TOOL THAT TRANSLATES PRINTED TEXT TO BRAILLE

By

Abraham Han

Blas Alejandro Calatayud Cerezo

Samuel Foley

Final Report for ECE 445, Senior Design, Spring 2023

TA: Raman Singh

3rd May 2023

Project No. 76

Abstract

The project aims to create a device that displays physical text as physical braille for the visually impaired. Most, if not all functionality was achieved in all separate subsystems, however system integration was not. The following report details the goals set at the start of the project, technical specifications of the project, what was successfully accomplished in the subsystems, what and how things failed in the system integration, the costs of the project, and finally, our concluding thoughts with regards to the project as a whole.

Contents

1. Introduction

1.1 Problem

According to the World Health Organization, there are around 39 million people who are legally blind around the world. The question we attempted to address is how can we aid the visually impaired navigate a world that is designed for the abled? Our answer to this question was to make information available to the visually impaired.

1.2 Solution

We attempted to create a device that converts physical text into physical braille. The device is a box with a camera sensor on the bottom, pins that stick out the top, and everything else necessary to power and drive the device inside the box.

The device is operated by holding the box over the text that the user wants displayed. The box takes an image of the text, and the braille pins start spelling out the text that has been read by the camera sensor.

1.3 High-Level Requirement

1. Text character to braille display accuracy is at least 90%
2. Battery life lasts 8 hours for an all-day battery life
3. All pins extend to 0.35±0.1 cm high in less than 1 second

1.4 Block diagram

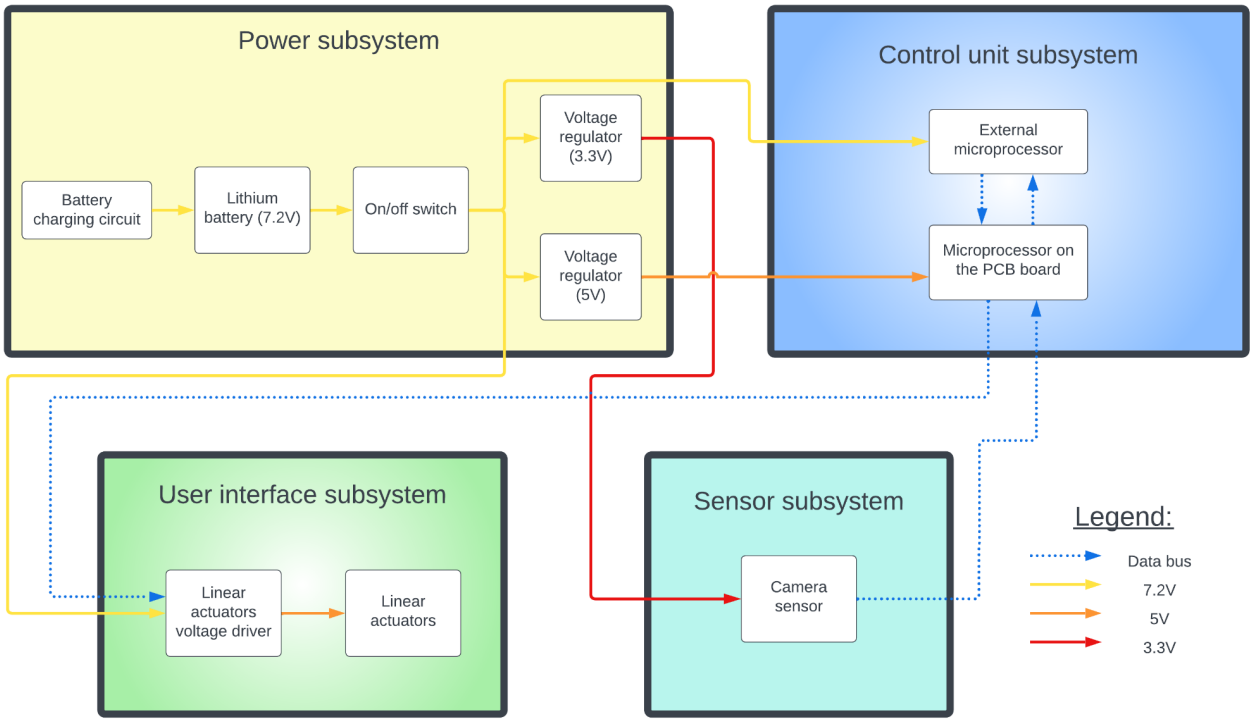

Figure 1. Block diagram

2 Design

2.1 Power Management Subsystem

The three core aspects of our power management subsystems are the battery charging circuit, the 5V regulator, and the 3.3V regulator. This was slightly changed from our very first design which was to use a 3.7 volt battery, with a 5 volt boost converter. We decided that using a 7.2 Volt battery would be easier as we could power the solenoids directly from the battery, and use a 5 volt regulator for the power to the microcontroller. Our 5 and 3.3 volt regulators used appropriate voltage regulator ICs with capacitors connecting the input and output to ground. Our battery charging circuit used an adjustable voltage regulator which would be set before charging to safely charge the battery. We also planned to use a USB-C to try and charge the battery, but we decided to leave it out of the final product as soldering it was nearly impossible for us to do.

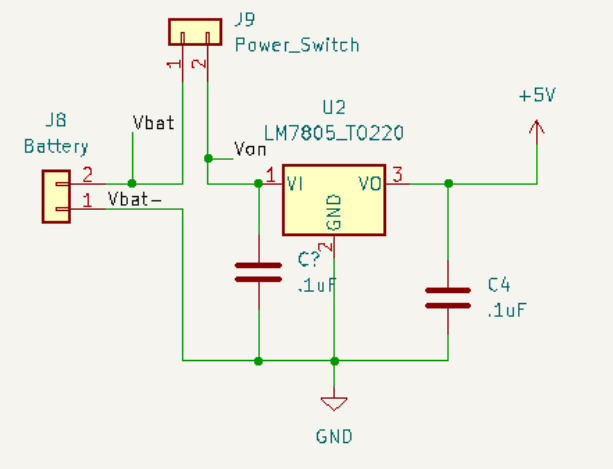

Figure 2. Schematic of the 5V regulator

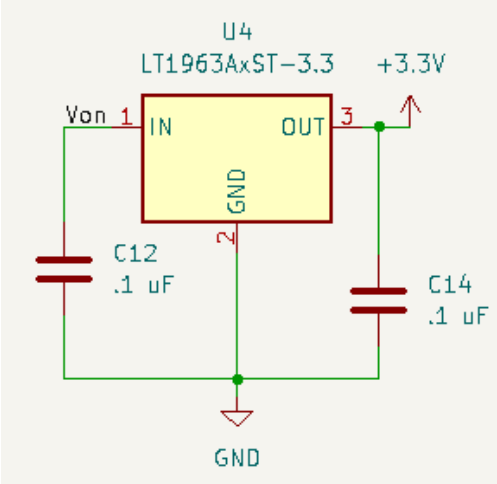

Figure 3. Schematic of the 3.3V regulator

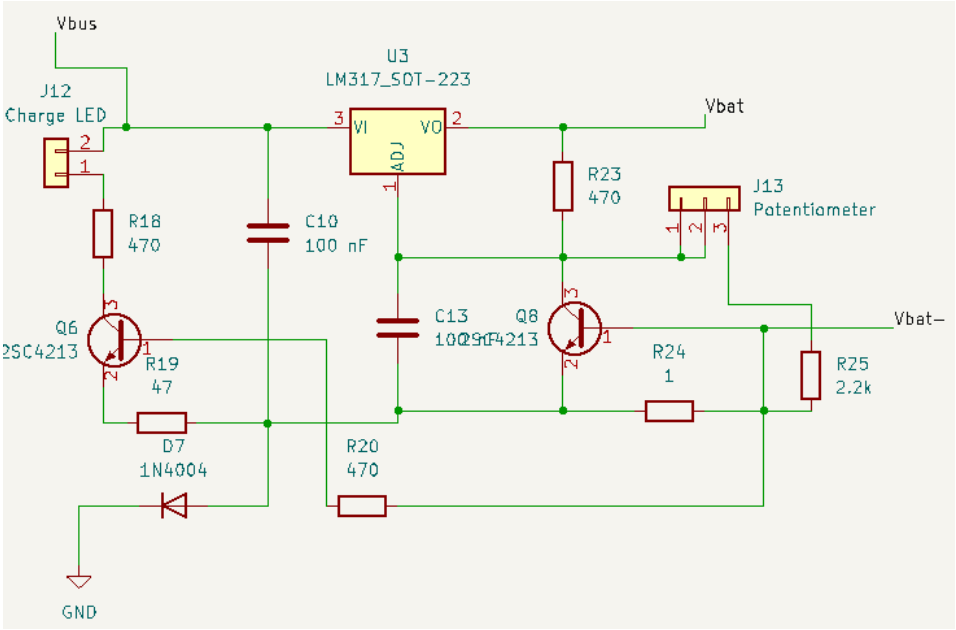

Figure 4. Schematic of the battery charging circuit

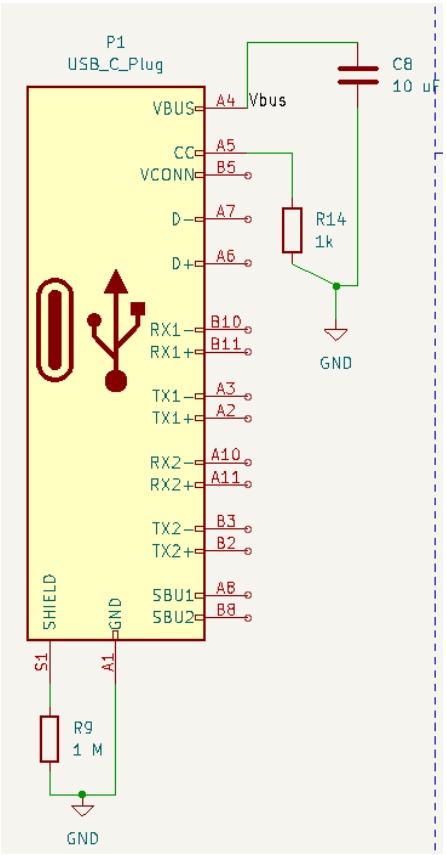

Figure 5. USB-C connections

2.2 Sensor Subsystem

For the Sensor Subsystem we choose to use the OV7670 camera. The reason we chose this camera was because of its affordability and we knew that it would be compatible with our microcontroller. We also looked at some raspberry pi cameras, which were a bit nice but decided against them due to the reason mentioned above. The code we used to receive images from the camera, modified for the arduino uno, can be seen in Appendix B.

To connect the camera to the microcontroller to the PCB, we used an 18-pin connector footprint so that we could make the connection between the pcb and camera using jumper wires. Each part of the 18-pin connector was connected to either a port on the Atmega32U4, to 3.3 volts, or to ground.

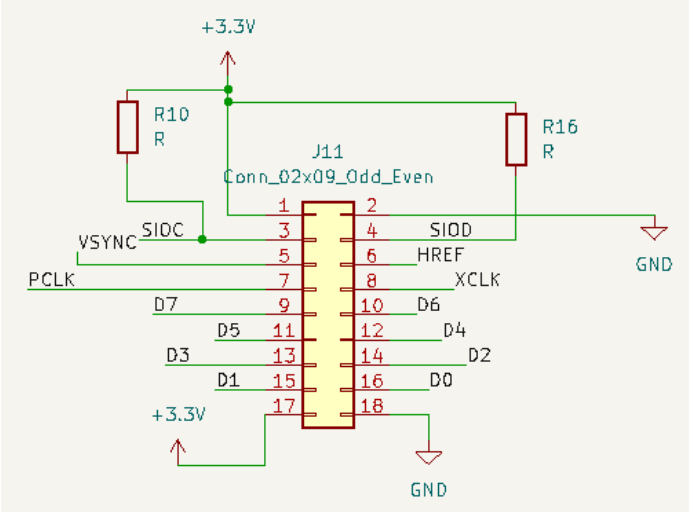

Figure 6. Camera connections

2.3 Control Unit Subsystem

At the heart of the control unit subsystem is the Optical Character Recognition (OCR) engine. Tesseract was the OCR engine of choice 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 microcontroller. Due to reasons discussed later, the interfacing between the microcontroller and the microprocessor on the PCB was not accomplished, however, the intended design was to have them communicate these character strings via the serial data transmission line on the GPIO header.

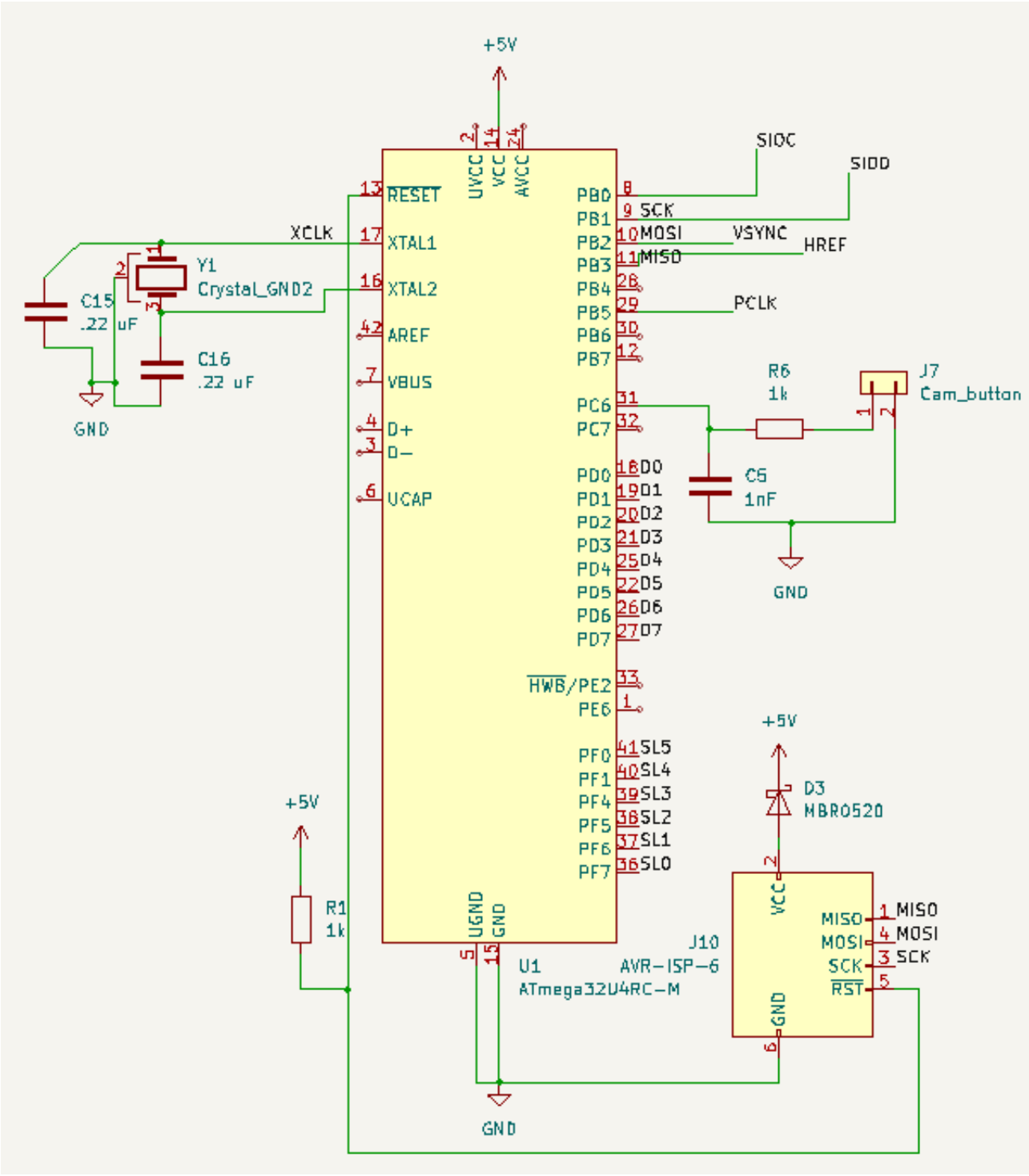

Figure 7. Circuit for the Atmega32U4 microcontroller, oscillator and camera button circuit

2.4 User Interface Subsystem

Our first step in designing the user interface system was to figure out what type of motor we should use to make the pins of our braille character. In our initial talk with the machine shop,we came to the conclusion that solenoids, with the pin being the braille dot, would work best. The next design step was designing a circuit to turn the solenoids on and off with the microcontroller, but where the solenoids would be receiving current directly from the battery as each solenoid requires around 1 amp of current, requiring a maximum of 5 amps which our microcontroller would not be able to supply. The idea we came up with was to use a NMOS transistor, where the output of the microcontroller would be connected to the gate voltage, the solenoid would be connected between the drain of the transistor and the positive terminal of the battery, and the source would be connected to ground.

Our final Design for our solenoid drivers can be seen in Figure 8. In addition to our initial design design we had a voltage divider circuit with a parallel capacitor. The voltage divider ended up not being necessary, which is why we made the first resistor ten times less than the second resistor. The capacitors were to reduce the effect of noise. The diodes are in parallel with the solenoids and facing the opposite direction to eliminate discharge from the solenoid into the rest of the circuit when the solenoid is turned off.

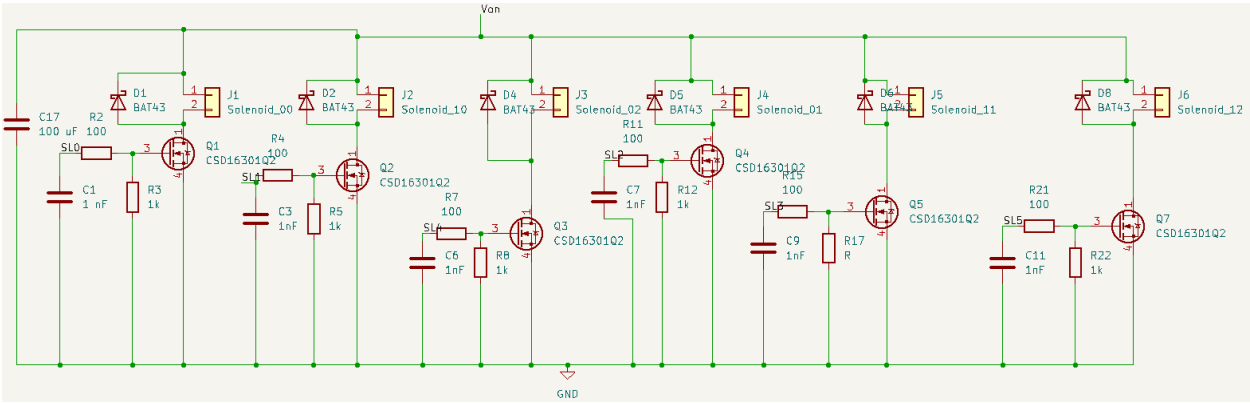

Figure 8. Driver circuit for our six solenoids

3. Design Verification

Unfortunately, we were not able to fully integrate each subsystem with each other to produce a working final product. This is in large part due to not being able to program the microcontroller on the pcb. However, we were able to get each of our subsystems working independently from each other.

3.1 Power Management Subsystem

As mentioned previously, the three core aspects of our power management subsystems are the battery charging circuit, the 5V regulator, and the 3.3V regulator and each of them has their own design verification.

3.1.1 Voltage regulators

We used a voltmeter to test both voltage regulators in our circuit. After plugging in the battery and turning on the power switch we measured the voltages between the ground plane on the PCB board and the output node on each voltage regulator. In figure 9 below we can see the output voltage of the 5V regulator, while in figure 10 we can see the output voltage of the 3.3V regulator:

Figure 9. Output voltage of the 5V regulator

Figure 10. Output voltage of the 3.3V regulator

3.1.2 Battery Charging Circuit

Due to difficulties with the USB-C connector, we decided to use one of the power supplies in the lab to test our battery charging circuit. We used 12V, and tested the charger by connecting the positive end of the power supply to a wire soldered to Vbus and the negative end of the power supply to ground. After using the potentiometer to adjust the output voltage of the voltage regulator, we connected both the battery and the power supply, with the switch to the rest of the circuit turned off. We recorded the battery voltage before charging and after leaving the power supply connected for about a minute. As can be seen in figure 11, the overall voltage of the battery increased by about 14 mV.

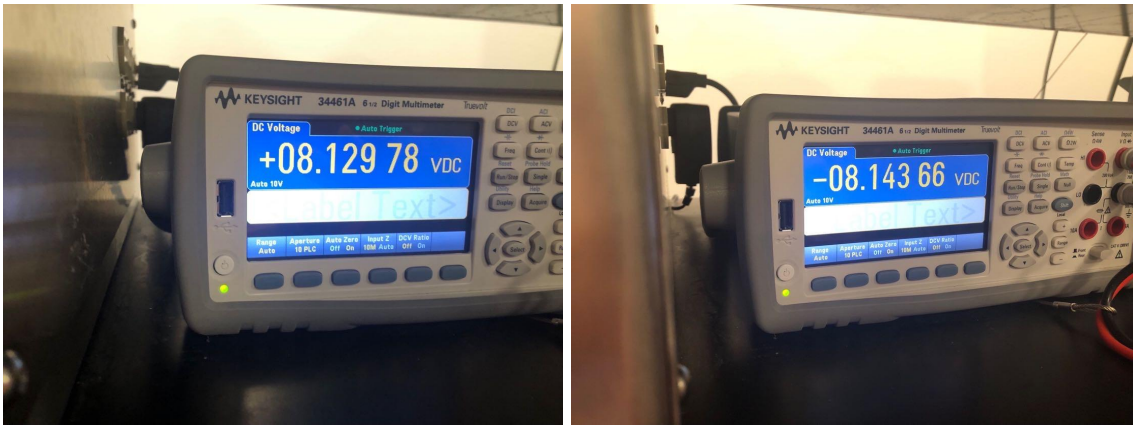

Figure 11. Battery voltages before (left) and after (right) charging for a short period of time

3.2 Sensor Subsystem

As we were unable to program the pcb microcontroller, we decided to run the code we had for the camera on the arduino uno. Using the arduino uno allowed us to connect it to a laptop and display the images we were taking with the camera, which can be seen below in figure 12.

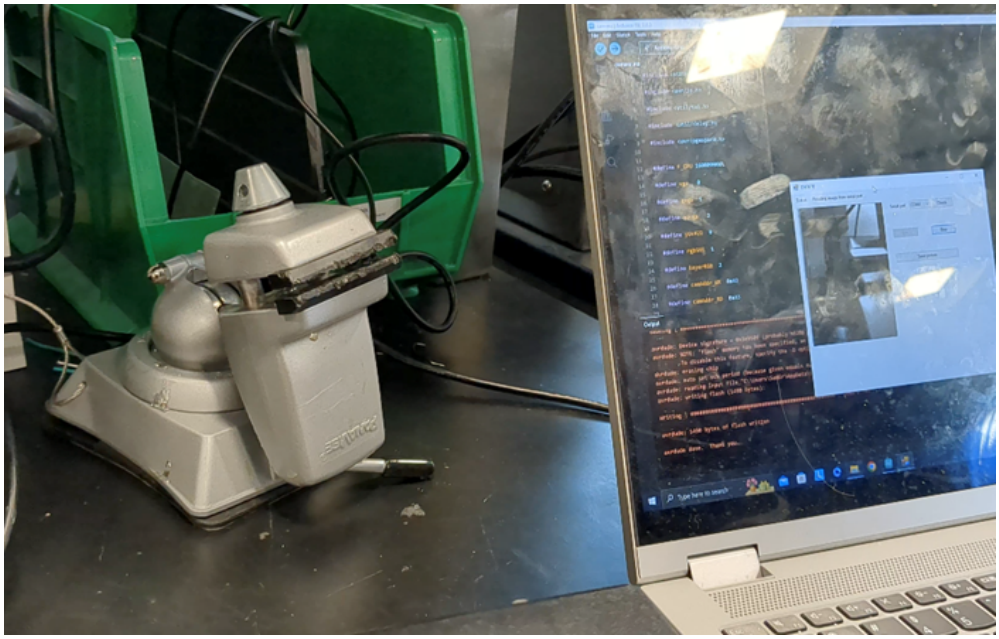

Figure 12. Laptop receiving images taken by the camera using the arduino uno

3.3 Control Unit Subsystem

A dataset of images of single characters (A-Z, 0-9), labeled by folders was acquired to test the accuracy of the OCR. A script crawled through the dataset and fed each image through the OCR engine without any preprocessing to establish a baseline accuracy of 67%. The accuracy was calculated by a simple division (# of images accurately labeled / # of images fed through total). In order to meet the high-level accuracy requirement of 90%, three preprocessing methods were integrated using the open source OpenCV image/video processing library. Grayscaling changed each of the pictures to pure black and white and increased the accuracy by 3%. Noise removal got rid of unwanted specs that weren't in the body of the text character and increased the accuracy by 5%. Two methods of thresholding, simple binary and Otsu's binarization were used to dramatically increase the contrast of the image, making the text character better visible against the background. Simple binary set a manual value while Otsu's binarization calculated an ideal thresholding value based on the information from each image. Both thresholding methods working in tandem increased the accuracy by 14%.

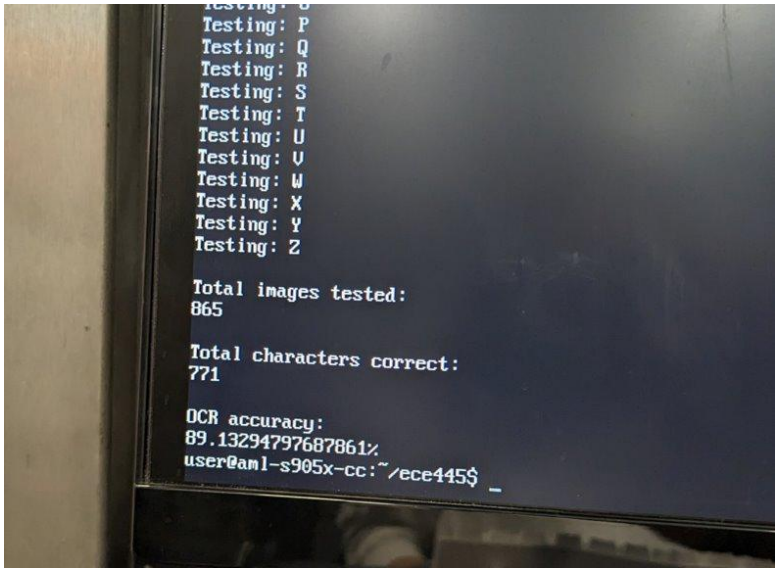

Figure 13. OCR accuracy results calculated on microcontroller

3.4 User Interface Subsystem

To test the user interface system, we soldered wires to the nodes where the outputs of the pcb microcontroller connected to the RC circuit. These wires were connected to an Arduino Uno which was programmed to continuously output each letter of the alphabet, one letter at a time. Both the arduino uno and the solenoids and the Arduino Uno were powered by our battery and we were able to get each letter to output correctly. An image from the test can be seen in figure 13 which shows the letter Q with the solenoids.

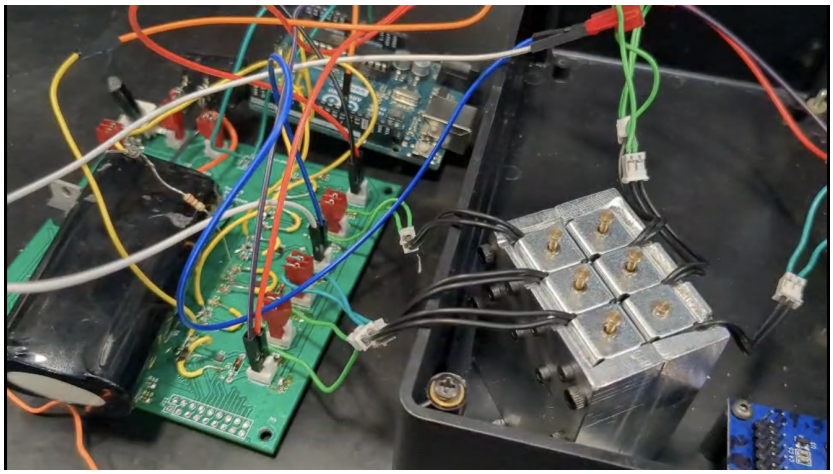

Figure 14. Letter Q displayed on solenoid

4. Costs

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

4.1 Parts

The table below shows most of these electronic components and their prices:

Description	Manufacturer	Part #	Quantity	Total actual cost ($)
Linear Actuator	Sparkfun	ROB-11015	6	29.70
Atmega32u4 (microcontroller on the PCB board)	Atmel	32u4	1	5.22
Arduino Uno (microcontroller for PCB board)	Arduino Uno	A000066	1	27.60
Le Potato Microcontroller (microcontroller 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. Parts cost

4.2 Labor

To carry out this project we would expect a salary of around 30 $/hour.
Each team member will work an estimate of 100 hours on their own to complete their work.
That means that we would expect a total salary of around 30 $/hour * 2.5 * 100 hours = $7,500 per team member.
If we multiply this salary times the number of people in the team we get a total amount of $7,500 * 3 people = **$22,500 in labor cost**.

Work Type	**Samuel's Hours**	**Abraham's Hours**	**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

Table 2. Hours spent by each team member on each work type

The machine shop in UIUC will be used while carrying out this project. According to their webpage [3], their cost is $38.17/hour. After talking with the machine shop and giving them our parts, we estimate they took about 6 hrs from those couple of days to complete our project.

Therefore, the total cost for the UIUC machine shop should be $38.17 * 6 hours = **$229.02.**

The total parts cost, as seen on table 1 is **$167.97**

Therefore, the total cost of the project is equal to:

$$Total\ cost = labor + machine\ shop + parts = \$22,500 + \$229.02 + \$167.97 = \$22,896.99$$

5. Conclusion

5.1 Conclusion

In the process of developing our project our largest obstacles were in implementing full system integration. We fell short on this aspect of the project and therefore were unable to produce a device that accomplished our initial goal of displaying physical text in physical braille. However, each subsystem met most, if not all, subsystem requirements. We were able to demonstrate functionality from each of the individual subsystems.

5.2 Ethical considerations

We followed the IEEE Code of Ethics [2] in the development and design of our project. We firstly focused on having good teamwork. We treated everyone in an equal manner and with respect. Everyone had access to all the information that was used in the development of our project, and everyone was kept up to date with the progress of the project. We placed a large importance on the safety of each team member and the user by minimizing the safety risk of the battery we used for our device. The safety concerns were addressed through the design of the charging system, slowing the rate of charging, and the way that the battery was housed to avoid any damage or puncture to the battery. In the pursuit of aiding the visually impaired, we recognize the ethical concerns of providing accurate information to the users of our device. We recognize that a user may depend on the accuracy of the data that is read in order to inform their decisions in their life, potentially life threatening decisions. To address this we wholly maximized the accuracy of our device as best as we could.

5.3 Future work

The first step in any future work for this project would be to achieve integration of all of our subsystems to provide a working product. This would start with a redesign of the PCB layout, as we made errors with connecting grounding planes, which we believe caused us to be able to program the microcontroller.

After achieving a working prototype we would want to work on improving the camera sensor subsystem. Our current camera has a max resolution of 640x480 which makes getting images that we can get characters difficult. Our camera speed as well is also unimpressive as it takes around 20 seconds to get an image, which is a lot slower than we want it to be.

References

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

[2]. IEEE. “”IEEE Code of Ethics”.” (2020), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (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 (visited on 02/23/2023).

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

[5]“Solenoid Driver Circuit,” *circuitdigest.com*. https://circuitdigest.com/electronic-circuits/solenoid-driver-circuit-diagram (accessed May 03, 2023).

[6].“How to add USB-C to your projects - PCB Design Tutorial - PCBway,” *www.pcbway.com*. https://www.pcbway.com/blog/PCB_Design_Tutorial/How_to_add_USB_C_to_your_projects.html (accessed May 03, 2023).

[7].Punkisnail, “Li-ion Battery Charger,” *Instructables*. https://www.instructables.com/Li-ion-Battery-Charger/ (accessed May 03, 2023).

[8].“How to Use OV7670 Camera Module with Arduino Uno,” *Circuitdigest.com*, 2019. https://circuitdigest.com/microcontroller-projects/how-to-use-ov7670-camera-module-with-arduino

Appendix A: Requirement and Verification Table

Requirement	Verification	Verification status (Y or N)
The Voltage regulator will limit the Voltage to the correct value for each system component.	Use an oscilloscope to measure voltage through voltage regulators under normal operation. Expected measurement should be 3.3±0.1V and 5.0±0.2 V	Y
The Power Management Subsystem will be able to safely charge the battery	The voltage regulator increases the voltage to 7.2 V 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.	Y
The camera must be able to take at least a 480p resolution photo and send the data to the Control Unit system whenever the take photo button is pressed.	The OCR testing software receives 480p resolution photos on photo button press.	Y Used laptop with serial connection rather than button
The ML algorithm on the 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 ML algorithm will be run against this dataset to determine its accuracy.	Y
The character text data is converted into signals that are sent to the motors to form Braille characters for all 63 Braille characters	Test software will be written in order to replicate converted signals from the ML algorithm. All 63 Braille character signals will be sent to the motors one by one and the correct formation will be verified by eye.	Y
The Motors must be able to lift the pins 0.35±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.	Y

Table 3. System requirements and verifications

Appendix B: Camera 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); //Earlier it had the value:writeReg(0x11, 12); New
```

```
version works better for me :) !!!!
}

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) + "%")
```