GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

TOOL THAT TRANSLATES PRINTED TEXT TO BRAILLE

Autor: Blas Alejandro Calatayud Cerezo

Director: Viktor Gruev

Madrid

Junio 2023

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

“TOOL THAT TRANSLATES PRINTED TEXT TO BRAILLE”

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2022 / 2023 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Blas Alejandro Calatayud Cerezo Fecha: 13 / 06 / 2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Viktor Gruev Fecha: 14 / 06 / 2023

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

TOOL THAT TRANSLATES PRINTED TEXT TO BRAILLE

Autor: Blas Alejandro Calatayud Cerezo

Director: Viktor Gruev

Madrid

Junio 2023

Acknowledgments

I would like to thank my family, friends, Universidad Pontificia Comillas, and the University of Illinois at Urbana-Champaign for their support throughout this academic journey.

I am deeply grateful to my family and friends for their continuous support. Their encouragement and belief in my abilities has been invaluable, providing me with the strength to overcome challenges and achieve my goals.

I would also like to extend my sincere appreciation to both Universidad Pontificia Comillas and the University of Illinois at Urbana-Champaign. Universidad Pontificia Comillas has provided a magnificent academic environment, with exceptional faculty members and staff who have guided and mentored me. Similarly, the University of Illinois at Urbana-Champaign granted me the opportunity to collaborate on this project during my exchange program, exposing me to new perspectives, prestigious professors, and excellent lab facilities with an incredible range of tools available.

Finally, I would like to thank all the individuals who have contributed directly or indirectly to this project. Their constructive feedback has significantly improved the quality of my work.

Tool that translates printed text to braille

Autor: Blas Alejandro Calatayud Cerezo

Director: Viktor Gruev

Entidad colaboradora: University of Illinois at Urbana-Champaign

Resumen del proyecto

El objetivo principal de este proyecto es diseñar y construir un dispositivo capaz de realizar una traducción en tiempo real de un texto impreso a braille. Esto ayudaría a las personas que solo pueden leer braille a ser más independientes en su vida diaria.

El dispositivo final sería en una pequeña caja que se podría sostener con una mano. En su interior contendría una cámara que se utilizaría para hacer fotografías del texto escrito y seis actuadores lineales que se usarían para recrear los caracteres en braille. La persona que utilice este dispositivo sujetaría la caja con una mano y la colocaría encima del texto escrito.

Palabras clave: braille, traducción, PCB (placa de circuito impreso por sus siglas en inglés), OCR (reconocimiento óptico de caracteres por sus siglas en inglés), cámara, actuador lineal.

1. Introducción

Según la Organización Mundial de la Salud, actualmente hay alrededor de 39 millones de personas legalmente ciegas en todo el mundo. Estas personas no pueden leer las letras del alfabeto tradicional, ya que no pueden verlas. En su lugar, utilizan el sistema braille. El braille es una forma de lenguaje escrito para personas ciegas que les permite leer texto con sus dedos. Cada carácter se recrea en una celda que contiene seis puntos divididos en tres filas con dos puntos por fila. Algunos de esos puntos están más elevados que los demás para formar un carácter, número o signo de puntuación específico. La imagen de abajo muestra el alfabeto braille:

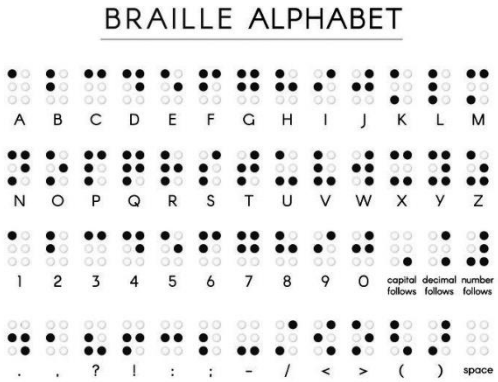

Ilustración 1. *Alfabeto braille.*

Actualmente no hay muchos recursos disponibles para que las personas que solo pueden leer braille puedan leer también el texto impreso de libros o revistas.

Este proyecto consistirá en desarrollar una herramienta que pueda colocarse sobre el texto impreso y traducirlo al braille en tiempo real, para que las personas ciegas puedan leerlo.

1.1. Objetivos principales

Este proyecto tiene como objetivo cumplir con estos tres requisitos principales:

- **La precisión de la traducción de caracteres de texto a braille debe ser de un 90% aproximadamente:** El diseño final del proyecto deberá garantizar una traducción precisa relacionando correctamente los caracteres del texto impreso con los caracteres y símbolos braille correspondientes. El diseño final será probado múltiples ocasiones para intentar lograr una precisión de traducción del 90%.
- **La duración de la batería debe ser suficiente como para durar un día completo con una sola carga:** El diseño final del proyecto debe ser portátil. Por lo tanto, el consumo de energía de todo el dispositivo debe ser lo suficientemente bajo como para permitir que la batería dure todo el día.
- **Todos los pines se deben extender una altura legible (mínimo de 0,2 cm) en menos de 1 segundo desde que se activan:** El diseño final del proyecto debe procesar el texto impreso y producir la salida braille en tiempo real con un retraso mínimo, pero dejando tiempo suficiente para que el usuario pueda leer el carácter braille formado por los pines.

2. Diseño

2.1. Diseño del hardware

Se decidió dividir este proyecto en cuatro subsistemas. Cada subsistema llevará a cabo una función concreta del proyecto. Si todos los subsistemas pueden cumplir sus funciones de manera independiente, entonces el proyecto debería ser capaz de funcionar como un todo. La siguiente imagen muestra los cuatro subsistemas en los que se divide el proyecto:

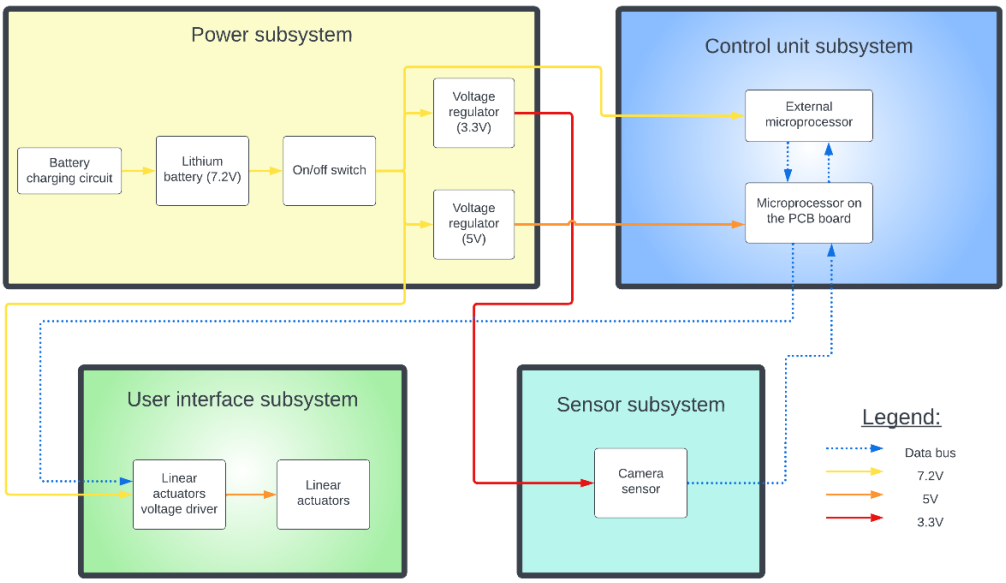

Ilustración 2. *Diagrama de bloques del proyecto.*

A continuación se muestra una breve explicación de cada subsistema y de las funciones que llevará a cabo:

1) **Subsistema de gestión de la energía:** el subsistema de energía alimentará todos los demás subsistemas. Estará compuesto por una batería recargable de polímero de litio que suministrará aproximadamente 7,2V, un interruptor de encendido/apagado del dispositivo, dos reguladores de voltaje diferentes que reducirán la tensión de la batería a 5V y 3,3V respectivamente, y un conector USB-C utilizado para cargar la batería a través de un circuito de carga de batería.
2) **Subsistema de sensores:** este subsistema contendrá un sensor de cámara, que se utilizará para hacer fotografías del texto impreso.
3) **Subsistema de unidad de control:** una placa PCB personalizada (placa de circuito impreso por sus siglas en inglés) contendrá un microcontrolador que interactuará con el sensor de cámara y con los actuadores lineales. Un segundo microprocesador más potente realizará OCR (reconocimiento óptico de caracteres por sus siglas en inglés) para identificar todos los caracteres presentes en cada palabra.
4) **Subsistema de interfaz de usuario:** seis actuadores lineales se moverán hacia arriba y hacia abajo de manera síncrona para recrear caracteres braille. El usuario podrá leer esos caracteres utilizando su dedo.

2.2. Diseño exterior

Se creó un diseño exterior que contendría todos los componentes electrónicos utilizados en el proyecto utilizando el software *Autodesk Fusion 360*. El diseño final se puede ver a continuación:

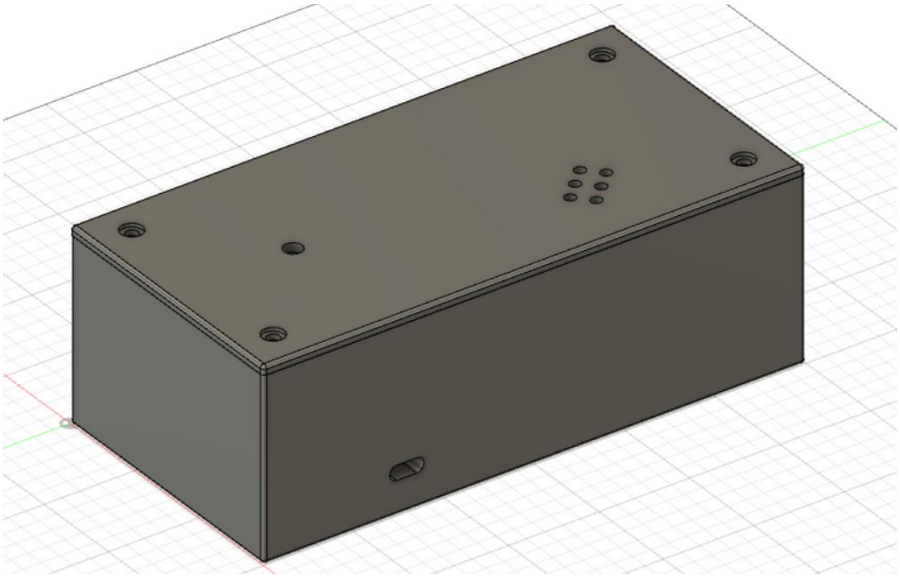

Ilustración 3. *Diseño exterior completo.*

El diseño exterior final del proyecto se asemeja a una caja rectangular. Esto se debe a que se buscaba un diseño sencillo y fácil de sujetar con una mano. La caja está dividida en dos partes que se mantienen unidas mediante 4 tornillos: una tapa y un contenedor.

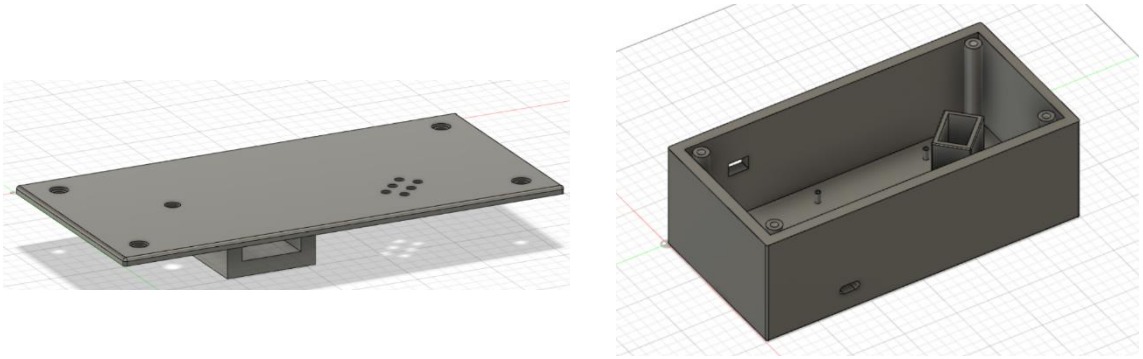

Ilustración 4. *Tapa (izquierda) y contenedor (derecha) diseñados.*

3. Resultados

Se logró y se demostró una funcionalidad completa de cada subsistema individual, cumpliendo con los requisitos de cada subsistema y con los objetivos principales del proyecto. La siguiente tabla resume algunos de estos logros:

Requisito	**Verificación**	**Logro (Sí o No)**
Los reguladores de voltaje limitarán el voltaje al valor correcto.	Se utilizará un voltímetro para medir la diferencia de tensión presente en ambos lados de cada regulador de voltaje.	Sí
El subsistema de gestión de energía podrá cargar la batería de forma segura.	Se medirá el voltaje de la batería utilizando un voltímetro para asegurarse de que el aumento de voltaje sea el esperado mientras carga.	Sí
La cámara debe ser capaz de tomar una fotografía con una resolución de 480p y enviar los datos al subsistema de unidad de control.	El software de prueba OCR recibirá fotos con una resolución de 480p.	Sí
El algoritmo OCR en el microprocesador externo debe ser capaz de analizar la imagen y reconocer los caracteres presentes en el texto con una precisión del 90%.	El software OCR utilizará un conjunto de datos de prueba compuesto por imágenes de caracteres con una resolución de 480p. El algoritmo OCR se ejecutará en este conjunto de datos para determinar su precisión.	Sí
Los actuadores lineales deben ser capaces de elevar los pines a una altura de 0,20±0,1 cm y bajarlos en menos de un segundo al formar los caracteres braille.	Se medirá la altura de elevación cuando todos los solenoides estén encendidos y se utilizará una regla para verificar la altura.	Sí

Tabla de datos 1. *Resumen de algunos requisitos y verificaciones de los subsistemas*

Sin embargo, el objetivo principal del proyecto era diseñar una herramienta capaz de realizar una traducción en tiempo real de texto impreso a braille.

Desafortunadamente, no fue posible integrar completamente todos los subsistemas entre sí para producir un producto final funcional. Esto se debe principalmente a la incapacidad de programar el ATmega32U4 debido a un problema con las conexiones de puesta a tierra diseñadas en la PCB. Para poder demostrar que todos los subsistemas funcionaban de forma individual, se utilizó en su lugar un microprocesador Arduino Uno.

4. Conclusión

Durante el desarrollo de este proyecto hubo que enfrentarse a un obstáculo significativo, las limitaciones del Arduino Uno para funcionar como dispositivo USB. Al carecer este microprocesador de un controlador USB incorporado como el ATmega32U4, no se pudo realizar una integración completa del sistema con el microprocesador Libre AML-S905X-CC. Por lo que fue imposible lograr confeccionar un dispositivo final que cumpliese con la función principal del proyecto, la de realizar una traducción en tiempo real de texto impreso a braille.

Sin embargo, es importante destacar que todos los subsistemas individuales cumplieron satisfactoriamente con sus requisitos y se demostró su funcionalidad con imágenes. A su vez, se lograron satisfactoriamente los objetivos principales del proyecto y se demostró su correcto funcionamiento.

Para transformar este prototipo en un dispositivo completamente operativo capaz de realizar una traducción en tiempo real de texto impreso a braille, quedan dos pasos clave. En primer lugar, se debe rediseñar las conexiones de tierra en la PCB diseñada para poder utilizar el microprocesador ATmega32U4. En segundo lugar, se debe desarrollar el código necesario para facilitar la comunicación entre los microprocesadores ATmega32U4 y Libre AML-S905X-CC. Afortunadamente, abordar estos desafíos es un proceso sencillo, y es muy desafortunado que el problema con el plano de tierra no se detectase antes durante la fase de pruebas.

A pesar de los obstáculos encontrados durante la integración de los subsistemas, el proyecto ha sentado una base sólida y, con estos ajustes finales, está bien posicionado para lograr el objetivo deseado y convertirse en un dispositivo completamente funcional.

5. Bibliografía

[1]. World Health Organization (WHO). “Blindness and vision impairment”. (2022). [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (last visited on 02/09/2023).

[2]. Circuitdigest. “Solenoid Driver Circuit”. [Online]. Available: https://circuitdigest.com/electronic-circuits/solenoid-driver-circuit-diagram (last visited on 05/03/2023).

[3]. Punkisnail. “Li-ion Battery Charger”. [Online]. Available: https://www.instructables.com/Li-ion-Battery-Charger/ (last visited on 05/03/2023).

[4]. Circuitdigest. “How to Use OV7670 Camera Module with Arduino Uno”. (2019). [Online]. Available: https://circuitdigest.com/microcontroller-projects/how-to-use-ov7670-camera-module-with-arduino (last visited on 05/03/2023).

[5]. SMtech. “9V to 5V converters”. [Online]. Available: https://somanytech.com/9v-to-5v-converter-circuit/ (last visited on 06/13/2023).

[6]. G7smy. Karl. “Solenoids on the Arduino with MOSFET power”. (2015). [Online]. Available: https://www.g7smy.co.uk/2015/02/solenoids-on-the-arduino-with-mosfet-power/ (last visited on 06/13/2023).

[7]. SHDesigns. Henion, Scott. “Lithium poly charge circuit”. (2003). [Online]. Available: http://shdesigns.org/lionchg.shtml (last visited on 06/13/2023).

[8]. American Foundation for the Blind (AFB). “What is braille?”. [Online]. Available: https://www.afb.org/blindness-and-low-vision/braille/what-braille (last visited on 04/20/2023).

Tool that translates printed text to braille

Author: Blas Alejandro Calatayud Cerezo

Supervisor: Viktor Gruev

Collaborating entity: University of Illinois at Urbana-Champaign

Abstract

The main objective of the project is to design and build a tool capable of performing a real time translation from printed text to braille. This would help people who are only able to read braille to become more independent by allowing them to participate more than they do now in aspects of daily life such as in their education or their work.

The developed tool would consist of a handheld box containing a camera sensor used to take pictures of the letters in a word and six linear actuators used to output the braille characters. The user would hold this handheld device with one hand and place it on top of written words.

Keywords: braille, translation, PCB (printed circuit board), OCR (optical character recognition), camera, linear actuator.

1. Introduction

According to the World Health Organization, there are currently around 39 million people who are legally blind around the world. These people are unable to read the letters in the alphabet, as they cannot see them. They use the braille system instead. Braille is a form of written language for blind people that allows them to read text with their fingers. Each character is recreated in a cell that contains six dots divided in three rows with two dots per row. Some of those dots are raised higher than the others to form a specific character, number, or punctuation mark. The image bellow shows the braille alphabet:

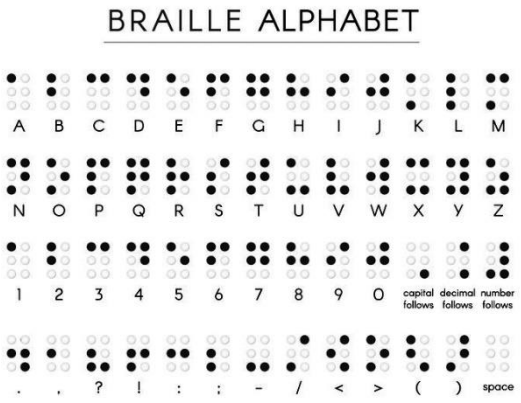

Illustration 1. *Braille alphabet*.

Right now, there are not many resources available for people who can only read braille to read physical written text from a book or magazine.

This project will focus on developing a tool that can be placed over printed text and translate it to braille in real time so that blind people can read it.

1.1. High-level requirements

This project aims to accomplish these three high-level objectives:

- **Text character to braille display accuracy is around 90%:** The final design of the project would need to ensure an accurate translation by correctly relating characters in printed text to the corresponding braille characters and symbols. The final design will be tested numerous times to attempt to achieve a 90% translation accuracy.
- **Battery life lasts on a single charge for an all-day battery life:** the final design of the project should be portable. Therefore, the power consumption of the whole device should be low enough to allow the battery to last a whole day.
- **All pins extend to a readable high (minimum of 0.2 cm) in less than 1 second since they are activated:** the final design of the project would need to process the printed text and produce the braille output in real time with a minimal delay, but leaving sufficient time for the user to read the braille character made by the pins.

2. Design

2.1. Hardware design

It was decided to divide this project into four smaller subsystems. Each subsystem will carry out a concrete function of the project. If all the subsystems can carry out their functions independently, then the project should be able to work as a whole. The following image shows the four subsystems in which the project is divided:

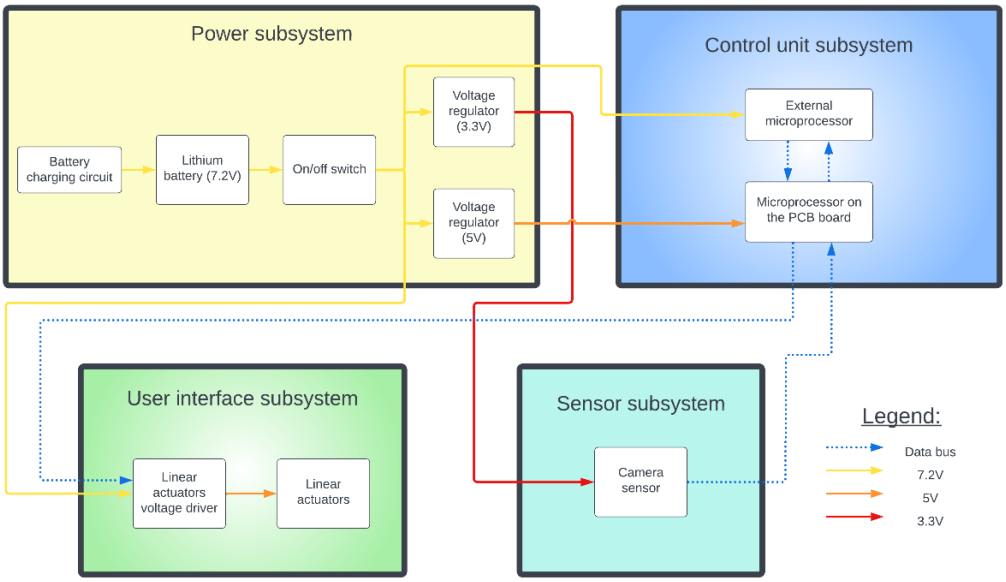

Illustration 2. *Block diagram of the project.*

Here is a brief explanation of each subsystem and the functions they will carry out:

1) **Power management subsystem:** the power subsystem will power the whole system. It will consist of a rechargeable Lithium Polymer (LiPo) battery supplying around 7.2V, an on/off switch, two different voltage regulators that step down the battery voltage to 5V and 3.3V respectively and a USB-C used to charge the battery through a battery charging circuit.
2) **Sensor subsystem:** this subsystem will contain the camera sensor, which will be used to take pictures of the printed text.
3) **Control unit subsystem:** a custom PCB board will contain a microcontroller which will interact with the camera sensor and with the linear actuators. A second and more powerful microprocessor will perform OCR (Optical Character Recognition) to identify all the characters in a word.
4) **User interface subsystem:** six linear actuators will move up and down in a synchronous way to recreate braille characters. The user will be able to read those braille characters using their finger.

2.2. Physical design

A physical design that would contain all the electronic components used in the project was developed using the *Autodesk Fusion 360* software. The final design can be seen bellow.

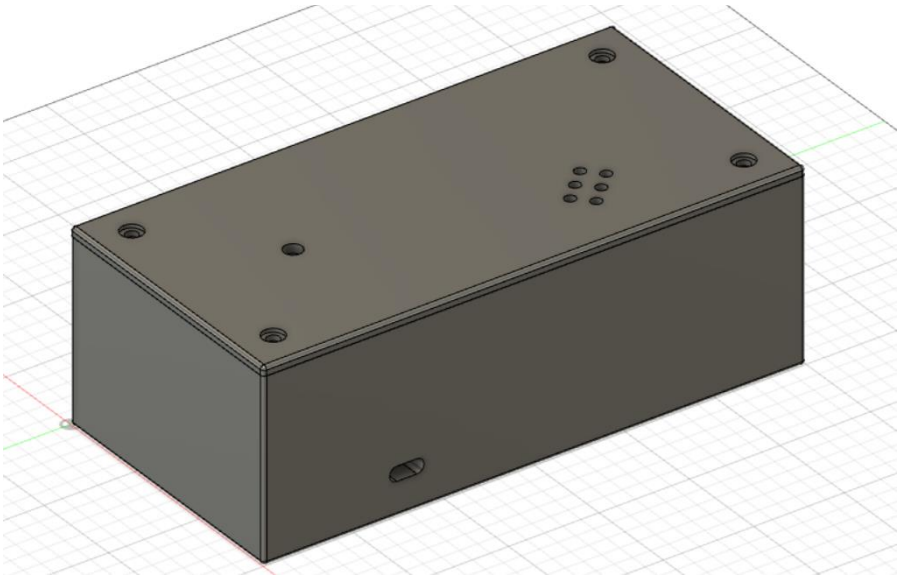

Illustration 3. *Complete physical design.*

The final physical design of the project looks like a rectangular box. This is because a simple design easy to hold with one hand was wanted. The box is divided into two parts which are hold together using 4 screws: a cover and a container.

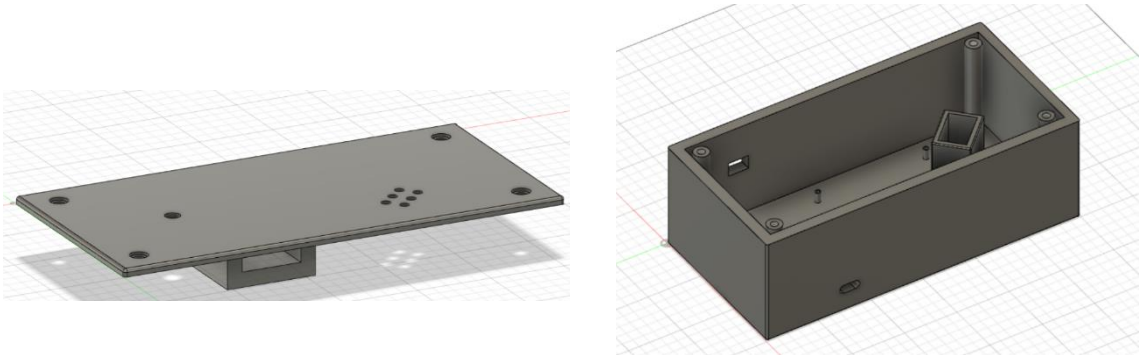

Illustration 4. *Cover (left) and container (right) designed.*

3. Results

A complete functionality of each individual subsystem was achieved and demonstrated as all the subsystem requirements and high-level requirements were met. The table below summarizes some of these achievements:

Requirement	Verification	Accomplished (Yes or No)
The voltage regulators will limit the voltage to the correct value.	Use a voltmeter to measure voltage through voltage regulators under normal operation.	Yes
The power management subsystem will be able to safely charge the battery	The voltage can be measured during testing using a voltmeter to ensure there is the expected voltage increase.	Yes
The camera must be able to take a 480p resolution photo and send the data to the control unit system.	The OCR testing software receives 480p resolution photos.	Yes
The OCR algorithm on the external microprocessor must be able to analyze image data and convert to character texts with 90% accuracy rate.	The OCR testing suite will use a test dataset of 480p resolution images of characters, the OCR algorithm will be run against this dataset to determine its accuracy.	Yes
The linear actuators must be able to lift the pins 0.20$\pm$0.1 cm high and to lower them in less than 1 second when forming the braille characters.	Measure rise height when all solenoids are powered on and use a ruler to check height.	Yes

Tabular data 1. *Summary of some subsystem requirements and verifications.*

However, the main objective of the project was to design a tool capable of performing a real time translation from printed text to braille.

Unfortunately, it was not possible to fully integrate every subsystem with each other to produce a final working product. This is mainly because of the inability to program the ATmega32U4 due to an issue with the grounding connections designed on the custom PCB. In order to be able to demonstrate that all the subsystems worked individually an Arduino Uno microprocessor was used instead of the ATmega32U4.

4. Conclusion

In the process of developing this project the largest obstacles were faced when interfacing all the subsystems into a final device. The inability of the Arduino Uno to act as a USB controller in contrast to the ATmega32U4 led to not been able to achieve a full system integration.

The ATmega32U4 has a built-in USB controller which allows it to ack as a USB device. This feature was going to be used to interface the ATmega32U4 with the external Libre AML-S905X-CC microprocessor. The microcontroller on the Arduino Uno (ATmega328P) does not have this feature and therefore it was not possible to connect it with the Libre AML-S905X-CC microprocessor.

However, all the individual subsystems requirements were met, and it was possible to demonstrate their functionality with pictures. This means that all the subsystems of the project successfully worked on their own. Moreover, all the high-level requirements of the project were met and proved.

Therefore, the only thing needed to obtain a final working device that accomplishes the task of performing a real time translation from printed text to braille is to redesign the grounding connections on the custom PCB and to design the code that carries out the interface between the ATmega32U4 and the Libre AML-S905X-CC microprocessors. This is a straightforward procedure that could have been done if the issue on the grounding plane of the PCB had been discovered earlier in the debugging process.

Despite the obstacles encountered while trying to achieve a full subsystem integration, the project has established a solid foundation, and with these final adjustments, it is well positioned to achieve the desired objective and become a fully functional device.

5. Bibliography

[1]. World Health Organization (WHO). “Blindness and vision impairment”. (2022). [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (last visited on 02/09/2023).

[2]. Circuitdigest. “Solenoid Driver Circuit”. [Online]. Available: https://circuitdigest.com/electronic-circuits/solenoid-driver-circuit-diagram (last visited on 05/03/2023).

[3]. Punkisnail. “Li-ion Battery Charger”. [Online]. Available: https://www.instructables.com/Li-ion-Battery-Charger/ (last visited on 05/03/2023).

[4]. Circuitdigest. “How to Use OV7670 Camera Module with Arduino Uno”. (2019). [Online]. Available: https://circuitdigest.com/microcontroller-projects/how-to-use-ov7670-camera-module-with-arduino (last visited on 05/03/2023).

[5]. SMtech. “9V to 5V converters”. [Online]. Available: https://somanytech.com/9v-to-5v-converter-circuit/ (last visited on 06/13/2023).

[6]. G7smy. Karl. “Solenoids on the Arduino with MOSFET power”. (2015). [Online]. Available: https://www.g7smy.co.uk/2015/02/solenoids-on-the-arduino-with-mosfet-power/ (last visited on 06/13/2023).

[7]. SHDesigns. Henion, Scott. “Lithium poly charge circuit”. (2003). [Online]. Available: http://shdesigns.org/lionchg.shtml (last visited on 06/13/2023).

[8]. American Foundation for the Blind (AFB). “What is braille?”. [Online]. Available: https://www.afb.org/blindness-and-low-vision/braille/what-braille (last visited on 04/20/2023).

Contents

List of figures

List of tables

Chapter 1. Introduction

According to the World Health Organization, there are currently around 39 million people who are legally blind around the world. These people are unable to read the letters in the alphabet, as they cannot see them. They use the braille system instead. Braille is a form of written language for blind people that allows them to read text with their fingers. Each character is recreated in a cell that contains six dots divided in three rows with two dots per row. Some of those dots are raised higher than the others to form a specific character, number, or punctuation mark. The image bellow shows the braille alphabet:

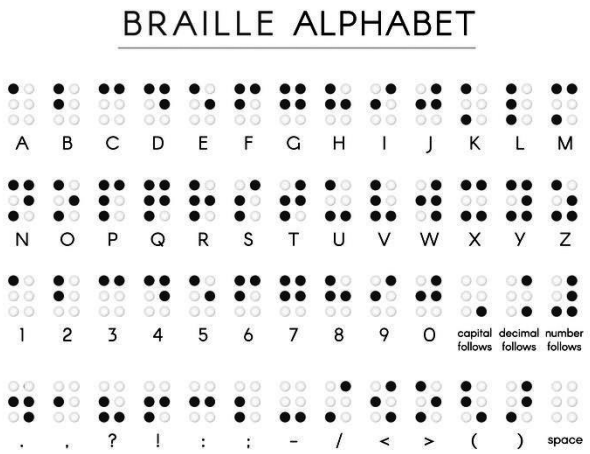

Figure 1. Braille alphabet.

Right now, there are not many resources available for people who can only read braille to read physical written text from a book or magazine.

This project will focus on developing a tool that can be placed over printed text and translate it to braille in real time so that blind people can read it.

The developed tool would consist of a handheld box containing a camera sensor used to take pictures of the letters in a word. The user would hold this handheld device with one hand and place it on top of written words. The box would also contain a PCB board with a microcontroller and an external battery module. It will receive the images taken by the camera, process them to recognize every letter on the word and finally output in braille the characters of that word one by one using pins that can be pushed up and down with linear actuators to recreate braille characters.

The person using this device will place one of their fingers on top of the moving pins used to create the braille characters to read the printed text.

After showing all the braille characters in a word, the user would simply move to the next word for it to be shown in braille.

1.1. Visual aid

The image below shows a representation of how the final project would look like:

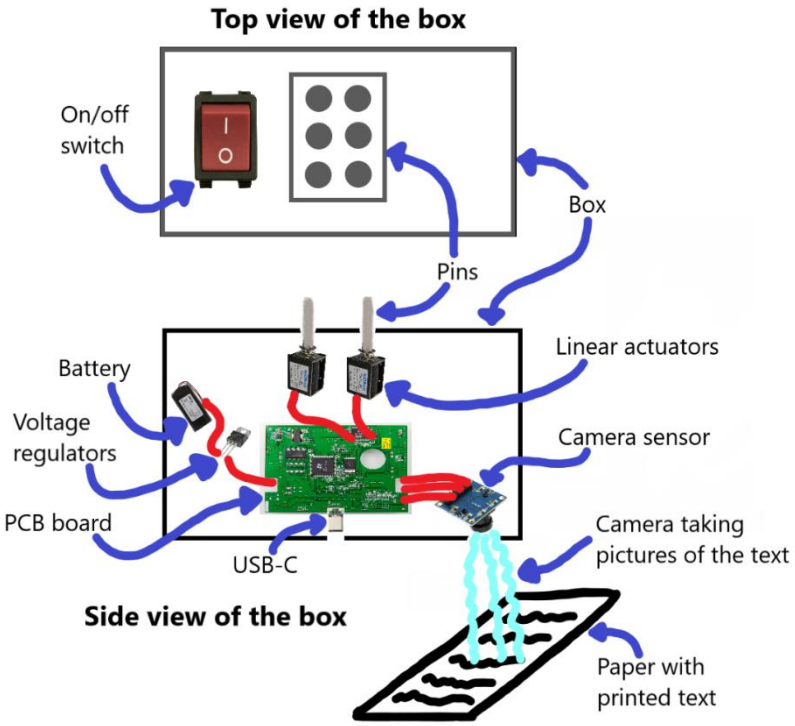

Figure 2. High-level overview of the final design.

The full list of components chosen to carry out the project can be found on appendix A.

1.2. High-level requirements

As said previously, the main objective of the project is to design a tool capable of performing a real time translation from printed text to braille.

In order to do this, this project aims to accomplish these three high-level objectives:

- **Text character to braille display accuracy is around 90%:** The final design of the project would need to ensure an accurate translation by correctly relating characters in printed text to the corresponding braille characters and symbols. The final design will be tested numerous times to attempt to achieve a 90% translation accuracy.
- **Battery life lasts on a single charge for an all-day battery life:** the final design of the project should be portable. Therefore, the power consumption of the whole device should be low enough to allow the battery to last a whole day.
- **All pins extend to a readable high (minimum of 0.2 cm) in less than 1 second since they are activated:** the final design of the project would need to process the printed text and produce the braille output in real time with a minimal delay, but leaving sufficient time for the user to read the braille character made by the pins.

Chapter 2. Sustainable development goals (SDG)

The Sustainable Development Goals (SDG) are 17 objectives that the United Nations general assembly developed in 2015 to be able to achieve a more sustainable and equal world by 2030.

This project is aligned with the following Sustainable Development Goals:

- **Goal 4. Quality education:** Blind and visually impaired people tend to face many difficulties when accessing education and learning opportunities. This is because there is limited access to written text in braille for them to read. By developing a real time printed text to braille translator, the access to education and learning for this people would be increased.

- **Goal 10. Reduced inequalities:** it is common for blind and visually impaired people to face social and economic inequalities such as limited access to education, employment, and other similar opportunities. By developing a real time printed text to braille translator, we could help reduce these inequalities by increasing access to education and therefore increasing employment opportunities and by allowing visually impaired people to be more independent.

Chapter 3. Design

3.1. Hardware design

It was decided to divide this project into four smaller subsystems. Each subsystem will carry out a concrete function of the project. If all the subsystems can carry out their functions independently, then the project should be able to work as a whole. The following image shows the four subsystems in which the project is divided:

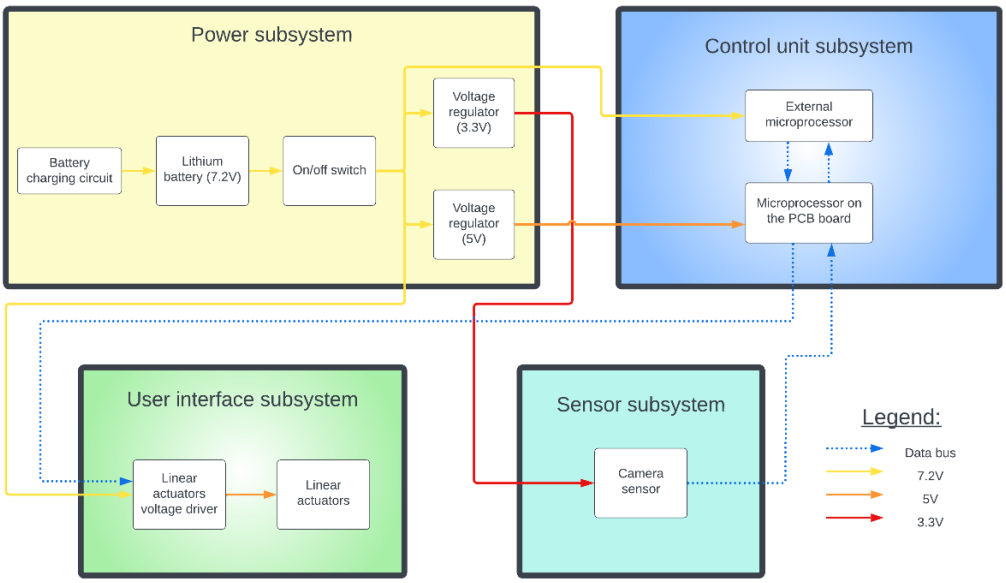

Figure 3. Block diagram of the project.

Here is a brief explanation of each subsystem and the functions they will carry out:

1) **Power management subsystem:** the power subsystem will power the whole system. It will consist of a rechargeable Lithium Polymer (LiPo) battery supplying around 7.2V, an on/off switch, two different voltage regulators that step down the battery voltage to 5V and 3.3V respectively and a USB-C used to charge the battery through a battery charging circuit.
2) **Sensor subsystem:** this subsystem will contain the camera sensor, which will be used to take pictures of the printed text.
3) **Control unit subsystem:** a custom PCB board will contain a microcontroller which will interact with the camera sensor and with the linear actuators. A second and more powerful microprocessor will perform OCR (Optical Character Recognition) to identify all the characters in a word.
4) **User interface subsystem:** six linear actuators will move up and down in a synchronous way to recreate braille characters. The user will be able to read those braille characters using their finger.

3.1.1. Power Management Subsystem

The three main parts of the power management subsystems are the 5V regulator, the 3.3V regulator and the battery charging circuit.

Some of the chosen components used to carry out the project work at 5V while others work at 3.3V, therefore we need to step down the 7.2V nominal voltage from the battery to be able to connect all the components. We also need to design a circuit to be able to charge the battery; a USB -C port was used to do this.

These are all the designed circuits for this subsystem:

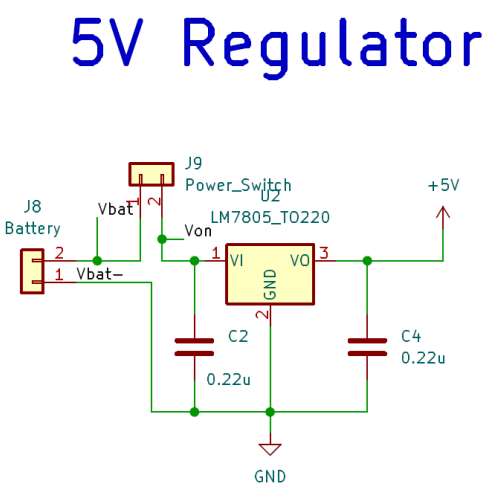

Figure 4. 5V regulator design (including the battery and switch connectors).

The 5V regulator chosen is the LM7805. The allowed range of input voltages to this regulator is from 7V to 18V and the output voltage is always around 5V. The connection circuit of the voltage regulator and the values of the capacitors are the ones suggested on the device datasheet:

- J8 is a two-pin connector for the battery.
- The signal *Vbat* is the battery voltage with respect to ground (around 7.2V).
- J9 is another two-pin connector in which the on/off switch for the whole device will be connected. The on/off switch can block a voltage up to 125V and a current up to 10A, which should be enough for our device.
- The signal *Von* is the voltage after the on/off switch (either 7.2V or 0V).
- The capacitors C2 and C4 ensure that the input and output signals do not have any strange peaks and are as close to a DC signal as possible.
  $C2 = C4 = 0.22\ \mu F$.

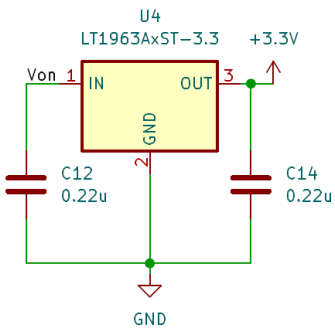

Figure 5. 3.3V regulator design.

The circuit of the 3.3V regulator seen on figure 5 is very similar to the circuit of the 5V regulator. It receives the signal *Von* into the LT1963A voltage regulator. The allowed range of input voltages to this regulator is from 5V to 9V and the output voltage is always around 3.3V. The connection circuit of the voltage regulator and the values of the capacitors are the ones suggested on the device datasheet.

The value of the capacitors is $C12 = C14 = 0.22\ \mu F$.

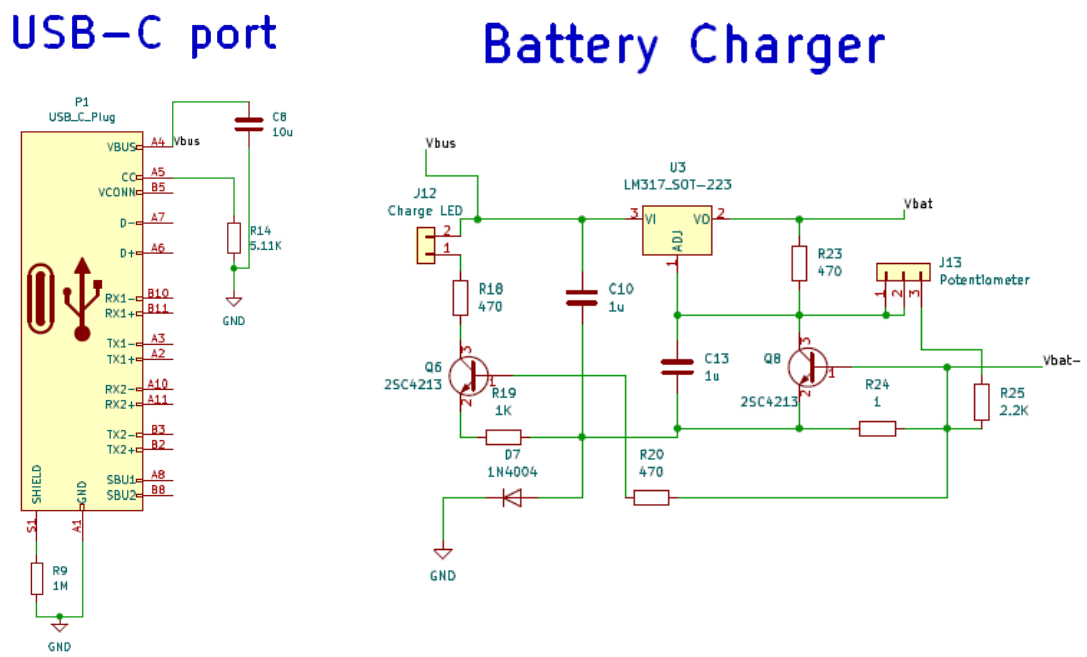

Figure 6. USB – C port and battery charger design.

Figure 6 shows the designed connections for the USB-C port and the battery charger.

The only pins that have been connected in the USB-C port are those that are used to power it. This is because it will not be used as a data transmitter or receiver. The USB-C will only be used to charge the battery. A capacitor was also added to ensure that the input signal does not have any strange peaks and is as close to a DC signal as possible.

The battery charger circuit has a LM317 voltage regulator, which can output a voltage from 1.25V to 37V by regulating the potentiometer connected to it. In our case we want it to output the battery's voltage, which is around 7.2V. It was decided to use a potentiometer instead of a regular voltage divider in the battery charger circuit because the battery voltage can vary while it's been charged and discharged, so we need to be able to adjust the output voltage of the battery charger to make sure we charge the battery in a safe way. A LED was also added to the battery charger circuit. It would turn on when the USB-C is connected and therefore it would be used as an indicator of the battery being charged.

On the other hand, the LM317 voltage regulator has a maximum output current of 1.5A, which is the maximum charging current that the battery can support according to its datasheet. This means that if we regulate the output voltage appropriately the battery will never overcharge.

It is important to mention that the battery charging circuit requires an input voltage *Vbus* higher than the output voltage to work, therefore the USB-C must be connected to a 12V power supply and not a common 5V one.

After designing the battery charging circuit, a simulation was carried out using the software *LTspice* to verify that it worked as expected. An image of the complete simulation can be seen on figure 7.

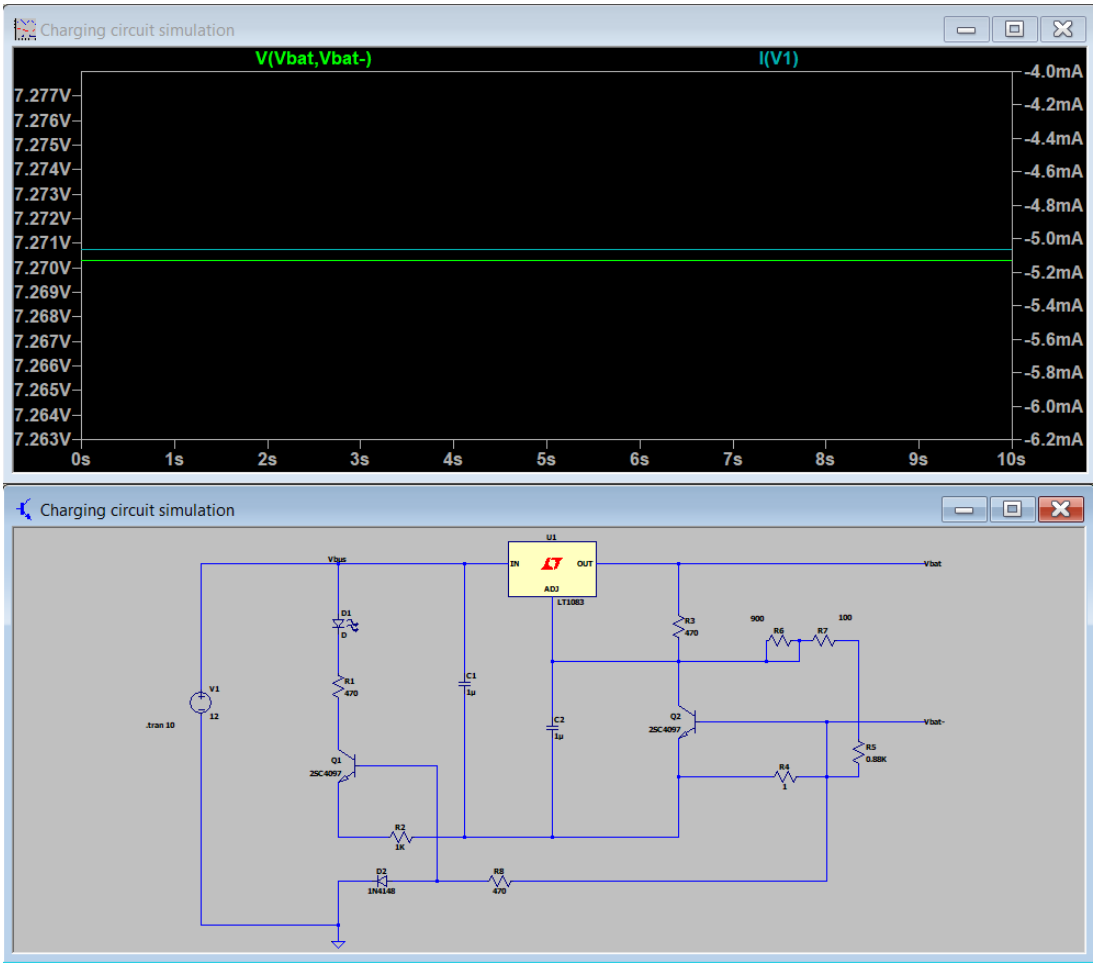

Figure 7. *LTspice* simulation of the battery charger circuit.

In this simulation a 12V input battery was used to simulate a USB-C connected to a 12V power supply. This 12V input voltage is then connected to the designed battery charger circuit.

In the graphs we can see that the voltage across the battery terminals (*Vbat* and *Vbat-*) is around 7.2V. This output voltage could be modified by adjusting the potentiometer depending on what the real battery voltage is as explained before.

3.1.2. Sensor Subsystem

The only component of this subsystem is the camera sensor, which will be used to take pictures of the printed text. The chosen camera sensor was the OV7670 camera. The reason this camera was chosen was because of its affordability and because it was compatible with the chosen microcontroller. In figure 8, we can see the connections designed for the OV7670 camera sensor.

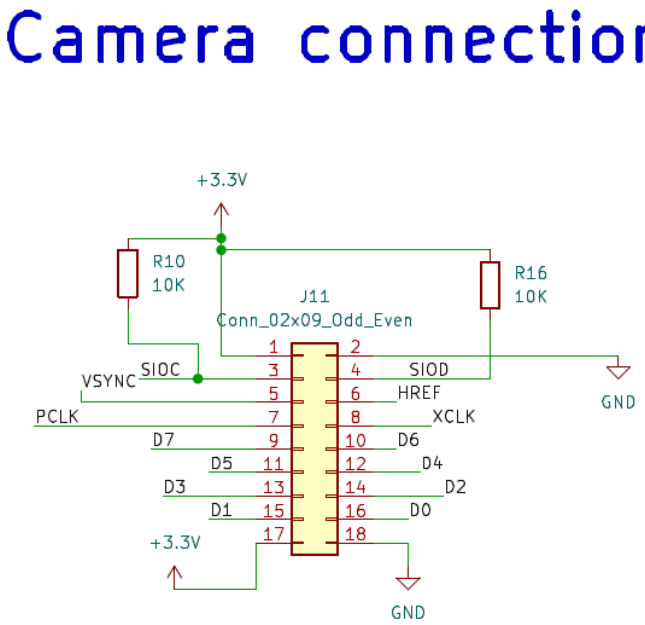

Figure 8. Camera connections.

To connect the camera to the microcontroller, an 18-pin connector footprint was used. The camera will be connected to the 3.3V voltage supply that comes out of the 3.3V voltage regulator. The inputs and outputs are digital pins coming from the microcontroller. A 16MHz clock will also be connected as an input to the camera, as specified on the device datasheet. The value of the resistors chosen are also the ones recommended on the datasheet.

The code used to receive images from the camera, modified for the Arduino Uno, can be seen in Appendix B.

3.1.3. Control Unit Subsystem

The control unit subsystem will be in charge of performing the Optical Character Recognition (OCR) task. It will consist of two different microprocessors that will interface between them. The ATmega32U4 microcontroller will receive the images taken by the OV7670 camera and send them to the Libre AML-S905X-CC microprocessor, a second and more powerful microprocessor that will carry out the OCR task.

In figure 9, we can see the connections designed for the ATmega32U4 microcontroller.

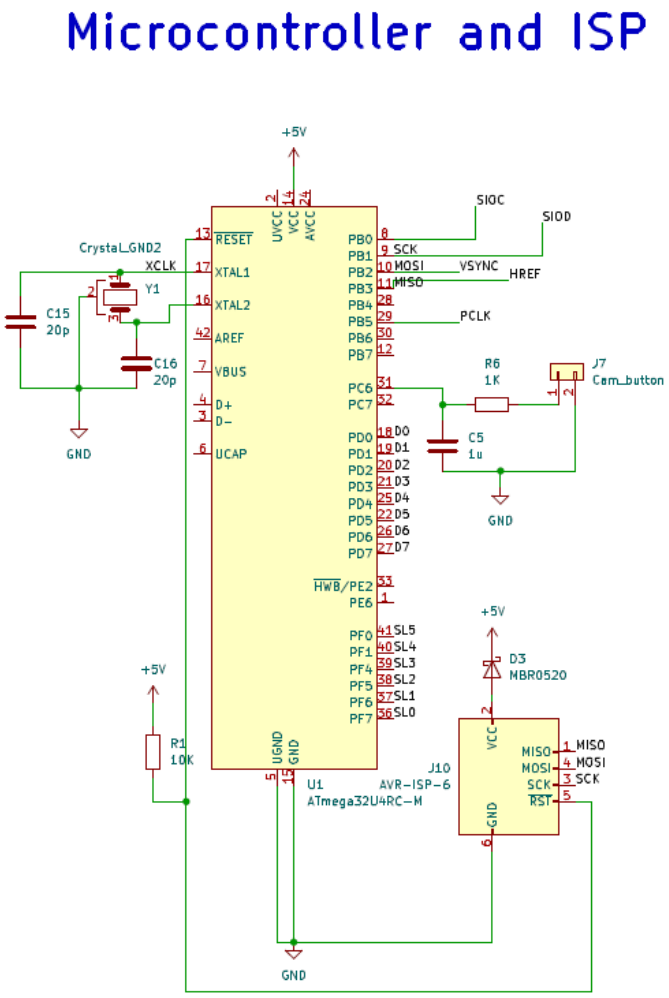

Figure 9. ATmega32U4 and ISP connections.

A 16MHz clock was used as an input to the microcontroller as specified on the device datasheet. An in-system programming (ISP) header was also added to be able to program the microcontroller. The microcontroller and the ISP header are connected to the 5V voltage supply that comes from the 5V voltage regulator.

The Libre AML-S905X-CC microprocessor is an already developed board, on figure 10 we can see an image of it.

Figure 10. Libre AML-S905X-CC microprocessor.

Tesseract was the OCR engine chosen because it was well-documented as an open-source machine learning (ML) based OCR engine, accurate, and easy to interface with.

A simple python script was written to wrap around the OCR engine and deal with the preprocessing of the images. Given an image file, the script runs preprocessing algorithms on the image, then runs the processed image through the OCR engine, outputting the ASCII characters as a string. The python script is running on Ubuntu 22.04 on the Libre AML-S905X-CC microprocessor.

The full python script can be found on appendix C.

3.1.4. User Interface Subsystem

It was decided to use six linear actuators to output the braille characters. In order to do this, a driver circuit had to be designed. This is because we had to turn on and off the linear actuators with the microcontroller, but each linear actuator requires around 1A of current which our ATmega32U4 microcontroller could not supply on its output pins.

It was decided to use an NMOS transistor per linear actuator, where the output of the microcontroller would be connected to the gate voltage through an input resistance, the linear actuator would be connected between the drain of the transistor and the positive terminal of the battery, and the source would be connected to ground. Figure 11 shows the connection used for one linear actuator.

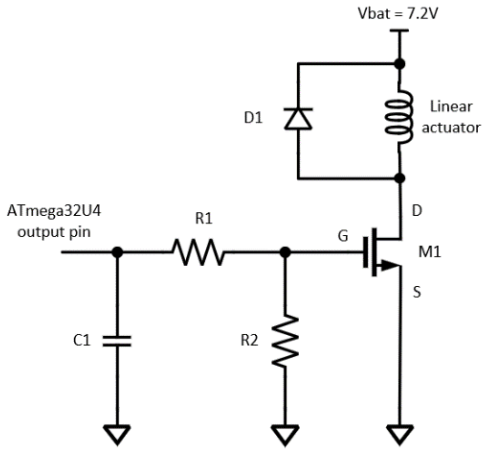

Figure 11. Linear actuator voltage driver circuit.

The components used in the voltage driver circuit are:

- D1: the diode will protect the NMOS device from getting an inductive voltage when the solenoid is turned off.
- M1: the NMOS device will be used as a switch to control the voltage and current driven into the solenoid.
- R1: will limit the current coming from the input of the microcontroller to protect it from high currents.
- R2 will keep the gate G of the NMOS device closed when the input in the microcontroller has been set to low. The overdrive voltage of the NMOS device chosen is $V_{GS(TH)}$ = 1.1V.
- C1: will make sure that the signal does not have any strange peaks and is as close to a DC signal as possible.

The maximum output current for the pins in the microcontroller chosen (ATmega32U4) is 40mA. Therefore, it was decided to limit the current to 30mA to make sure we are not exceeding that limit. The output voltage of the pins is 5V. Using Ohm's law we find R1 to be:

$$R1 = \frac{V}{I} = \frac{5V}{30mA} = 167\ \Omega \qquad \textit{(Equation 1)}$$

For the value of R2 a high resistor value was chosen to make sure that the gate of the NMOS device is closed when the input pin is set to low.

$$R2 = 100\ k\Omega$$

Finally, it was decided to use a capacitor C1 of value:

$$C1 = 1\ nF$$

The final design for the complete solenoid driver circuit can be seen in Figure 12.

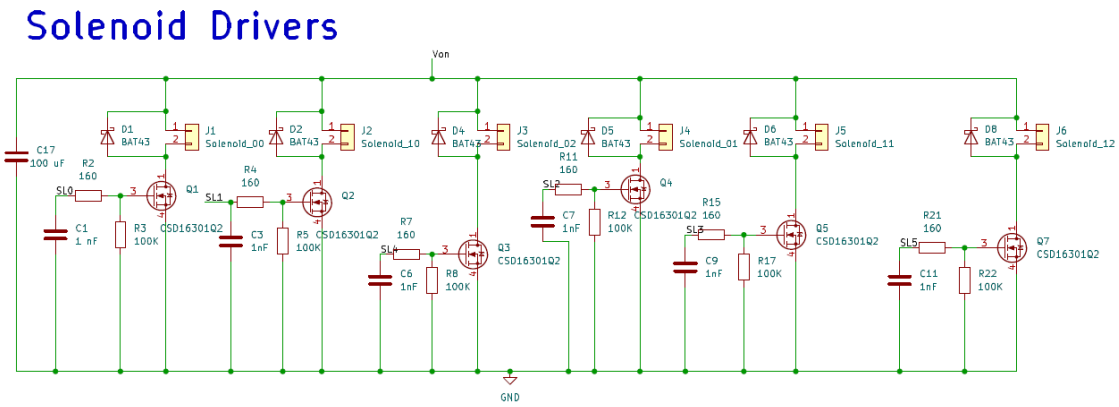

Figure 12. Solenoid driver circuit.

In this schematic, we can see that there are six power system circuits (one for each solenoid) connected between ground and Von. Von is the battery voltage when the on/off switch is turned on, the battery voltage is equal to 7.2V.

It is worth mentioning that there is a capacitor connected in parallel with all the voltage driver circuits to avoid any strange peaks on the power traces and make sure that the voltages as close to a DC signal as possible. This capacitor is C17 and its value is 100µF as seen on figure 12.

After designing the linear actuator driver circuit, a simulation was carried out using the software *LTspice* to verify that it worked as expected. An image of the complete simulation can be seen on figure 13.

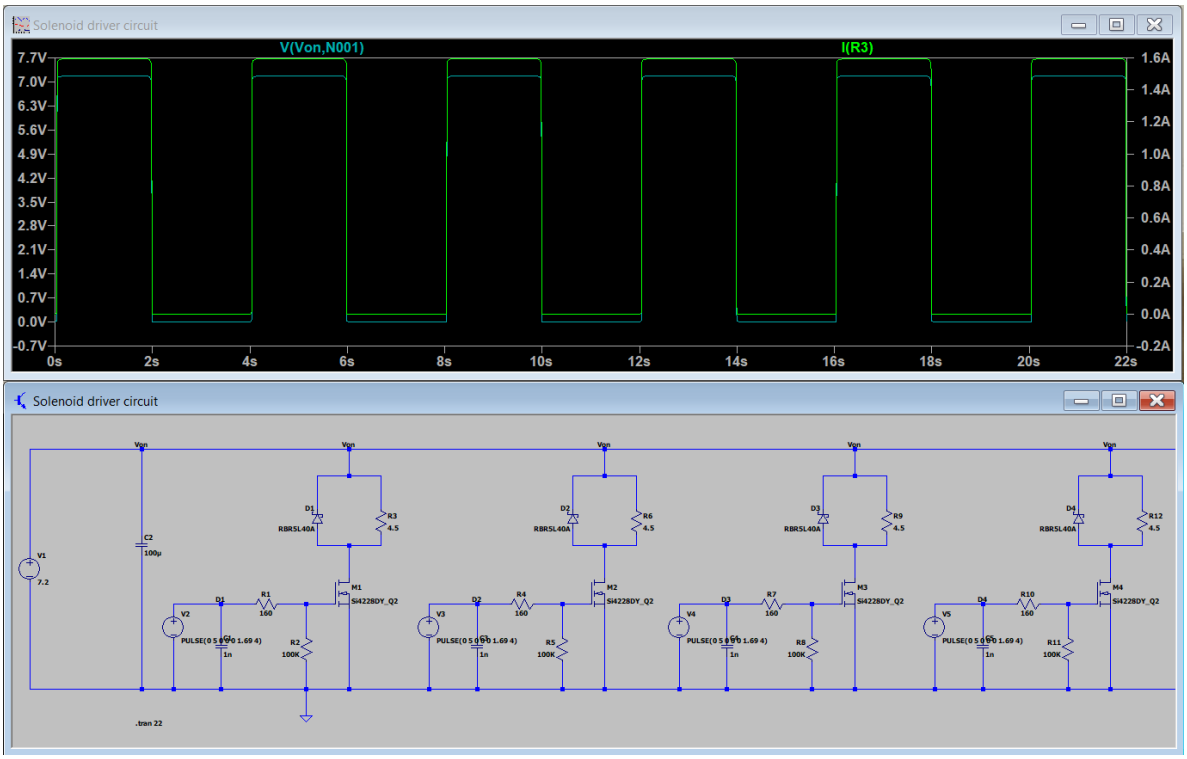

Figure 13. Solenoid driver circuit simulation.

In this simulation an ideal 7.2V input battery was used to simulate the real 7.2V battery and it was connected to the designed voltage driver circuit. The linear actuators were simulated by using their DC resistance value of 4.5Ω according to the device datasheet.

The signals coming from the output pins of the ATmega32U4 microcontroller were simulated using 5V pulses that are turned on for 2 seconds and then turned off for another 2 seconds.

In the graphs we can see that the voltage across the linear actuators when they are turned on is around 7.2V, which is a reasonable voltage according to the device datasheet, while the current is around 1.6A, which is the expected current.

When the signal coming from the microcontroller is set to off, the voltage and the current on the linear actuators is set to 0V and 0A respectively.

3.1.5. PCB layout

All the designed hardware circuits (except the external microprocessor used to carry out the OCR task) will be manufactured in a custom printed circuit bord (PCB) using the *KiCad* software. The manufacturer of the custom PCB board will be *PCBWay*.

Figure 14 shows the PCB layout designed, which includes all the circuits explained in section 3.1.

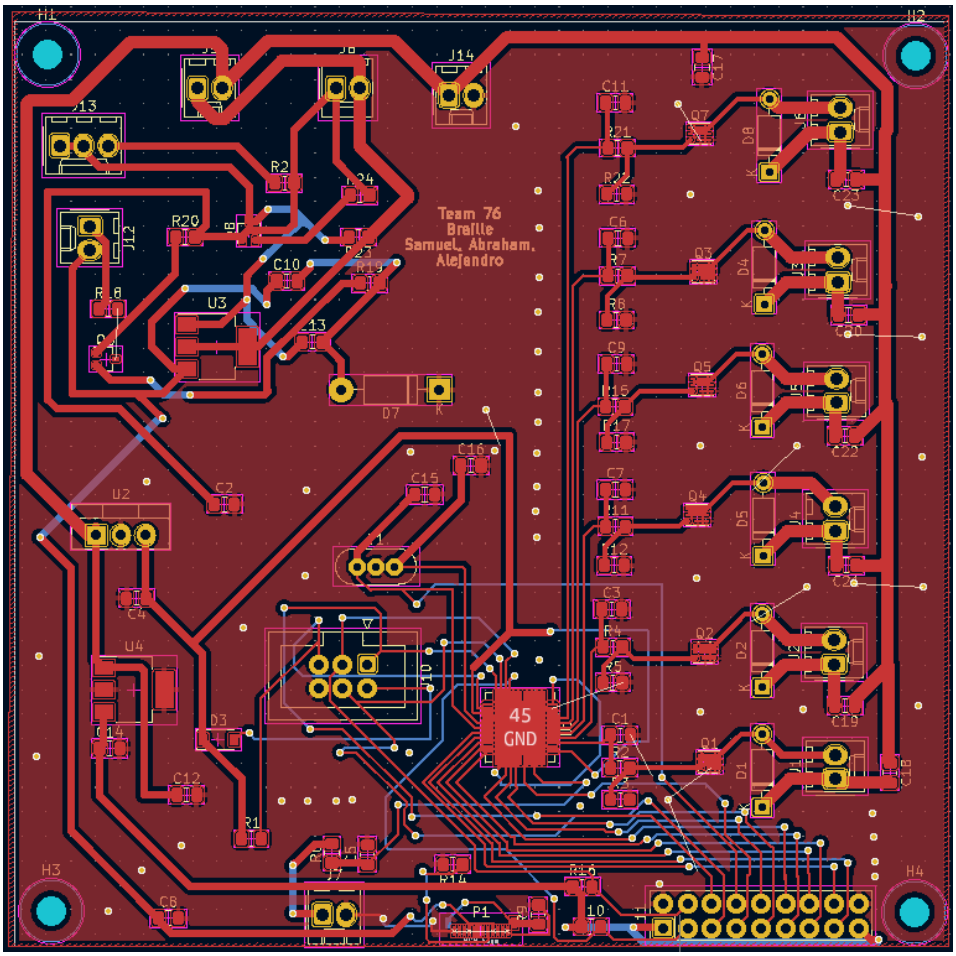

Figure 14. PCB layout.

It was decided to make the power traces in the layout much wider than the data lines because a much bigger current will flow through those traces when compared to the data lines.

A more complete schematic showing the PCB layout and its dimensions can be found on appendix D.

3.2. Physical design

A physical design that would contain the PCB and all the electronic components used in the project was developed using the *Autodesk Fusion 360* software. The final design can be seen on figure 15.

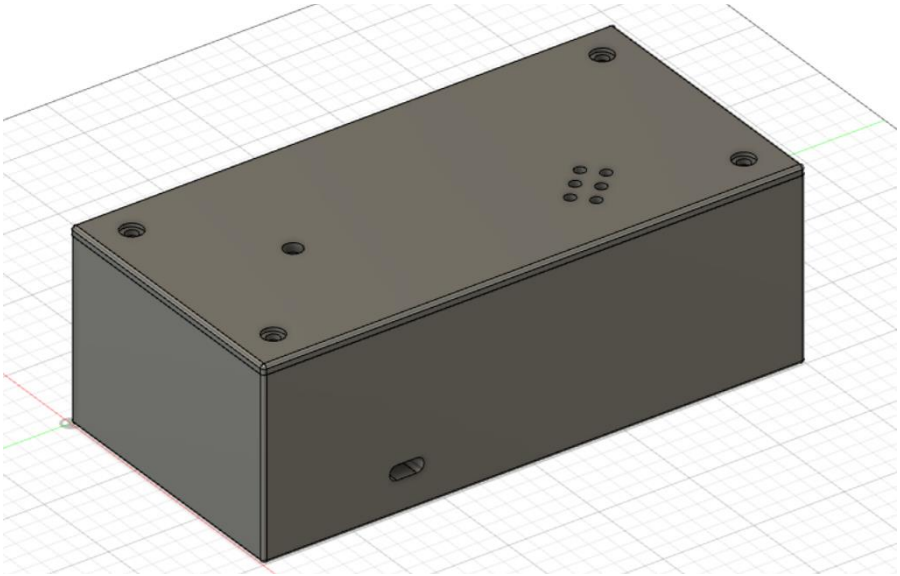

Figure 15. Complete physical design.

The final physical design of the project looks like a rectangular box. This is because a simple design easy to hold with one hand was wanted. The box is divided into two parts which are hold together using 4 screws: a cover and a container. Two views of the container can be seen on figure 16.

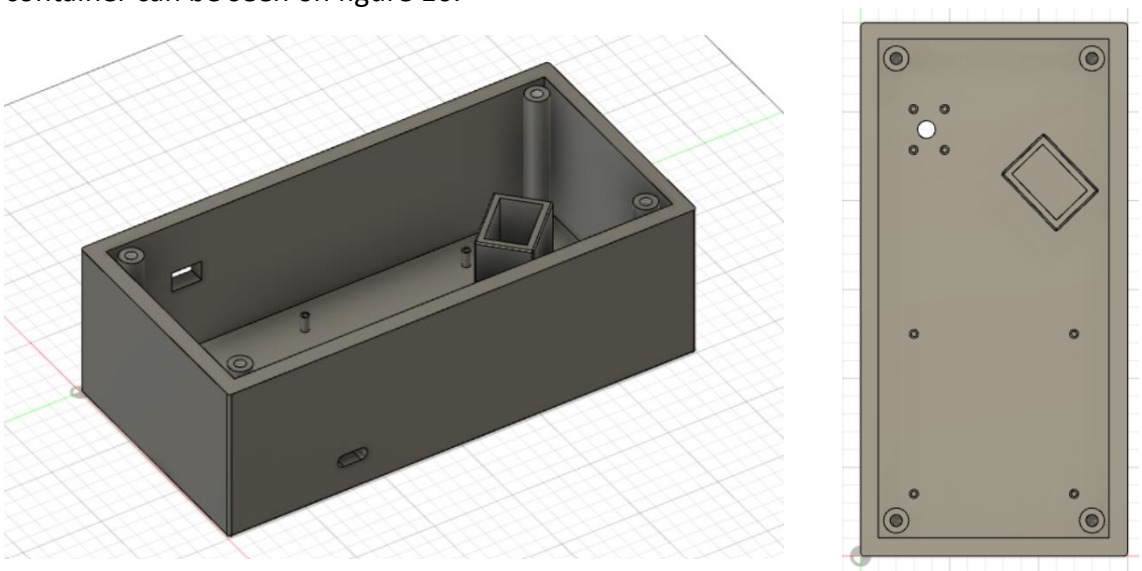

Figure 16. Side view (left) and top view (right) of the box container.

The container has 4 cylinders with a hole for the screws to be placed in, one on each corner. It also has a rectangular hole on one side for the on/off switch to be placed in and another hole on the other side to be able to connect the USB-C to an external 12V voltage supply.

The designed PCB will be placed inside the box using another 4 screws that will be sliced through the mounting holes that the PCB has into the 4 small cylinders that can be seen inside the box container, at the bottom part of the top view in figure 16.

The linear actuators will be hold in place inside the small box that can be seen at the top right part of the top view in figure 16.

The camera sensor will be placed at the top left part of the top view in figure 16 using 4 screws. The lens of the camera will be placed through the circular hole.

The second part of the box is the cover. Two views of it can be seen on figure 17.

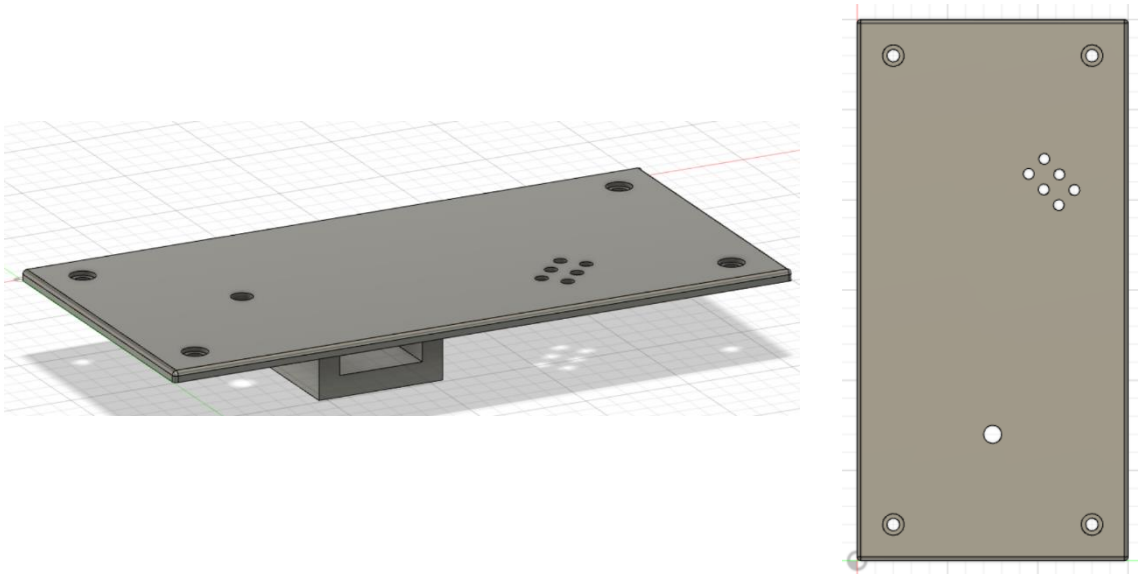

Figure 17. Side view (left) and top view (right) of the box cover.

The cover has a small U-shape box bellow it as seen on the side view. The battery will be placed inside that box.

The hole seen at the bottom part of the top view is where the LED of the charging circuit will be placed.

A complete drawing of the cover and the container with all their dimensions can be found on appendix E.

The machine shop at the University of Illinois at Urbana-Champaign was used to build the physical design.

The completely built physical design can be seen on figure 18.

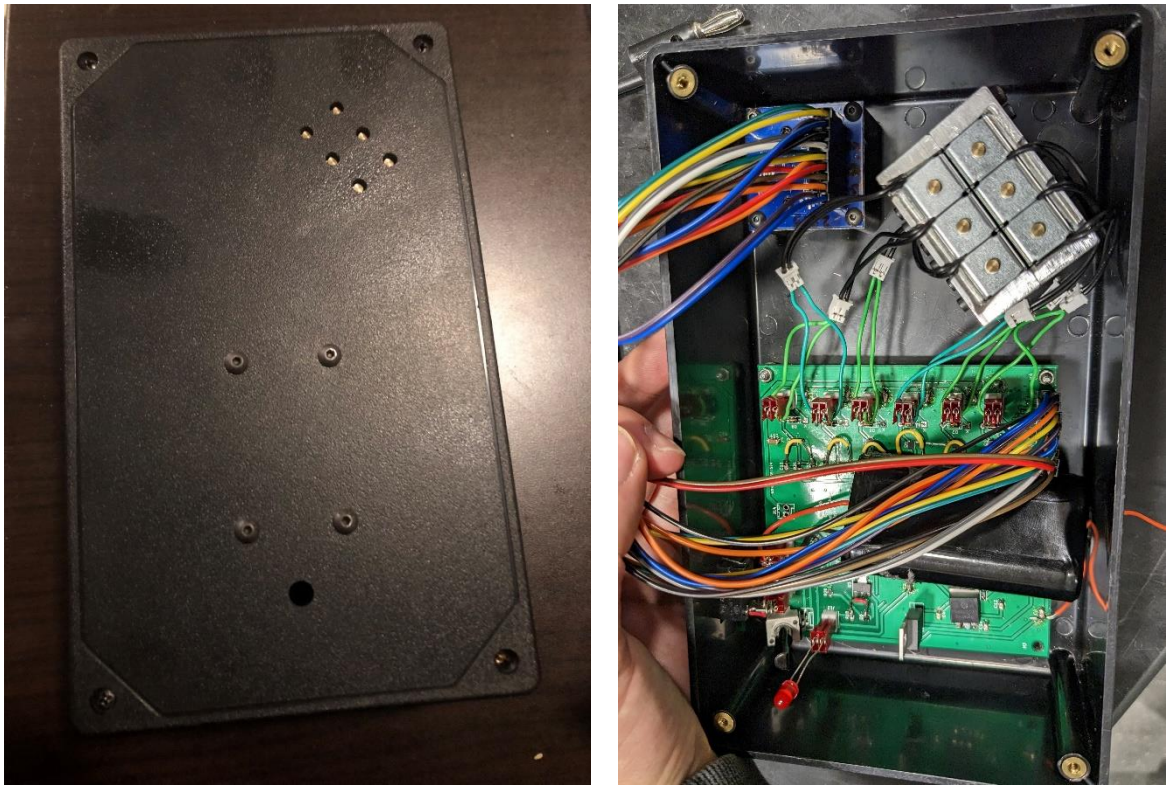

Figure 18. Completely built physical design with the cover (left) and without it (right).

3.3. Power consumption design

As mentioned in the introduction, one of the high-level requirements of this project is being able to develop a final device with a battery life that can last all day on a single charge. This is because the developed device should be portable.

An explanation of the calculations done to obtain an approximation of the expected battery discharging time and the average power consumption that the final device will have will be explained in this section.

A distinction was done between two operation modes: working mode and standby mode.

- Working mode: this mode refers to those times in which the on/off switch will be on and therefore the device will be fully working.
- Standby mode: this mode refers to those times in which the on/off switch will be turned off.

To determine the battery's discharging time the following formula will be used:

$$Discharging\ time\ (h) = \\ = Battery\ capacity\ (Ah) \cdot \frac{Battery\ voltage\ (V)}{Device\ average\ power\ consumption\ (W)} \cdot 0.9 \qquad \textit{(Equation 2)}$$

As seen on equation 2, a 0.9 multiplier is added to take into account the average power efficiency of Lithium Polymer (LiPo) batteries.

The device average power consumption during working mode will be the sum of the average power consumption of the following electronic components:

- ATmega32U4 microprocessor ≈ 50 mW
- Camera sensor ≈ 50 mW
- Linear actuators ≈ 2.5 · 5 · 0.5 W = 6.25W
- Libre AML – S905X – CC microprocessor ≈ 1W

The average power consumption of the linear actuators was obtained by taking into account that the most commonly used letters in the braille alphabets require 2 to 3 pins to be lifted at a time (2.5 on the equation), each linear actuator uses 5W of power when turned on (5 on the equation) and they will be lifted during half of the time and be off during the other half (0.5 on the equation).

It's important to mention that the power consumption of the voltage regulators used is not taken into account in this calculation because their power consumption is relatively low when compared to the rest of the components and they can be neglected.

The chosen battery for the project is a rechargeable Lithium Polymer (LiPo) battery with a typical capacity of 3350mAh and a nominal voltage of 7.2V according to its datasheet (the concrete model chosen can be found on appendix A).

Therefore, the estimated discharging time during the working mode is:

$$\boldsymbol{Discharging\ time} = 3.350\ (Ah) \cdot \frac{7.2V}{50 \cdot 10^{-3} + 50 \cdot 10^{-3} + 2.5 \cdot 5 \cdot 0.5 + 1} \cdot 0.9 \approx \boldsymbol{3h}$$

On the other hand, the estimated power consumption during the standby mode will be very close to 0W, this is because the power switch will be turn off and the discharging time will be close to infinity (very long).

Hence, according to these calculations we should be able to make a device with a battery life that lasts all-day on a single charge as long as the usage is not very intense. However, this is just an average calculation, we must take into account that the power consumption can vary depending on various factors such as the workload and the system configuration.

Chapter 4. Costs

In order to carry out this project, some electronic components were purchased through major electronic components distributors, such as *Digi-Key*, *Mouser Electronics*, *Texas Instruments,* and others. The machine shop at the University of Illinois at Urbana-Champaign was also used to design and build the containment unit of the project.

4.1. Electronic components

The total cost of all the electronic components used is **$167.97.**

A full list containing all the electronic components used in this project and their prices can be found on appendix A.

4.2. Labor

Based on the average salary of a graduate student from the Electrical and Computer Engineering department (ECE) of the University of Illinois at Urbana-Champaign, we would expect a salary of around 30 $/hour to carry out this project.

To complete the project each team member will work an estimate of 10 hours a week on their own to complete their work. Considering that the team has 3 members, and each team member has been working on this project a total of 10 weeks, we would expect a total salary of around:

$$30\,\frac{\$}{hour}\cdot 10\,\frac{hours}{week}\cdot 10\,weeks\cdot 2.5 = \$\mathbf{7,500}\ \boldsymbol{per\ team\ member} \quad \textit{(Equation 3)}$$

As seen on equation 3, a 2.5 multiplier is added to the expected salary to take into account the expenses related with using the equipment in the laboratories and for all the cost associated with scaling up the project to a company for its distribution.

If we multiply this salary times the number of people in the team, we get a total amount of:

$$\$7{,}500\ per\ member\cdot 3\ members = \$\mathbf{22,500}\ \boldsymbol{in\ labor\ cost} \quad \textit{(Equation 4)}$$

The total hours spent by each team member on each work type can be found on appendix F.

4.3. Machine shop cost

The machine shop at the University of Illinois at Urbana-Champaign was used to design and build the containment unit of the project. According to their official webpage, their cost is $38.17 per hour. After talking with the machine shop and giving them the electronic components, it took them about 6 hours to complete the physical design of the project.

Therefore, the total costs associated with the machine shop should be:

$$38.17 \frac{\$}{hour} \cdot 6\ hours = \$\mathbf{229.02} \qquad \textit{(Equation 5)}$$

4.4. Total project cost

Considering all costs explained on chapter 4, the total cost of the project is equal to:

$$\boldsymbol{Total\ cost} = electronic\ components + labor + machine\ shop =$$

$$= \$167.97 + \$22{,}500 + \$229.02 = \$\mathbf{22,896.99} \qquad \textit{(Equation 6)}$$

Chapter 5. Implementation into a final device

Once the custom PCB containing all the hardware for the project was designed and manufactured, the electronic components had been ordered and received and the physical design had been built it was time to put everything together into a final device. In this chapter, the process of creating a functional device from the design will be explained.

5.1. Soldering

The first thing that was done after receiving all the electronic components was to solder them into the custom PCB. Two different soldering techniques were used to carry out this process:

1. **Soldering pen with soldering iron:** this is the most common used soldering technique. A soldering pen with a hot end melts the soldering iron to be able to solder the electronic component onto the PCB. A picture of the soldering pen used to carry out this project can be seen on figure 19.

Figure 19. Soldering pen used in this project.

This technique was used in the project to solder big electronic components, such as resistors, capacitors, diodes, voltage regulators, …

2. **Hot air gun with soldering paste:** soldering paste is applied to the pads of the PCB, then the electronic component is placed on top of the paste and finally hot air is applied to melt the paste using a hot air gun. A picture of the hot air gun and the soldering paste used in this project can be seen on figure 20.

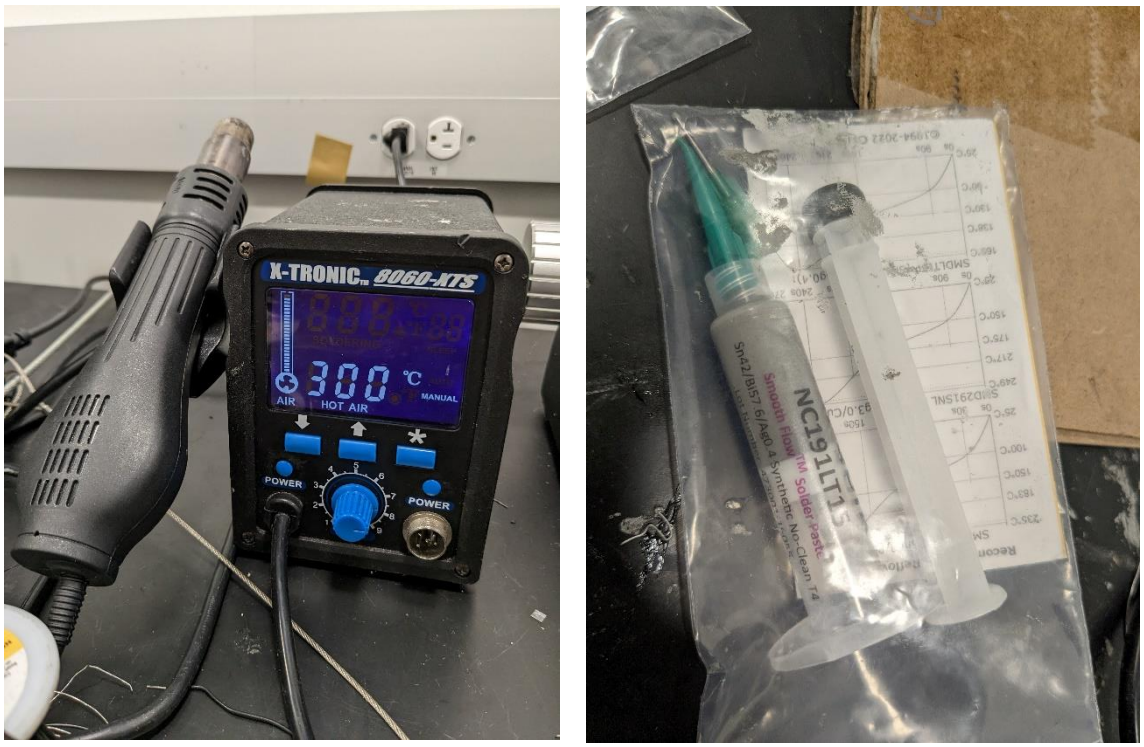

Figure 20. Soldering gun (left) and soldering paste (right) used.

The hot air gun was always kept at a temperature of 300ºC to avoid damaging the electronic components with higher soldering temperatures.

This technique was used in the project to solder small electronic components, such as NMOS, BJT's, the ATmega32U4 microcontroller, …

5.2. Assembling all components into the physical enclosure

After soldering all the electronic components onto the PCB it was time to place everything inside the designed box. Figure 21 shows an image of the final physical design with all the components already assembled.

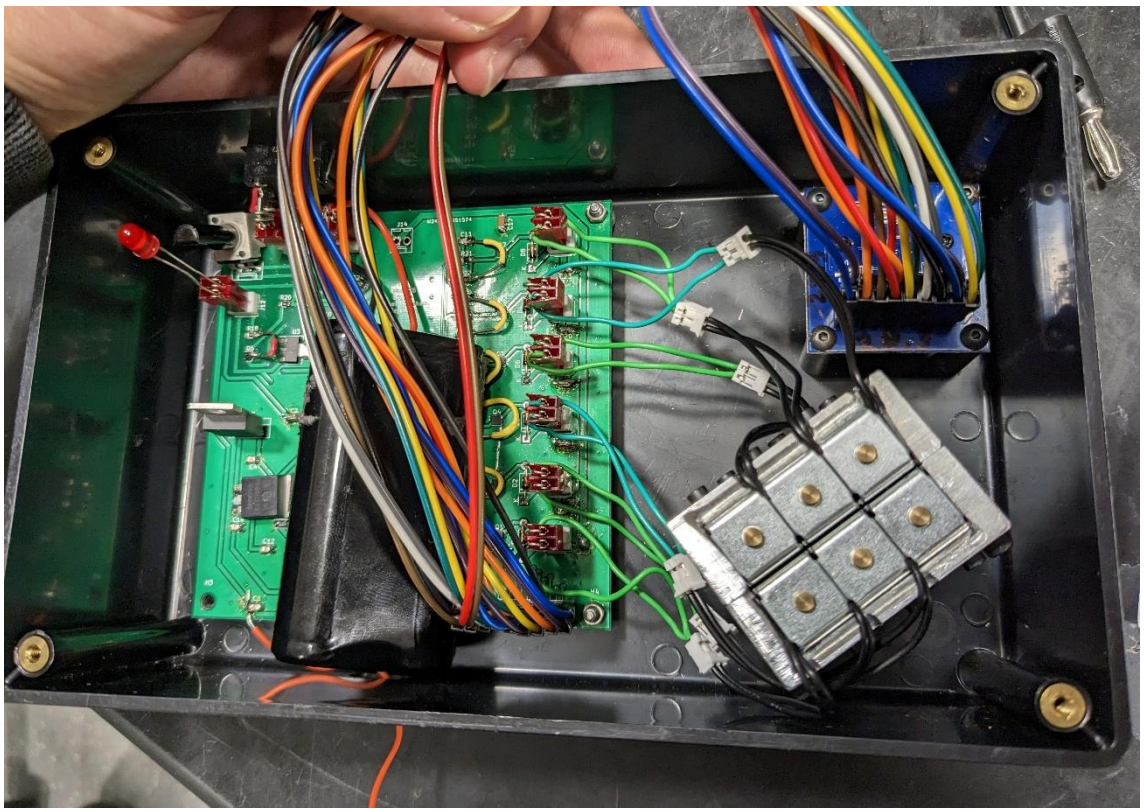

Figure 21. Physical box assembled.

As seen on figure 21, jumper wires were used to connect the camera sensor to its designed connections on the PCB. Jumper wires were also used to connect the linear actuators to their designed connections on the PCB.

The PCB and the camera sensor were assembled onto the box by using screws, one for each mounting hole.

5.3. Programing the microcontroller

The *Arduino IDE* software application was used to program the ATmega32U4 microcontroller on the custom PCB. It was decided to use this software because it provides an intuitive user interface and because the ATmega32U4 is the same microcontroller used in the Arduino Leonardo board; therefore, uploading the code to the microcontroller on the custom PCB was a straightforward process.

The in-system programming (ISP) header on the PCB was connected to the computer with the *Arduino IDE* software using a USB to 6 pin connector wire and the designed code (which can be seen on appendix B) was uploaded to the ATmega32U4 microcontroller. After doing this something unexpected happened. An error kept appearing on the *Arduino IDE* software every time the code was uploaded. A connection could not be established between the computer and the ATmega32U4 microcontroller on the PCB.

At first, it was decided to desolder and solder again the ATmega32U4 microcontroller, as a soldering problem could be the issue because the pins are small and hard to solder, but after doing this the problem was not solved.

Then, the hardware circuits designed on the PCB were checked for possible mistakes that could have been made and the problem was found. The ground pin on the ATmega32U4 microcontroller was not connected to the ground plane on the PCB. This can be seen on figure 22.

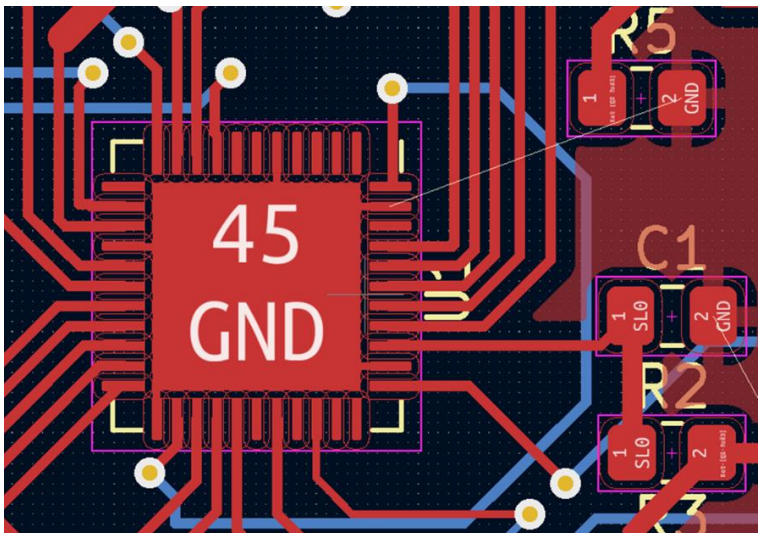

Figure 22. Issue with the ground connection of the ATmega32U4 microcontroller.

In figure 22 we can see a straight white line which connects the ground pin of the ATmega32U4 microcontroller to the ground plane of the PCB. This white line is a suggested connection that the *KiCad* software encourages the user to make, but is not a real connection that has been made.

After figuring out what the problem was, we tried to solder a jumper wire that connected both ground nodes. However, we were not able to do this because the pins on the ATmega32U4 were too small.

At this point in time it was not possible to order a new custom PCB with both ground nodes connected. Therefore, it was decided to use an external Arduino Uno board as a microcontroller for the project instead of using the ATmega32U4. If this was done, the *Arduino IDE* software could still be used to program the microcontroller and the only connections that had to be modified on the custom PCB where those that connected the microcontroller with the linear actuators and with the camera sensor. In order to make these connections, jumper wires were soldered at the input of each linear actuator voltage driver circuit and connected to the Arduino Uno microprocessor. The same thing was done with the camera connections.

After making these modifications to the device the code was successfully uploaded to the Arduino Uno microprocessor and the system worked as expected. A picture of the final design containing the Arduino Uno can be seen on figure 23.

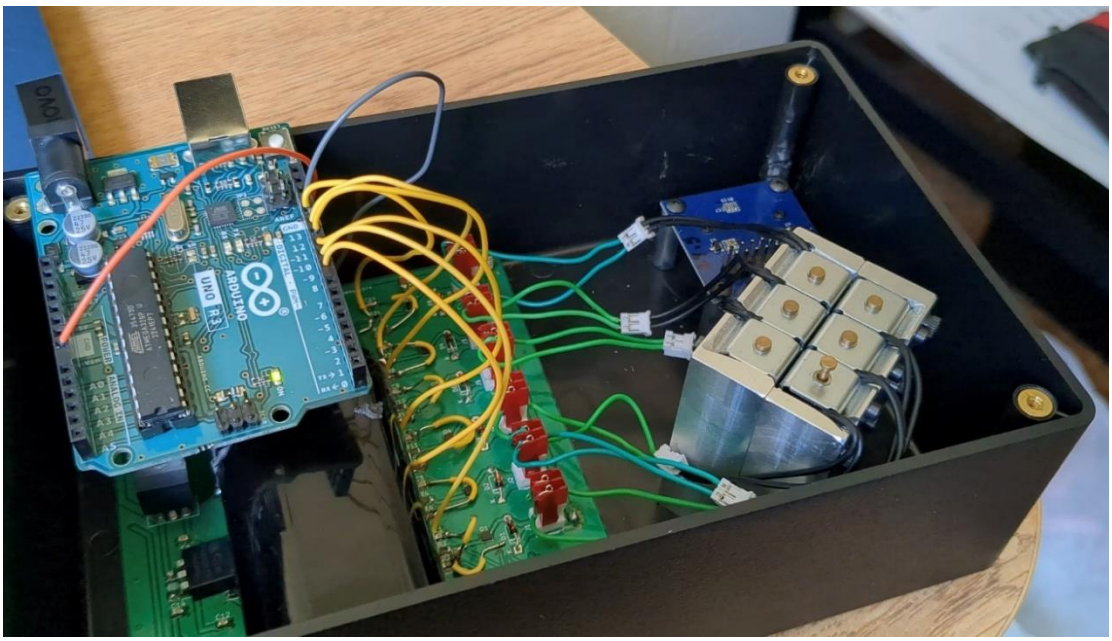

Figure 23. Final design containing the Arduino Uno as a microcontroller.

Chapter 6. Results

Throughout this chapter the obtained results will be explained, the accomplished objectives of the project will be proved with images and the unaccomplished ones will be explained in detail with the main reasons being described.

First, the obtained results on the functionality of each individual subsystem will be shown and explained. Then, an explanation of the level of achievement of the project as a whole will be given.

6.1. Power management subsystem results

As mentioned on the hardware design section, the three core aspects of the power management subsystems are the 5V regulator, the 3.3V regulator and the battery charging circuit. Each of them has their own design verification.

A voltmeter was used to test both voltage regulators of the power management subsystem. After connecting the battery used in the project and turning on the power switch, the voltages between the ground plane on the PCB and the output node on each voltage regulator were measured. In figure 24 we can see the output voltage of the 5V regulator and the output voltage of the 3.3V regulator.

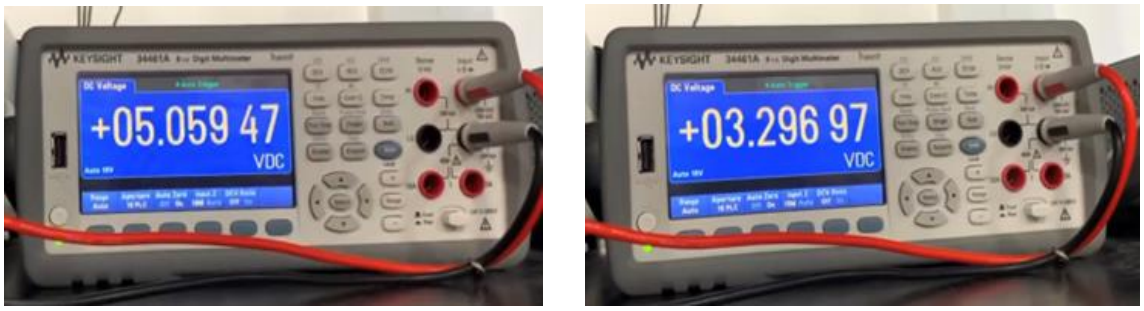

Figure 24. Output voltage of the 5V regulator (left) and the 3.3V regulator (right).

As seen on figure 24, both voltage regulators worked properly. This means that the functionality of this core aspect of the power management subsystem was achieved.

In order to verify the functionality of the battery charging circuit, a 12V supply voltage was connected to the USB-C on the PCB. After adjusting the output voltage of the battery charger circuit using the potentiometer to the current battery voltage, the battery was connected to the PCB with the power switch turned off.

The battery voltage was recorded using a voltmeter before charging and after leaving the battery charger connected for one minute. We can see both voltage values bellow, in figure 25.

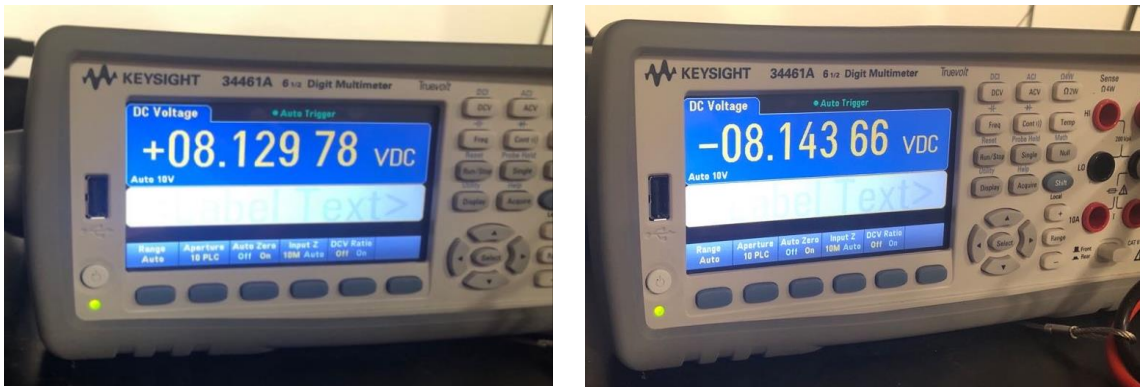

Figure 25. Battery voltages before (left) and after (right) charging the battery.

As seen in figure 25, the overall voltage of the battery increased by 14 mV, which means that the battery had been charged successfully.

Therefore, as explained in this section, all the power management subsystem main parts worked as expected.

6.2. Sensor subsystem results

The designed code for the camera sensor was uploaded to the Arduino Uno microprocessor. The camera sensor was requested to automatically take a picture every 10 seconds. In order to verify that the sensor subsystem was working properly, the *Arduino IDE* software was used on the computer to display the images that the camera sensor was taking. A test picture taken by the camera sensor of an object in the laboratory can be seen on figure 26.

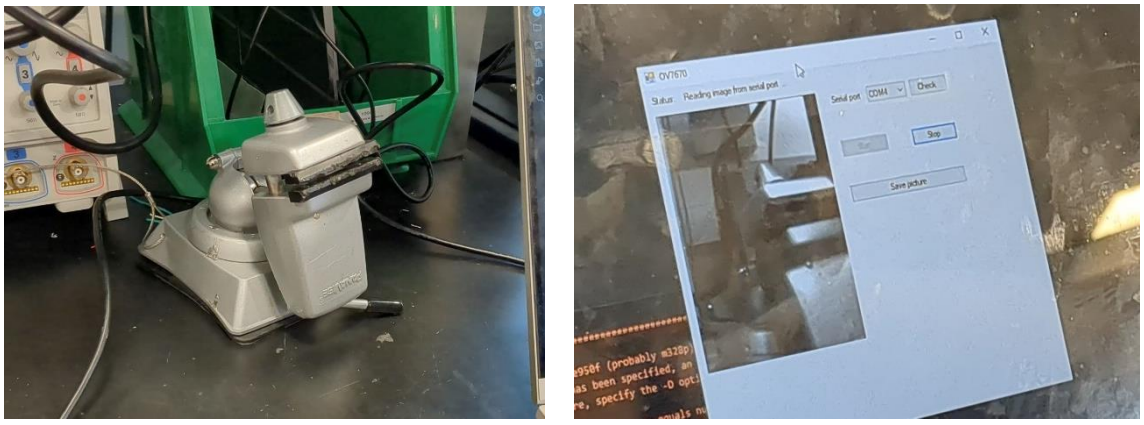

Figure 26. Laptop receiving images taken by the camera sensor.

Therefore, the sensor subsystem worked as expected, as it was able to take pictures with a sufficient image quality.

6.3. Control unit subsystem results

In order to test the control unit subsystem and the accuracy of the OCR engine, a dataset containing images of single characters (A-Z, 0-9) labeled by folders was used.

A designed script crawled through the dataset and fed each image to the OCR engine. Two different counters were added to the script. One of them would count the total number of images fed to the OCR engine, while the other would count the amount of times that the OCR engine properly recognized the character that had been inputted. Then, both numbers would be divided to obtain the accuracy as seen on equation 7.

$$OCR\ accuracy\ \% = \frac{number\ of\ images\ properly\ labeled}{number\ of\ images\ tested} \cdot 100 \qquad \textit{(Equation 7)}$$

The designed script used to test the OCR engine can be found on appendix C.

An image of the designed script running on the Libre AML-S905X-CC microprocessor can be seen on figure 27.

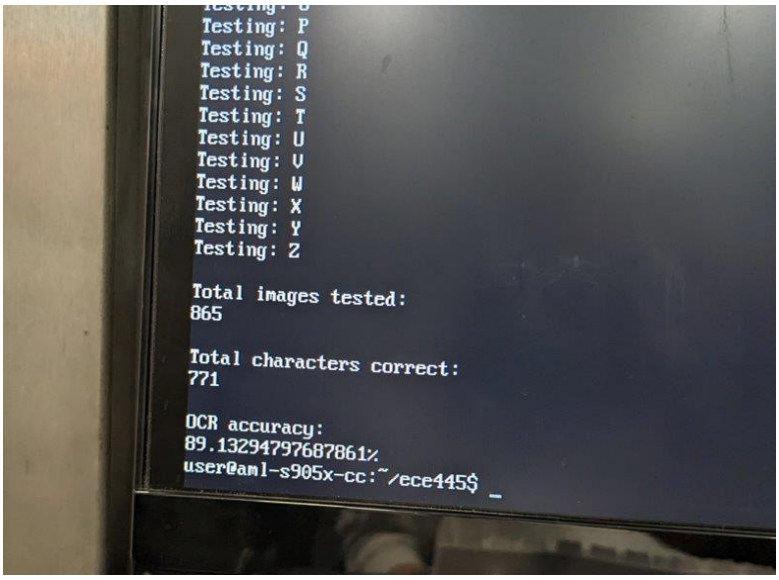

Figure 27. OCR accuracy results.

As seen on figure 27, the OCR accuracy achieved that time that the script was ran was approximately 89.13%. This percentage would vary between 85% and 92% depending on the images that were fed to the script.

Therefore, the control unit subsystem worked as expected, as the OCR engine was successfully ran using the designed script on the Libre AML-S905X-CC microprocessor and the obtained accuracy was around 90%.

6.4. User interface subsystem results

To test the user interface subsystem, the Arduino Uno was programmed to output each character of the braille alphabet, one character at a time. Both the Arduino Uno and the linear actuators were powered using the LiPo battery. An image from the test can be seen in figure 28 which shows the letter F been displayed on the linear actuators.

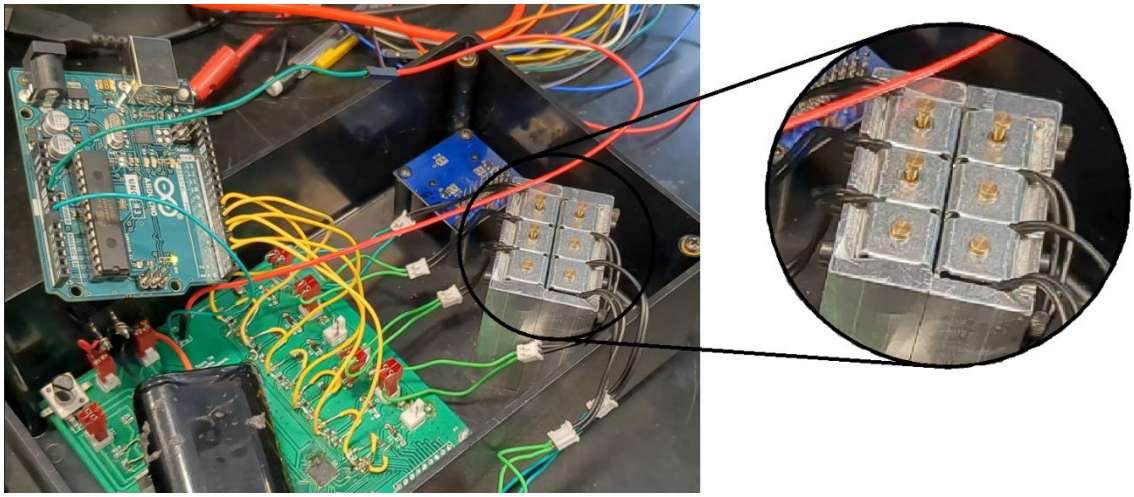

Figure 28. Letter F displayed on the linear actuators.

A ruler was used to measure the rise height of all the linear actuators when they were powered on. All the linear actuators where able to lift their pins 0.3cm on average.

All braille characters were properly displayed on the linear actuators and all pins extended to a readable high (minimum of 0.2 cm) in less than 1 second since they were activated. This means that the user interface subsystem worked as expected.

6.5. Overall project results

As explained in this chapter, a complete functionality of each individual subsystem was achieved and demonstrated as all the subsystem requirements and high-level requirements were met.

A table containing all the subsystem requirements verifications carried out can be found on appendix H.

However, the main objective of the project was to design a tool capable of performing a real time translation from printed text to braille.
Unfortunately, it was not possible to fully integrate every subsystem with each other to produce a final working product. This is mainly because of the inability to program the ATmega32U4 on the PCB.

The ATmega32U4 has a built-in USB controller which allows it to ack as a USB device. This feature was going to be used to interface the ATmega32U4 with the Libre AML-S905X-CC microprocessor as explained in the hardware design chapter. The microcontroller on the Arduino Uno (ATmega328P) does not have this feature and therefore it was not possible to connect it with the Libre AML-S905X-CC microprocessor.

Chapter 7. Conclusion

In the process of developing this project the largest obstacles were faced when implementing a full system integration. The inability of the Arduino Uno to act as a USB controller in contrast to the ATmega32U4 led to not been able to achieve a final device.

However, all the individual subsystems requirements were met, and it was possible to demonstrate their functionality with pictures. This means that all the subsystems of the project successfully worked on their own. Moreover, all the high-level requirements of the project were met and proved.

Therefore, the only thing needed to obtain a final working device that accomplishes the task of performing a real time translation from printed text to braille is to redesign the grounding connections on the custom PCB and to design the code that carries out the interface between the ATmega32U4 and the Libre AML-S905X-CC microprocessors. This is a straightforward procedure that could have been done if the issue on the grounding plane of the PCB had been discovered earlier in the debugging process.

7.1 Future work

The following optimizations would be carried out to improve the project's overall functionality in any future work done:

- **Achieve a full subsystem integration:** the first step in any future work for this project would be to achieve a full integration of all the subsystems to develop a final working product. The most efficient way to do this would be to redesign the PCB layout to connect all the grounding planes together. The ATmega32U4 microprocessor would then be used instead of the Arduino Uno and it would interface with the Libre AML-S905X-CC microprocessor.
- **Improve the sensor subsystem:** the camera sensor used in this project has a maximum image resolution of 640x480p. Been able to recognize characters from these images can become a laborious and hard task. One possible solution would be to use a scanner sensor (like the one most printers have nowadays) instead of using a camera sensor. By doing this, all the printed text would be scanned at once (without having to take multiple pictures) and the image quality would be improved.

Chapter 8. Ethical considerations and safety

The IEEE Code of Ethics [2] was mainly followed while carrying out this project. The main aspects to consider are:

- Image data must be collected in order to carry out the translation from text characters to braille. Image data is sensitive and private information. The privacy of users is a very important consideration; therefore, the integrity of the image data must be protected. All image data that is collected will be kept on the project's system, and never shared on the internet. This way, we will minimize the number of access points to the data, lowering the risk of a data leak.
- This project uses Lithium Polymer (LiPo) batteries in the power subsystem. LiPo batteries are notorious for being a potential safety and fire hazard when punctured, mishandled, or charged in a way that is not intended. Therefore, it would be our responsibility to avoid such a safety risk as much as possible. We would address this through the design of the power subsystem housing, and the way that the battery charging system is designed.
- As a device that aims to provide better accessibility for the visually impaired, we recognize the consequences of having an inaccurate reading of real-world data and therefore an inaccurate translation to braille. A user may depend on the accuracy of the data that is read to make decisions in their life, potentially life-threatening decisions. To minimize this risk, we will try to make our device as accurate as possible.
- We will ensure good teamwork, treating everyone in an equal manner and with respect. We will also make sure that all team members have access to all the project related documents.

Chapter 9. Bibliography

[1]. World Health Organization (WHO). "Blindness and vision impairment". (2022). [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (last visited on 02/09/2023).

[2]. IEEE. "IEEE Code of Ethics". (2020). [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (last visited on 02/09/2023).

[3]. Machine Shop, School of Chemical Sciences at UIUC. [Online]. Available: https://scs.illinois.edu/resources/cores-scs-service-facilities/machine-shop (last visited on 02/23/2023).

[4]. Salary averages. The Grainger college of engineering. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages (last visited on 02/23/2023).

[5]. Circuitdigest. "Solenoid Driver Circuit". [Online]. Available: https://circuitdigest.com/electronic-circuits/solenoid-driver-circuit-diagram (last visited on 05/03/2023).

[6]. PCBWay. "How to add USB-C to your projects". [Online]. Available: https://www.pcbway.com/blog/PCB_Design_Tutorial/How_to_add_USB_C_to_your_projects.html (last visited on 05/03/2023).

[7]. Punkisnail. "Li-ion Battery Charger". [Online]. Available: https://www.instructables.com/Li-ion-Battery-Charger/ (last visited on 05/03/2023).

[8]. Circuitdigest. "How to Use OV7670 Camera Module with Arduino Uno". (2019). [Online]. Available: https://circuitdigest.com/microcontroller-projects/how-to-use-ov7670-camera-module-with-arduino (last visited on 05/03/2023).

[9]. SMtech. "9V to 5V converters". [Online]. Available: https://somanytech.com/9v-to-5v-converter-circuit/ (last visited on 06/13/2023).

[10]. Bad Casserole. Neil. “Component selection and schematic for a custom ATmega32U4 PCB”. (2021). [Online]. Available: https://www.badcasserole.com/component-selection-and-schematic-for-a-custom-atmega32u4-pcb/ (last visited on 06/13/2023).

[11]. G7smy. Karl. “Solenoids on the Arduino with MOSFET power”. (2015). [Online]. Available: https://www.g7smy.co.uk/2015/02/solenoids-on-the-arduino-with-mosfet-power/ (last visited on 06/13/2023).

[12]. SHDesigns. Henion, Scott. “Lithium poly charge circuit”. (2003). [Online]. Available: http://shdesigns.org/lionchg.shtml (last visited on 06/13/2023).

[13]. American Foundation for the Blind (AFB). “What is braille?”. [Online]. Available: https://www.afb.org/blindness-and-low-vision/braille/what-braille (last visited on 04/20/2023).

[14]. United Nations Sustainable Development Goals (SDG). “Sustainable Development Goals”. [Online]. Available: https://www.un.org/sustainabledevelopment/ (last visited on 04/20/2023).

Appendix A. List of electronic components

The table below shows all the electronic components used to carry out this project and their prices:

Description	Manufacturer	Part number	Quantity	Total cost ($ USD)
Linear Actuator	Sparkfun	ROB-11015	6	29.70
Atmega32u4 (microcontroller on the PCB board)	Atmel	ATmega32U4	1	5.22
Arduino Uno (microprocessor on the PCB board)	Arduino Uno	A000066	1	27.60
Le Potato Microcontroller (microprocessor for OCR testing)	Libre	AML-S905X-CC	1	35.00
USB AVR Programmer	MDFLY	PGM-AV0010FBA	1	11.95
Camera	Olimex	OV7670	1	5.70
Battery	Jauch Quartz	LI18650JP2S1P	1	14.95
3V regulator	Onsemi	NCP565D2T33R4G	1	2.38
5V regulator	Texas Instruments	LM7805CT/NOPB	1	1.41
On/Off switch	ZF	SRB22A2DBBNN	1	1.41

USB-C	Molex	1054440011	1	3.01
Oscillator	Abracon	AWSCR-16.00CV-T	1	0.46
6 pin connector header	Würth	61200621621	1	0.48
2 pin connector header	Molex	0022232021	11	1.67
100µF capacitor	TDK	445-6007-6-ND	7	7.84
10µF capacitor	Murata	GRM21BC81E106ME51L	1	0.23
1µF capacitor	Kemet	C0805C105M4RAC7800	2	0.46
0.22µF capacitor	Kyocera	08055C224KAT2A	4	0.76
1000pF capacitor	Kyocera	478-1328-6-ND	6	0.72
20pF capacitor	Murata	GQM2195C2E200JB12D	2	1.22
1MΩ resistor	Vishay	749-1721-6-ND	1	0.36
100KΩ resistor	TE connectivity	CPF0805B100KE1	6	3.72
10KΩ resistor	Yageo	RC0805JR-1310KL	3	0.27
5.11KΩ resistor	Panasonic	ERJ-6ENF5111V	1	0.10
2.2KΩ resistor	Panasonic	ERJ-P06J222V	1	0.12

1KΩ resistor	Panasonic	ERJ-P06J102V	3	0.30
470Ω resistor	Panasonic	ERJ-P06J471V	3	0.36
160Ω resistor	Vishay	541-160CDKR-ND	6	0.54
47Ω resistor	TE Connectivity	CRGCQ0603F47R	1	0.15
1Ω resistor	Panasonic	ERJ-6GEYJ1R0V	1	0.09
50V diode	Diotec	1N4001	1	0.40
30V diode	Nexperia	632723300011	6	1.32
20V diode	Micro commercial	MBR0520L-TP	1	0.45
1KΩ potentiometer	Bourns	PTV09A-4225F-B102	1	0.89
Heat sink	Ohmite	RA-T2X-25E	1	2.09
BJTs	Toshiba	2SC4213BTE85LF	2	0.80
NMOS Transistor	Texas Instruments	CSD16301Q2	6	3.84
Total				**$167.97**

Table 1. Electronic components and their prices.

Appendix B. Camera sensor code

```
#include <stdint.h>
#include <avr/io.h>
#include <util/twi.h>
#include <util/delay.h>
#include <avr/pgmspace.h>

#define F_CPU 16000000UL
#define vga   0
#define qvga  1
#define qqvga   2
#define yuv422  0
#define rgb565  1
#define bayerRGB  2
#define camAddr_WR  0x42
#define camAddr_RD  0x43

/* Registers */
#define REG_GAIN    0x00  /* Gain lower 8 bits (rest in vref) */
#define REG_BLUE    0x01  /* blue gain */
#define REG_RED       0x02  /* red gain */
#define REG_VREF    0x03  /* Pieces of GAIN, VSTART, VSTOP */
#define REG_COM1    0x04  /* Control 1 */
#define COM1_CCIR656  0x40    /* CCIR656 enable */
#define REG_BAVE    0x05  /* U/B Average level */
#define REG_GbAVE   0x06  /* Y/Gb Average level */
#define REG_AECHH   0x07  /* AEC MS 5 bits */
#define REG_RAVE    0x08  /* V/R Average level */
#define REG_COM2    0x09  /* Control 2 */
#define COM2_SSLEEP         0x10  /* Soft sleep mode */
#define REG_PID           0x0a  /* Product ID MSB */
#define REG_VER           0x0b  /* Product ID LSB */
#define REG_COM3    0x0c  /* Control 3 */
#define COM3_SWAP         0x40  /* Byte swap */
#define COM3_SCALEEN          0x08  /* Enable scaling */
#define COM3_DCWEN          0x04  /* Enable downsamp/crop/window */
#define REG_COM4    0x0d  /* Control 4 */
#define REG_COM5    0x0e  /* All "reserved" */
#define REG_COM6    0x0f  /* Control 6 */
#define REG_AECH    0x10  /* More bits of AEC value */
#define REG_CLKRC   0x11  /* Clocl control */
#define CLK_EXT           0x40  /* Use external clock directly */
#define CLK_SCALE   0x3f  /* Mask for internal clock scale */
#define REG_COM7    0x12  /* Control 7 */ //REG mean address.
#define COM7_RESET          0x80  /* Register reset */
#define COM7_FMT_MASK         0x38
#define COM7_FMT_VGA          0x00
#define COM7_FMT_CIF          0x20  /* CIF format */
#define COM7_FMT_QVGA         0x10  /* QVGA format */
#define COM7_FMT_QCIF         0x08  /* QCIF format */
#define COM7_RGB          0x04  /* bits 0 and 2 - RGB format */
#define COM7_YUV          0x00  /* YUV */
#define COM7_BAYER          0x01  /* Bayer format */
#define COM7_PBAYER         0x05  /* "Processed bayer" */
#define REG_COM8    0x13  /* Control 8 */
#define COM8_FASTAEC          0x80  /* Enable fast AGC/AEC */
#define COM8_AECSTEP          0x40  /* Unlimited AEC step size */
```

```
#define COM8_BFILT    0x20  /* Band filter enable */
#define COM8_AGC    0x04  /* Auto gain enable */
#define COM8_AWB    0x02  /* White balance enable */
#define COM8_AEC    0x01  /* Auto exposure enable */
#define REG_COM9    0x14  /* Control 9- gain ceiling */
#define REG_COM10   0x15  /* Control 10 */
#define COM10_HSYNC         0x40  /* HSYNC instead of HREF */
#define COM10_PCLK_HB         0x20  /* Suppress PCLK on horiz blank */
#define COM10_HREF_REV          0x08  /* Reverse HREF */
#define COM10_VS_LEAD         0x04  /* VSYNC on clock leading edge */
#define COM10_VS_NEG          0x02  /* VSYNC negative */
#define COM10_HS_NEG          0x01  /* HSYNC negative */
#define REG_HSTART    0x17  /* Horiz start high bits */
#define REG_HSTOP   0x18  /* Horiz stop high bits */
#define REG_VSTART    0x19  /* Vert start high bits */
#define REG_VSTOP   0x1a  /* Vert stop high bits */
#define REG_PSHFT   0x1b  /* Pixel delay after HREF */
#define REG_MIDH    0x1c  /* Manuf. ID high */
#define REG_MIDL    0x1d  /* Manuf. ID low */
#define REG_MVFP    0x1e  /* Mirror / vflip */
#define MVFP_MIRROR         0x20  /* Mirror image */
#define MVFP_FLIP   0x10  /* Vertical flip */
#define REG_AEW           0x24  /* AGC upper limit */
#define REG_AEB           0x25    /* AGC lower limit */
#define REG_VPT           0x26  /* AGC/AEC fast mode op region */
#define REG_HSYST   0x30  /* HSYNC rising edge delay */
#define REG_HSYEN   0x31  /* HSYNC falling edge delay */
#define REG_HREF    0x32  /* HREF pieces */
#define REG_TSLB    0x3a  /* lots of stuff */
#define TSLB_YLAST    0x04  /* UYVY or VYUY - see com13 */
#define REG_COM11   0x3b  /* Control 11 */
#define COM11_NIGHT         0x80  /* NIght mode enable */
#define COM11_NMFR          0x60  /* Two bit NM frame rate */
#define COM11_HZAUTO          0x10  /* Auto detect 50/60 Hz */
#define COM11_50HZ          0x08  /* Manual 50Hz select */
#define COM11_EXP   0x02
#define REG_COM12   0x3c  /* Control 12 */
#define COM12_HREF          0x80  /* HREF always */
#define REG_COM13   0x3d  /* Control 13 */
#define COM13_GAMMA         0x80  /* Gamma enable */
#define COM13_UVSAT         0x40  /* UV saturation auto adjustment */
#define COM13_UVSWAP          0x01  /* V before U - w/TSLB */
#define REG_COM14   0x3e  /* Control 14 */
#define COM14_DCWEN         0x10  /* DCW/PCLK-scale enable */
#define REG_EDGE    0x3f  /* Edge enhancement factor */
#define REG_COM15   0x40  /* Control 15 */
#define COM15_R10F0         0x00  /* Data range 10 to F0 */
#define COM15_R01FE         0x80  /*      01 to FE */
#define COM15_R00FF         0xc0  /*      00 to FF */
#define COM15_RGB565          0x10  /* RGB565 output */
#define COM15_RGB555          0x30  /* RGB555 output */
#define REG_COM16   0x41  /* Control 16 */
#define COM16_AWBGAIN         0x08  /* AWB gain enable */
#define REG_COM17   0x42  /* Control 17 */
#define COM17_AECWIN          0xc0  /* AEC window - must match COM4 */
#define COM17_CBAR          0x08  /* DSP Color bar */
/*
* This matrix defines how the colors are generated, must be
* tweaked to adjust hue and saturation.
```

```
*
* Order: v-red, v-green, v-blue, u-red, u-green, u-blue
* They are nine-bit signed quantities, with the sign bit
* stored in0x58.Sign for v-red is bit 0, and up from there.
*/
#define REG_CMATRIX_BASE  0x4f
#define CMATRIX_LEN           6
#define REG_CMATRIX_SIGN  0x58
#define REG_BRIGHT    0x55  /* Brightness */
#define REG_CONTRAS         0x56  /* Contrast control */
#define REG_GFIX    0x69  /* Fix gain control */
#define REG_REG76   0x76  /* OV's name */
#define R76_BLKPCOR         0x80  /* Black pixel correction enable */
#define R76_WHTPCOR         0x40  /* White pixel correction enable */
#define REG_RGB444          0x8c  /* RGB 444 control */
#define R444_ENABLE         0x02  /* Turn on RGB444, overrides 5x5 */
#define R444_RGBX   0x01  /* Empty nibble at end */
#define REG_HAECC1    0x9f  /* Hist AEC/AGC control 1 */
#define REG_HAECC2    0xa0  /* Hist AEC/AGC control 2 */
#define REG_BD50MAX         0xa5  /* 50hz banding step limit */
#define REG_HAECC3    0xa6  /* Hist AEC/AGC control 3 */
#define REG_HAECC4    0xa7  /* Hist AEC/AGC control 4 */
#define REG_HAECC5    0xa8  /* Hist AEC/AGC control 5 */
#define REG_HAECC6    0xa9  /* Hist AEC/AGC control 6 */
#define REG_HAECC7    0xaa  /* Hist AEC/AGC control 7 */
#define REG_BD60MAX         0xab  /* 60hz banding step limit */
#define REG_GAIN    0x00  /* Gain lower 8 bits (rest in vref) */
#define REG_BLUE    0x01  /* blue gain */
#define REG_RED           0x02  /* red gain */
#define REG_VREF    0x03  /* Pieces of GAIN, VSTART, VSTOP */
#define REG_COM1    0x04  /* Control 1 */
#define COM1_CCIR656          0x40  /* CCIR656 enable */
#define REG_BAVE    0x05  /* U/B Average level */
#define REG_GbAVE   0x06  /* Y/Gb Average level */
#define REG_AECHH   0x07  /* AEC MS 5 bits */
#define REG_RAVE    0x08  /* V/R Average level */
#define REG_COM2    0x09  /* Control 2 */
#define COM2_SSLEEP         0x10  /* Soft sleep mode */
#define REG_PID           0x0a  /* Product ID MSB */
#define REG_VER           0x0b  /* Product ID LSB */
#define REG_COM3    0x0c  /* Control 3 */
#define COM3_SWAP         0x40  /* Byte swap */
#define COM3_SCALEEN          0x08  /* Enable scaling */
#define COM3_DCWEN          0x04  /* Enable downsamp/crop/window */
#define REG_COM4    0x0d  /* Control 4 */
#define REG_COM5    0x0e  /* All "reserved" */
#define REG_COM6    0x0f  /* Control 6 */
#define REG_AECH    0x10  /* More bits of AEC value */
#define REG_CLKRC   0x11  /* Clocl control */
#define CLK_EXT           0x40  /* Use external clock directly */
#define CLK_SCALE   0x3f  /* Mask for internal clock scale */
#define REG_COM7    0x12  /* Control 7 */
#define COM7_RESET          0x80  /* Register reset */
#define COM7_FMT_MASK         0x38
#define COM7_FMT_VGA          0x00
#define COM7_FMT_CIF          0x20  /* CIF format */
#define COM7_FMT_QVGA         0x10  /* QVGA format */
#define COM7_FMT_QCIF         0x08  /* QCIF format */
#define COM7_RGB    0x04  /* bits 0 and 2 - RGB format */
```

```
#define COM7_YUV    0x00  /* YUV */
#define COM7_BAYER          0x01  /* Bayer format */
#define COM7_PBAYER         0x05  /* "Processed bayer" */
#define REG_COM8    0x13  /* Control 8 */
#define COM8_FASTAEC          0x80  /* Enable fast AGC/AEC */
#define COM8_AECSTEP          0x40  /* Unlimited AEC step size */
#define COM8_BFILT    0x20  /* Band filter enable */
#define COM8_AGC    0x04  /* Auto gain enable */
#define COM8_AWB    0x02  /* White balance enable */
#define COM8_AEC    0x01  /* Auto exposure enable */
#define REG_COM9    0x14  /* Control 9- gain ceiling */
#define REG_COM10   0x15  /* Control 10 */
#define COM10_HSYNC         0x40  /* HSYNC instead of HREF */
#define COM10_PCLK_HB         0x20  /* Suppress PCLK on horiz blank */
#define COM10_HREF_REV          0x08  /* Reverse HREF */
#define COM10_VS_LEAD           0x04  /* VSYNC on clock leading edge */
#define COM10_VS_NEG          0x02  /* VSYNC negative */
#define COM10_HS_NEG          0x01  /* HSYNC negative */
#define REG_HSTART    0x17  /* Horiz start high bits */
#define REG_HSTOP   0x18  /* Horiz stop high bits */
#define REG_VSTART    0x19  /* Vert start high bits */
#define REG_VSTOP   0x1a  /* Vert stop high bits */
#define REG_PSHFT   0x1b  /* Pixel delay after HREF */
#define REG_MIDH    0x1c  /* Manuf. ID high */
#define REG_MIDL    0x1d  /* Manuf. ID low */
#define REG_MVFP    0x1e  /* Mirror / vflip */
#define MVFP_MIRROR         0x20  /* Mirror image */
#define MVFP_FLIP   0x10  /* Vertical flip */
#define REG_AEW           0x24  /* AGC upper limit */
#define REG_AEB           0x25  /* AGC lower limit */
#define REG_VPT           0x26  /* AGC/AEC fast mode op region */
#define REG_HSYST   0x30  /* HSYNC rising edge delay */
#define REG_HSYEN   0x31  /* HSYNC falling edge delay */
#define REG_HREF    0x32  /* HREF pieces */
#define REG_TSLB    0x3a  /* lots of stuff */
#define TSLB_YLAST    0x04  /* UYVY or VYUY - see com13 */
#define REG_COM11   0x3b  /* Control 11 */
#define COM11_NIGHT         0x80  /* NIght mode enable */
#define COM11_NMFR          0x60  /* Two bit NM frame rate */
#define COM11_HZAUTO          0x10  /* Auto detect 50/60 Hz */
#define COM11_50HZ          0x08  /* Manual 50Hz select */
#define COM11_EXP   0x02
#define REG_COM12   0x3c  /* Control 12 */
#define COM12_HREF          0x80  /* HREF always */
#define REG_COM13   0x3d  /* Control 13 */
#define COM13_GAMMA         0x80  /* Gamma enable */
#define COM13_UVSAT         0x40  /* UV saturation auto adjustment */
#define COM13_UVSWAP          0x01  /* V before U - w/TSLB */
#define REG_COM14   0x3e  /* Control 14 */
#define COM14_DCWEN         0x10  /* DCW/PCLK-scale enable */
#define REG_EDGE    0x3f  /* Edge enhancement factor */
#define REG_COM15   0x40  /* Control 15 */
#define COM15_R10F0         0x00  /* Data range 10 to F0 */
#define COM15_R01FE         0x80  /*      01 to FE */
#define COM15_R00FF         0xc0  /*      00 to FF */
#define COM15_RGB565          0x10  /* RGB565 output */
#define COM15_RGB555          0x30  /* RGB555 output */
#define REG_COM16   0x41  /* Control 16 */
#define COM16_AWBGAIN         0x08  /* AWB gain enable */
```

```
#define REG_COM17   0x42  /* Control 17 */
#define COM17_AECWIN          0xc0  /* AEC window - must match COM4 */
#define COM17_CBAR          0x08  /* DSP Color bar */
#define CMATRIX_LEN             6
#define REG_BRIGHT    0x55  /* Brightness */
#define REG_REG76   0x76  /* OV's name */
#define R76_BLKPCOR         0x80  /* Black pixel correction enable */
#define R76_WHTPCOR         0x40  /* White pixel correction enable */
#define REG_RGB444          0x8c  /* RGB 444 control */
#define R444_ENABLE         0x02  /* Turn on RGB444, overrides 5x5 */
#define R444_RGBX   0x01  /* Empty nibble at end */
#define REG_HAECC1    0x9f  /* Hist AEC/AGC control 1 */
#define REG_HAECC2    0xa0  /* Hist AEC/AGC control 2 */
#define REG_BD50MAX         0xa5  /* 50hz banding step limit */
#define REG_HAECC3    0xa6  /* Hist AEC/AGC control 3 */
#define REG_HAECC4    0xa7  /* Hist AEC/AGC control 4 */
#define REG_HAECC5    0xa8  /* Hist AEC/AGC control 5 */
#define REG_HAECC6    0xa9  /* Hist AEC/AGC control 6 */
#define REG_HAECC7    0xaa  /* Hist AEC/AGC control 7 */
#define REG_BD60MAX         0xab  /* 60hz banding step limit */
#define MTX1            0x4f  /* Matrix Coefficient 1 */
#define MTX2            0x50  /* Matrix Coefficient 2 */
#define MTX3            0x51  /* Matrix Coefficient 3 */
#define MTX4            0x52  /* Matrix Coefficient 4 */
#define MTX5            0x53  /* Matrix Coefficient 5 */
#define MTX6            0x54  /* Matrix Coefficient 6 */
#define REG_CONTRAS         0x56  /* Contrast control */
#define MTXS            0x58  /* Matrix Coefficient Sign */
#define AWBC7           0x59  /* AWB Control 7 */
#define AWBC8           0x5a  /* AWB Control 8 */
#define AWBC9           0x5b  /* AWB Control 9 */
#define AWBC10            0x5c  /* AWB Control 10 */
#define AWBC11            0x5d  /* AWB Control 11 */
#define AWBC12            0x5e  /* AWB Control 12 */
#define REG_GFI           0x69  /* Fix gain control */
#define GGAIN           0x6a  /* G Channel AWB Gain */
#define DBLV          0x6b
#define AWBCTR3           0x6c  /* AWB Control 3 */
#define AWBCTR2           0x6d  /* AWB Control 2 */
#define AWBCTR1          0x6e  /* AWB Control 1 */
#define AWBCTR0           0x6f  /* AWB Control 0 */

struct regval_list{
  uint8_t reg_num;
  uint16_t value;
};

const struct regval_list qvga_ov7670[] PROGMEM = {
  { REG_COM14, 0x19 },
  { 0x72, 0x11 },
  { 0x73, 0xf1 },
  { REG_HSTART, 0x16 },
  { REG_HSTOP, 0x04 },
  { REG_HREF, 0xa4 },
  { REG_VSTART, 0x02 },
  { REG_VSTOP, 0x7a },
  { REG_VREF, 0x0a },

  { 0xff, 0xff }, /* END MARKER */
```

```
};

const struct regval_list yuv422_ov7670[] PROGMEM = {
  { REG_COM7, 0x0 },  /* Selects YUV mode */
  { REG_RGB444, 0 },  /* No RGB444 please */
  { REG_COM1, 0 },
  { REG_COM15, COM15_R00FF },
  { REG_COM9, 0x6A }, /* 128x gain ceiling; 0x8 is reserved bit */
  { 0x4f, 0x80 },    /* "matrix coefficient 1" */
  { 0x50, 0x80 },    /* "matrix coefficient 2" */
  { 0x51, 0 },     /* vb */
  { 0x52, 0x22 },    /* "matrix coefficient 4" */
  { 0x53, 0x5e },    /* "matrix coefficient 5" */
  { 0x54, 0x80 },    /* "matrix coefficient 6" */
  { REG_COM13, COM13_UVSAT },

  { 0xff, 0xff },    /* END MARKER */
};

const struct regval_list ov7670_default_regs[] PROGMEM = {//from the linux
driver
  { REG_COM7, COM7_RESET },
  { REG_TSLB, 0x04 }, /* OV */
  { REG_COM7, 0 },  /* VGA */
  /*
  * Set the hardware window.  These values from OV don't entirely
  * make sense - hstop is less than hstart.  But they work...
  */
  { REG_HSTART, 0x13 }, { REG_HSTOP, 0x01 },
  { REG_HREF, 0xb6 }, { REG_VSTART, 0x02 },
  { REG_VSTOP, 0x7a }, { REG_VREF, 0x0a },

  { REG_COM3, 0 }, { REG_COM14, 0 },

  /* Mystery scaling numbers */
  { 0x70, 0x3a }, { 0x71, 0x35 },
  { 0x72, 0x11 }, { 0x73, 0xf0 },
  { 0xa2,/* 0x02 changed to 1*/1 }, { REG_COM10, 0x0 },

  /* Gamma curve values */
  { 0x7a, 0x20 }, { 0x7b, 0x10 },
  { 0x7c, 0x1e }, { 0x7d, 0x35 },
  { 0x7e, 0x5a }, { 0x7f, 0x69 },
  { 0x80, 0x76 }, { 0x81, 0x80 },
  { 0x82, 0x88 }, { 0x83, 0x8f },
  { 0x84, 0x96 }, { 0x85, 0xa3 },
  { 0x86, 0xaf }, { 0x87, 0xc4 },
  { 0x88, 0xd7 }, { 0x89, 0xe8 },

  /* AGC and AEC parameters.  Note we start by disabling those features,
  then turn them only after tweaking the values. */
  { REG_COM8, COM8_FASTAEC | COM8_AECSTEP },
  { REG_GAIN, 0 }, { REG_AECH, 0 },
  { REG_COM4, 0x40 }, /* magic reserved bit */
  { REG_COM9, 0x18 }, /* 4x gain + magic rsvd bit */
  { REG_BD50MAX, 0x05 }, { REG_BD60MAX, 0x07 },
  { REG_AEW, 0x95 }, { REG_AEB, 0x33 },
  { REG_VPT, 0xe3 }, { REG_HAECC1, 0x78 },
  { REG_HAECC2, 0x68 }, { 0xa1, 0x03 }, /* magic */
```

```
{ REG_HAECC3, 0xd8 }, { REG_HAECC4, 0xd8 },
{ REG_HAECC5, 0xf0 }, { REG_HAECC6, 0x90 },
{ REG_HAECC7, 0x94 },
{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP | COM8_AGC | COM8_AEC },
{ 0x30, 0 }, { 0x31, 0 },//disable some delays

/* Almost all of these are magic "reserved" values.  */
{ REG_COM5, 0x61 }, { REG_COM6, 0x4b },
{ 0x16, 0x02 }, { REG_MVFP, 0x07 },
{ 0x21, 0x02 }, { 0x22, 0x91 },
{ 0x29, 0x07 }, { 0x33, 0x0b },
{ 0x35, 0x0b }, { 0x37, 0x1d },
{ 0x38, 0x71 }, { 0x39, 0x2a },
{ REG_COM12, 0x78 }, { 0x4d, 0x40 },
{ 0x4e, 0x20 }, { REG_GFIX, 0 },
/*{0x6b, 0x4a},*/{ 0x74, 0x10 },
{ 0x8d, 0x4f }, { 0x8e, 0 },
{ 0x8f, 0 }, { 0x90, 0 },
{ 0x91, 0 }, { 0x96, 0 },
{ 0x9a, 0 }, { 0xb0, 0x84 },
{ 0xb1, 0x0c }, { 0xb2, 0x0e },
{ 0xb3, 0x82 }, { 0xb8, 0x0a },
/* More reserved magic, some of which tweaks white balance */
{ 0x43, 0x0a }, { 0x44, 0xf0 },
{ 0x45, 0x34 }, { 0x46, 0x58 },
{ 0x47, 0x28 }, { 0x48, 0x3a },
{ 0x59, 0x88 }, { 0x5a, 0x88 },
{ 0x5b, 0x44 }, { 0x5c, 0x67 },
{ 0x5d, 0x49 }, { 0x5e, 0x0e },
{ 0x6c, 0x0a }, { 0x6d, 0x55 }
{ 0x6e, 0x11 }, { 0x6f, 0x9e }, /* it was 0x9F "9e for advance AWB" */
{ 0x6a, 0x40 }, { REG_BLUE, 0x40 },
{ REG_RED, 0x60 },

{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP | COM8_AGC | COM8_AEC | COM8_AWB },
/* Matrix coefficients */
{ 0x4f, 0x80 }, { 0x50, 0x80 },
{ 0x51, 0 },    { 0x52, 0x22 },
{ 0x53, 0x5e }, { 0x54, 0x80 },
{ 0x58, 0x9e },

{ REG_COM16, COM16_AWBGAIN }, { REG_EDGE, 0 },
{ 0x75, 0x05 }, { REG_REG76, 0xe1 },
{ 0x4c, 0 },     { 0x77, 0x01 },

{ REG_COM13, /*0xc3*/0x48 }, { 0x4b, 0x09 },
{ 0xc9, 0x60 },   /*{REG_COM16, 0x38},*/
{ 0x56, 0x40 },

{ 0x34, 0x11 }, { REG_COM11, COM11_EXP | COM11_HZAUTO },
{ 0xa4, 0x82/*Was 0x88*/ }, { 0x96, 0 },
{ 0x97, 0x30 }, { 0x98, 0x20 },
{ 0x99, 0x30 }, { 0x9a, 0x84 },
{ 0x9b, 0x29 }, { 0x9c, 0x03 },
{ 0x9d, 0x4c }, { 0x9e, 0x3f },
{ 0x78, 0x04 },
/* Extra-weird stuff.  Some sort of multiplexor register */
{ 0x79, 0x01 }, { 0xc8, 0xf0 },
{ 0x79, 0x0f }, { 0xc8, 0x00 },
```

```
  { 0x79, 0x10 }, { 0xc8, 0x7e },
  { 0x79, 0x0a }, { 0xc8, 0x80 },
  { 0x79, 0x0b }, { 0xc8, 0x01 },
  { 0x79, 0x0c }, { 0xc8, 0x0f },
  { 0x79, 0x0d }, { 0xc8, 0x20 },
  { 0x79, 0x09 }, { 0xc8, 0x80 },
  { 0x79, 0x02 }, { 0xc8, 0xc0 },
  { 0x79, 0x03 }, { 0xc8, 0x40 },
  { 0x79, 0x05 }, { 0xc8, 0x30 },
  { 0x79, 0x26 },
  { 0xff, 0xff }, /* END MARKER */

};

void error_led(void){
  DDRB |= 32;//make sure led is output
  while (1){//wait for reset
    PORTB ^= 32;// toggle led
    _delay_ms(100);
  }
}

void twiStart(void){
  TWCR = _BV(TWINT) | _BV(TWSTA) | _BV(TWEN);//send start
  while (!(TWCR & (1 << TWINT)));//wait for start to be transmitted
  if ((TWSR & 0xF8) != TW_START)
    error_led();
}

void twiWriteByte(uint8_t DATA, uint8_t type){
  TWDR = DATA;
  TWCR = _BV(TWINT) | _BV(TWEN);
  while (!(TWCR & (1 << TWINT))) {}
  if ((TWSR & 0xF8) != type)
    error_led();
}

void twiAddr(uint8_t addr, uint8_t typeTWI){
  TWDR = addr;//send address
  TWCR = _BV(TWINT) | _BV(TWEN);    /* clear interrupt to start transmission
*/
  while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
  if ((TWSR & 0xF8) != typeTWI)
    error_led();
}

void writeReg(uint8_t reg, uint8_t dat){
  //send start condition
  twiStart();
  twiAddr(camAddr_WR, TW_MT_SLA_ACK);
  twiWriteByte(reg, TW_MT_DATA_ACK);
  twiWriteByte(dat, TW_MT_DATA_ACK);

  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop
  _delay_ms(1);
}

static uint8_t twiRd(uint8_t nack){
```

```
  if (nack){
    TWCR = _BV(TWINT) | _BV(TWEN);
    while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
    if ((TWSR & 0xF8) != TW_MR_DATA_NACK)
      error_led();
    return TWDR;
  }

  else{
    TWCR = _BV(TWINT) | _BV(TWEN) | _BV(TWEA);
    while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
    if ((TWSR & 0xF8) != TW_MR_DATA_ACK)
      error_led();
    return TWDR;
  }
}

uint8_t rdReg(uint8_t reg){

  uint8_t dat;
  twiStart();
  twiAddr(camAddr_WR, TW_MT_SLA_ACK);
  twiWriteByte(reg, TW_MT_DATA_ACK);

  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop

  _delay_ms(1);
  twiStart();
  twiAddr(camAddr_RD, TW_MR_SLA_ACK);
  dat = twiRd(1);
  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop
  _delay_ms(1);
  return dat;
}

void wrSensorRegs8_8(const struct regval_list reglist[]){

  uint8_t reg_addr, reg_val;
  const struct regval_list *next = reglist;

  while ((reg_addr != 0xff) | (reg_val != 0xff)){
    reg_addr = pgm_read_byte(&next->reg_num);
    reg_val = pgm_read_byte(&next->value);
   writeReg(reg_addr, reg_val);
    next++;
  }
}

void setColor(void){
  wrSensorRegs8_8(yuv422_ov7670);
 // wrSensorRegs8_8(qvga_ov7670);
}

void setResolution(void){
 writeReg(REG_COM3, 4); // REG_COM3 enable scaling
  wrSensorRegs8_8(qvga_ov7670);
}

void camInit(void){
```

```
 writeReg(0x12, 0x80);
  _delay_ms(100);
  wrSensorRegs8_8(ov7670_default_regs);
 writeReg(REG_COM10, 32);//PCLK does not toggle on HBLANK.
}

void arduinoUnoInut(void) {

  cli();//disable interrupts
    /* Setup the 8mhz PWM clock
  * This will be on pin 11*/

  DDRB |= (1 << 3);//pin 11
  ASSR &= ~(_BV(EXCLK) | _BV(AS2));
  TCCR2A = (1 << COM2A0) | (1 << WGM21) | (1 << WGM20);
  TCCR2B = (1 << WGM22) | (1 << CS20);
  OCR2A = 2;//(F_CPU)/(2*(X+1))
  DDRC &= ~15;//low d0-d3 camera
  DDRD &= ~252;//d7-d4 and interrupt pins

  _delay_ms(3000);
    //set up twi for 100khz
  TWSR &= ~3;//disable prescaler for TWI

  TWBR = 72;//set to 100khz
    //enable serial
  UBRR0H = 0;
  UBRR0L = 1;//0 = 2M baud rate. 1 = 1M baud. 3 = 0.5M. 7 = 250k 207 is 9600
baud rate.
  UCSR0A |= 2;//double speed aysnc
  UCSR0B = (1 << RXEN0) | (1 << TXEN0);//Enable receiver and transmitter
  UCSR0C = 6;//async 1 stop bit 8bit char no parity bits
}

void StringPgm(const char * str){
  do{
      while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit
      UDR0 = pgm_read_byte_near(str);
      while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit
  } while (pgm_read_byte_near(++str));
}

static void captureImg(uint16_t wg, uint16_t hg){
  uint16_t y, x;
  StringPgm(PSTR("*RDY*"));
  while (!(PIND & 8));//wait for high
  while ((PIND & 8));//wait for low
    y = hg;
  while (y--){
        x = wg;
      //while (!(PIND & 256));//wait for high
    while (x--){
      while ((PIND & 4));//wait for low
            UDR0 = (PINC & 15) | (PIND & 240);
          while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit
      while (!(PIND & 4));//wait for high
      while ((PIND & 4));//wait for low
      while (!(PIND & 4));//wait for high
    }
```

```
    //  while ((PIND & 256));//wait for low
  }
    _delay_ms(100);
}

void setup(){
  arduinoUnoInut();
  camInit();
  setResolution();
  setColor();
 writeReg(0x11, 31);
}

void loop(){
  captureImg(320, 240);
}
```

Appendix C. OCR code

```
import cv2
import numpy as np
import os
import pytesseract

####### PREPROCESSING FUNCTIONS #######
# get grayscale image
def get_grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# noise removal
def remove_noise(image):
    return cv2.medianBlur(image,5)

#thresholding
def thresholding(image):
    return cv2.threshold(image, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)[1]

#dilation
def dilate(image):
    kernel = np.ones((5,5),np.uint8)
    return cv2.dilate(image, kernel, iterations = 1)

#erosion
def erode(image):
    kernel = np.ones((5,5),np.uint8)
    return cv2.erode(image, kernel, iterations = 1)

#opening - erosion followed by dilation
def opening(image):
    kernel = np.ones((5,5),np.uint8)
    return cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)

#canny edge detection
def canny(image):
    return cv2.Canny(image, 100, 200)

#skew correction
def deskew(image):
    coords = np.column_stack(np.where(image > 0))
    angle = cv2.minAreaRect(coords)[-1]
    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(image, M, (w, h), flags=cv2.INTER_CUBIC,
borderMode=cv2.BORDER_REPLICATE)
    return rotated

#template matching
def match_template(image, template):
    return cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)
```

```
####### END PREPROCESSING FUNCTIONS #######

testedCount = 0
correctCount = 0
flip = 1

# Walk through all files in test_dataset
for root, dirs, files in os.walk(r'testing_data'):
    dirs.sort()

    # Extract the correct char value from test data's directory name
    currChar = root[len(root)-1].upper()
    print("Testing: " + currChar)

    # Walk through each file in test data's directory
    for file in files:
        testedCount += 1
        if file.endswith('.png'):
            filePath = os.path.join(root, file)
            # open file as img object
            img = cv2.imread(filePath)
            img = get_grayscale(img)
            img = remove_noise(img)
            img = thresholding(img)

            # Adding custom options
            custom_config = r'--oem 3 --psm 6'
            testChar = pytesseract.image_to_string(img,
config=custom_config).upper().strip()

            if (testChar == currChar):
                correctCount += 1


print("\nTotal images tested: ")
print(testedCount)
print("\nTotal characters correct: ")
print(correctCount)
print("\nOCR accuracy: ")
print(str((correctCount/testedCount) * 100) + "%")
```

Appendix D. PCB schematic and PCB layout

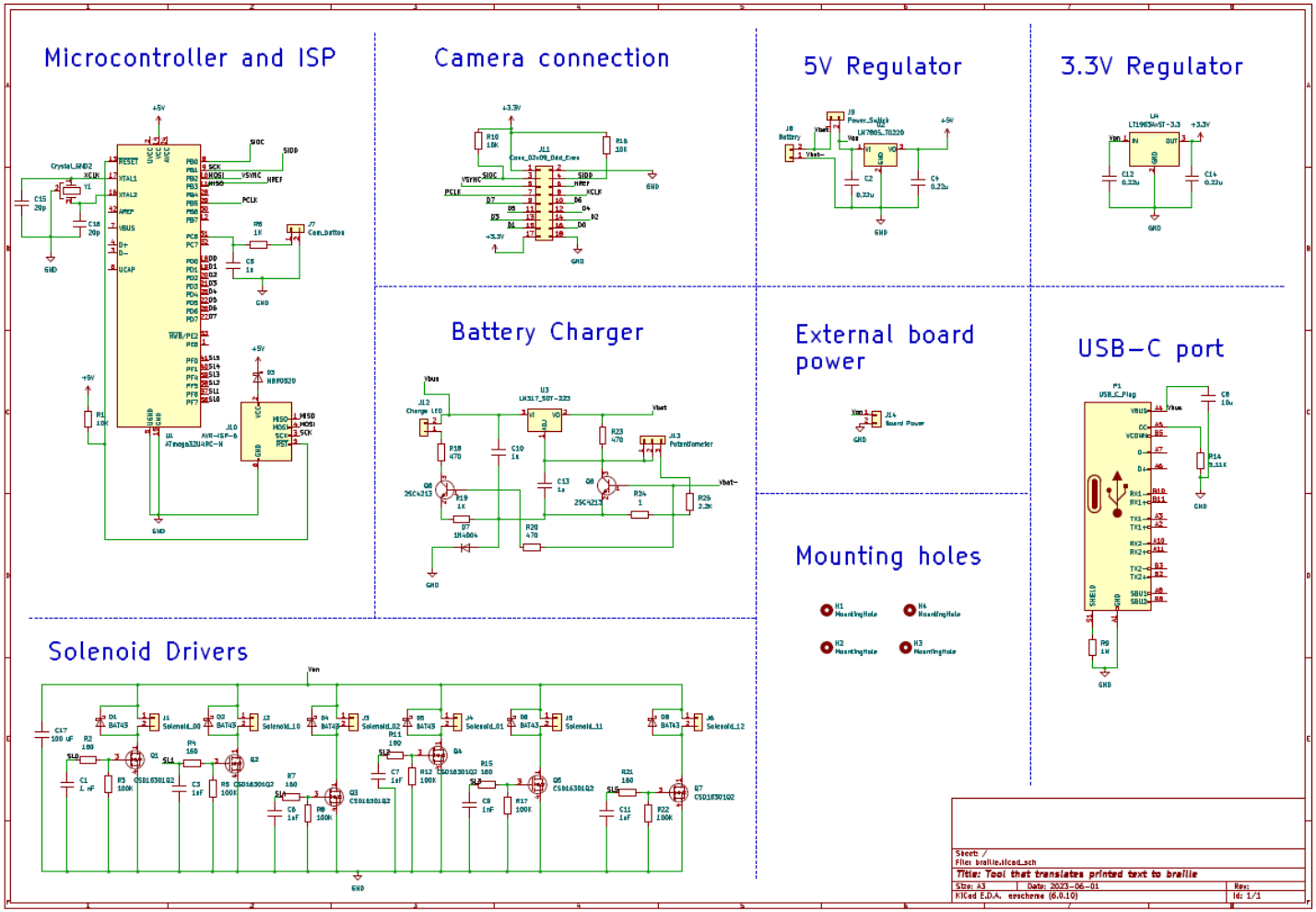

Figure 29. PCB schematic.

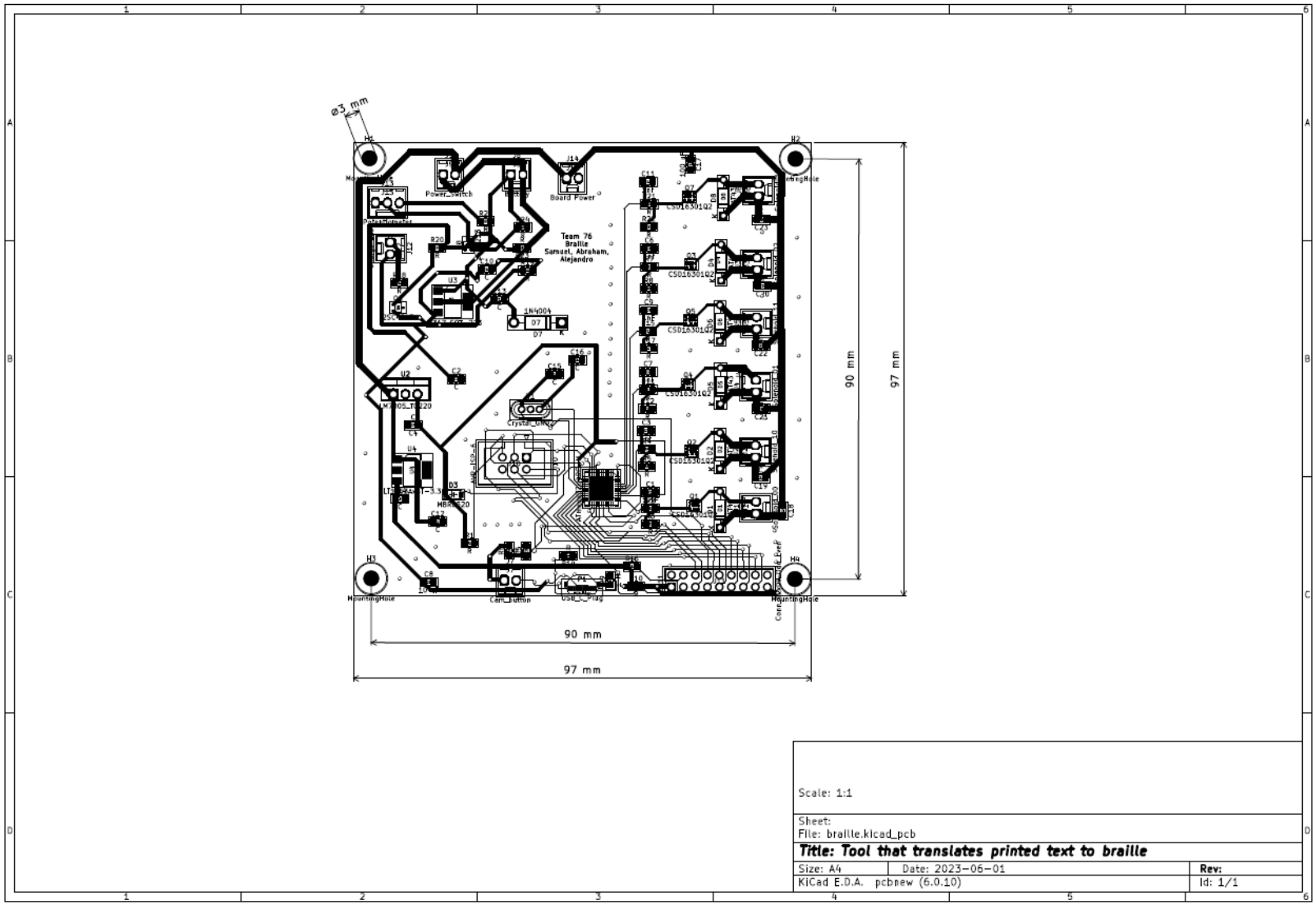

Figure 30. PCB layout.

Appendix E. Physical design drawings

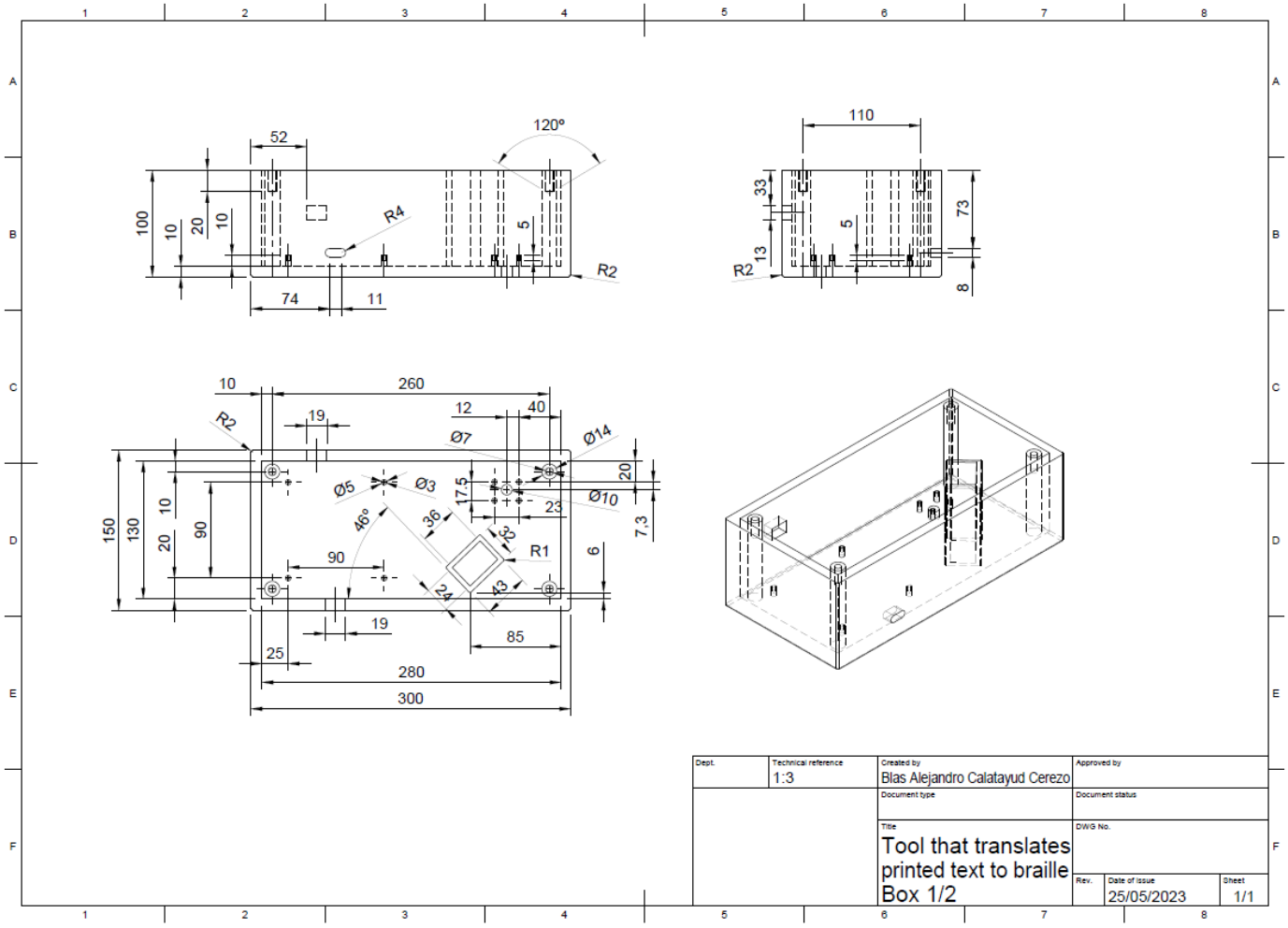

Figure 31. Container drawing.

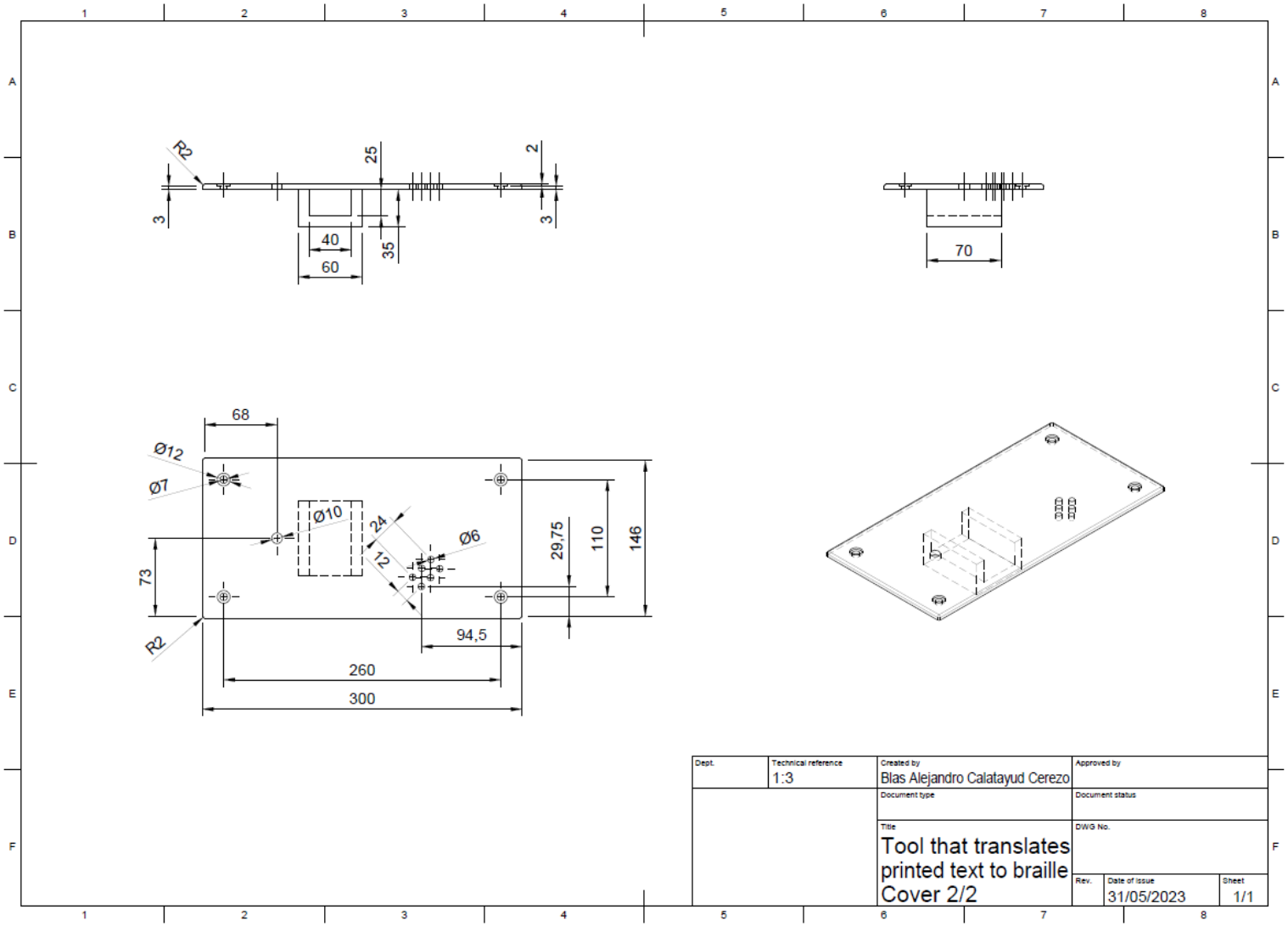

Figure 32. Cover drawing.

Appendix F. Labor hours distribution

Work Type	**Samuel's hours**	**Abraham's hours**	**Blas Alejandro's hours**
PCB design	25	0	25
Soldering	20	0	20
OCR Programming	0	35	0
Debugging	30	40	30
Writing reports and documentation	25	25	25
Total	**100**	**100**	**100**

Table 2. Approximate number of hours spent by each team member on each work.

Appendix G. Personal weekly schedule

The schedule below summarizes the tasks I was in charge of doing each week:

Week (month/day)	Work to be done
02/20	• Start ordering the components needed for the project. • Begin researching and designing the circuits that will be used in the power subsystem and the linear actuator voltage driver circuit. • Learn how to use *KiCad* (the software used to design the PCB board).
02/27	• Design the circuits that will be used in the power subsystem and the linear actuator voltage driver circuit. • Help with the design of the camera and microcontroller connections to be made on the PCB.
03/06	• Finish the design of all the circuits in *KiCad*. • Do the PCB layout and order the PCB board. • Order more components needed for the project.
03/13	Spring break
03/20	• Build and test the first prototype by soldering and testing the components on the PCB board.
03/27	• Fix potential problems in the PCB board (design a new PCB if necessary).
04/03	• Build the final design and test it. Solve any potential issues.
04/10	• Final adjustments for mock demo and testing.
04/17	• Improve the project based on the feedback received in the mock demo. • Final Adjustments for final demo and testing.
04/24	• Final presentation at the University of Illinois at Urbana-Champaign
05/01	• Work on the final paper for the University of Illinois at Urbana-Champaign
Rest of the time	• Work on the final paper for ICAI and final presentation at ICAI

Table 3. Personal weekly schedule for the project.

Appendix H. Subsystems requirement and verification table

The table below summarizes all the verifications carried out to check whether each subsystem requirements had been met.

Requirement	Verification	Accomplishment (Yes or No)
The voltage regulators will limit the voltage to the correct value for each system component.	Use a voltmeter to measure voltage through voltage regulators under normal operation. Expected measurement should be 3.3±0.1V and 5.0±0.2 V	Yes
The power management subsystem will be able to safely charge the battery	The potentiometer regulates the voltage to the current battery voltage and can be used to charge the LiPo battery. The voltage can be measured during testing using a voltmeter to ensure there is the expected voltage.	Yes
The camera must be able to take a 480p resolution photo and send the data to the control unit system.	The OCR testing software receives 480p resolution photos.	Yes
The OCR algorithm on the external microprocessor must be able to analyze image data and convert to character texts with 90% accuracy rate.	The OCR testing suite will use a test dataset of 480p resolution images of characters, the OCR algorithm will be run against this dataset to determine its accuracy.	Yes

The character text data is converted into signals that are sent to the motors to form braille characters.	Test software will be written in order to replicate converted signals from the OCR algorithm. All braille character signals will be sent to the linear actuators one by one and the correct formation will be verified by eye.	Yes
The linear actuators must be able to lift the pins 0.20±0.1 cm high and to lower them in less than 1 second when forming the braille characters.	Measure rise height when all solenoids are powered on and use a ruler to check height.	Yes

Table 4. Subsystem requirements and verifications.