCR character recognition surpassing 90% accuracy and motor controls guiding mail to the correct mailbox with over 80% accuracy, the system effectively addresses mail mismanagement issues. Additionally, the web server's customizable interface allows users to tailor their mailbox settings, further enhancing the mail sorting experience. This project demonstrates the potential of technology to streamline and improve the mail delivery process for both recipients and mail delivery workers.

We were able to successfully complete our requirements as spelled out in the document. We were able to integrate all the individual subsystems into a single working product. We were very uncertain about the Optical Character Recognition system and how effective it would be in our setup. Luckily, we were able to fine tune the pipeline so we can get about 95% character accuracy. Another uncertainty in our project was the PCB. We were making changes to the electronics set up constantly throughout the project, and we got our final PCB just days before the demo.

We believe that this project is fairly ethical, safe to the public and is beneficial for everyone involved. Referencing the IEEE Code of Ethics, our project would comply with all of the requirements, however, we can see some possible violations if this project is carried out as intended. The first foreseeable issue comes into play when unauthorized users try to access the residents' private data from our database such as name, address, and the photos of the mail received. Another similar issue can arise as an unauthorized user could change the dataset for residents in terms of what mail they block and receive. This could be a possible violation of the IEEE Code of Ethics Section I-1 [4]. We strive for our project to be non-discriminatory, lawful, and well-cited. Overall, we seek to create a product that can be efficient and helpful for mail industry workers.

Moving forward, we aim to enhance the OCR capabilities to accurately recognize handwritten text, further improving the system's effectiveness in mail sorting. This would require a much more robust image processing pipeline, and a much more reliable camera setup. The current setup has too much variability in lighting, so it is hard to find thresholding values that work for all images. Another modification would be to extend the design to accommodate 16 boxes. This would require one more level of paddles, and would make the design taller. The most immediate change would be to clean up the wiring of our current setup. This would require a new PCB with adjusted pin outs.

2.6 References

1. "7 Steps of Image Pre-Processing to Improve OCR Using Python." *NextGen Invent Corporation*, <https://www.nextgeninvent.com/blogs/7-steps-of-image-pre-processing-to-improve-ocr-using-python-2/#:~:txt=Seven%20steps%20to%20perform%20image%20pre-processing%20for%20OCR> .
2. "GitHub - letter_shredder_pi." Angelo Santos, Lisa Pachikara, Sahas Munamala, https://www.github.com/angelos4/letter_shredder_pi
3. "GitHub - letter_shredder_controller." Angelo Santos, Lisa Pachikara, Sahas Munamala, https://www.github.com/angelos4/letter_shredder_controller
4. "Hitec RCD USA - HS-311 Servo." Hitec RCD USA, 4 May 2023, <https://hitecrcd.com/products/servos/analog/sport-2/hs-311/product>.
5. IEEE. "IEEE Code of Ethics." IEEE, Institute of Electronics and Electrical Engineers, <https://www.ieee.org/about/corporate/governance/p7-8.html>.
6. "Python Web Server with Flask." Raspberry Pi Foundation, 2 May 2021, <https://projects.raspberrypi.org/en/projects/python-web-server-with-flask>.
7. "Raspberry Pi and Arduino Serial Communication." Robotics Back-End, <https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/>.
8. "SQL Statements & Structure." MariaDB Knowledge Base, <https://mariadb.com/kb/en/sql-statements-structure/>.
9. "Power Consumption Benchmarks." Pi Dramble, <https://www.pidramble.com/wiki/benchmarks/power-consumption>.
10. "Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf." Microchip Technology Inc. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

Appendix A Requirement and Verification Table

| Requirements: | Verification: | Requirement Fulfilled: |
|--|--|------------------------|
| The cameras should be triggered to capture data into the raspberry pi when mail has entered the slot. | With only the image capturing unit active along with the laser detection sensors, we will test how often an image is captured when mail is placed in. | Y |
| The mail should remain in the slot and the camera should not capture an image if the prior mail has not yet reached it's destination. | This requirement can be tested by modifying the motors in the sorting subsystem. If we allow one of the motors to block the mail from entering the box and place another mail in the divot, we will then be able to check if the mail has left recognition module. | Y |
| Text recognition from the OCR module should produce the correct text from the mail 85 +/- 5% of the time. | We will create a sample set of 20 different mail printouts. We will then compare the output from our OCR tests to the data on the printouts to evaluate the failure rate of this module. | Y |
| The Raspberry Pi should host a webserver that will interact with this database to update the names of the users that are meant to receive the mail along with the destinations in terms of the mailbox number. | We will connect to the web server from a website, change the name of the recipient in mailbox #1. Then, drop a piece of mail with that new name, and observe that the mail enters that mailbox#1. We will also place mail for the recipient name that was changed and observe that the mail enters the trash slot. | Y |
| The webserver should also be able to update and modify the blacklisted senders for each of the users dynamically to allow user control for mail. | We will add/modify the blacklisted list and place mail from a sender in that blacklist into the chute. We should see that the mail enters the trash slot. | Y |

| | | |
|--|--|---|
| Raspberry Pi should be able to capture images and store locally on a signal from the microcontroller to begin visual processing. | Insert a piece of mail into the mail sorter so that the microcontroller can trigger a picture to be captured. Inspect the image is saved properly via ssh. | Y |
| Raspberry Pi can send signals to the microcontroller for the motor controls based on data captured from the cameras. | Insert a piece of mail, and let the OCR software do its thing. Inspect through the system log that the microcontroller received the instructions to move the servos to guide the mail. | Y |
| Microcontroller can load signal from Raspberry Pi to control motors to move the paddles for the mail to move into the correct mailbox. | Insert a piece of mail with a specific name. Verify on the raspberry pi log that the correct name and mailbox were extracted and selected. Then Visually inspect that the paddles are moved to guide the mail. | Y |
| The motor dropoff only releases the mail once it has been successfully scanned. | Run a piece of mail through the system, and verify using the system log stored on the raspberry pi that the image was processed and the mail slot determined before the motors moved. | Y |
| The paddles are in the precise orientation that allows the mail to enter its respective box. | Run a piece of given mail through the system and visually check that the paddles get set to the correct position. | Y |
| The mail should remain in the slot if the prior mail has not yet reached it's destination. | This requirement can be tested by modifying the motors in the sorting subsystem. If we allow one of the motors to block the mail from entering the box and place another mail in the divot, we will then be able to check if the mail has left recognition module. | Y |
| This subsystem must be able to take in power from an AC to DC converter connected to an outlet. | Run the motors on the same powerline and check that the voltage remains at 5V using a voltmeter. | Y |

| | | |
|--|--|----------|
| <p>The voltage and energy provided from this converter must be enough to provide stable power to all the components, along with the appropriate voltages needed by any of the devices.</p> | <p>Test the amperage across the different motors using an ammeter and see that each one has at least 1 amp when all motors are being run simultaneously.</p> | <p>Y</p> |
|--|--|----------|

Appendix B Mail Sensing Results

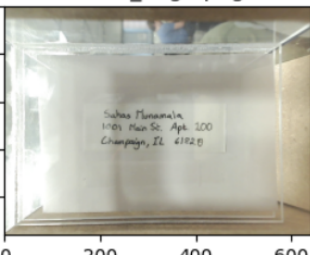
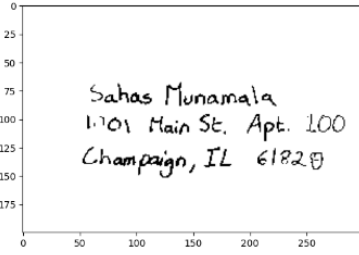

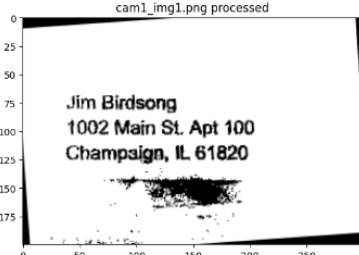
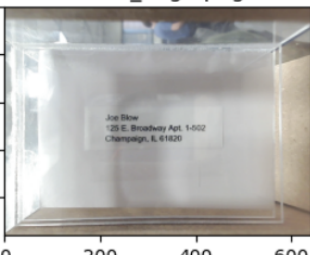
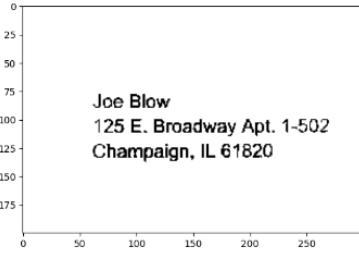
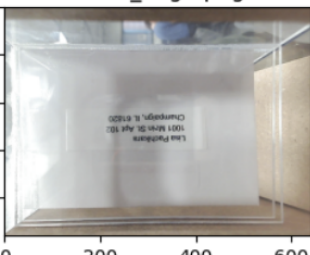
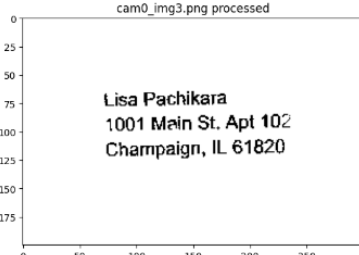
| | | |
|--|---|---|
|  <p>cam0_img8.png</p> |  | <p>Sahas Munamala OV Hain St. Apt. LOO Champaign, IL 61820</p> |
|  <p>cam1_img1.png</p> |  | <p>Jim Birdsong 1002 Main St. Apt 100 Champaign, IL, 61820</p> |
|  <p>cam0_img2.png</p> |  | <p>Joe Blow 125 E. Broadway Apt. 1-502 Champaign, IL 61820</p> |
|  <p>cam0_img3.png</p> |  | <p>Lisa Pachikara 1001 Main St. Apt 102 Champaign, IL 61820</p> |

Table 2. Results for the tests of our Optical Character Recognition System. The first column is the raw images taken by the cameras, the second column is the output of our pipeline before sending the data to pytesseract for text extraction. And the last column is the results of OCR.

