Refreshable Text to Braille Display

Jacob Zhang, Dionysius Xaverio, Tanin Padungkirtsakul, Sarang Kashyap

4/30/2025

GTA: Manu Ramesh
Instructor: Ryan Beasley

Design Document 3

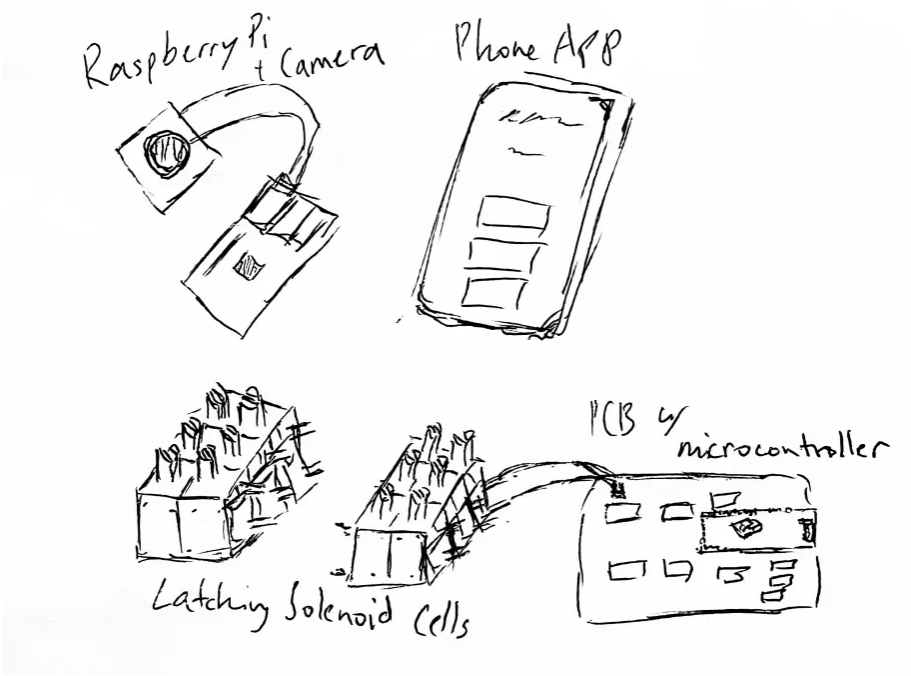


Figure 0.0.1: Sketch of Display

Contents

List of Figures

List of Tables

Revision Log

Date	Revision	Changes
1/29/2025	1	Design Document 1: • User Story • Requirements • Activity/Block Diagrams • Executive Description
3/16/2025	2	Design Document 2: • Updated Activity and Block Diagrams • Theory of Operation
4/20/2025	3	Design Document 3: • Specification Measurements
4/30/2025	4	Final Design Document: • Integration Approach • System Photographs • Final Status

Table 1: Revision Log

Glossary

- **Bluetooth** Short-range wireless technology standard that is used for exchanging data between fixed and mobile devices over short distances and building personal area networks
- **Bluetooth Low Energy (BLE)** A version of Bluetooth designed to have low energy consumption. The system remains in sleep mode until the data transfer is initiated.
- **Microcontroller** Small computer in a single integrated circuit
- **SPI (Serial Peripheral Interface)** Interface bus used for data transmitting between microcontroller and peripherals

1 Introduction

1.1 Executive Description

The objective of this project is to create a device that can output the braille translation from some input source of text. The goal is to be a tool to improve accessibility for visually impaired to be able to get the feeling of reading, rather than some form of audio feedback, or where audio feedback is not convenient. The source of the text will come from either manual input from an app, or from text detection with a camera.

1.2 User Story

Rob is running a conference and wants his good friend Cecil to come. Cecil is visually impaired, however, but Rob wants Cecil to have as good an experience as possible. There are multiple events occurring at the same time in different rooms. Thus, Rob wants to set up an easily changeable Braille signboard that can be set between events to help Cecil and other visually impaired people easily navigate the conference and find where the event they want to attend is located. To quickly set the Braille on the signboard in between events, Rob uses a convenient and intuitive app, allowing him to edit with just a phone, and even at distance. To further aid Cecil, Rob also wants to have a camera with the ability to read the many signs and posters placed around the conference walls.

2 Design Requirements

2.1 Requirements

1. The device must be able to display readable braille characters
2. The device must be able to output braille characters in accordance with text inputs from the user accurately
3. The output of the device must be able to take text input wirelessly using a smartphone.
4. The device must be able to process images containing text and translate it into braille

2.2 Factors influencing requirements

2.2.1 Public Health, Safety, and Welfare

1. User Safety: The device must be designed to prevent injury during use, particularly, since it will be physically interacted with by visually impaired individuals, all moving parts (e.g., actuators) must be safely enclosed, and materials should be smooth and free of sharp edges.
2. Electrical Safety: Components must comply with safety standards (e.g., low-voltage operation, overheating protection, short-circuit prevention) to avoid electric shock or burns.

2.2.2 Global Factors

1. Localization: The design should support the standard English alphabet as text input and the Unified English Braille standards for the output to be globally applicable.

2.2.3 Cultural Factors

1. User Interface Design: The user interface of the application should be designed to ensure it is user-friendly and appealing to a diverse range of users.
2. Acceptance of Assistive Technology: Some cultures may have stigma or pride associated with the use of assistive devices. The design should have a discreet and dignified appearance to promote acceptance.

2.2.4 Social Factors

1. Inclusion and Accessibility: The device could promote educational and social inclusion by allowing blind users to read printed or digital content independently.

2. Community Engagement: Design decisions should be informed by the visually impaired community to ensure that the product meets real-world needs and preferences.

2.2.5 Environmental Factors

1. Material Sustainability: Where possible, materials should be recyclable or biodegradable. The product should avoid hazardous materials like lead or PVC.

2. Energy Efficiency: The device should minimize power usage, especially in off-grid or low-resource environments. This could be achieved by using low-power microcontrollers and integrating sleep mode into the device

3. Longevity and Repairability: A longer device life cycle and the ability to perform basic maintenance or repairs (e.g., replace a solenoid or battery) reduce e-waste and improve sustainability.

2.2.6 Economical Factors

1. Affordability: The device must be cost-effective, especially for deployment in low-income or resource-constrained settings where accessibility tools are critically needed but often unavailable.

2. Manufacturing Costs: Choosing widely available, low-cost components and considering modular design can reduce production costs and simplify assembly.

3 System Overview

3.1 System Block Diagram

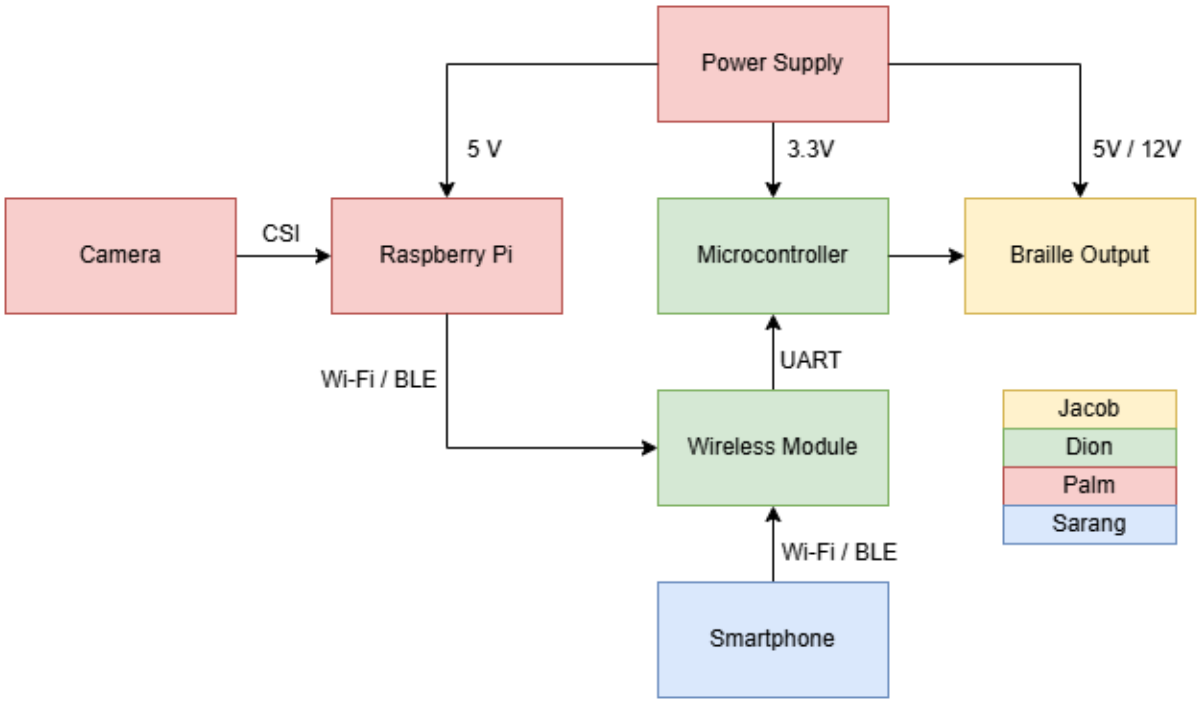


Figure 3.1.1: System Block Diagram

3.2 System Activity Diagram

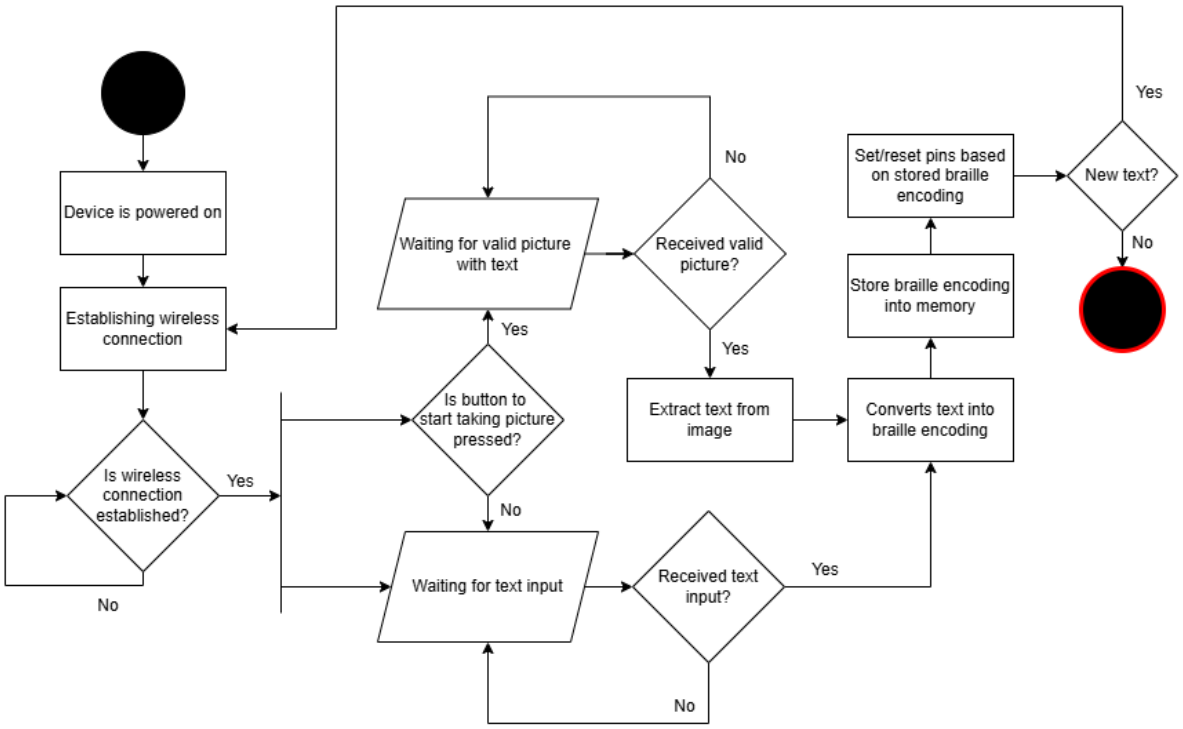


Figure 3.2.1: System Activity Diagram

3.3 Integration Approach

The system is designed to convert captured images or manually entered text into a tactile Braille display. A camera module extracts text from photos using OCR, while a mobile app provides another input option by sending text directly to the microcontroller. Once the text is received, the microcontroller parses it into Braille encoding and prepares the corresponding output.

To control the display, the microcontroller selects the appropriate Braille pin using GPIO signals, multiplexers, and a motor driver. Pulses are sent to latching solenoids, which raise or lower the pins based on the required Braille pattern. By using latching solenoids, the system maintains the display without needing continuous power.

3.4 System Photographs

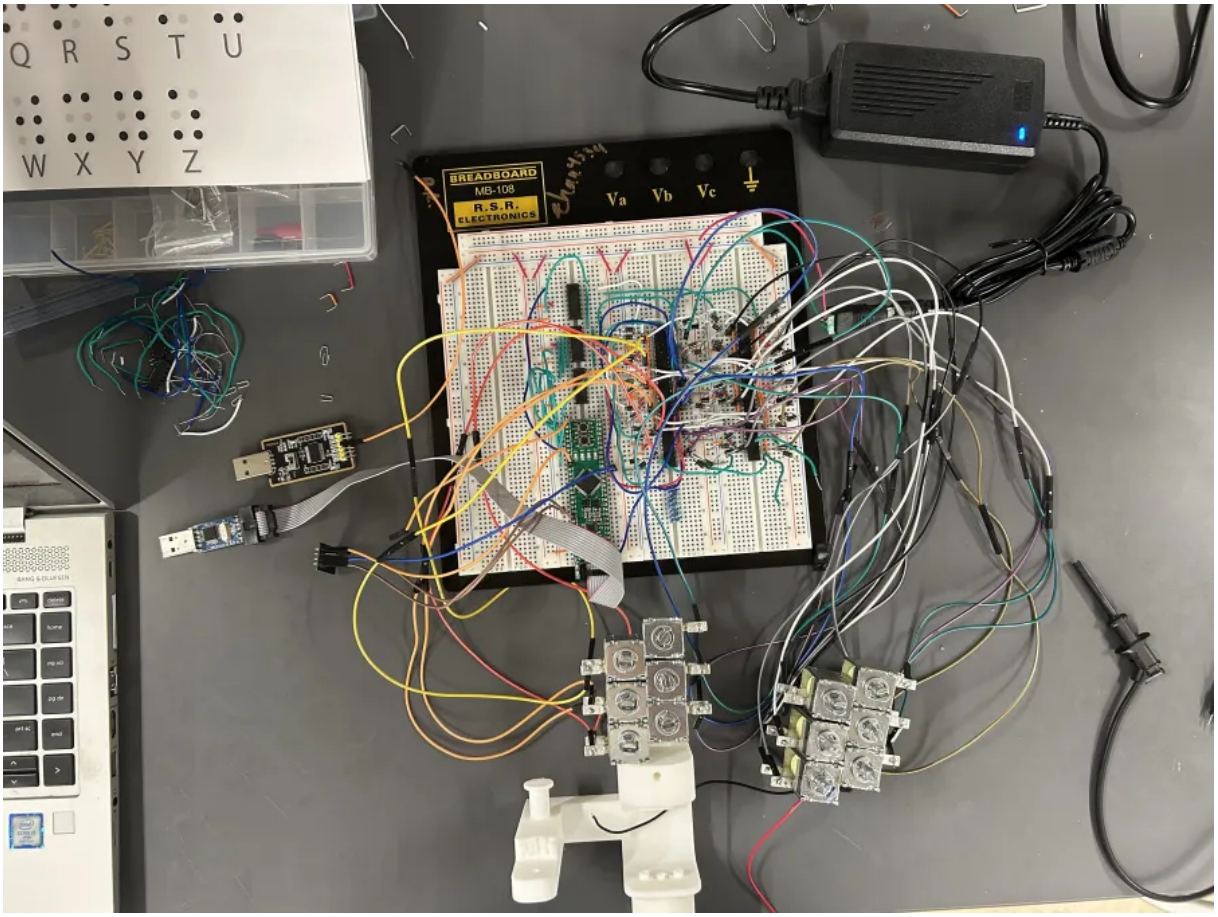


Figure 3.4.1: Photo including the Outputs, Circuit, Power, Microcontroller, WiFi, and Bluetooth ports

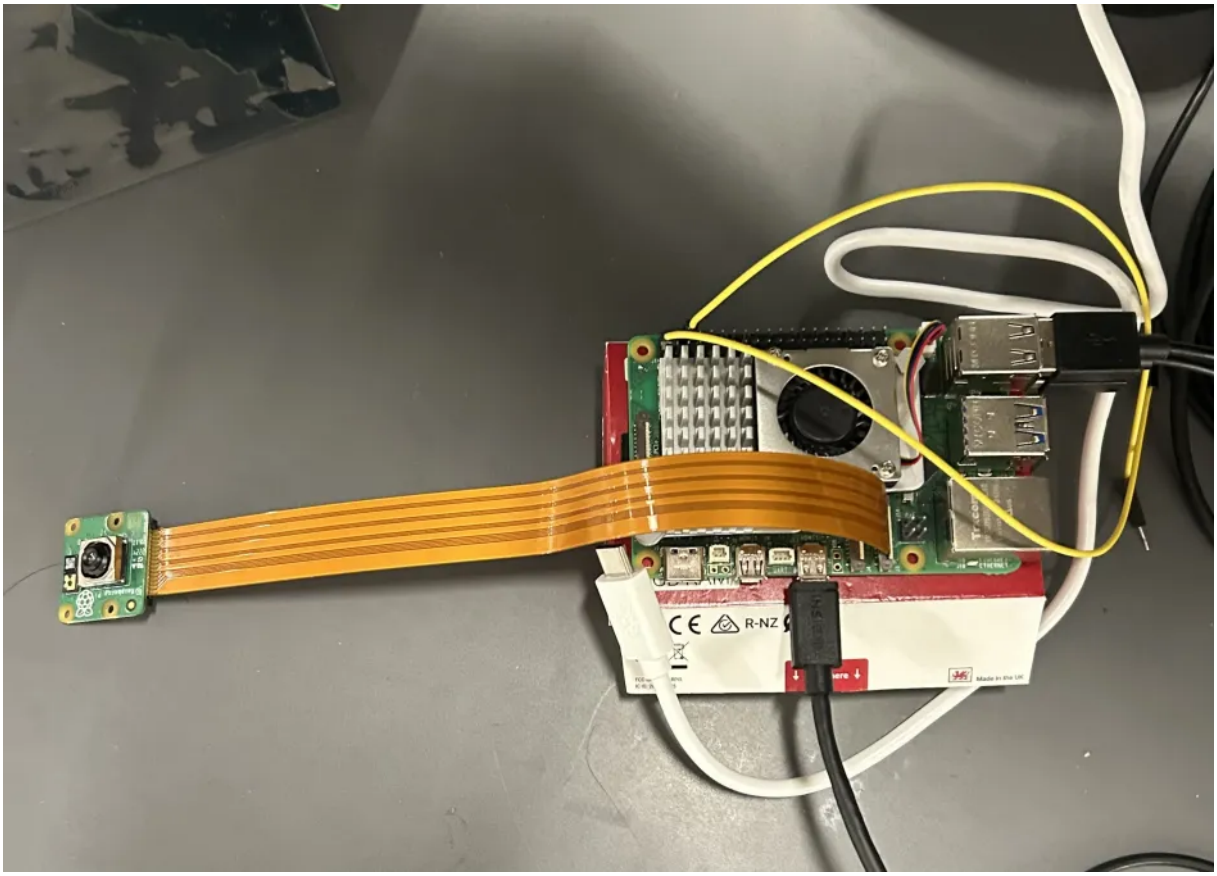


Figure 3.4.2: Photo of Camera and Raspberry Pi

4 Subsystems

4.1 Subsystem 1: Braille Output

Owner: Jacob Zhang

4.1.1 Subsystem Diagrams

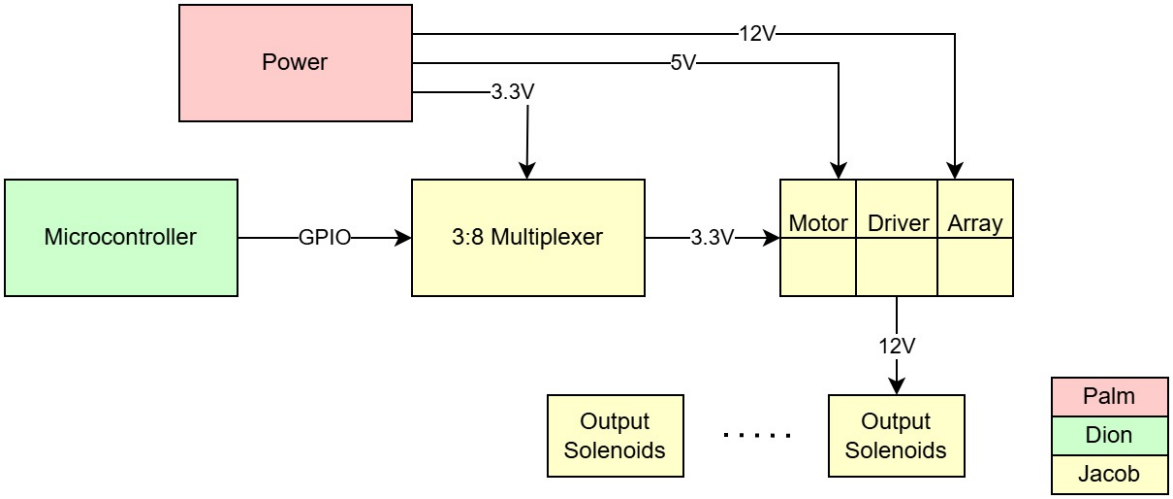


Figure 4.1.1: Subsystem Block Diagram

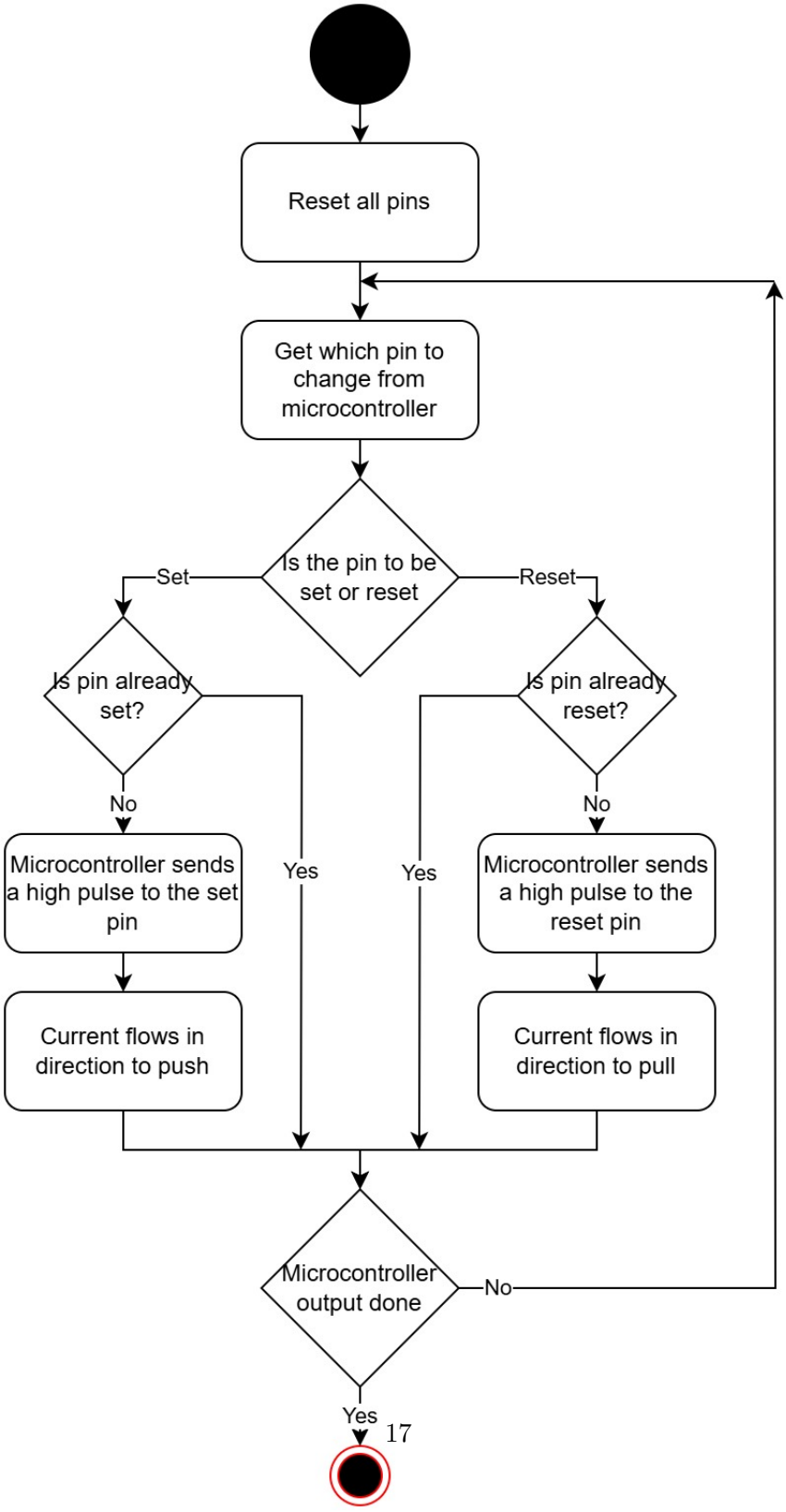


Figure 4.1.2: Subsystem Activity Diagram

4.1.2 Specifications

1. The current flowing while idle should be less than 50 mA.
   We want to limit the amount of power used while idle.

2. The current flowing through each solenoid should be less than 1.5A
   This specification is set in consideration of the specification of the motor drivers.

3. The voltage difference through the solenoid should be 10 to 14V when activating the solenoid.
   This specification is set since the solenoid is 12V, giving some leeway.

4. The voltage difference through the solenoid should be between -10 to -14V when resetting the solenoid.
   This specification is set since the solenoid is 12V, giving some leeway.

4.1.3 Subsystem Interactions

The subsystem will take pulses from the GPIO pins on the microcontroller in order to determine which pin is to be modified, and whether it should be set or reset.

The subsystem draws power from the power subsystem. The solenoid motors used require 12V, multiplexers require 3V, and motor drivers require 5V.

4.1.4 Core ECE Design Tasks

- **ECE 362**: Will help in designing and building the circuit.

4.1.5 Schematics

See Figure 4.1.3.

4.1.6 Parts

- 3:8 Mux - 74HC238
- Motor Driver - L293E
- Latching Solenoids - DSML-0630-12C
- Diodes - 1N4007

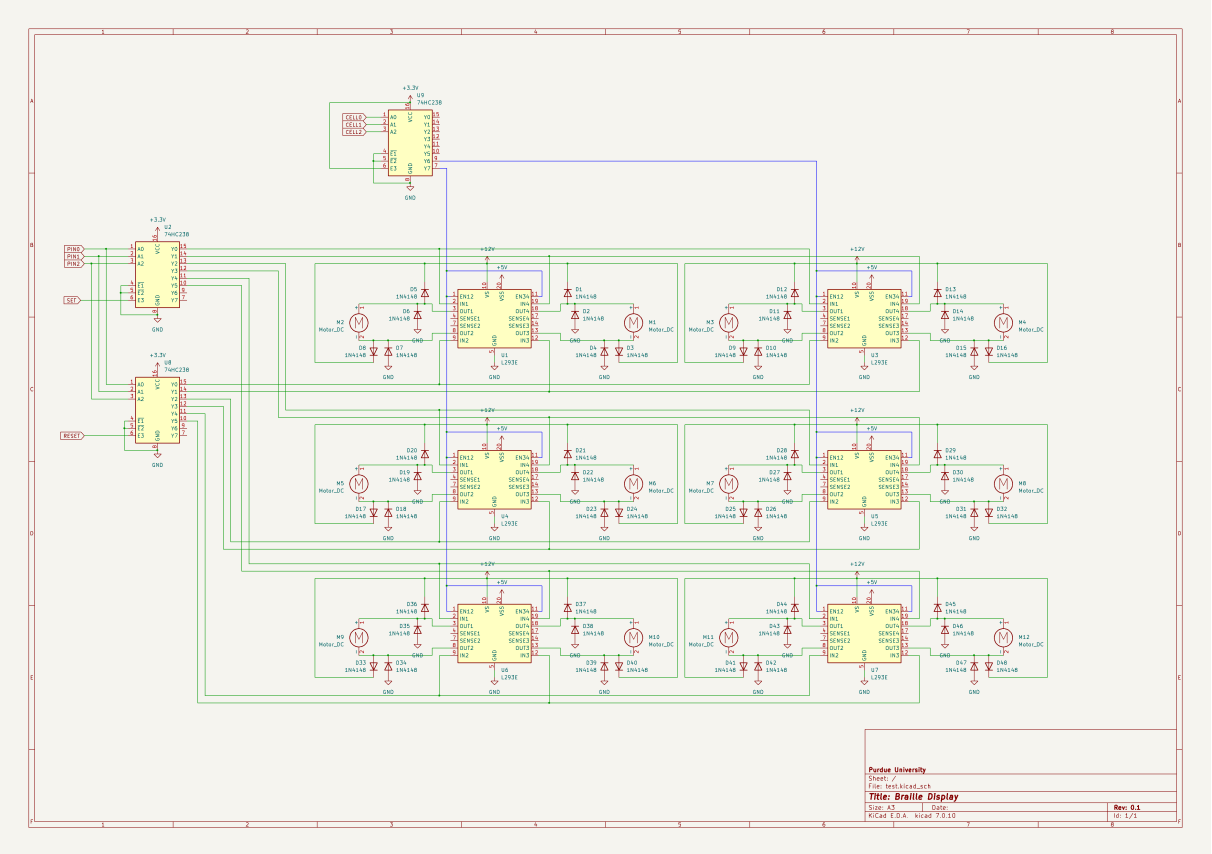


Figure 4.1.3: Circuit Schematic for Braille Output

4.1.7 Theory of Operation/Algorithm

The microcontroller will send out pulses using the GPIO pins. 6 GPIO pins are used for determining which cell and which Braille pin is being targeted. An additional two pins are used for set and reset, which feed in to the enable pins on the multiplexers used for choosing which Braille pin position. When either set or reset is pulsed high, the other will not be, so only one multiplexer will be enabled each pulse.

The output of the multiplexers is then fed into the input channels of the motor driver connected to the corresponding pin. Since only one multiplexer is enabled at a time, there will be a high in one input channel, and low in the other, with the direction determined by which is enabled. If the solenoid is currently not set, it will then be set, and if a pulse is sent with the opposite high and low configuration, the solenoid will then reset.

4.1.8 Specifications Measurement

Measurement setup and measurement value photos are provided below.

1. The current flowing while idle should be less than 50 mA.
   -The measured current while idle is .016 mA.

2. The current flowing through each solenoid should be less than 1.5A
   -The measured current when activated is 257 mA.

3. The voltage difference through the solenoid should be 10 to 14V when activating the solenoid.
   -The voltage difference is measured at +10.26 V.

4. The voltage difference through the solenoid should be between -10 to -14V when resetting the solenoid.
   -The voltage difference is measured at -10.02 V.

Photo proof is included below.

4.1.9 Standards

- **IEEE 120-1989**: Provides the method for measuring voltage, current, power, necessary for circuits.

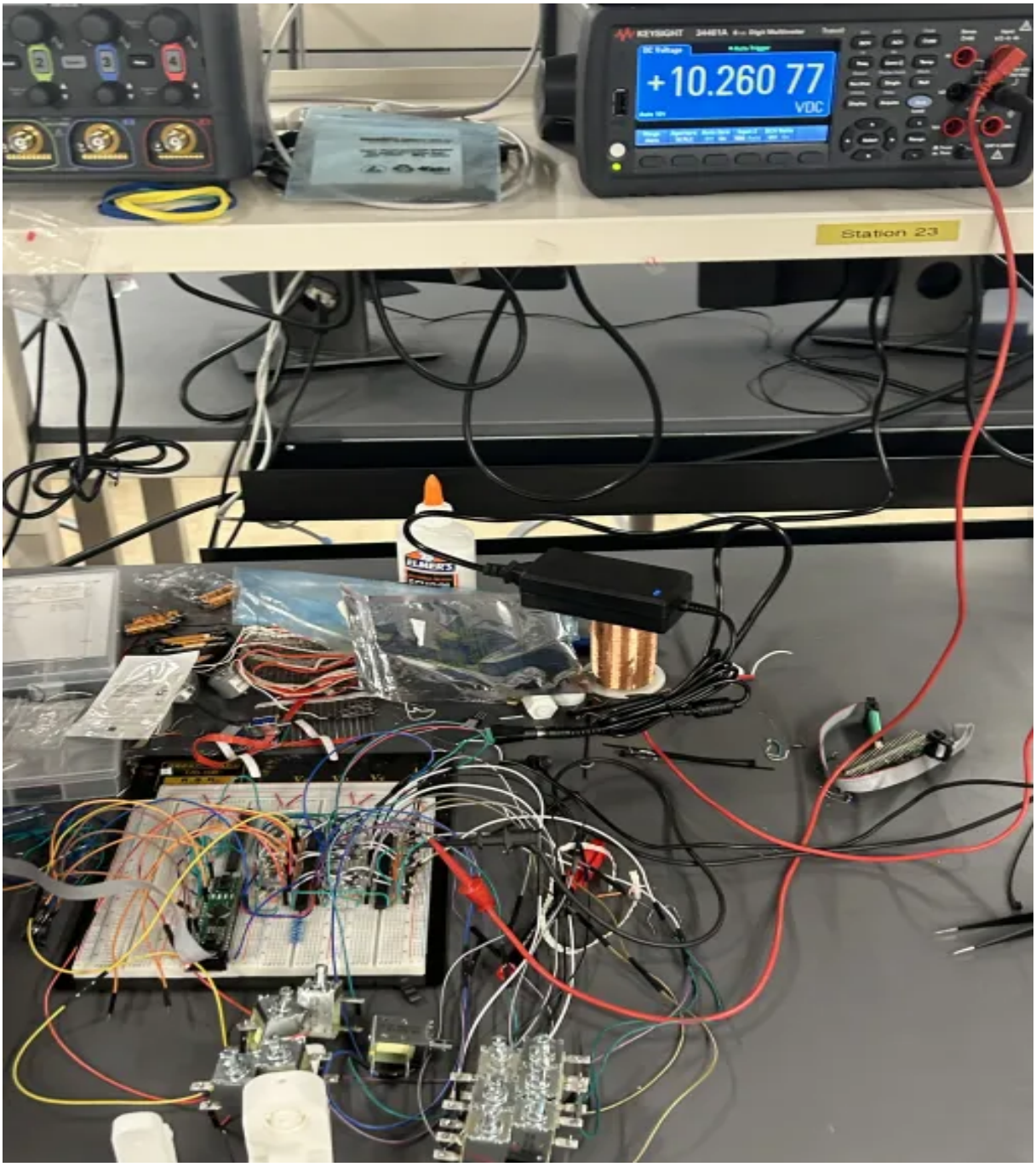


Figure 4.1.4: Setup for measurement with specification 3 showing

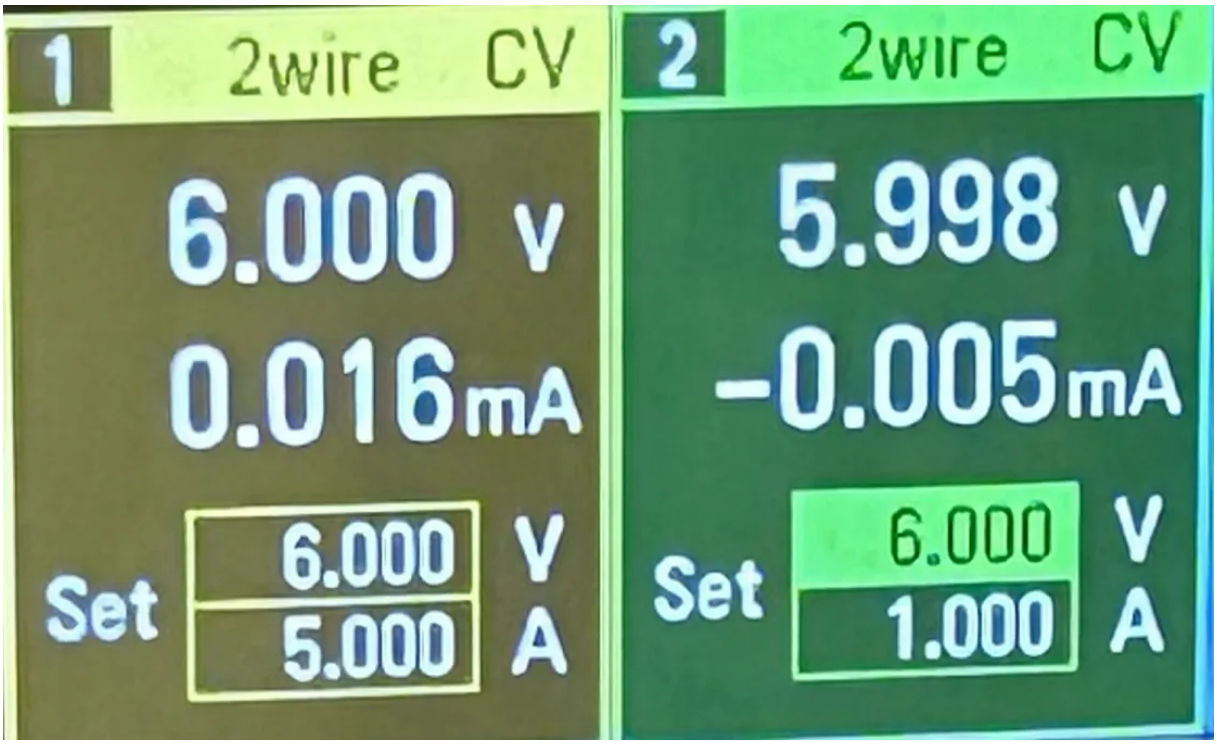


Figure 4.1.5: Measurement for specification 1

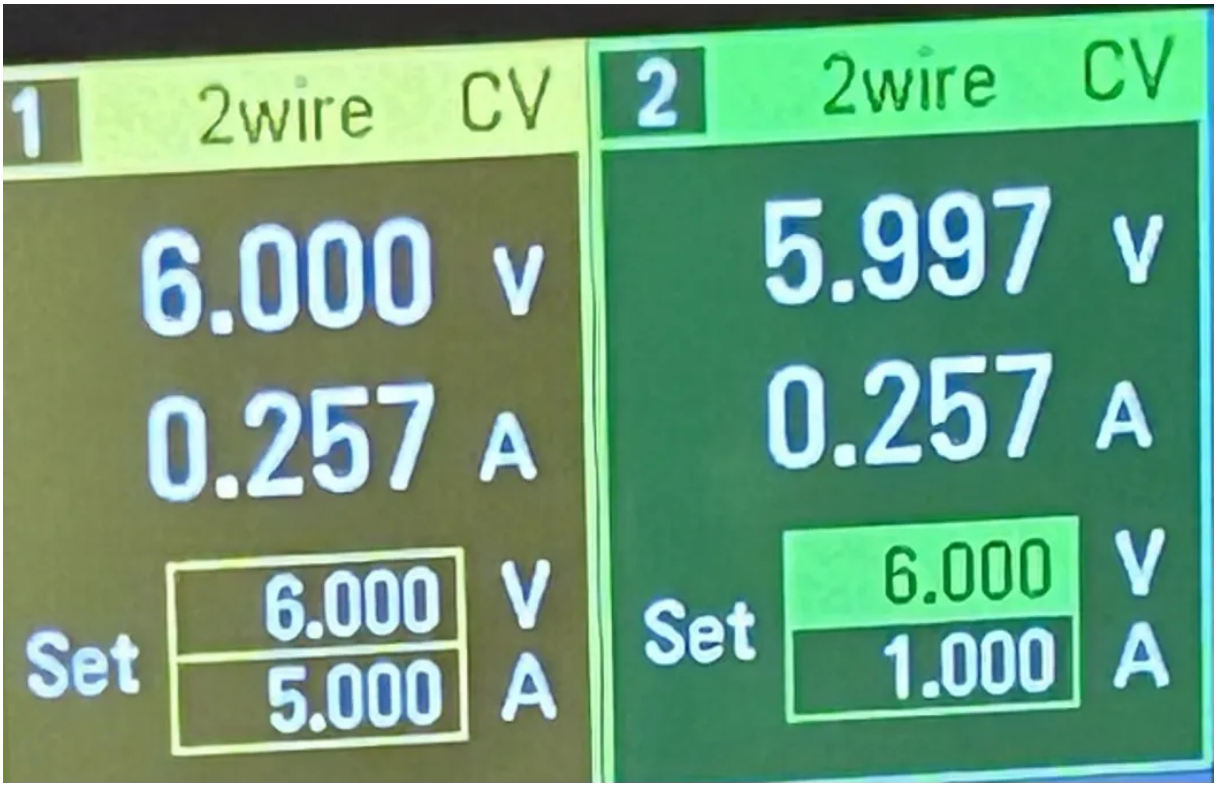


Figure 4.1.6: Measurement for specification 2

Figure 4.1.7: Measurement for specification 3

Figure 4.1.8: Measurement for specification 4

4.2 Subsystem 2: Power Supply and Computer Vision

Owner: Tanin Padungkirtsakul (Palm)

4.2.1 Subsystem Diagrams

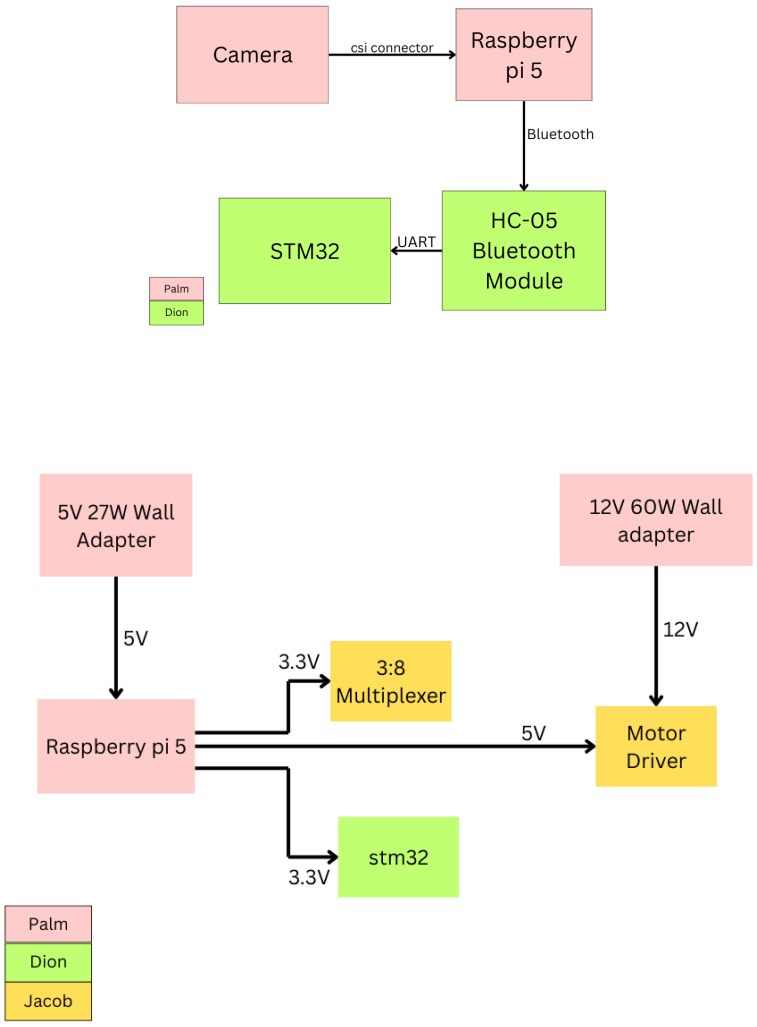


Figure 4.2.1: Subsystem Block Diagram

part	voltatge (V)	current (mA)	power (W)	number needed	subtotal power
stm32f091rct6	3.3	0.053	0.0001749	1	0.0001749
raspberry pi 5	5	800	4	1	4
l293e	12	6	0.072	6	0.432
74hc238	3.3	50	0.165	3	0.495
dsml0630 12c	12	400	4.8	12	57.6
hc05	3.3	30	0.099	1	0.099
esp01	3.3	80	0.264	1	0.264

power supply	total power
3.3 V	0.8581749
5 V	4
12 V	58.032

Figure 4.2.2: Subsystem Block Diagram

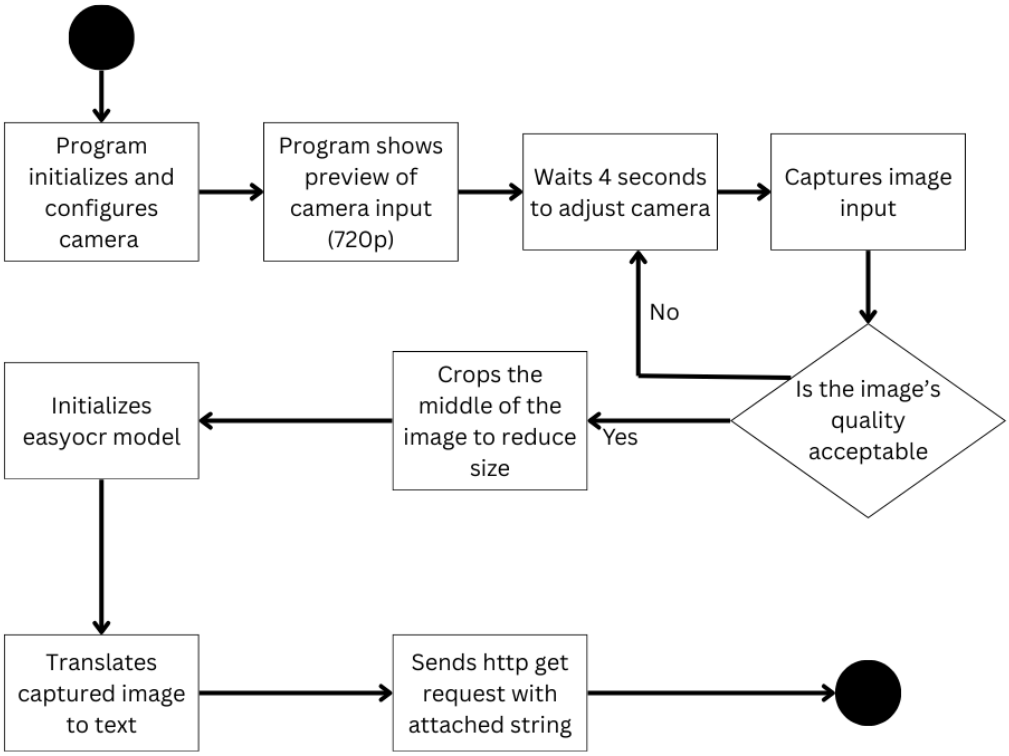


Figure 4.2.3: Activity Diagram

4.2.2 Specifications

1. Camera resolution must be 12 - 16 MP.

   A standard high quality camera, such as the one in a mobile phone, has around 12 MP. This will allow the captured image to be high quality, allowing for more accurate recognition while not being too expensive on the computation, and in a literal way.

2. Program should have character error rate between 2- 5 percent

   A widely accepted standard for the error rate for character recognition is below 5 percent. Keep in mind that this score is for a perfectly captured document, having sharp contrast, perfectly aligned, well lit, etc.

3. Software must take less than 10 seconds to translate

4. The Raspberry Pi must be able to supply 3.3V and 5V as required by the STM32, decoders, and the motor drivers.

5. The other power supply must be able to provide 12V and 60W as required by the motor drivers.

4.2.3 Subsystem Interactions

The computer vision program will send a message output to the translation subsystem. The power supply will power the microcontroller and the solenoid.

4.2.4 Core ECE Design Tasks

- **ECE 20875**: Python basics and convolutional neural network, which is used in the easyOCR library
- **ECE 362**: Familiarize with STM32, including basic operational condition, etc.

4.2.5 Parts

- Raspberry Pi 5
- Pi Camera Module 3
- Raspberry Pi Power Adapter 25W
- Raspberry Pi 5 Active Cooler
- AC/DC WALL MOUNT ADAPTER 12V
- Barrel jack adapter

4.2.6 Algorithm

There are many ways to approach Optical Character Recognition (OCR), many of which are open source and free to use. To save time and effort, we decided to use a python library called easyOCR, which has been proven to be very fast and accurate. The process begins with preprocessing, where the input image is converted to grayscale, normalized, and resized. For text detection, EasyOCR uses the CRAFT (Character Region Awareness for Text Detection) model, which identifies regions containing text and generates boxes around them. These boxes are refined using Non-Maximum Suppression (NMS) to filter out unnecessary detections. In text recognition, EasyOCR uses a CRNN (Convolutional Recurrent Neural Network), which combines CNNs for feature extraction, RNNs (BiLSTM) for sequence modeling, and CTC (Connectionist Temporal Classification) loss to convert predicted sequences into readable text.

4.2.7 Theory of Operation

The Raspberry Pi handles camera input for the subsystem. When an image is captured, it is first preprocessed—resized or enhanced as needed—then passed to EasyOCR, a deep learning-based OCR engine. EasyOCR first detects text regions using the CRAFT model, which identifies areas likely to contain characters. These regions are then passed to a CRNN (Convolutional Recurrent Neural Network), which reads the text using convolutional and recurrent layers, followed by a CTC decoder to produce the final string output.

Once the text is extracted, the Raspberry Pi sends it to the STM32 microcontroller using an API call over a serial communication interface. The STM32 receives the translated text and can then use it to trigger further processes, such as displaying the data or controlling other hardware components. This setup leverages the Raspberry Pi's strength in AI processing and the STM32's reliability in real-time embedded control.

4.2.8 Specifications Measurement

1. Camera resolution must be 12 - 16 MP.

   - The Pi Camera has 12 MP

2. Software must take less than 10 seconds to translate

   - EasyOCR model takes less than 10 seconds to generate text not counting time to load the model

3. The Power supply for raspberry pi must be able to supply 5V 25W as required by the raspberry Pi

   - The power supply has specifications 5V and 25.5 W which is more than required

4. The other power supply must be able to provide 12V and 60W as required by the motor drivers.

   The specification of the power supply meets the standard of 12V 60W.

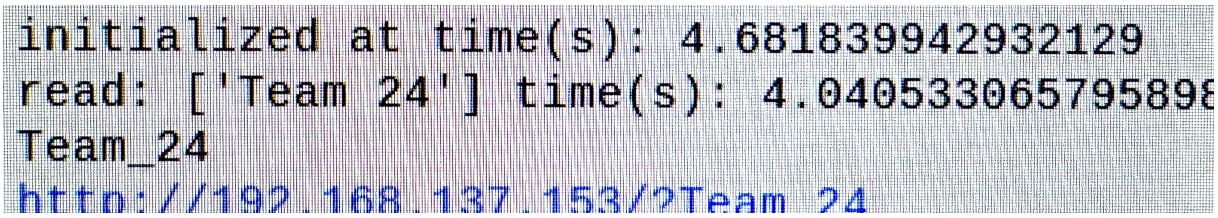


Figure 4.2.4: Measurement for specification 2

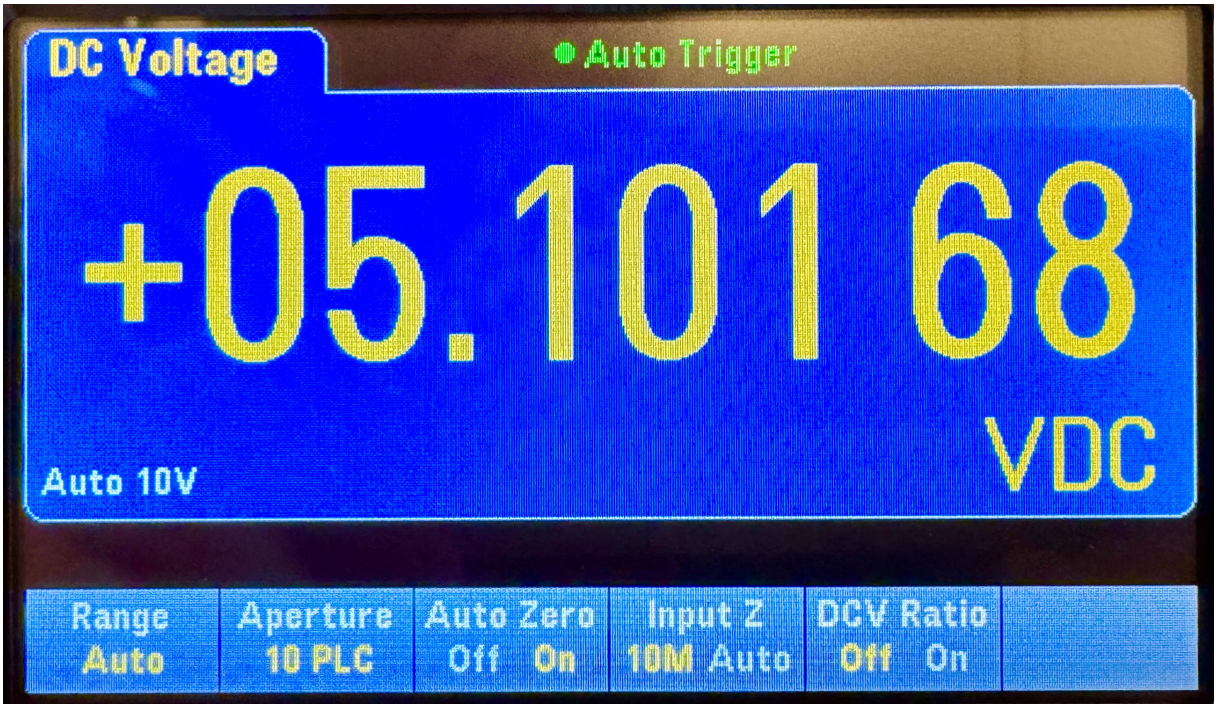


Figure 4.2.5: Measurement for specification 3

4.2.9 Standards

- **ISO/IEC 30116:2016**: This standard establishes a methodology to measure the accuracy of OCR output, including metrics like character recognition rates, error types, and confidence scores.

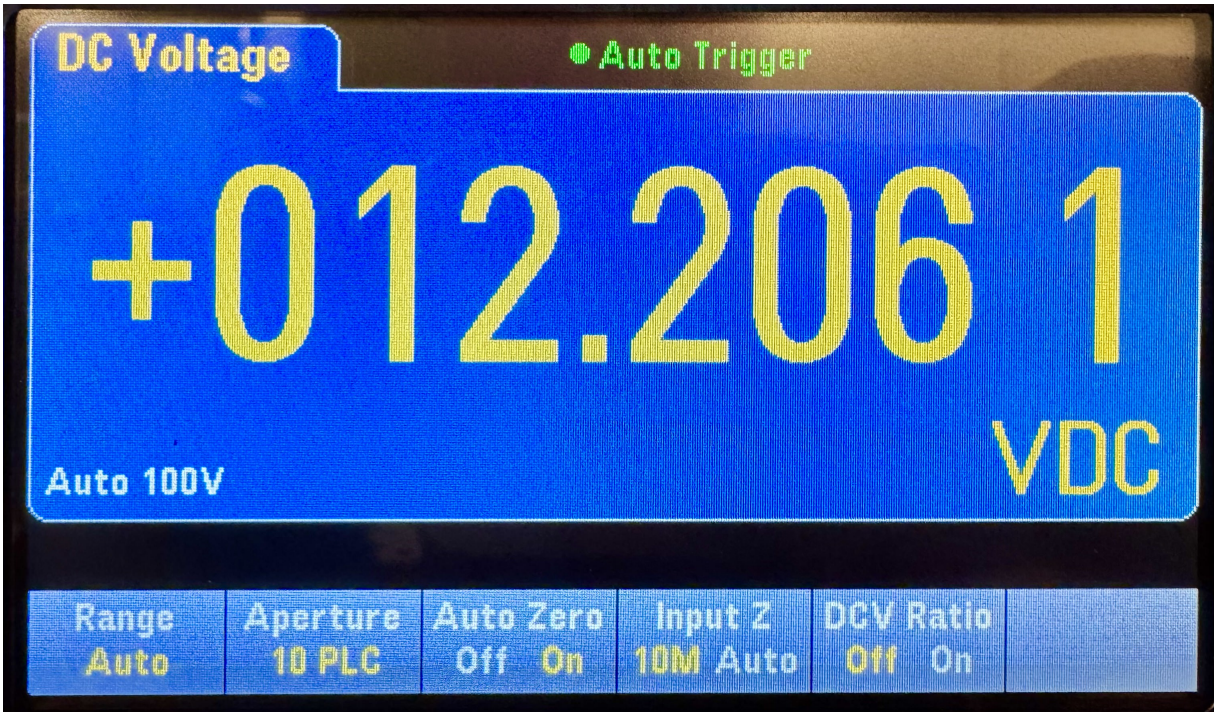


Figure 4.2.6: Measurement for specification 4

- **ISO 1073-1:1976**: Defines OCR-A, which was one of the earliest standardized machine-readable fonts, designed to be both easily distinguishable by computers and readable by humans.

4.3 Subsystem 3: Microcontroller and Wireless Communication

Owner: Dionysius Xaverio

4.3.1 Subsystem Diagrams

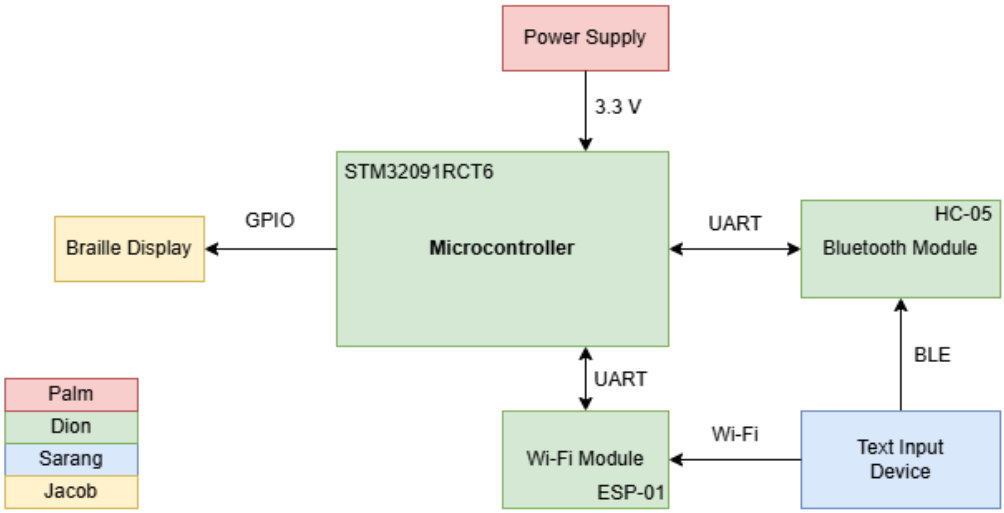


Figure 4.3.1: Microcontroller and Wireless Communication Block Diagram

4.3.2 Specifications

- The compilation code size for the microcontroller should be less than 1 MB
- The wireless communication should have a minimum range of 10 meters
- The microcontroller should be able to receive and process the signal from the wireless communication module in less than 100 miliseconds

4.3.3 Subsystem Interactions

This subsystem interacts with the following:

1. Input System (Text Source):
   Receives text input via mobile app using BLE or Wi-Fi
2. Processing Unit (Braille Translator):
   Converts text into braille encoding using translation algorithms

3. Actuation System (Braille Output Mechanism):
   Sends processed braille data to control the actuators responsible for rendering the braille characters.

4.3.4 Core ECE Design Tasks

- **ECE 36200**: This course uses the STM32 and teaches concepts related to the usage of a microcontroller. The same microcontroller family is used for this project.
- **ECE 26400**: Topics in C programming were introduced in this course, which teaches concepts for embedded C, the programming language used for the microcontroller.

4.3.5 Parts

- Microcontroller: STM32F091RCT6
- Bluetooth Module: HC-05
- Wi-Fi Module: ESP-01

4.3.6 Theory of Operations/Algorithm

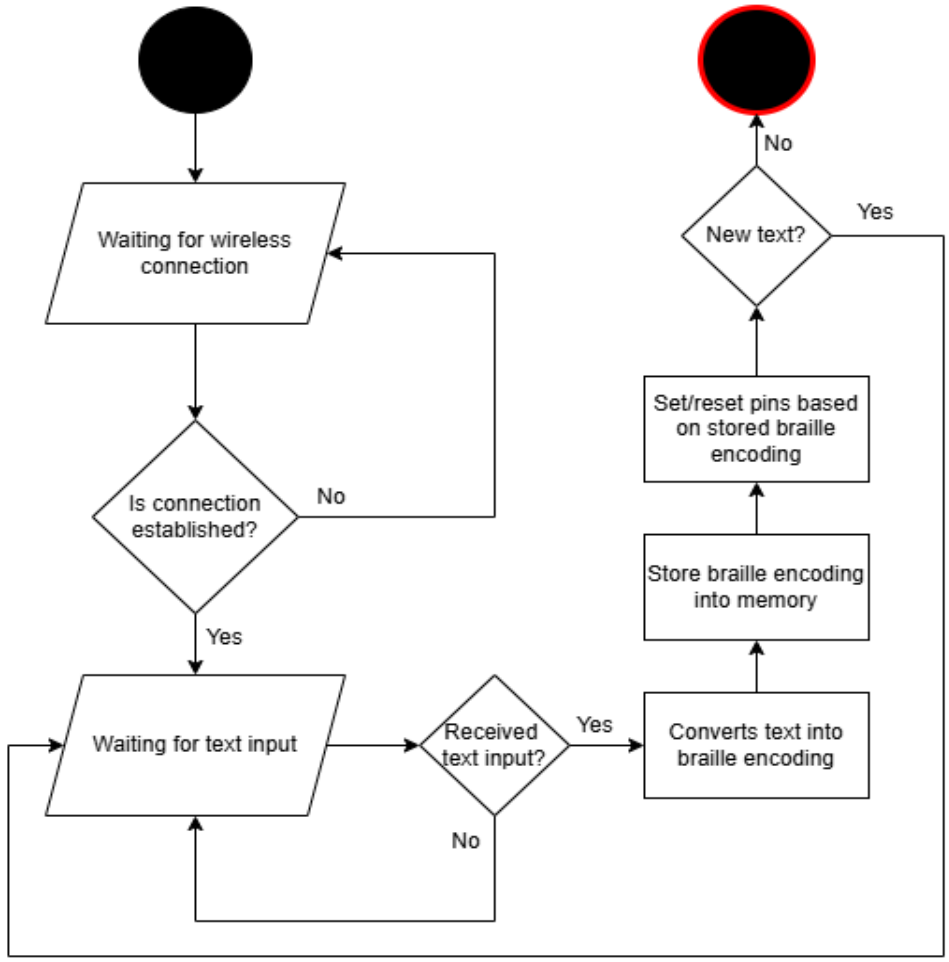


Figure 4.3.2: Microcontroller and Wireless Communication Activity Diagram

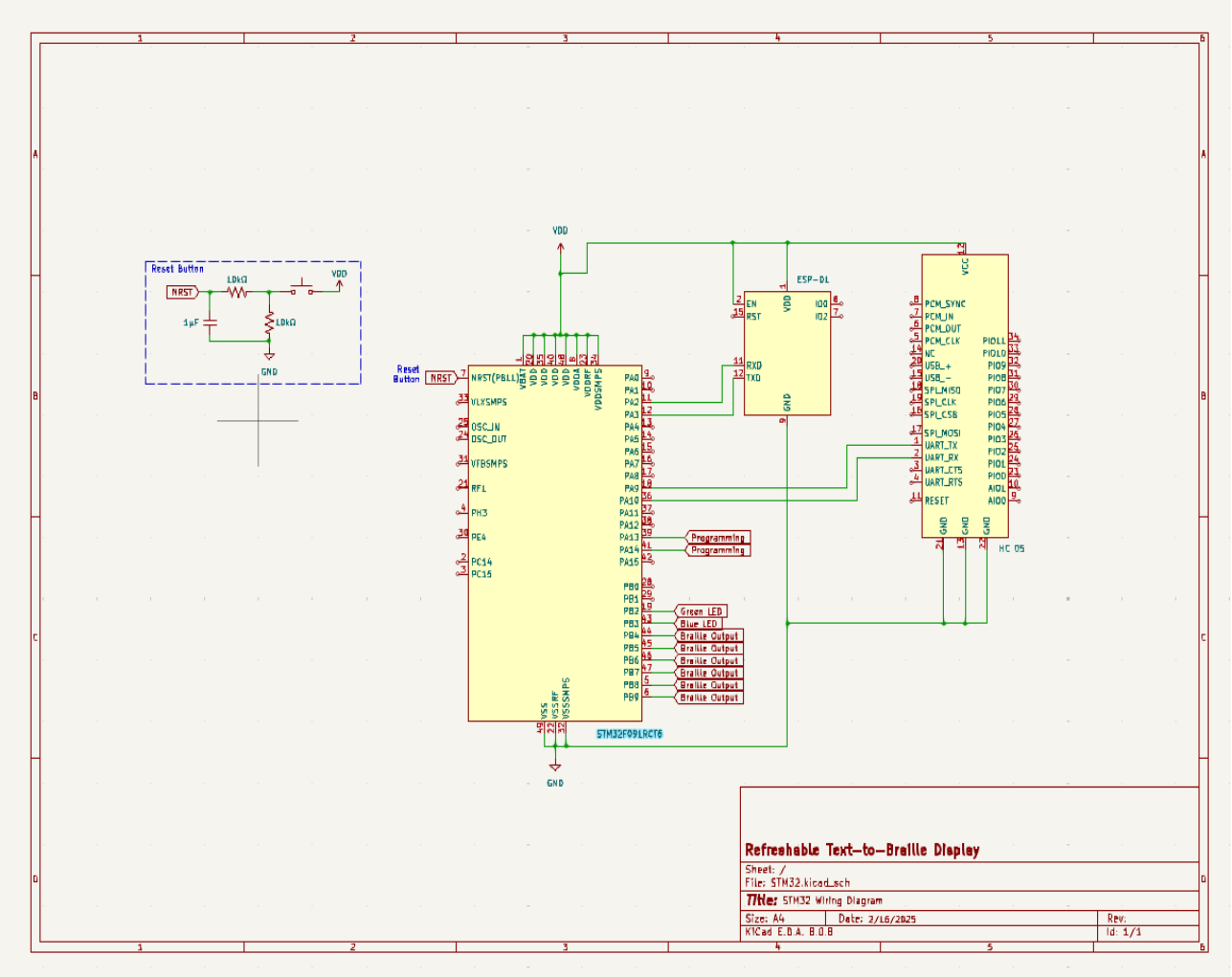


Figure 4.3.3: Microcontroller and Wireless Wiring Diagram

The microcontroller and wireless communication subsystem is responsible for receiving text input, processing it into Braille, and transmitting the data to the braille output mechanism. The purpose of this subsystem is to ensure efficient data handling, low-latency communication, and compliance with accessibility standards. Additionally, the wireless communication of the device can support both Wi-Fi and Bluetooth (BLE) connections. As seen in Figure 4.3.3, the Bluetooth Module is connected to UART1 (PA9 and PA10) and the Wi-Fi Module is connected to UART2 (PA2 and PA3). This configuration allows the device to be controlled via Bluetooth and Wi-Fi simultaneously. To handle the simultaneous connections, the microcontroller will process each of the incoming string in a queue and will display the newest received string to the braille display.

To ensure system reliability, the microcontroller also implemenets several error detection and handling mechanism:

- Wi-Fi Communication Failure: If Wi-Fi connection is unable to connect to a specified host or the connection is lost, the device will continuously attempt reconnection while alerting the user through a blinking green status LED.
- Bluetooth Communication Failure: If Bluetooth connections fails or no device is connected via Bluetooth, the device will continuously re-establish connection while alerting user through a blinking blue LED.

The system workflow for the Microcontroller and Wireless Communication subsystem is as follows:

1. Initialization:
    - Power on the microcontroller and initialize the internal components, including UART, I2C, SPI, GPIO, and wireless modules.
    - Perform self-check to ensure that all peripherals (input interface, wireless communication modules, and actuator control circuit) are functioning correctly.
    - Configure wireless modules to initiate pairing (Bluetooth) or connect to a preconfigured network (Wi-Fi)
2. Text reception:
    - Receive input text string via BLE or Wi-Fi.
    - The received text is temporarily stored in a buffer for processing.
3. Text to Braille translation:
    - Convert received text into the corresponding braille encoding.
    - The text is processed character by character, mapping each to a specific braille representation using a pre-defined lookup table.
    - The translated braille encoding is then stored in memory.
4. Controlling braille display:
    - Turn on/off GPIOs based on the translated braille encoding.
5. Feedback and Monitoring:
    - Continuously monitor transmission status and adjust based on user input.
    - Provide error handling for connectivity issues.

4.3.7 Specifications Measurement

1. The compilation code size for the microcontroller should be less than 1 MB
    - The size of the compilation code is 897 KB, which is less than 1 MB
2. The wireless communication should have a minimum range of 10 meters
    - The wireless module is still able to relay and receive information when it is 10 meters away from the source of the signal.
3. The microcontroller should be able to receive and process the signal from the wireless communication module in less than 100 miliseconds
    - Figure 4.3.4 shows that a signal is processed in approximately 90 milliseconds.

Figure 4.3.4: Signal Processing Delay of the Microcontroller

4.3.8 Standards

- **Braille Encoding Standards:**:
    - Follows Unicode Braille Patterns (U+2800–U+28FF)
- **Wireless Standards:**:
    - Bluetooth SIG Specification v5.0
    - IEEE 802.11 (Local Area Network Technical Standard)
- **Microcontroller Standards:**:
    - ISO 9001 Certified Microcontrollers
    - IEC 61508 (Functional Safety)
- **Accessibility Compliance:**:
    - ADA (Americans with Disabilities Act) Guidelines
    - WCAG (Web Content Accessibility Guidelines) for text input considerations

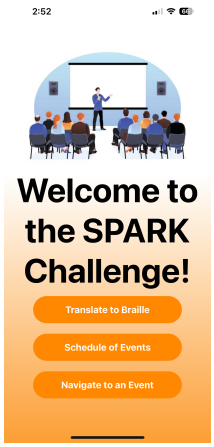
4.3.9 Code Base

Link to the Github Repository can be found here:
https://github.com/DionysiusXaverio/RefreshableBrailleDisplayMicrocontroller

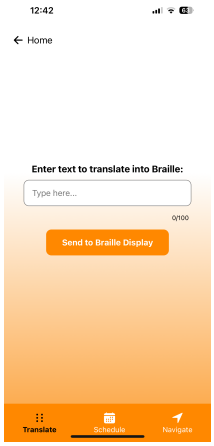
4.4 Subsystem 4: Mobile Application

Owner: Sarang Kashyap

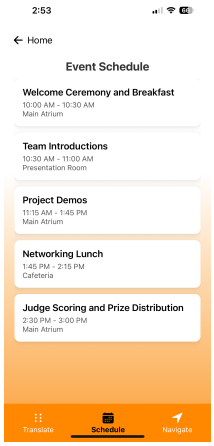
4.4.1 Subsystem Diagrams



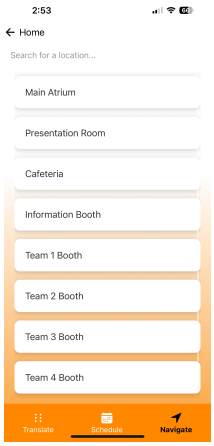
(a) Homepage



(b) Translate to Braille screen



(c) Schedule of Events screen



(d) Navigation to Events screen

Figure 4.4.1: Screens from the TactileCon Mobile App

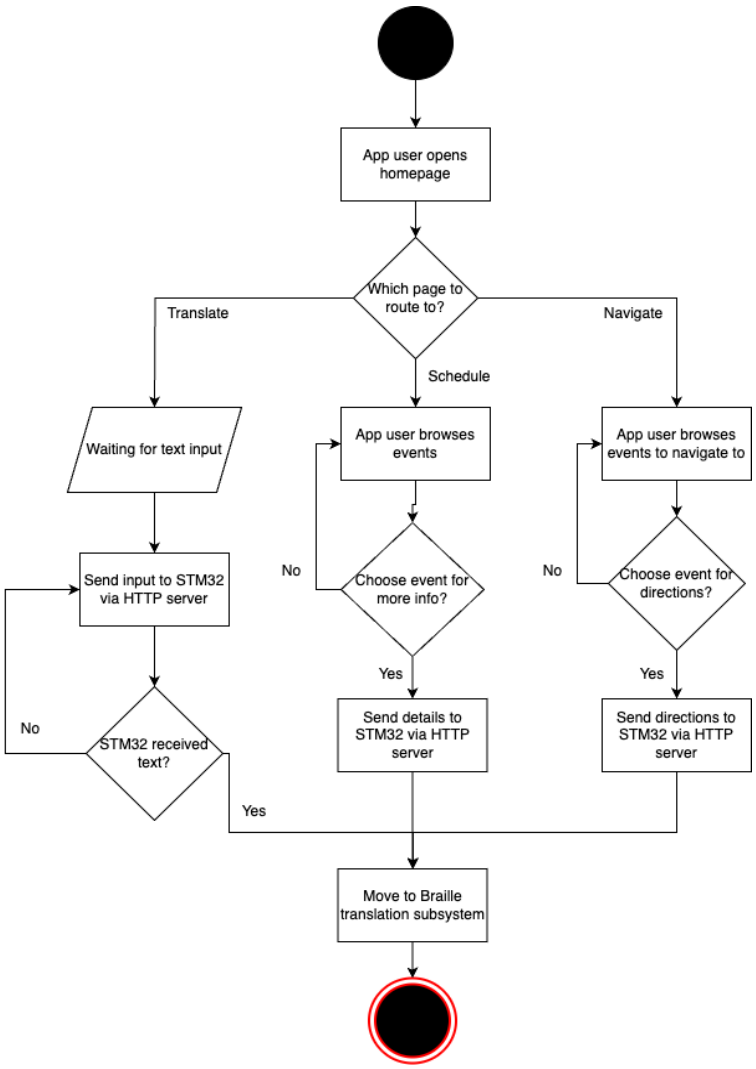


Figure 4.4.2: Mobile App Activity Diagram

4.4.2 Specifications

1. The input box should only accept up to 100 characters.
2. The app should send text to the microcontroller under 0.5 seconds (500 milliseconds).
3. If the text includes anything other than accepted characters (letters, spaces, commas, periods, exclamation marks, question marks), an error message should be sent within 0.5 seconds (500 milliseconds).

4.4.3 Subsystem Interactions

This subsystem interacts with the microcontroller subsystem by transmitting the text for Braille translation via a Wifi-enabled HTTP server, which listens to GET requests for incoming text strings. This translated text is finally displayed on the Braille display.

4.4.4 Core ECE Design Tasks

- **ECE 362**: Will help in understanding how to seamlessly communicate with the STM32.

4.4.5 Parts

Since this is a software-only subsystem, I will not be using any hardware components. However, the software framework I will be using to create the mobile app is React Native, along with prototype designs that will be made using Figma. This is a cross-platform application which means that it can be used on an Android, iPhone, and the Web.

4.4.6 Codebase

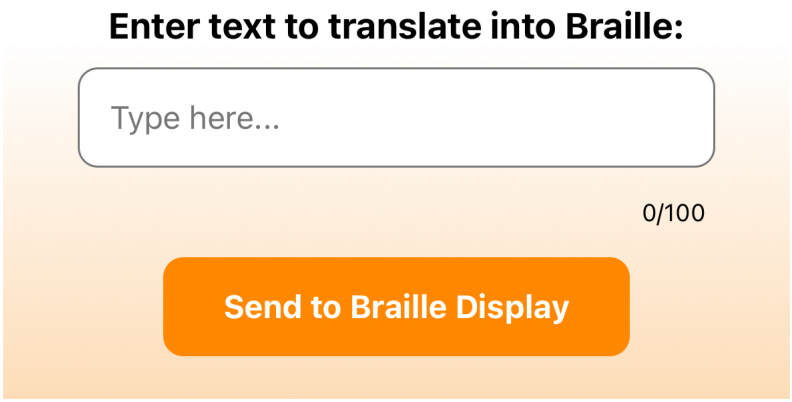
Link to Github repo: https://github.com/sgkashya/TactileCon.

4.4.7 Theory of Operation

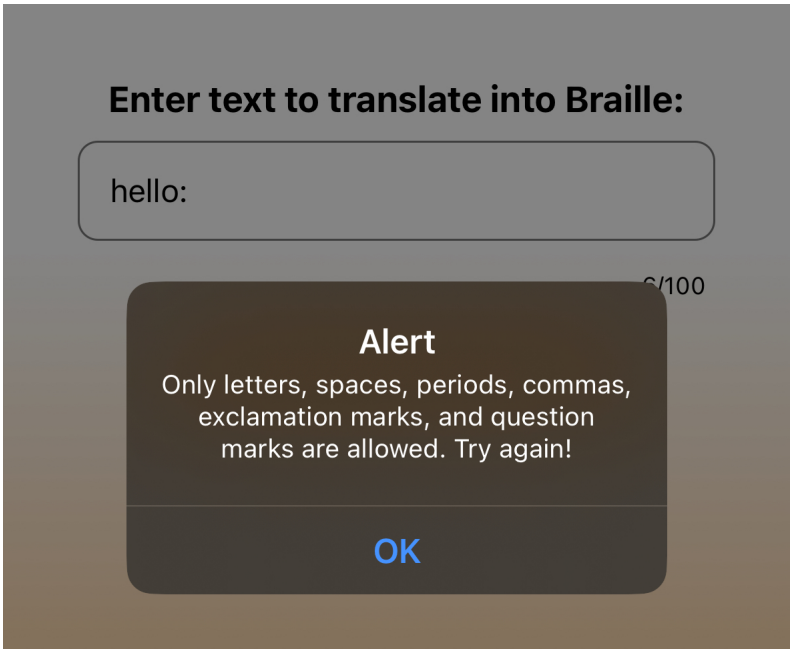
The user, who is the event organizer, will enter the app where they will be greeted by a welcome message about the main event. Then, they can decide which of the three pages they want to navigate to: 'Translate to Braille', 'Schedule of Events', or 'Navigate to an Event'. The first page offers users a text box where they can enter text that they want translated into Braille, which will be communicated via a Wifi-enabled HTTP server to the STM32 microcontroller. This STM32 will run its translation algorithm to display the Braille characters on the Braille signboard. The second page allows the app user to choose an event that they want more information about, and then transmit those details to the signboard to be displayed in Braille. The third and final page will allow users to choose an event that they want directions to, and those directions will be displayed on the Braille

signboard. All in all, the entire purpose of this app is for the event organizers to assist the visually impaired attendees make their way around the events. They will be the main users of this app, as they can choose what information the visually impaired attendees want translated on the signboard.

4.4.8 Specifications Measurement

1. The input box should only accept up to 100 characters.
    - This image shows the character counter under the input box.
    - 

2. The app should send text to the microcontroller under 1 second (1000 milliseconds).
    - This image shows how fast the text was sent (937.02 ms).
    - 

3. If the text includes anything other than accepted characters (letters, spaces, commas, periods, exclamation marks, question marks), an error message should be sent within 1 second (1000 milliseconds).
    - This image shows the error message displaying right after an unacceptable character is typed.
    - 

4.4.9 Standards

- **ISO 12207 and ISO 15288**: Focused on software life cycle processes.

5 PCB Layout

5.1 PCB Schematics

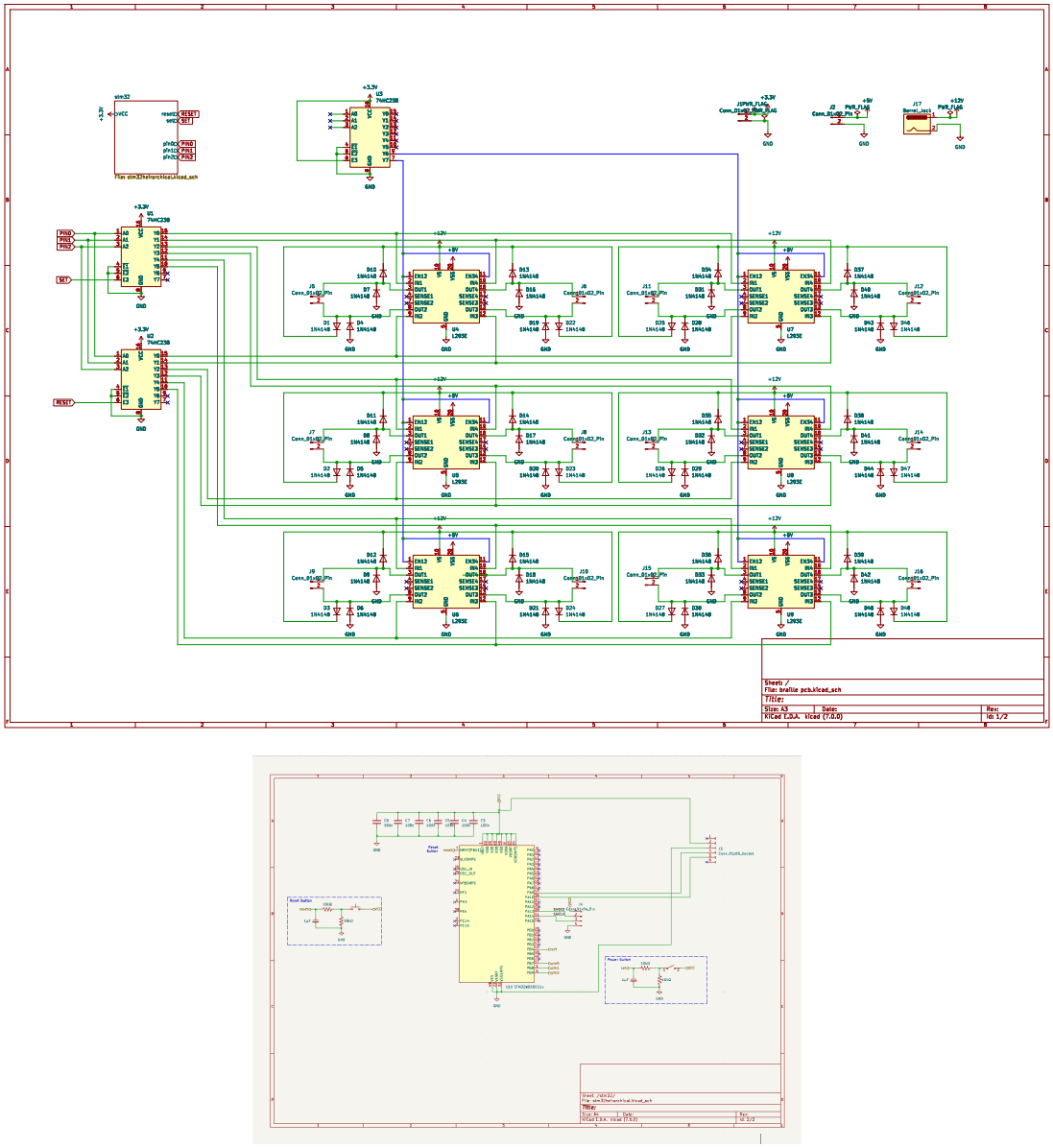


Figure 5.1.1: PCB Schematic

5.2 PCB Layout

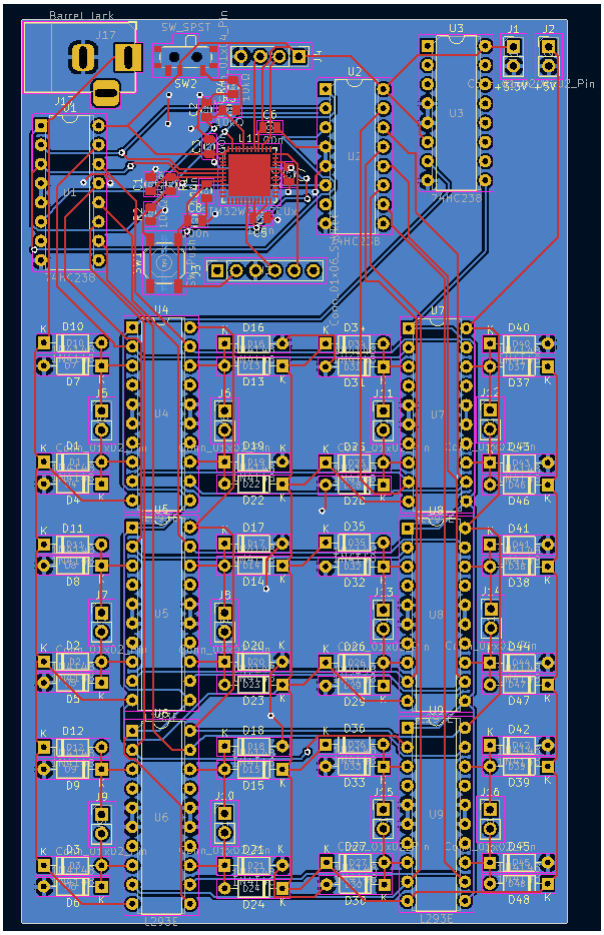


Figure 5.2.1: PCB Layout

6 Final Status of Requirements

1. The device must be able to display readable braille characters
   Met: The device has the ability to display these characters using a lookup table.

2. The device must be able to output braille characters in accordance with text inputs from the user accurately
   Met: The microcontroller has been integrated with the outputs and can receive any text the microcontroller has access to.

3. The mobile app should be able to communicate with the microcontroller via a Wifi connection.
   Met: The app is able to send data to the microcontroller.

4. The device must be able to take text input wirelessly using a smartphone.
   Met: The device is currently able to receive the text input from the smartphone.

5. The device must be able to process images containing text and translate it into braille
   Met: The device is able to process text from images, send it to the microcontroller, and display the output as braille characters.

7 Team Structure

7.1 Team Member 1

Sarang Kashyap
Major: Computer Engineering
Contact: sgkashya@purdue.edu
Team Role: Mobile App
Bio: Sarang is a senior in Computer Engineering from Pleasanton, California. He has professional experience in software engineering and cybersecurity.

7.2 Team Member 2

Jacob Zhang
Major: Computer Engineering
Contact: zhan4534@purdue.edu
Team Role: Braille Output
Bio: Jacob is from Carmel, Indiana. He has a concentration in AI/ML and a minor in mathematics.

7.3 Team Member 3

Dionysius Xaverio
Major: Computer Engineering
Contact: dxaverio@purdue.edu
Team Role: Microcontroller and Wireless Communication
Bio: Dionysius Xaverio is a Senior in the Computer Engineering program at Purdue University. Previously, he has taken multiple classes regarding Computer Networking and Security, and also has professional experience in Database Management.

7.4 Team Member 4

Tanin Padungkirtsakul (Palm)
Major: Computer Engineering
Contact: tpadungk@purdue.edu
Team Role: Computer Vision and Power Supply
Bio: Palm is a senior in ECE, from Bangkok, Thailand. He has a lot of experience in ML.