Should Just Work/The Lonely Robo

Wade Hisiro, Joshua Laney, Nicholas Mohammad, Christopher Truong, Sean Wolfe

December 16, 2019

Capstone Design ECE 4440 / ECE4991

Signatures

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Statement of work:

Wade Hisiro:

My main task was designing the embedded software for the MSP. This included setting up system clocking using an external crystal and interacting with the go button, LED indicator, motor drivers, limit switches, and solenoid and l. This work included initializing them appropriately as inputs/outputs and setting up declarations to set their outputs high and low. My job also included setting up UART communication on the MSP and I helped Chris set up and debug the UART on the Jetson Nano. Furthermore, I designed the protocol that the system used for communicating between the MSP and the Jetson, which consisted of the MSP sending a reserved word to the Nano to let it know it was ready for the word search. Then, the Jetson would send a byte for each x and y starting and ending locations. When the MSP was done highlighting, it would send the reserved byte to the Jetson to indicate it was ready for another. This process continued until the Jetson no longer had any coordinates to send, so it would instead send the reserved byte letting the MSP know the word search has been completed. Additionally, I had to write the code to achieve this process.

Most of my other work dealt with defining the MSP coordinate system and the motor movement. I had done calculations to define the number of steps between coordinate points to allow for the maximum precision based on the max coordinate size (limited by a byte for UART) and the linear distance travelled per step. However, this had to be modified after testing revealed more steps had to be taken per point in order to make the motor movement more consistent. Additional work included creating methods for moving the motors to various XY locations, homing, and highlighting words. I also worked with Josh to convert real-world distances into steps so the Jetson could properly direct the MSP and for testing the various components after they were soldered onto the PCB. Additionally, I collaborated with Nick and Sean on ideas for the highlighter holder.

Joshua Laney:

My main task was designing the full electrical system. I first picked all the main components including the specific MSP, stepper motors, and motor drivers. The stepper motors were chosen because they were designed to work with the mechanical XY table form OpenBuilds. The motor drivers were chosen to support the voltage and current requirements of the motors, and because they are from the same manufacturer, and use the same method of communication as the motor drivers in the Introduction to Embedded course. The MSP was then chosen to provide enough GPIO pins for the motors, as well as other peripherals like the solenoid, limit switches, go-button, and the UART communication. To support the functional components, I developed four different power networks. 12V for the motors, and the LED lighting strips, 5V 5A to serve as a dedicated power supply for the Nvidia Jetson, 5V 2A to power the solenoid, and 3.3V 2A to power the MSP, and to server as logic level for the switches and motor drivers. With the components selected I then laid out the schematic in Multisim, designed the PCB in Ultiboard, and solder the parts onto the board.

In addition, the designed the electrical side of the project I served as an extra set of hands in assembling the XY table form open builds, helped write some of the computer vision pipeline, and helped debug some of the embedded code. The main part of the software pipeline I worked on was related to converting the location of points in the image, to real world distances, and then from real-world distances to steps the stepper motors should take. This was done by using a set of five fiducial points on a sheet of paper, and hard coding the relative distance (in centimeters) between them. To convert real world distance into steps, the pen was programmed to draw lines of varying step counts, and their distances measured to produce a measure of cm/step.

Nicholas Mohammad:

My primary task was to serve on the mechanical system team, specifically in the specification and construction of the XY table. The project required a relatively precise mechanical highlighting system with at most +/-.5 cm accuracy, otherwise the highlighting would be noticeably off. Also, the table also needed to allow enough space on the chassis to mount the camera, and a plate that would allow for easy mounting of our pen-solenoid holder. Finally, the table also needed to accommodate the space for a standard A4 sheet of paper while providing enough room for the home position of the pen holder to be out of the camera's field of view. With these constraints in mind, I found a standard kit supplied by OpenBuilds that was large enough to fit our spatial needs, came with motors that provided tolerances well within our desired range, and an easy-to-use mounting plate for our pen holder. Once the parts came in, I worked with Josh to assemble the table and construct a mounting setup for the camera.

My secondary task for the project was software, primarily in regards to puzzle solving, increasing word bank accuracy, identifying and rotating character bounding boxes, and line drawing for the XY table. I translated the CS2150 word puzzle algorithm to python code that could be easily integrated with the main pipeline. Furthermore, I added a permutative solution feature, where words not found from the scanned in word bank would be run through a spell checker and letter swap out table in order to combat read-in errors. As for computer vision, I worked with Chris and Josh to wrap characters in bounding boxes to identify their centers. We also worked on transforming those center points through rotations, scaling, and translations in order to calculate their proper image coordinates. Lastly, I wrote the Bresenham line algorithm in the embedded code that was responsible for getting the motors to highlight a word.

Christopher Truong:

My primary task was designing and implementing the computer vision and software pipeline for imaging, parsing and solving the word search puzzle. More specifically, I worked on the image processing steps which were necessary to clean up our input, the tuning of Tesseract to improve accuracy, image calibration, writing the code to transform image to world coordinates, tuning image to world calibration, and writing the Jetson side UART functionality. All of these functions were important to the project as a whole because they comprise the core functionality of the system - solving a word search puzzle.

Sean Wolfe:

My main task involved designing custom 3D-printed parts for our system. The first of these two is the "connector", a part that connected the highlighter (used for solving the puzzle) to the solenoid that lifts the highlighter off the stage vertically. This task required copious amounts of precise measurements to make sure that two parts were properly joined by the part. Because of the part's small size (~.6 cubic inches in volume), high quality calipers were needed to acquire the proper dimensions of the design. In addition, the design needed to allow for the highlighter to be replaced in the case that it runs out of ink. To overcome this, I researched several approaches to this issue before settling on the use of a collet that can collapse around the end of the highlighter and fastened with a small zip-tie.

The second part I designed is the "holder". This structure holds the highlighter-solenoid fixture and can be fastened to the XY-table. The holder also includes a shaft meant to mitigate the jostling of the highlighter as it moves across our XY-grid. To design this part, I consulted numerous CAD files of the aforementioned XY-table to make sure that its dimensions for mounting were aligned with our XY-table. Consulting these files allowed me to guarantee that the size of the holder would not hamper the camera's view of the sheet of paper containing the puzzle and word bank. In addition, I found a way to house the heavy solenoid within the holder with using walls. These walls prevent the jostling of the solenoid during the completion of the puzzle.

Table of Contents

Table of Figures

(This should list the page of each figure used in your document, including the full caption.) Word has tools to help you do this very easily)

Abstract

We built a word search solving robot. By taping a word search down to the bed of the machine, an automated highlighter is used to highlight the location of each word in the word bank on the puzzle grid of letters. This is done by taking a single picture of the word search, and using computer vision and optical character recognition (OCR) to solve the puzzle. From there the solution is mapped back into physical space, and the start and end coordinates of each word are sent to a microcontroller that controls the actual highlighting of the words by using stepper motors to position the highlighter and a solenoid to pick up and drop the highlighter.

Background

In the process of deciding on a capstone, we were inspired by a project we saw in a previous semester - the Scrabble playing robot. Such a project was intriguing because it presented challenges across the spectrum of engineering, as robotics problems tend to do. We felt that this idea captured the essence of computer engineering in a project that was feasible to complete in a single semester, so we generated a project that was similar in principle, but less burdening in the mechanical aspects.

While this project is a derivative of other puzzle playing robots, the particular task we chose seems to be unique, at least in the scope of things that have been published on the internet and scraped by Google. Speculatively, the reason being that this word searching task hasn't been done is that it is either too trivial or too useless a task -- much more advanced robots have been created in the past, but that would be outside of the scope of this class.

For our particular implementation, we decided to choose a different route in terms of computing hardware selection. As the primary method of computation, we plan on using an NVIDIA Jetson board, in conjunction with a separate board to handle control of the stepper motors and limit switches. The Jetson board allows us to run Linux and use all of the software and libraries available therein, allowing us much more flexibility compared to something like the myRIO.

This project draws on concepts from Intro to Embedded/Advanced Embedded, CS 2150, Computer Vision, and the FUN series. We'll utilize embedded programming techniques, a fast word search algorithm implemented in CS 2150, concepts from computer vision, and circuit design methods learned in FUN 1 to FUN 3.

Constraints

Design Constraints

The design of our PCB was constrained by several factors. There were only 1-2 chances per month during the project to submit the design of our board to the manufacturer. This gave our team little room for error in terms of the electrical design. This design constraint also applied to the submissions of our CAD files for 3D printed parts. In addition, our first PCB took two weeks to arrive at our workstation. This long turnaround time added to the stress level of our project and halted our ability to test the system as a whole until the last few weeks of the semester.

Economic and Cost Constraints

Our only economic constraint is that our professor required that all electrical components be ordered through Digikey or Mouser Electronics. While this stipulation constrained the parts we could use for our system, it never caused a major issue. The cost constraints of the project prevented our group from using slotted through-holes on our PCB. The issues caused by this constraint led to our team having to manually solder jumper wires to a power connector.

External Standards

Our system requires a Type 1 NEMA standard enclosure. This standard regards electrical connections that are potentially hazardous for an indoor setting. We adhered to this standard by using an enclosure for our circuit board and Jetson Nano when all members of the team were absent from our workstation [1].

The robot was powered off of converted AC line voltage. To properly handle this voltage, a desktop, class 1, AC to DC power convertor was used. It featured a NEMA 15-5 class connector [2]. With our system's large power supply, we knew that safety could be a concern.

The PCB was designed adhered adhering to the IPC-2221 family of printed circuit board design standards [3]. These standards dictate design rules such as minimum trace thickness, as well as trace spacing. These helped ensure strong connections without noise. For our system's PCB, we made sure that all of the components were spaced safely about our board. For example, the connection for our power supply is placed on the opposite side of the circuit that controls the high-powered solenoid. Meeting these safety standards also helped us prevent mistakes in implementation.

Tools Employed

On the embedded side, Code Composer Studio 9.2 was used for writing the C code used on the MSP [4]. Code composer was a valuable tool because its debugging tools helped a lot throughout the project to understand and fix undesired behavior. The debugging functionalities were initially unfamiliar, which provided a good for learning a lot about how to use them. Additionally, a NI VirtualBench was used with a MSP430FR2476 Launchpad to measure the actual pin outputs and simulate inputs before all of the components arrived [5][6]. This was especially useful for confirming the clock frequency after setting it up with an external crystal, the PWM period, and the UART messages and baud rate. Additionally, a TI MSP-FET Flash Emulation Tool was used to flash the C code onto the MSP on the PCB to run on the actual system. Lastly, Microsoft Excel was used for calculations related to the clock system and configuring the coordinate system.

Electrically, the overall schematic of the board was designed in Multisim,[7] and the PCB was laid out in Ultiboard [8]. For every component in Multisim, a library model was created containing the number of pins, a description, a link to digikey, and the cost of the component. These models were then linked to their associated PCB footprints in Ultiboard. The required skills to use these tools was learned throughout the ECE fundamentals series. PCB design, best practices, and component selection were learned through a summer design internship.

On the software side, all of the code was written in Python3 with the text editors our choice. We used OpenCV 4.1.1 and Google Tesseract OCR 4.1 as our vision library and optical character recognition engine respectively [9][10].

All 3D printed parts were designed using SolidWorks 2019 [11]. LinkedIn Learning's "SolidWorks 2019 Essential Training"course was consulted regularly to learn the basic principles of mechanical design and SolidWorks [12].

Ethical, Social, and Economic Concerns

Environmental Impact

Our product has some negative environmental impacts, with the most significant being that it requires electricity as opposed to a human filling out the word search. The level to which this matters depends on the energy source of the power, but it is considerably more likely that the word search solver has a non-zero carbon footprint. Some steps were taken to reduce power consumption when the device is not in use, but admittedly more could have been done such as turning off the LED strips when the system was inactive.

Some of the other negative aspects are that it needs to use paper and highlighters in order to work. The robot required quite a bit of paper for tasks like testing and calibration while it was being created and it now requires a sheet of paper in order to run. Paper usage is not particularly environmentally friendly as it requires chopping down trees, pollution and energy in order to create it, and can make help fill landfills and produce methane if not recycled [13]. In order to limit some of these effects, all of the paper used so far has been recycled and all future paper should be recycled as well. The highlighter is a consumable made of non-recyclable materials, further filling landfills with plastics and other non-biodegradable materials. The highlighter should be capped when not in use in order to maximize lifetime and reduce the number of them required. Additionally, some tape is used to hold the paper down, but this should not amount to much.

For the other materials, the design is better in regards to the environment. Some of the used parts were recycled from previous semester's projects. Additionally, the metal parts from the chassis could easily be used for future projects. The electrical components used have a small negative impact from chemicals and energy consumption while being manufactured. For this prototype, the effects are minimal but they might become significant if it were to become mass produced as many more of these components would be required.

Sustainability

The robot is not particularly sustainable but not necessarily unsustainable either. For device usage, it consumes power, highlighters, and paper. The most significant of these is the power consumption when the device is active. The pen actuation in particular draws excess power continuously in order to hold the highlighter above the paper. However, the solenoid would most likely consume less power than another motor would for pen actuation because the motor would draw current while in the process of highlighting while the solenoid does not. Additionally, once the puzzle is done, the highlighter can be taken to the side and just dropped so it doesn’t need to continue consuming power when the machine is not in use. The paper can be recycled once finished, thereby increasing the sustainability of this part. Unfortunately, the highlighter can not be easily recycled.

The materials used for production were also mixed in regards to sustainability. While the production of metal in general is not sustainable, it is mostly reusable. The aluminum used for the chassis can be reused in future projects. Additionally, other components like the camera, which was already secondhand, or the motors can also be repurposed. Some of the electrical components on the PCB may be able to be salvaged, but most will not. The plastic used for the 3D-printed highlighter holder is not very sustainable and can not be easily recycled.

Health and Safety

The robot does not pose any serious health risks as minimal human interaction is required. When human interaction is needed to change out the sheets of paper when the robot is done there is the potential for pinch points in the XY table. The heat from the solenoid, control board, and Jetson could cause burning hazards if left unchecked. Highlighters also emit fumes that could be toxic, but in such small quantities it should not pose any serious risk.

NEMA 5-15 or 5-20 for the AC wall connector [2]. The product will use standard 102V 60Hz and will have the appropriate fuses and kills switches in case of a large current pull.

Fumes from the highlighter pose a potential threat but should be well below OSHA limits (1926.55 App A) [15].

Potential pinch points with motors and the metal frame could pose a problem if someone puts their hand in the mechanism while its operating or when somebody is changing the paper (OSHA 1910.211(d)(44)) [16].

Manufacturability

The product seems to be relatively easy to manufacture based on the prototype when not considering the cost. The designs have already been made, so they would only need to be replicated. When taking the cost into account, the manufacturability of it becomes much more difficult. The total system cost is non-trivial and the system would need to be redesigned in order to lower costs.

The product used standard metal extrusions, belts, and gears from the same source (OpenBuilds) which aided in the assembly of the mechanical unit [17]. However, the product requires a custom mount for the highlighter holder that needs to be 3-D printed. The actual assembly is fairly quick as the system was able to be built by a team of two people in under two hours using a kit from OpenBuilds. The main drawback is that these mechanical parts cost over $300. The bulk of this comes from the chassis so alternative system designs will need to be considered.

The PCB has already been designed so it would only need to be reprinted and to have the components soldered onto it. The PCB used large parts and standard human-solderable packages (through-hole, 0806, dip-8 etc) as well as standard connectors (100mil pitch) which aided in the final assembly. The most difficult part to solder is the 48-pin MSP, which while tedious, is possible to hand solder. A pick and place machine could also be used to solder a majority of the parts, leaving only a few through-hole components for hand soldering. All of the electrical components used are standard and inexpensive.

The software has already been created, so it does not need any additional manufacturing. However, the Nvidia Jetson Nano is quite expensive and a replacement to a cheaper GPU or processor would need to be considered [18]. This switch would potentially require changes to the code which would take time, but be replicable after that.

Ethical Issues

There do not seem to be any large ethical issues with our project. Since the robot is only being designed to solve a word search, it is not displacing any workers and would not contribute anything meaningful to the world. The main ethical issues associated with the project come from the supply chain. This includes its carbon footprint as well as possible working conditions of those making the various parts. While an attempt was made to limit the power consumption, it still consumes a decent amount of power when operating. Since the project serves no real-world purposes and it uses electricity to accomplish something for fun, some might see it as a waste. Although a camera is used, it does not raise privacy concerns because it is always pointed towards the word search when in use and it is always being monitored during operation. Apart from these minor concerns, there seem to be no reason why any group of people, animals, or the environment should be harmed by this project.

Intellectual Property Issues

Our project is unique in its application and combination of processes, and thus has a high potential to be patentable. At its core it is a robot that controls a writing utensil based on visual input, which in itself is fairly unique, and the application of solving word searches is completely novel. In the broader sense of our project, it relates to an Intel patent “Calibration system for vision-based automatic writing implement”, specifically the 18th claim which cites [19]:

> An apparatus comprising:
> an articulated robot arm for drawing a specialized target on a substrate and for drawing or writing secured to a base;
> an image sensor for finding and recognizing the target; and
> a controller, having a storage medium, including a rule set to interact with a user and instructions to manipulate the robot arm,
> to receive and process an image from the image sensor;
> to determine an action based on the rule set and the image; and
> to move the robot arm based on a determination of the image based on the rule set.

Our robot uses a rule set to move a robotic arm to draw on a substrate based on the input from an image [19]. Where our robot differs from this claim is that our robot draws lines, not a "specialized target", and it's vision is thus not dependent on that target, but is instead dependent on the format of the word search it is solving.

The common device most similar to our project is the pick and place machine, which relies on a vision system that translates visual coordinates into real world physical coordinates, which then control a head through xy space. There are many patents related to the pick and place machine, such as the Panasonic "Apparatus and a method of mounting electronic components" [20]. The first claim of this patent states:

> An apparatus for mounting electronic components comprising:
> a transfer device for transferring an electronic component to a substrate;
> a camera for observing identification marks formed on at least four positions on the substrate;
> a position determining unit for detecting coordinates of the identification marks based on a result of the observations of said camera;
> a coordinate data storing unit for storing data of coordinates of a mounting position at which the electronic component is to be mounted;
> a positional correction arithmetic unit for calculating a correction value for coordinates of the mounting position for each of a plurality of combinations of three positions selected from among the at least four positions of the identification marks, based upon coordinate data for each of the selected three positions, and for obtaining a mean value of the correction values for use as a positional correction value; and
> a control unit for controlling said transfer means according to programmed data for the mounting position and the positional correction value.

Our robot also depends on a camera observation of position points, and calculates coordinates and correction s based on those points. The difference is that our system does not need the fiducial points to be on the word search, and instead references a previously taken calibration image, and relies on the assumption that the camera remained stationary in position. The claim also focuses on the transfer of an electronic component onto a substrate instead of on marking a page. While the method for doing so are similar (the pick and place using a suction head instead of a highlighter), the claim on the patent is too focused to conflict with our robot.

A significant component of our project is the XY table that moves the pen across the page below along with an attached solenoid responsible for raising and dropping the pen. At an abstracted level, our robot can be thought of as a device that draws on a page through the use of a mechanical system, just like General Electric Co's robotic pen [21].

The 7th describes several of the pen components and their functionality. Note that this is an independent claim because it does not explicitly reference and rely on descriptions set forth in any previous claims. It is as follows:

> A robotic pen comprising:
> a computer numerically controlled machine including a stage for mounting a workpiece for rotation and

orthogonal translation, the said stage permitting translation generally in a plane and rotation about an axis
generally parallel to said plane, and an elevator for translation from said stage;
a pen tip rotatably mounted to said elevator; and
a dispenser joined in flow communication with said pen tip for ejecting a stream of material atop said work piece.

Indeed, our robot also draws on whatever is put within the table through the assistance of a computer / microcontroller. However, General Electric Co's robotic pen does so by moving the paper in three dimensions with the pen stationery. In order to draw anything, the paper needs to be mounted onto a plane that will rotate and translate it to achieve the desired strokes. Our project differs in that it moves the pen to draw lines and keeps the paper stationary. The overall effect is similar, however, the method in which the task is achieved is significantly different. Given this difference, it is safe to say General Electric Co's robotic pen wouldn't keep our robot from being patentable due to the dissimilarity in how each robot goes about drawing on a sheet of paper.

Detailed Technical Description of Project

The product described in this report is a robotic word search solver. The project works by using computer vision to detect the word search and identifying the words in the word bank and the letters in the grid. From here, it solves the word search using these inputs and maps the letters to coordinates in the puzzle area. For each word in the solution map, the MCU by command of the Jetson will: (1) ensure that the highlighter is off the paper, (2) move highlighter to the start cell, (3) drop the highlighter, and (4) move the highlighter to the end cell. Upon completion of the puzzle, the MCU will ensure the highlighter is off the paper and use the LED indicator to show that the process has been completed. Figure 1 below shows the process flow for the entire system.

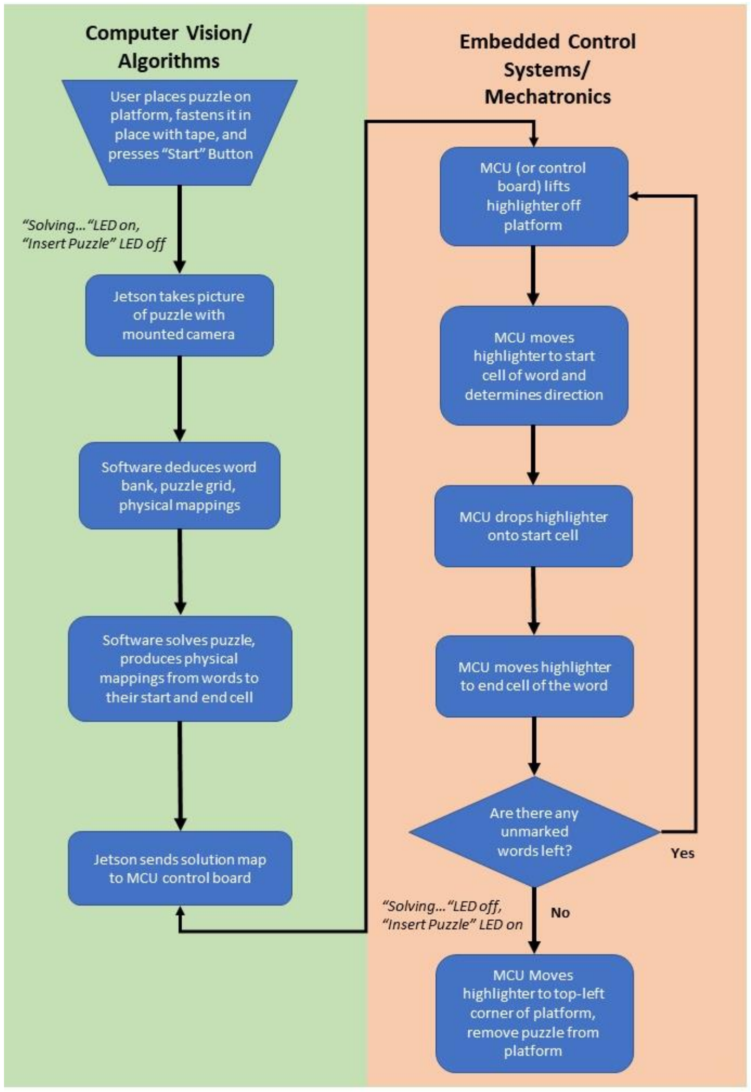

Computer Vision/ Algorithms
Embedded Control Systems/ Mechatronics
User places puzzle on platform, fastens it in place with tape, and presses "Start" Button
"Solving..."LED on, "Insert Puzzle" LED off
Jetson takes picture of puzzle with mounted camera
Software deduces word bank, puzzle grid, physical mappings
Software solves puzzle, produces physical mappings from words to their start and end cell
Jetson sends solution map to MCU control board
MCU (or control board) lifts highlighter off platform
MCU moves highlighter to start cell of word and determines direction
MCU drops highlighter onto start cell
MCU moves highlighter to end cell of the word
Are there any unmarked words left?
Yes
No
"Solving..."LED off, "Insert Puzzle" LED on
MCU Moves highlighter to top-left corner of platform, remove puzzle from platform

Figure 1 Software Flow Diagram

Electrical

Power management for the various subsystems will be handled by a custom printed circuit board (PCB). This PCB was created according to the block diagram shown in Figure 2 that illustrates how the various components of the system are related to each other. The dashed lines show which parts are powered by the PCB. The solid lines and direction arrows indicate which components are inputs to other parts and how they need to be connected electrically.

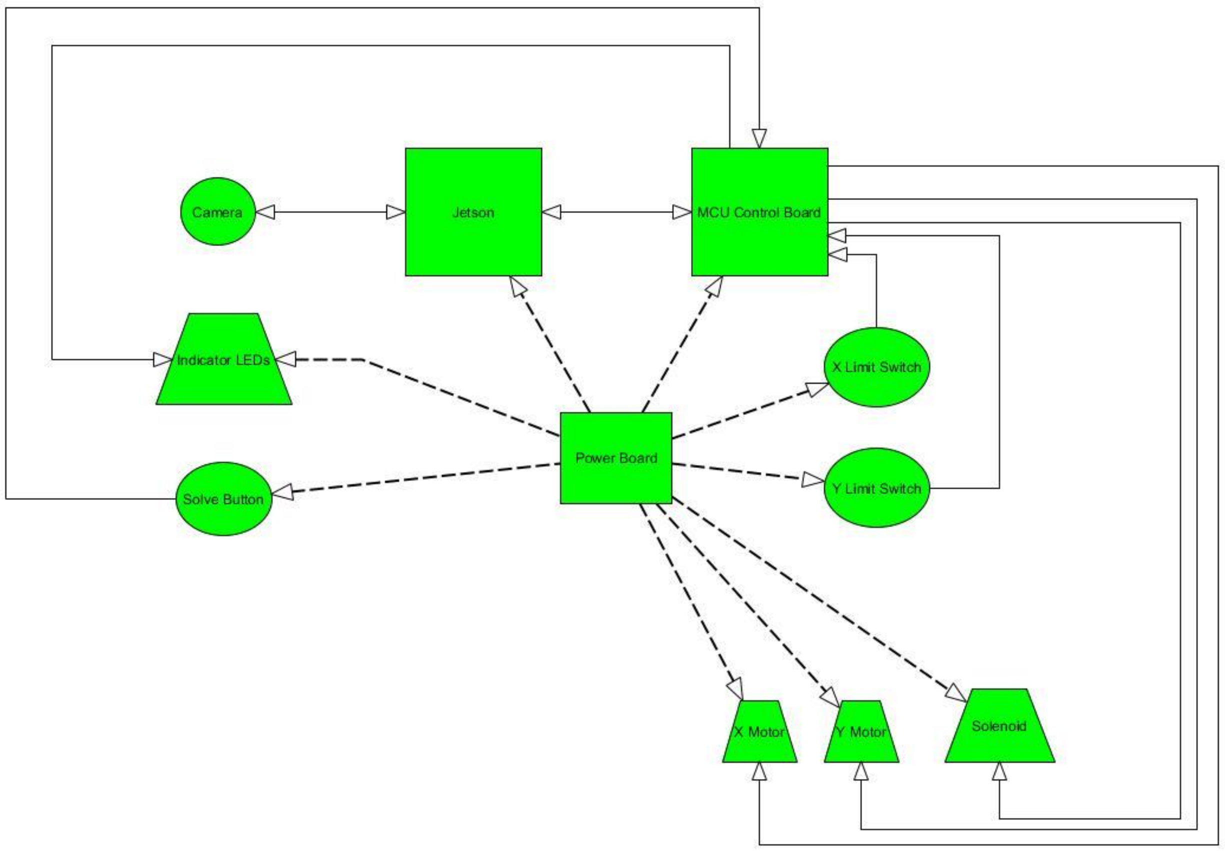


Figure 2 Electrical Block Diagram

Electrically, the whole system is powered by 12V 150W Desktop Class 1 AC to DC converter from XP Power [22]. This connected to the PCB through a 4 pin mini-din connector , which directly fed into a 15A fuse and a green power indicator LED [23][24][25]. As seen in Figure 3, the green LED has a 2.1V forward voltage drop so a 510Ω 0805 resistor was chosen to limit the current through the LED to ~20mA.

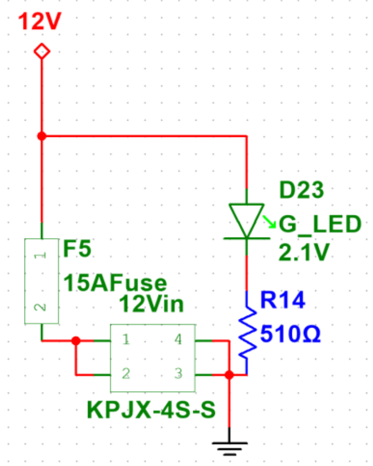


Figure 3 12V Input

The 12V line served as the backbone of the boards power system, and fed three different linear DC to DC regulators, the power for the motor drivers, and the page lighting LED strips.

The LED strips were chosen to be bright 6000K white LEDs to best evenly illuminate the word search and wash out as many shadows as possible. The LED strips connected directly to the 12V power rail, using a Wurth 2 position terminal block header [26] (Figure 4) Wurth terminal block headers are used for all offboard connections (except for the power input) [27]. They were chosen because of their easy to connect and disconnect plugs that conveniently screw down to clamp onto the connecting wire [28]. Also included in the LED strip connection was a 12V TVS to help ensure a clean signal to the LEDs and help protect the circuit from any ESD coming from the connection to the LEDs [29].

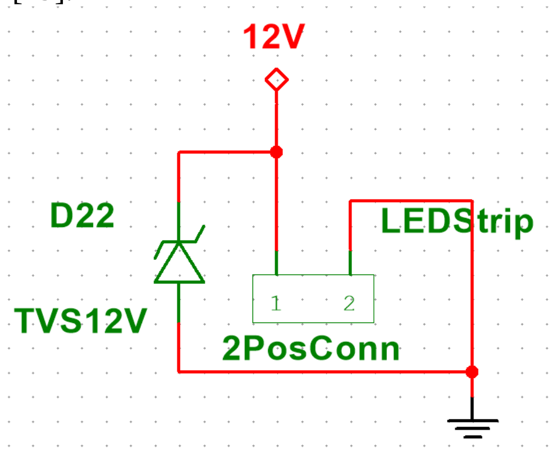


Figure 4 Connection to LED Strip

The first linear regulator replacement DC to DC convertor takes the 12V line down to a 5V 5A line (Figure 5) [30]. This 5V supply was then directly piped off board (Figure 6) and served as the dedicated power supply for the Nvidia Jetson Nano, connecting to the Nano through a barrel plug [31]. Linear regulator replacement DC to DC convertors were chosen to simplify the design process as all the supporting components to the power regulating IC come pre designed and assembled, making the whole power conversion process an easy to use through hole package, requiring only a few specified bypass capacitors. The regulator for the Jetson had to be tuned to 5V using a 1.47kΩ tunning resistor as specified by the data sheet. A 5A PTC fuse was placed on the input to the regulator, and a 5V TVS, and indicator LED were placed on the output [32][33]. The indicator LED was the same one used for the 12V input, but required a 150Ω resistor to set the current.

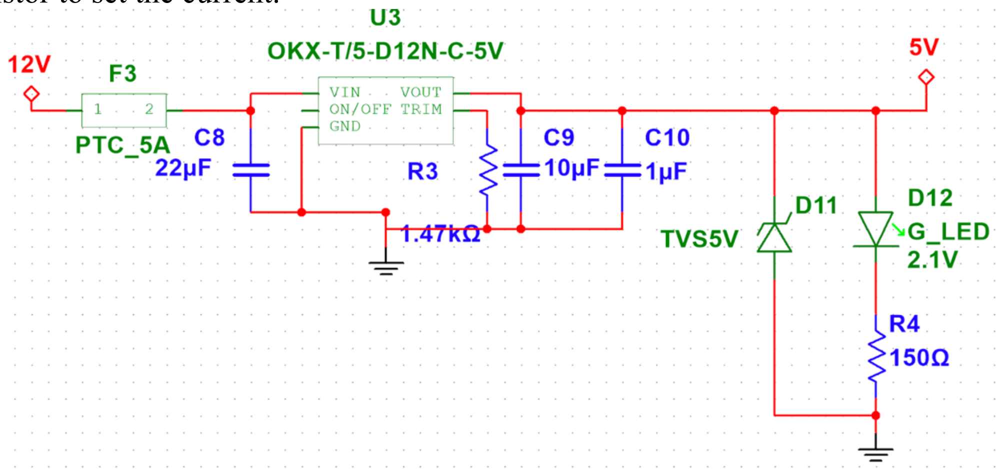


Figure 5 5V 5A DC to DC

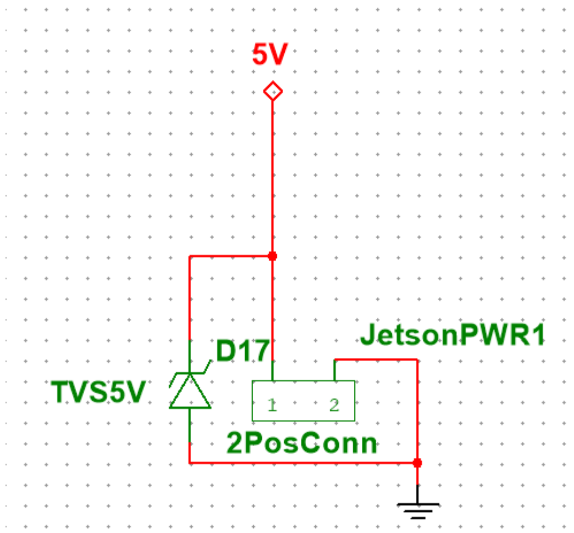


Figure 6 Nvidia Jetson Power Connection

The second linear regulator replacement DC to DC convertor again takes the voltage down to 5V, but this time with only 2A of accessible current (Figure 7)[34]. This regulator was used to power the solenoid. The Jetson power and solenoid power networks were chosen to be separated to avoid any power competition issues between the two, as well as any issues caused by possible back EMF produced by the solenoid. Here a 2A PTC is used on the input, and the same indicator LED is used with another 150Ω resistor limiting its current [35]. The regulator used was not adjustable and therefore only required a couple bypass capacitors and no trim resistor. A 5V TVS was also placed to help keep a steady 5V level.

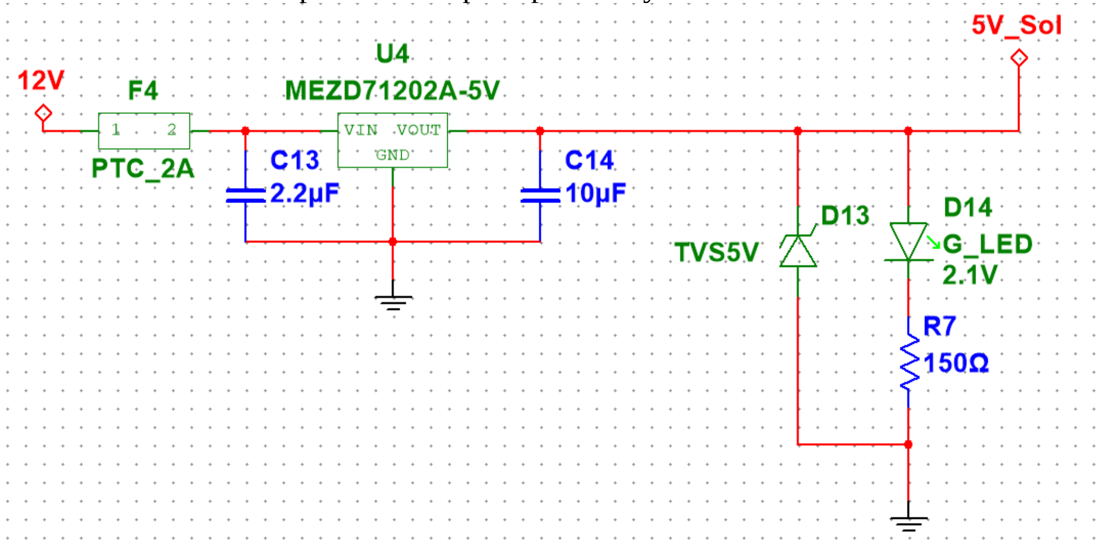


Figure 7 5V 2A DC to DC

The final linear regulator replacement DC to DC convertor drops the voltage down to 3.3V 2A (Figure 8) [36]. This is used as the power and logic level for the microcontroller and the logic level for the motor drivers. The supported circuitry for the convertor is almost identical to that of the 5V 2A convertor, as they are both from the same manufacturer. Here a 3.3V TVS is used, and a 68Ω resistor is used to set the current through the indicator LED [37].

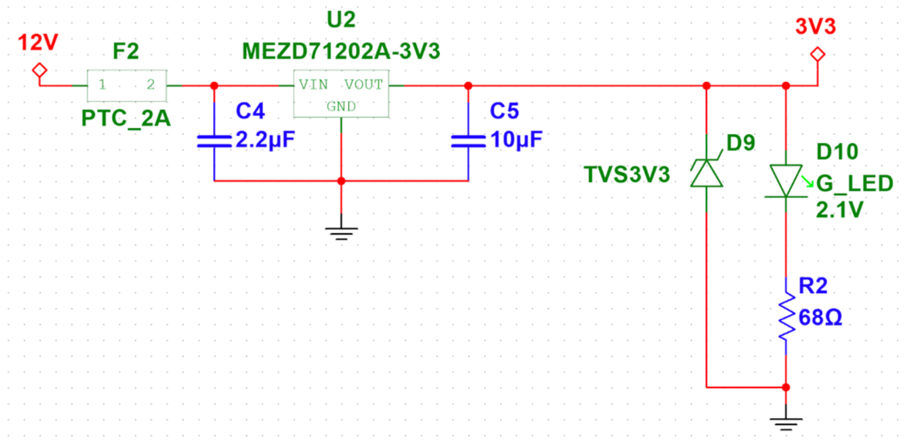


Figure 8 3.3V 2A DC to DC

The heart of the circuit is the MSP430FR2476TP [38]. It was chosen because of the large number of GPIO pins, flash storage, and RAM available, leaving lots of head room and allowing for liberal use of resources. Supporting the MSP (Figure 9) is a 32.768kHz external crystal oscillator to provide accurate timing for UART communication and stepper motor control, as well as the required Spi-By-Wire 2-Pin JTAG hardware specified in the microcontroller datasheet [39]. A 10mA fuse was added to power input of the MSP for added protection, and 3.3V TVS diodes were connected to the 14Pos JTAG connector and the 3-pin power routing header for added protection [40][41][42]. After initial testing the fuse was replaced with a 2A fuse (essential a short) as programing the MSP resulted in a higher than 10mA current draw. The 22pF capacitors supporting the crystal oscillator were chosen based on the 12.5pF load capacitance of the crystal, and the 2pF parasitic capacitance of the MSP pins. Using the equation C = 2*(xtal load capacitance) - pin parasitic capacitance. While for the selected components this equation produces a value of 23pF, 22pF capacitors were the closest values available for purchase. A test point was also connected to an open pin of the MSP to allow for easy debugging in programming.

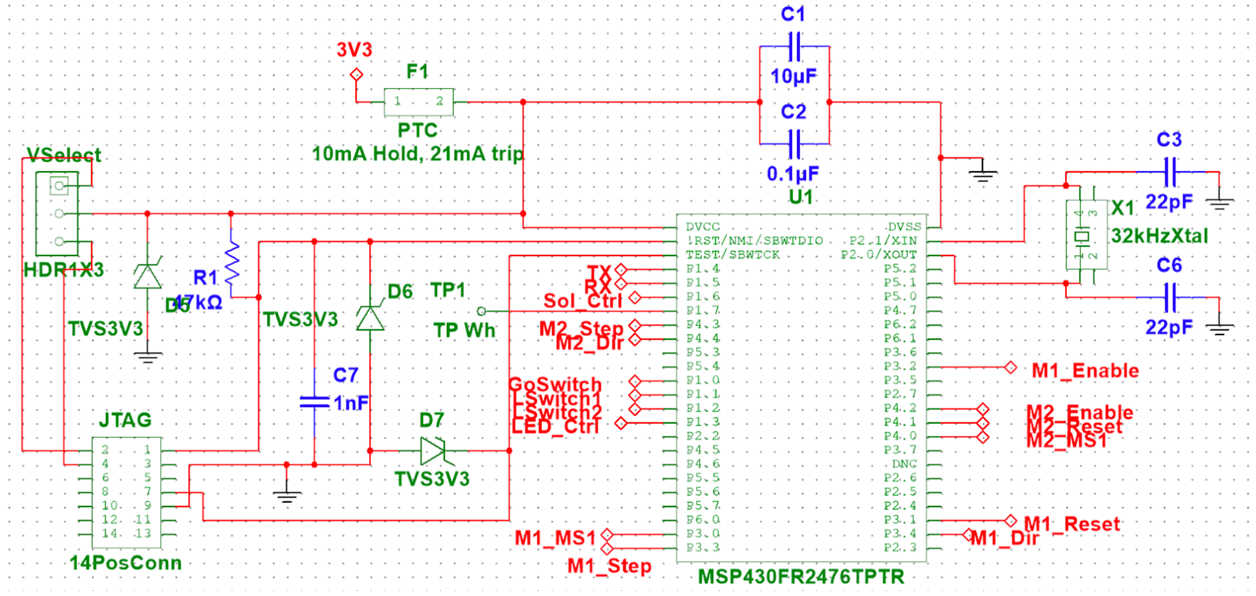


Figure 9 MSP430 Schematic

Wired to the MSP were two wurth connectors for the limit switches, a push button with integrated LED, UART communication to the Jetson, the two motor drivers, and the power transistor for the solenoid. The limit switches were wired to be active low by adding a pull up resistor to one side of the switch, and a connection to ground on the other (Figure 10). A 3.3V TVS was also included for ESD protection [43].

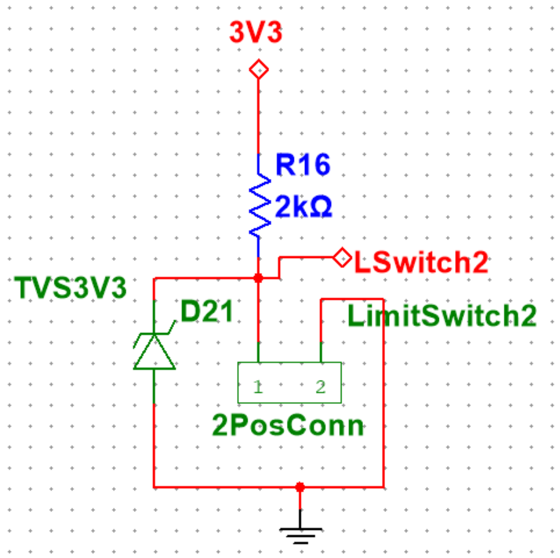


Figure 10 Limit Switch Connection

The push button was wired in a similar active low fashion to the limit switches (Figure 11) [44]. In addition, a BJT was wired to the LED to allow for its control by a pin on the MSP430. A 150Ω resistor was added in series with the LED to set the current to a reasonable amount for the 1.8V forward voltage drop. The resistor going into the base of the BJT was selected to help limit the base current.

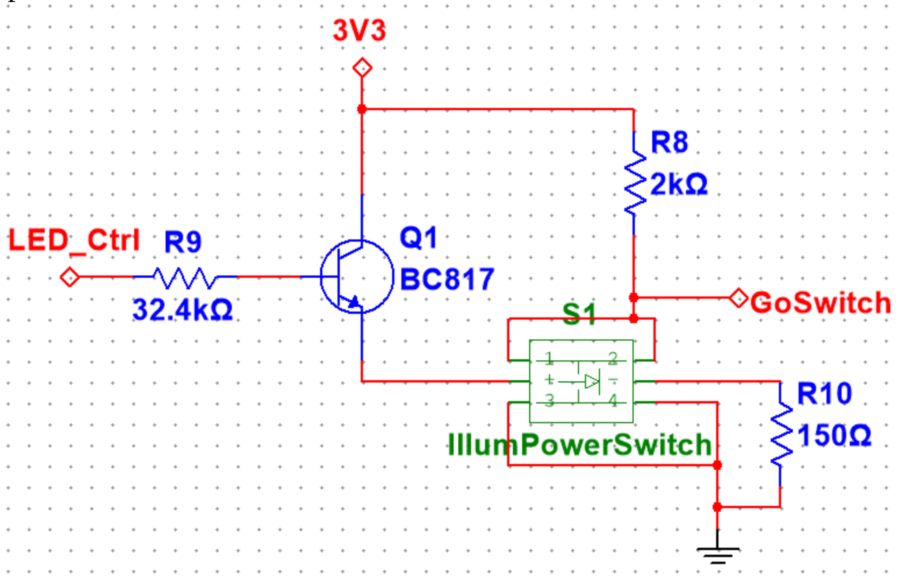


Figure 11 Pushbutton Schematic

The UART communication lines simply went to a Wurth off board connector (Figure 12). Again 3.3V TVS' were added for ESD protection. The ground connection between the Jetson and MSP was established through the Jetson power connection described earlier.

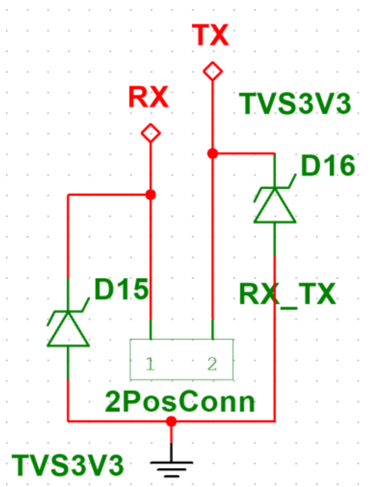


Figure 12 Off Board UART Connection

The solenoid was controlled using a large NPN Darlington power transistor [45][46]. A Schottky diode was also included in parallel with the solenoid to provide back EMF protection (Figure 13)[47]. A heatsink was added to the transistor for added protection, and a current limiting resistor was added to the base [48].

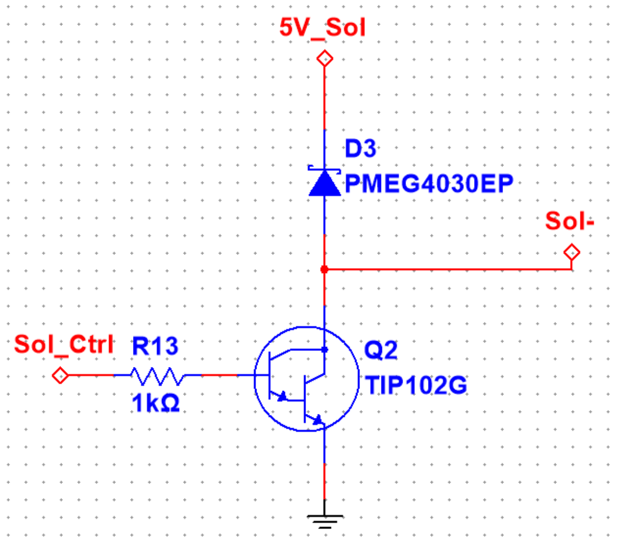


Figure 13 Solenoid Control Circuit

The NEMA 17 stepper motors were controlled using Allegro A3982 motor drivers [49][50]. Athe A3982s (Figure 14) were chosen because they are from the same manufacturer and use the same control method as the motor drivers programmed in advanced embedded. The bypass capacitors were selected following datasheet recommendations, and the logic pins were connected directly to the MSP430. The motor control lines connected to 4pin Wurth terminal blocks (Figure 15) and had ESD protection through a 4 pin to ground TVS diode package [51][52]. The 220mΩ current sense resistor was chosen based off the maximum current per phase rating the following equation from the datasheet.

$$R_s = \frac{V_{Ref}}{8I_{Max}}, R_s I_{Max} \leq 0.5$$

$$V_{Ref} = 3.3V, I_{Max} = 1.68A => R_s = 0.246\Omega$$

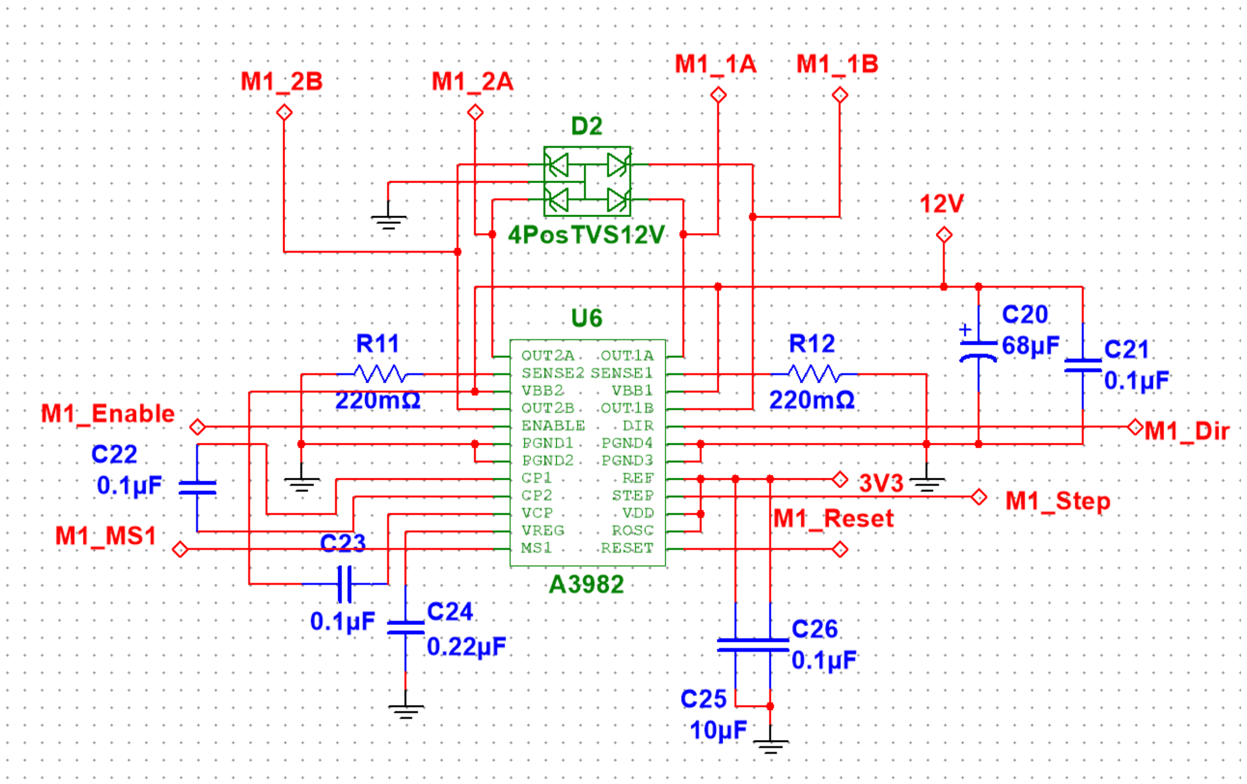


Figure 14 Motor Driver Schematic

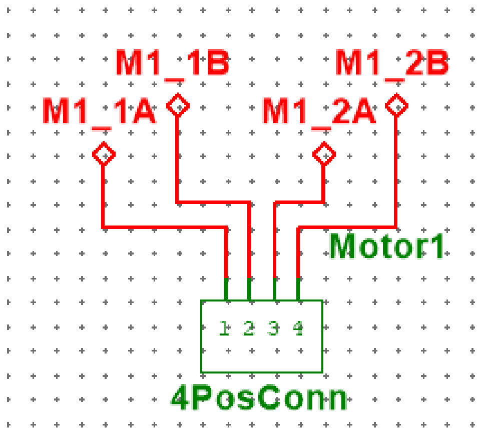


Figure 15 Off Board Motor Connectors

The schematic was laid out on a 5-inch by 5-inch PCB (Figure 16, Figure 17). This size was chosen to provide ample space for all the connectors coming off the board. The connectors were placed around the perimeter of the board, with similar connection types by each other (each motor next to each other, Jetson power next to UART, etc.).

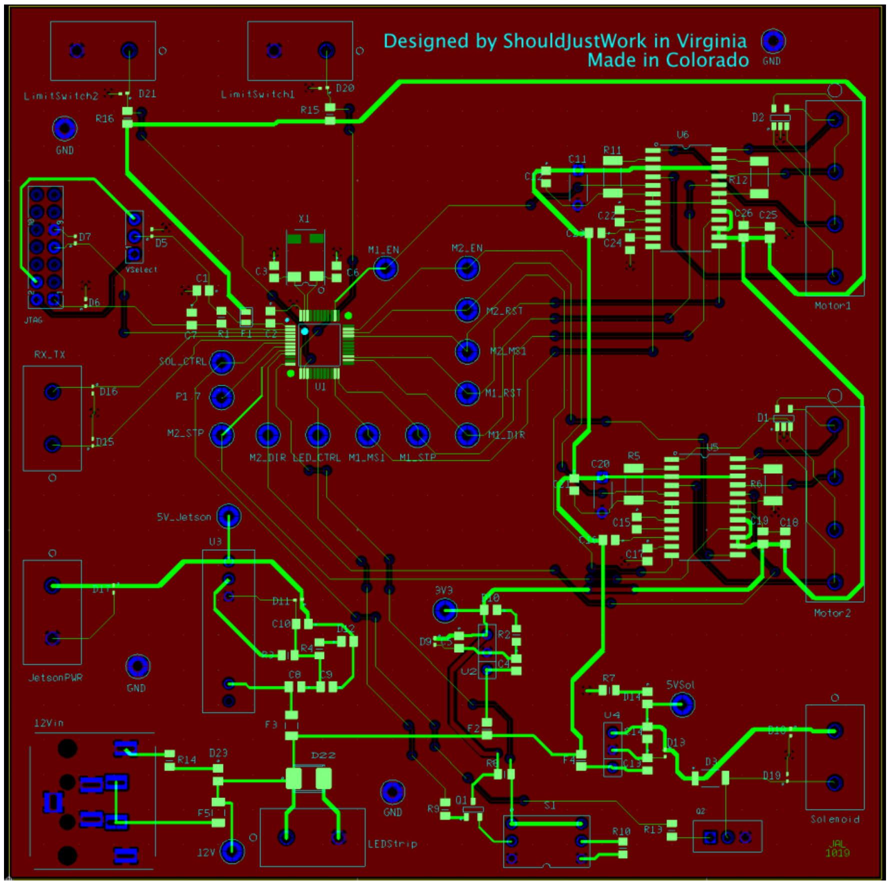


Figure 16 Full PCB Design

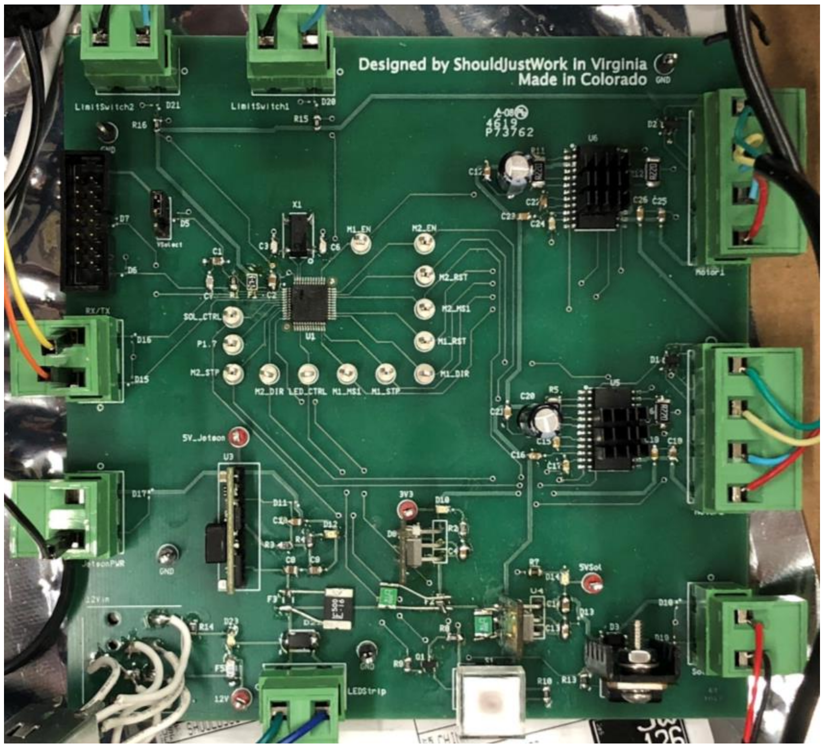


Figure 17 Soldered PCB

The most difficult part of the PCB layout was the fanout for the MSP. This included having test points for each data line coming from the MSP. For one section a vertical line on the top plane, horizontal line on the bottom plane routing convention was adopted to ease the routing process (Figure 18).

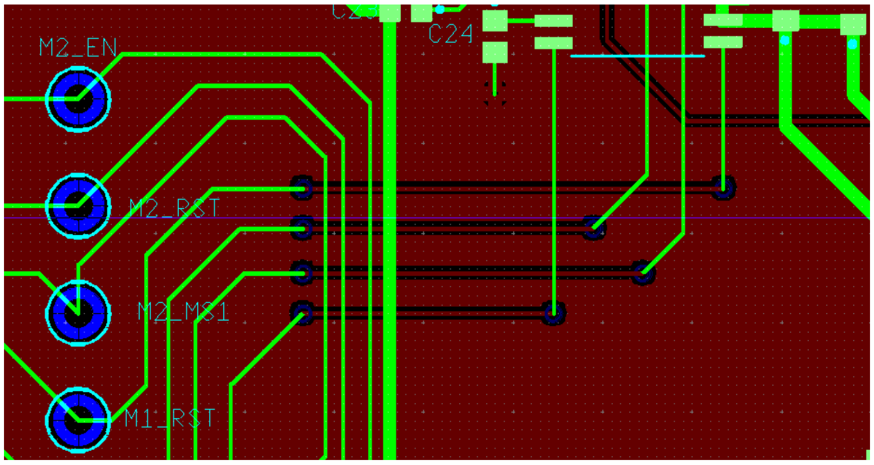


Figure 18 Trace Stitching

When the PCB came in it was determined the footprints for the TVS diode were too small to effectively solder, and the majority of them were left off. Another edit was that the 4-pin power connector for the 12V line required oblong through holes (Figure 19). The board manufacturer would not produce oblong through holes at the price tier of the PCB, so circular ones were used instead. This required jumper wires to be soldered from the board to the power connector as the connector's pins could not fit in the circular holes (Figure 20).

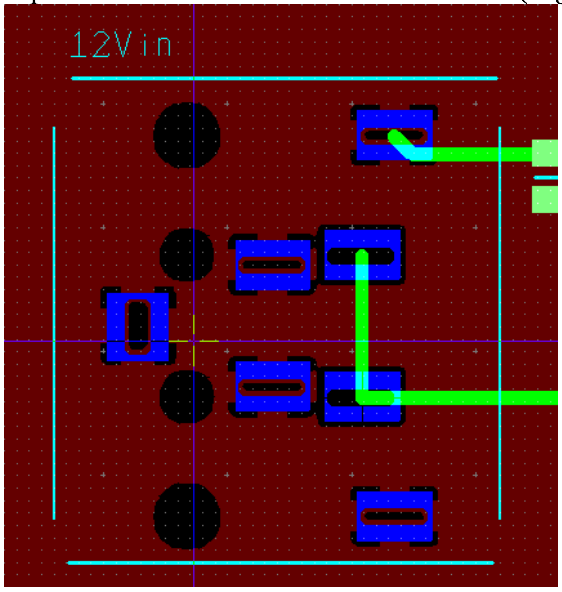


Figure 19 Power Connector Footprint

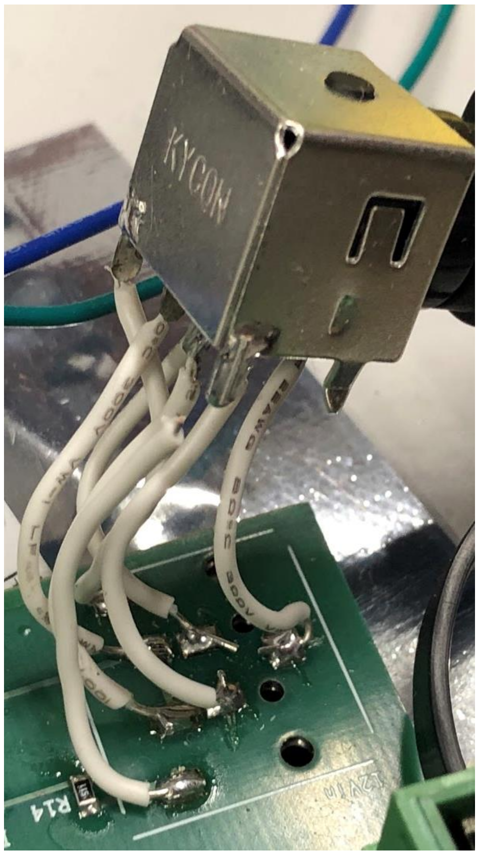


Figure 20 Power Connector Soldered Jumpers

Embedded Design

The code written for the MSP430FR2476 was organized by the component. For example, all of the code dealing with the solenoid was in the solenoid.h and solenoid.c files. This separation made it very easy to find specific pieces of code and to simplify the writing and reading of it because it breaks it down into one part at a time. The MSP430FR2476 was primarily in charge of highlighting the words. In order to do so, it controlled the solenoid to be able to pick up and put down the highlighter and it controlled two motor drivers for x and y movement. The solenoid was easy to operate as it just required setting the MSP pin used for the solenoid control high. This was possible because it controlled the gate on a transistor, and turning it high connects the drain to the source and allows current to flow in the solenoid. Powering the solenoid acted to lift the highlighter and turning it off would drop it. For the motor drivers, the data sheet was consulted to properly configure the five inputs: enable, reset, direction, step, and MS1. Somewhat counter-intuitively, enable was active low meaning that it had to be set low in order to enable motor driver outputs to drive the actual motors. When the device was not in operation, the enable pins for both motors would be set high so they would free up the motors. The reset controls had to be set high when the motors were meant to step because the step input would be ignored otherwise. The direction just controlled whether the motors would rotate clockwise or counterclockwise and the corresponding pin outputs were changed depending on whether the motors needed to more forward of backward in a direction. The transition from low to high on the step pin is an actual motor step, so this was used to control the actual steps taken. Lastly, MS1 was the control over whether the motors would take full or half steps. Originally, full steps were going to be used because it would make the motor movement faster. However, when testing the motors, it was discovered that they vibrated too much, so the switch was made to half stepping to reduce these vibrations. Furthermore, this choice also allowed for more precision in controlling the motor locations.

The MSP was also in charge of the go button. Initially, the go button was supposed to start the whole system, but this was changed once it was discovered that the computer vision pipeline required more human input that initially thought. In the future, the go button could be used to provide some of this input and the pipeline could be changed to require less human input. Once this change was made, the go button was just used once the Jetson had solved the puzzle to let it know that the MSP was ready to receive word locations. Interrupts were enabled on the pin used for the button so that a press would be handled right away and so that it did not need to constantly poll the pin. Additionally, the go button had an internal LED that was powered when the system was active to indicate that it was currently solving a puzzle.

The MSP used two limit switches, one for setting the x-axis and the other for the y-axis. As such, their usage allowed for a system origin as it would be the only point where both switches were active. The switches were active low meaning that pressing them caused the pin connected to them to read low. Since the limit switches were connected on the same port as the go button, using interrupts for them would have been more complicated since the interrupt service routine is called based only on the port. This could have been fixed by checking the bit in the interrupt handler, but it seemed cleaner to not do this. Furthermore, the limit switches could only ever be hit after taking a step while the go button could be pressed at any instant in time, so it made sense to just poll the corresponding switch pins after a step.

For setting up the MSP’s clock, it was decided to use a 32.768kHz external crystal oscillator for increased timing precision. This crystal was used as a reference for the frequency-locked loop (FLL). The FLL was used to stabilize the Digitally Controlled Oscillator’s (DCO) frequency by automatically adjusting it according to the reference clock. Using the equations shown in the MSP user guide’s “CS Operation” chapter, the FLL was set-up to stabilize the DCO at 8388608 Hz [53]. This number was chosen somewhat arbitrarily because it was in the middle of the range of frequencies. This frequency seemed suitable for the system’s needs as it would be faster than lower frequencies and the possible range for PWM periods seemed fine without requiring dividing the timer’s clock source input like the higher frequencies probably would have. Furthermore, this value is pretty standard as a multiple of 32768 and is included in configuration charts within the user’s guide leading to an easy setup. The output of the DCO was used as the source for the master and submaster clock. Timer A was configured to UP mode and to be sourced from the submaster clock. Timer A was used for controlling PWM on the stepper motor drivers by configuring two period registers, one for each motor, to interrupt the program after a set period and toggle the motor driver’s step.

Since the Jetson Nano is actually doing all the computer vision work and computing to solve the word search and determine the word locations, it has to be able to share these locations with the MSP. In order to achieve this, UART was chosen for the communication because it was conceptually simple, only the two devices needed to communicate, and speed was not an issue as transmissions are infrequent. It was decided to send the standard 8 bit messages instead of 7 so there could be more points in the coordinate system, increasing precision. We had initially thought about sending two bytes for every location to further increase the precision, but there seemed to be no need after doing the calculations and discovering that each point would already be a little over one millimeter away from each other. This precision seemed more than suitable and we figured it would be more limited by the Jetson's ability to measure the distances between points using computer vision. The baud rate was chosen to be 115200 because it is a standard rate that would be easy to implement and was actually the default on the Jetson. In fact, the MSP430's user guide had a chart on page 589 showing the required register setup to achieve this baud rate with the system's clock frequency of 8388608 Hz [53].

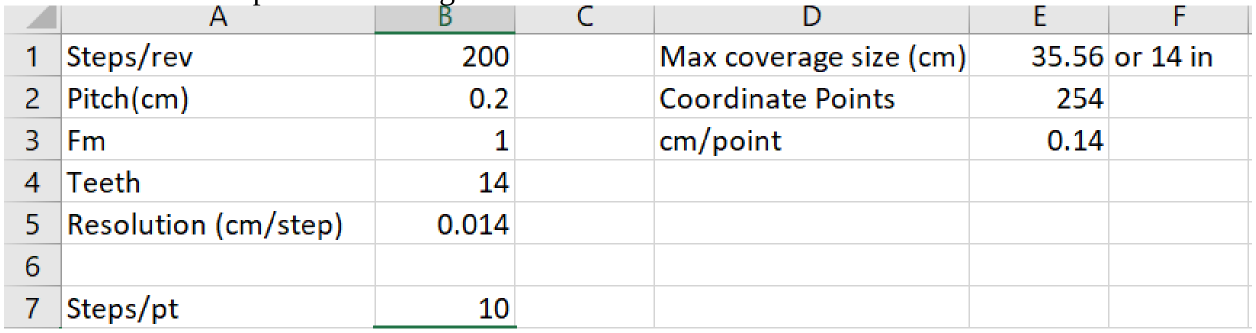
The protocol for communicating with the Jetson Nano was established as follows:

1. The MSP should send the reserved byte (0xFF) to the Nano once the system is first activated.
2. The Jetson will send four different bytes coordinate to the x and y locations of the beginning and end of words to highlight.
3. The MSP will ack that it received them with the reserved word and proceed to highlight the word.
4. Once the MSP highlights the word, it will send the reserved byte once again to let the Jetson know it is ready to receive once again.
5. Steps 2 - 4 will repeat until no more words are left. At this point, the Jetson will send the MSP the reserved byte to let it know that the word search has been finished.

In order to move to various locations, two functions (moveToX and moveToY) were created that setup the PWM for properly moving the x and y motors. This step consisted of setting the direction of movement for each motor and properly setting the enable and reset pins to actually allow outputs from the drivers. These functions also enabled interrupts for Timer A so that the PWM could actually be used. In each motor driver's respective interrupt service routine handler, there was logic to handle the actual movements. The ISRs were used for toggling the step output to the appropriate motor and for keeping track of where the motors currently were within the coordinate system. When the motors reached their destination, the routine was used to disable the motor driver output to disable the PWM. In order to actually be able to highlight diagonal words, the motors could not operate without regard for each other. As such, Bresenham's line algorithm was implemented to take small steps in the x and y directions by calculating how many points it should move at once in order to achieve a fairly straight line. A second option would have been to calculate different what period each motor should have so that they could achieve different angles, but this would have required more complex math for the MSP. Bresenham's was good because it was pretty simple to implement and only required very simple mathematical operations that the MSP could quickly solve. However, this method would produce more jagged lines when the steps are larger because it can only move to integer points, so it was decided to actually increase the number of coordinate points. However, the Jetson still had to send each coordinate point using only one byte, so the number of points available to the outside world could not increase. This was solved by abstracting the new coordinate point system by making one coordinate point equal to 4 internal coordinate points. For example, it the Jetson said to go to the point (10,10), the system would go to the point it called (40,40). This was simple and quick for the system to implement because it could simply take the Jetson's locations and shift them left by two to effectively multiply them by four. The greater number of points allowed for smoother diagonal lines because each point corresponding to a smaller real-world distance.

Another method was created to home the highlighter. This method setup the motors to move in one direction at a time toward its respective axis. After each step, the corresponding limit switch would be checked to see whether or not it was hit yet. Once the limit switch was hit, it would stop moving in that direction because it was on the axis. In order to achieve at least the minimum wait time between steps, a simple delay function was created that waited for an arbitrary number of for loop iterations because it did not need to be precise. The number of iterations was just fine-tuned until it could move as fast as it could without having any issues or stalling. The goal of this method was to return to the origin as smoothly and accurately as possible.

In order to determine the coordinate system and period for motor movement, some trial and error was involved. As previously mentioned, the coordinate system was initially calculated mathematically based on the step resolution, the size of the area that needed to be covered, and the number of coordinate points. Figure 21 below shows a snippet of an Excel file used for these calculations. However, these numbers had to be changed at various points such as switching from full to half steps and realizing the whole area could not be used due to camera constraints.



	A	B	C	D	E	F
1	Steps/rev	200		Max coverage size (cm)	35.56	or 14 in
2	Pitch(cm)	0.2		Coordinate Points	254	
3	Fm	1		cm/point	0.14	
4	Teeth	14				
5	Resolution (cm/step)	0.014				
6						
7	Steps/pt	10				

Figure 21 Example Calculations for Steps per Coordinate Point

In order to determine the period for motor PWM, different ones were tested to find what worked best. The desired trait was that it would be as fast as it could without stalling, vibrating too much, or missing steps. However, it was discovered that the use of Bresenham complicated things in regard to the period and number of steps between points. Since Bresenham required only moving several points at a time, and the resolution was already decreased, the motors were pretty much stopping as soon as they started. This seemed to result in inconsistent distances travelled per step and it was not able to return to fixed points by coordinate location once it started away from the origin, perhaps due to acceleration and deceleration issues. The inconsistency was fixed by slowing down the motor periods and increasing the number of steps between points. After these numbers were tweaked, it became remarkably consistent and could go to precise spots from anywhere else on the page. Initially, it was planned to use ramping for motor acceleration and deceleration, but this did not seem to make sense since it was starting and stopping so suddenly. Given that it was already very accurate, doing this did not seem needed.

Software

Here we will delve into the design of the software pipeline that images, parses, and makes sense of the word search puzzle. We will also specify what parameter settings worked best for us - these were derived experimentally, so they may or may not work for configurations different than ours, e.g if the word search puzzle format is different, lighting isn't controlled, or if the camera used is different.

To begin with, we have some software and hardware requirements that must be met in order to build our system. Firstly, the device chosen on which this software system will run must, at a bare minimum, have support for the OpenCV and Tesseract OCR libraries , and secondly, the device on which this software resides must have UART support, as this is how we will communicate found word positions to solve[9][10]. In our implementation, we used OpenCV 4.1.1 and Tesseract 4.1. Although it is likely, there is no guarantee that future version of these libraries will maintain backwards compatibility of their API. Our programming language of choice is Python 3, which we chose because of the vast libraries it offers and its usability. Most modern computing platforms will support Python 3, and you can use any text editor of your choice. Of course, using an interpreted language as opposed to a compiled language like C or C++ has its downsides - namely, performance comparatively suffers. For other features we've implemented to work, such as the spellcheck functionality, the Jamspell library must be supported as well [54]. This requires dependencies that are easier to install on a Linux based system, although it is possible to do so on a Windows based machine as well. To meet these requirements, we chose the Nvidia Jetson Nano, which is an embedded computing system that runs a variant of Ubuntu 18.04, has a built in GPU, and has a relatively powerful ARM based CPU [18]. The idea with the GPU was to take advantage of OpenCL acceleration in some OpenCV functions, although we ran out of time to implement this and the unaccelerated code was sufficiently fast enough to not need GPU usage.

At a high level, we capture an image with the webcam, separate out the puzzle and wordbank, then apply preprocessing to both images. From there it gets fed to Google Tesseract, which outputs the letter/word detections in addition to their pixel locations in the image. Part of the preprocessing involves a crop and a rotation of the image, so in order to get the coordinates from Tesseract back to the original orientation, we apply the inverse spatial transformations. The word search itself is solved using an exhaustive search algorithm where we iterate through every letter in the word bank and check in all possible directions for the existence of a word. If a word is found, we store that as start and end image coordinates. Finally, using parameters from a known calibration pattern, we invert the pinhole camera equation to transform from image coordinates back to our real world coordinates, which are then easily convertible to the coordinate space known by our XY system. Using a simple UART based API, we then communicate word start and endpoints to the MSP, which then draws each point pair, ACKing back to the Jetson once it's finished, which lets it know when to send the next coordinate pair. Once all of the found words in the puzzle are highlighted, the Jetson then opens a camera viewfinder and awaits the next puzzle input.

Camera calibration and image to world calibration

All cameras have some level of distortion, and while for our camera doesn't particularly have a particularly egregious level of distortion, it's still good practice to use a checkerboard or some other known pattern to estimate the distortion and intrinsic camera parameters. To do this, we printed out a standard calibration pattern as shown in Figure 22.

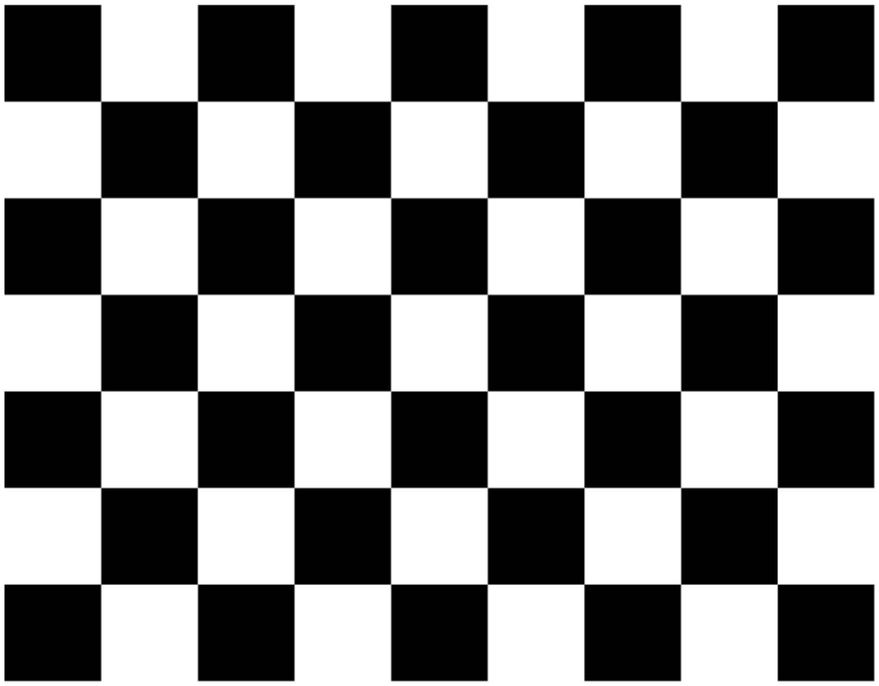


Figure 22 6 x 8 Calibration Checkerboard (internal rows and columns)

With this printout, we used the camera to take around 15 images of the checkerboard at various distances in various poses. OpenCV provides a function that looks for checkerboards in images and gives the internal points of the checkerboard called findChessboardCorners() [55]. We feed those points into another OpenCV provided function called calibrateCamera() which then outputs the intrinsic matrix and the distortion coefficients [55]. These parameters are then applied to the overall image using the OpenCV function undistort() [55].

Puzzle and Word Bank separation

This step is particularly important because of how our OCR engine works - the layout of the page is important to getting an accurate detection, and different modes of Tesseract work better for different types of text. In particular, the sequence-based machine learning method (LSTM) works better for real, English words, while the old school pattern matching based system works better on a letter by letter basis [56]. For example, if we were to try to parse the puzzle portion (the letter grid) with the newer, LSTM based method, we would receive “hallucinated” output, i.e the system would see English words in the soup of letters where there wouldn’t be any. On the flipside, this works a lot better for the word bank portion, as we’re almost always going to be seeing words that are from the English dictionary, something which the Tesseract model has been trained on.

To separate the puzzle from the word bank, we take advantage of a property found in all of the word search puzzles we chose - all the word grids are always surrounded by a square. Starting from the raw image, we convert it to grayscale, apply an erosion operation, then a Canny edge detector, then feed it into OpenCV’s findContours() function [57]. For the erosion operation, we use a 3x3 kernel of 1s, with an iteration count of 1. Effectively, this thickens lines in the image, which helps find contours more robustly (Figure 23). The input for findContours() must, at a minimum be thresholded. Canny edge detection does this in addition to clearly finding the edges of the square (Figure 24). We found that this increased the rate at which the square was detected by findContours(), an example of which is shown in (Figure 25).

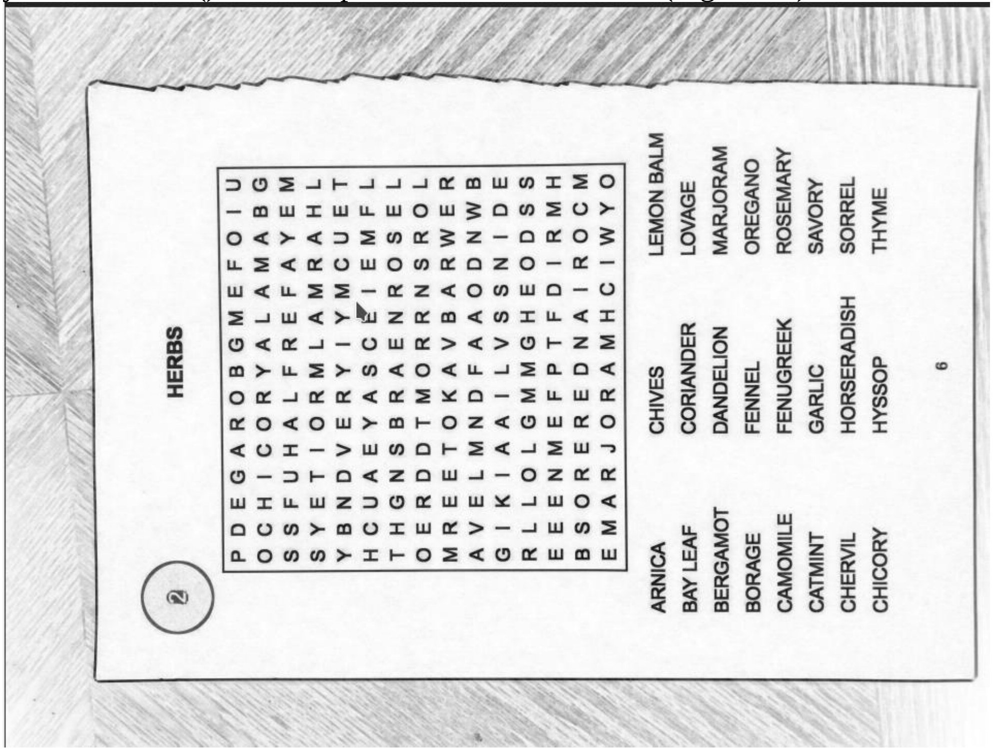


Figure 23 Eroded image

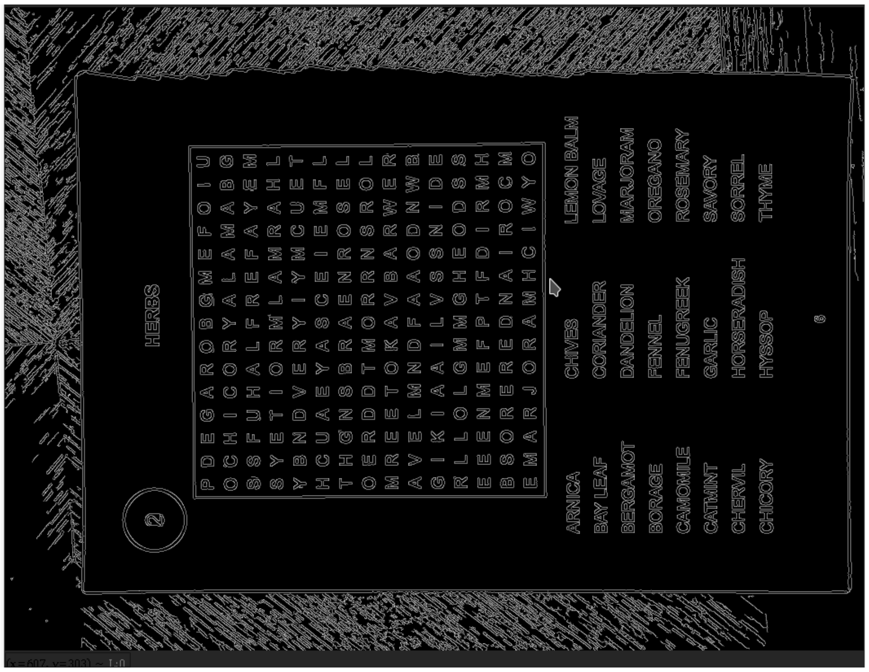


Figure 24 Output After Canny Edge Detection

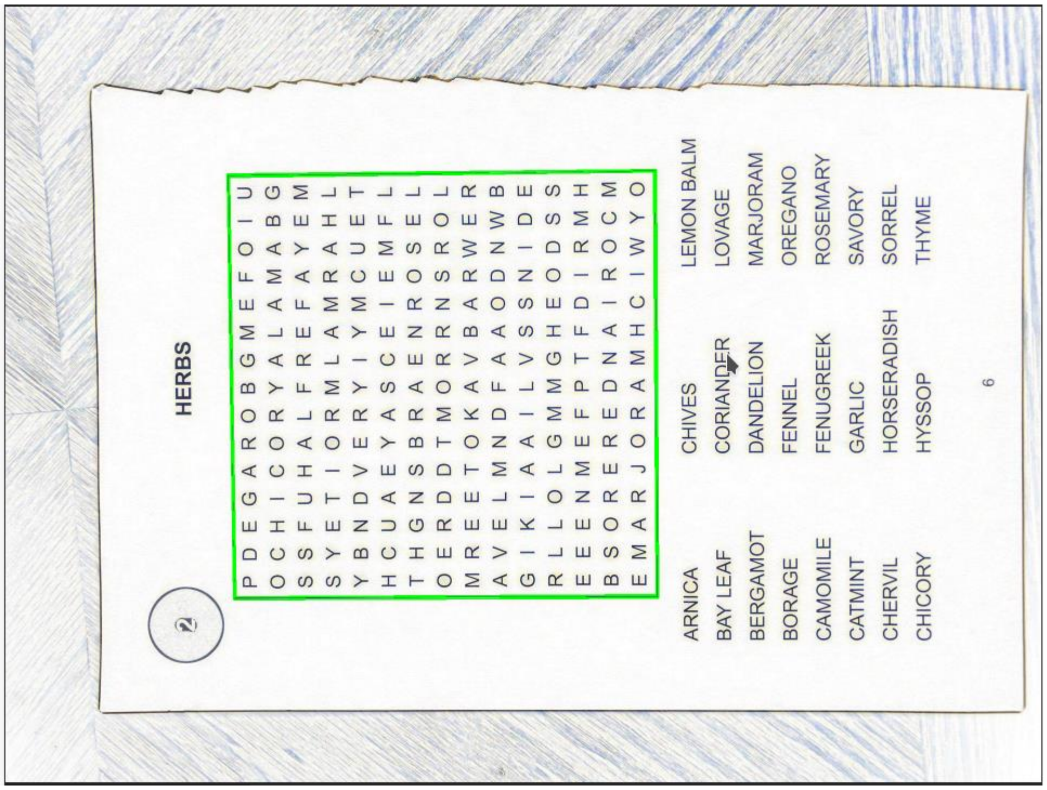


Figure 25 Found Puzzle Square

With the location of the square in image space, we then crop out the puzzle, and mask out the puzzle correspondingly with black on the rest of the image. This "rest of the image" will be our word bank, in other words, it's assumed that everything in the image that isn't the puzzle is the word bank.

Identifying words and their image coordinates

We use the Pytesseract library, which is simply a Python wrapper around Tesseract to more easily integrate it into our program.

Before we utilize Tesseract, there are a few preprocessing steps that are performed. In order of operation, they are, resize, applying a sharpening kernel to the word bank, grayscaling, otsu binarization, rotation, and deskew to the word grid. Tesseract expects an image DPI of at least 300, so we blow up the smaller input image to compensate. The sharpening filter helps improve detection accuracy if the input image happens to be out of focus and the rotation and deskew are to set the image upright relative to the screen. The binarization is important because it does the bulk of noise removal from the image, making words and outlines very clear and apparent. We chose the Otsu implementation because it worked well in a variety of lighting conditions, preventing us from having to hardcode in different binarization thresholds for different lighting conditions.

Tesseract is given two different inputs, one for the word puzzle and one for the word bank. For the word puzzle the config flags are given as follows: *--tessdata-dir "./protos_data/tessdata" -l eng --oem 0 --psm 6 -c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ load_system_dawg=0 load_freq_dawg=0 textord_heavy_nr=1.*

Let's break this down. The first arguments *--tessdata-dir* and *-l eng*, specificies which trained language model to use. Because we're using an older version of the OCR engine, as specified by *--oem 0*, we're required to supply our own model rather than the default. *--psm 6* sets the page segmentation mode to assume it's looking at a single uniform block of text [58]. *tessedit_char_whitelist* restricts detection output to the uppercase English output, as that encompasses every possible letter, we could encounter in the word bank. *load_system_dawg* and *load_freq_dawg* are set to False to tell Tesseract to ignore loading the system dictionary, as we only want per letter detections. Finally, *textord_heavy_nr* is set to True to enable more aggressive noise reduction. We found these parameters through experimentation, and all of the tweaks together make the output more accurate.
For the word bank:
--oem 3 --psm 3 -c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ textord_heavy_nr=1
It's quite similar, with the exception of *--oem 3* and *--psm 3*, which set the engine to the newer LSTM based mode, and sets page segmentation to an automatic mode, respectively [58].

Once the preprocessing steps were completed, bounding boxes for each letter were found via Pytesseract. From the bounding boxes, each letter's center point (measured in pixels) was calculated and used to represent the image coordinate of each letter. Each image coordinate was stored in a 2D matrix that will be used for further image coordinate calculation. Since the image was rotated 90 degrees plus a deskew angle, the center coordinates are also rotated by an equivalent amount. In order to resolve this discrepancy between the original image and the current one, the coordinates must be rotated back to the correct position.

$$Rv = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}.$$

Figure 26 Rotation Matrix Equation Given (x,y) Coordiantes

To perform the rotation, the above rotation matrix equation was used. Due to the fact that the origin of the image is in the top left of the image, and not the center, the coordinate points needed to be translated before rotation. Mathematically this looks like the following - let w be the width of the image and h be the height, then the new coordinates were $(x', y') = \left(x - \left\lfloor \frac{w}{2} \right\rfloor, y - \left\lfloor \frac{h}{2} \right\rfloor\right)$. Once each point was translated, they were rotated by the above matrix function and placed back into the 2D center coordinate matrix. Once each coordinate had been rotated, the offset that was removed was added back, and all that was left to do was scale them and perform one last translation.

The points needed to be scaled because the image being worked with was scaled up to three times the size of the original, which means each coordinate needed to be divided by three. Furthermore, the image was also cropped, meaning the origin of the current image is not equivalent to the origin of the previous image. Once the final translation was performed, each center coordinate was in the correct position, and the 2D matrix held the true image coordinates for each letter that are now ready to be converted to physical coordinates for the XY table.

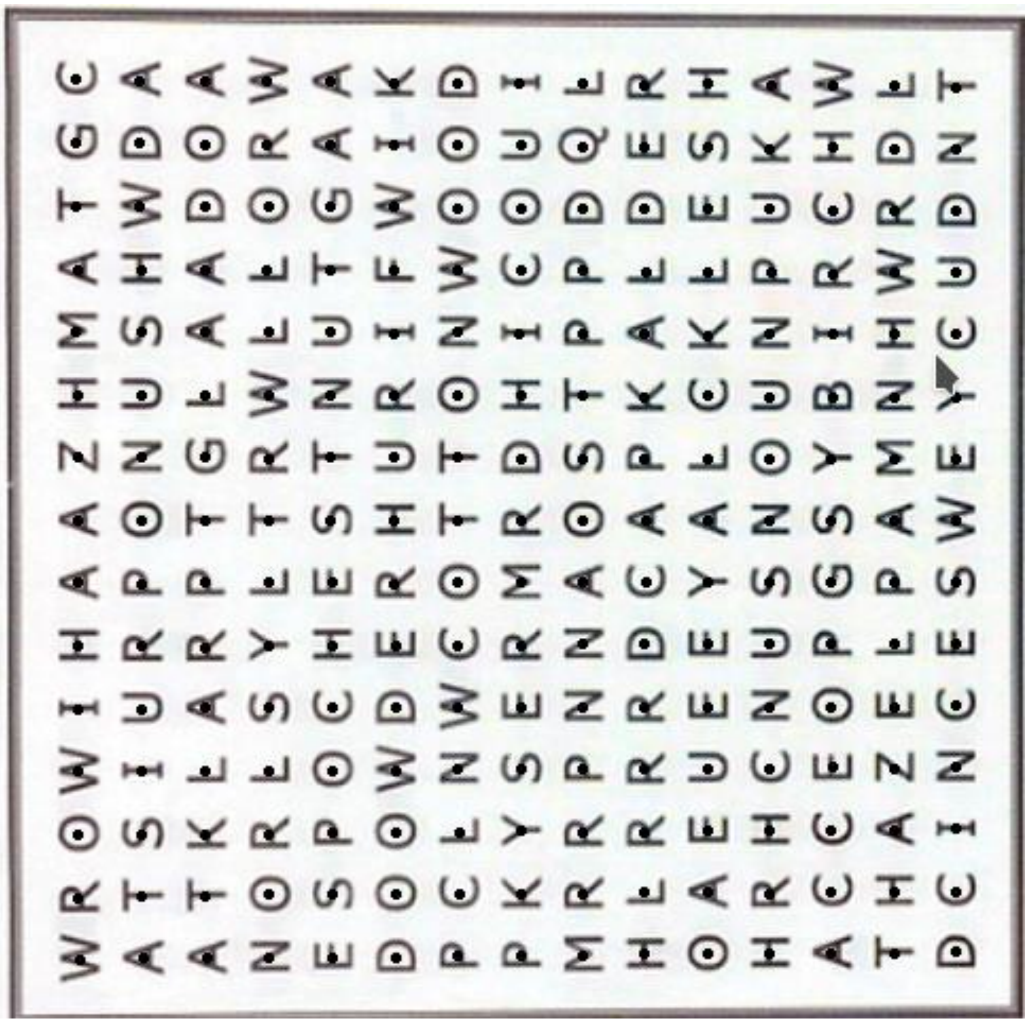


Figure 27 Image Center Points After Detection and Undoing the Image transformations

<u>Solving the Word Search Puzzle and Corrective Measures to Improve Accuracy</u>

Once the puzzle and word bank have been separated and read in via Google Tesseract, a brute-force approach was used to solve the word puzzle. The logic for the solver is as follows - iterate through each row and column of the puzzle, then iterate in each valid direction (up, down, left, right, diagonal), and finally, iterate through the length of the word puzzle. On each iteration of this algorithm, a temporary candidate word is read in from the puzzle, and checked against a dictionary of words from the bank. If the candidate does not match any of the words in the dictionary, it was discarded; if the candidate matched one of the words, it was stored along with its starting location. As an example, if the first letter of the word apple was in the fourth row, third column, the following touple would be pushed on to the solution array - (3, 2, 'apple'). The process continues until the iteration reaches the bottom right character, after which solving terminates.

Although the solution algorithm is fast and reliable, the interpretation of the word bank may not be, especially in less than ideal lighting conditions. In order to make the solution algorithm as robust as possible, a permutative solving algorithm had to be implemented. In this new algorithm, there are two main functions: letter swap out and spell checking.

The letter swap out algorithm relies on a table of letters that Tesseract has the highest chance of confusing. For example, the letters O and Q, or H and M. Often times an incorrect word simply had one or two letters misinterpreted when being read in from the page, and swap out to the most commonly confused letter would correct the issue. Thus, once the solver has gone through its initial pass through the puzzle, it returns a list of words found and a list of words not found. The latter is then fed into the swap out function, which generates every combination of possible character swaps for that word. For example, giving a word like ROBOT to the swapping function would return the following list of words - [RQBOT, ROBQT, RQBQT]. This is done for every word not found, generating a master list of all possible updated candidates to try on the word puzzle solver. If any one of the updated words was found, the corresponding initial word (in this case ROBOT) would be removed from the list of incorrect words. After this section of the algorithm was complete, the only words remaining are those that aren't valid, even after character swap out. Typically, these words are either misspelled (ROBOUT) or are missing characters (RBOT).

To combat simple misspellings and dropped characters, a third party spell checker known as JamSpell was used. This library was trained on 300,000 Wikipedia sentences and 300,000 news article sentences and could provide reasonable spell checkings out of the box. The runtime on JamSpell leaves a bit to be desired and thus was only resorted to if absolutely necessary (when swap out fails). Each word in the incorrect array is fed into Jamspell, which outputs a list of potential word candidates for each letter. Every candidate is then compiled in a list to try on the word puzzle solver. If any one of the candidate words was found, the corresponding initial word would be removed from the list of incorrect words. By the end of this iteration of the solving algorithm, only words that were significantly incorrect would be omitted from the word puzzle. At that point, the program outputs the list of unfound words to the terminal and continues on through the pipeline.

Converting Image Coordinates to World Coordinates to XY Table Coordinates

The process of getting from points in a 3D image to a 2D image plane can be modeled by the equation in Figure 28.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Figure 28 Pinhole Camera Equation

X,Y, and Z are the real world coordinates, with an extra homogeneous coordinate added to make the whole transformation representable by matrix multiplication. The first matrix on the right side of the equation, *A*, represents the intrinsic camera parameters derived from the initial checkerboard calibration step. This contains values such as the focal length, and the principal offset of the camera. The second matrix contains both the rotation and translation values that bring the 3D coordinates into the appropriate 2D space. Finally, on the left side of the equation we have a scaling factor, *s*, and the image coordinates *u* and *v*.

Inverting that equation, we get the equation in Figure 29

$$\left(s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} A^{-1} - t \right) R^{-1} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Figure 29 Pinhole Camera Equation Solved for World Coordinates

So then, we need to determine what *R* and *t* were originally to solve for the world coordinates. Luckily, one of the solutions to this problem is implemented in OpenCV already, within a function called SolvePNP(), which takes in a set of image coordinates along with their corresponding real world coordinates and the intrinsic camera matrix [55]. This function outputs *R*, which we then invert along with the intrinsic camera matrix. We determine the scaling factor *s* by choosing it such that it forces the x,y coordinate of a known point to be exactly how what it should be in real world coordinates. For example, we know that the point diagonally adjacent to our defined origin should be x = 26.1 and y = 18.1, so then we choose *s* such that the output of the inversion equals it exactly. We found that this method of choosing the scaling factor works quite robustly. In our implementation, we only use 5 calibration points, as seen in Figure 30, but this could be improved by using up to 9 calibration points.

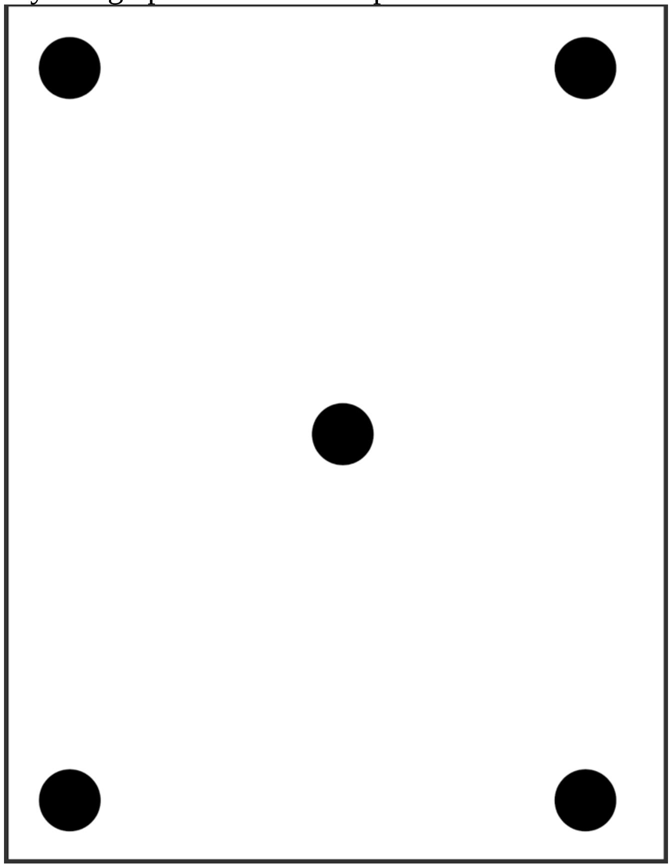


Figure 30 Image to World Calibration Image

Once *R* and *t* are known, those values are stored to later be applied to the image coordinates from solving the word search puzzle.

With the real-world coordinates (in centimeters) of the start and endpoints of each word to highlight, we perform the final step of converting them into step coordinates that the MSP understands. This comprises of adding the x,y offset between the XY table's (0,0) and the origin we've defined through the image calibration, and dividing the world coordinates by the number of centimeters per stepper motor step.

Mechanical - XY Table

The mechanical system comprised of two main components: the XY table and the pen holder. We will begin with the design of the XY Table, which was responsible for moving the highlighter and pen holder for the purpose of highlighting target words on the puzzle page. Our design requirements for the table were specced with several parameters in mind: size of A4 standard paper, spacious enough for the penholder origin to be out of the camera frame, sufficient room to add a camera mounting structure, provide space for LED strips, and small enough to fit on the capstone table. With these parameters in mind, we decided to go with a dual-motor, 20"x20" aluminum structure from OpenBuilds. The 20" sides allowed plenty of room for the A4 paper, and also provided enough space that the penholder would be out of image frame when homed. This was particularly helpful as the segmentation in our computer vision algorithm would be thrown off by the contours on the penholder mounting system. The size of the table also provided enough space for us to mount four LED arrays along the interior of the aluminum beams that allowed for sufficient lighting of the word puzzle. While the size of the XY table was important for the project, the components that made it up where also crucial to project success.

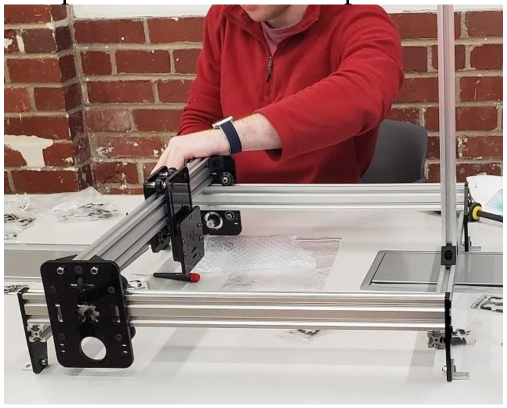


Figure 31 XY Table Fully Assembled

Our table was made up of several components that ensured structural integrity and guaranteed high performance functionally. The base chassis is comprised of two double barrel, 20" aluminum T slotted pipes that are screwed orthogonally above two single barrel 20" aluminum T slotted pipes. There is one more double barrel aluminum pipe that sits perpendicular to the first two and is fastened via angle brackets gantry plates. This pipe has a gantry along with a mounting plate that serves as the platform on which the penholder is mounted. As opposed to the other aluminum extrusions, this one will move along with the pen holder and is responsible for providing a second dimension of motion. These pipes all allow for the timing belts to slide through them and be screwed down at either end. This allows for the motors to directly pull the gantries along the length of the aluminum for precise pen movement.

Moving away from the aluminum extrusions, there are three gantry plates responsible for motion and they are seated on the double barrel aluminum pipes. They must be fastened to the thicker extrusions specifically because the wheel distances fit perfectly around them, which provides stabilized, smooth movement for the pen holder. Two of the gantries are equipped with a 14-tooth pulley that interlocks with the timing belts and connects to the motors. The reason all three don't have motors is due to the fact three motor-timing-belt systems were not necessary. Only a single motor was needed to move in each dimension since our payload (the penholder) was not heavy enough to warrant the extra pulling power that three motors would supply.

Attached to the moving aluminum extrusion is a gantry kit that also has a specialized mounting plate. Initially the penholder was going to be directly screwed into the plate for stability and integrity, however there were a shortage of screws. The solution to this was to use command strips that would directly connect the penholder to the mounting plate. Although the penholder no longer needed to be screwed in, the mounting plate was still an important as it provided a sliding mechanism for adjustability. The penholder would not allow the highlighter to reach that bottom of the page if it were directly mounted to the gantry plate, however, the vertical adjustments provided by the mounting plate allowed the penholder to be mounted at the perfect height for operation.

The final component of the XY table is the camera mounting system, which is simply two perpendicular pieces of single barrel aluminum connected via an angle bracket. The first aluminum extrusion was mounted in an upright, erect position on the side of the table opposite of the penholder origin. The primary reason for this design decision was the reduction in clutter on that side of the table as there were already a significant amount of wires there. The second single barrel aluminum pipe was fastened by an angle bracket approximately 10 inches vertically above the paper. This height gave the optimal framing of the paper on the camera, with almost nothing else in its FOV. Lastly, the camera was mounted to the structure through zip ties, which provided stable support to the camera that kept it from moving and sliding.

Mechanical - 3D-Printed Parts

Another mechanical engineering aspect of the project involves the design and physical implementation of two parts: the connector and the holder. The purpose of the connector is to join the solenoid and highlighter. The connector tightly fastens to both the highlighter and the solenoid so that when the solenoid throw is moved, it is able to pull the highlighter along with it. This action is critical for the system; as the system solves the puzzle, the highlighter must be pulled up off the page frequently. One side of the connector is tailored to grasp the throw of the solenoid. A cylindrical glove works with a rectangular piece in the center of the glove to secure the solenoid throw. For further security, a small hole (3 mm in diameter) through the glove and the rectangular piece mates perfectly with a hole in the solenoid's throw. We designed our connector such that a small "twist-tie" can be wound through this small hole to fasten the connector to the solenoid. Specific details and dimensions can be seen in Figure 32.

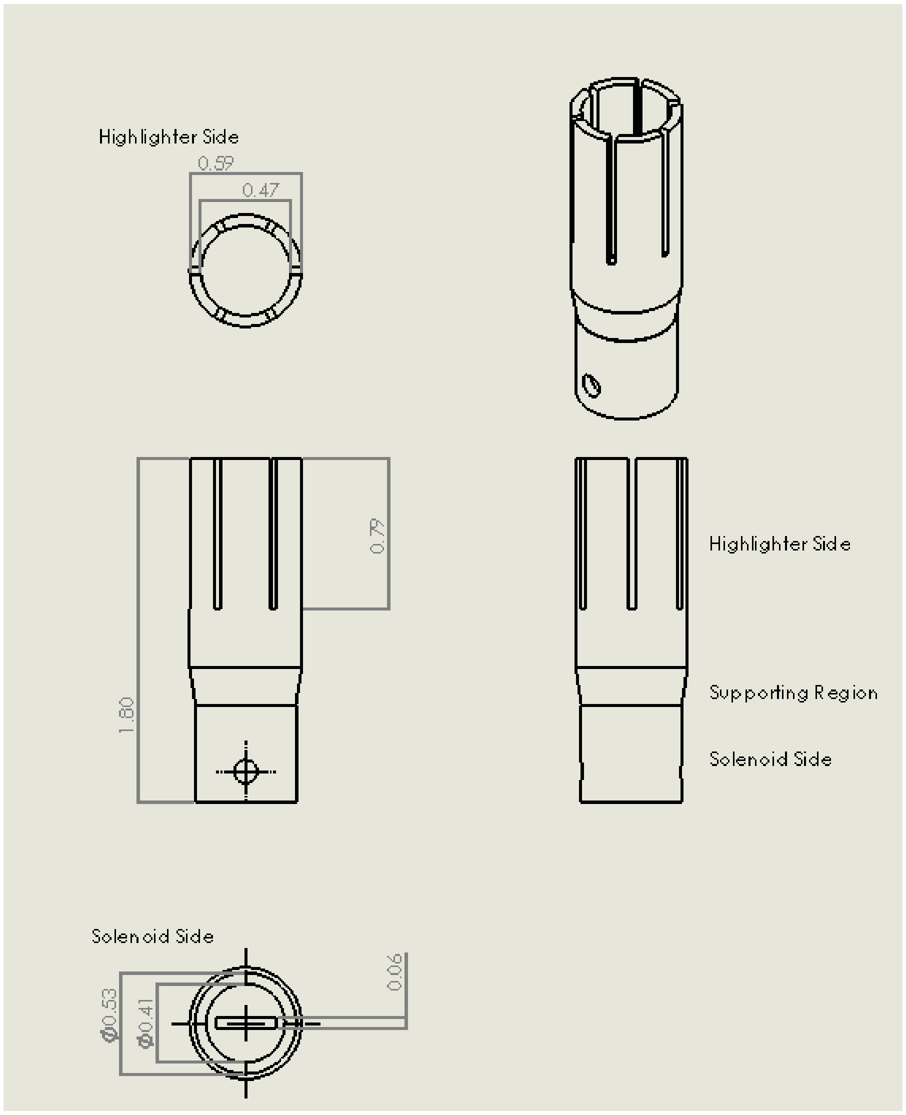


Figure 32 Connector Drawing (Inches)

The other side of the connector is responsible for fastening around the non-writing end of the highlighter. This side presents more of a challenge than the side including the solenoid for several reasons. One challenge to this side is that there were no publicly available CAD files for the highlighter we were using. This meant that all of the measurements on this side needed to be measured using rulers and calipers. Another challenge associated with this side involved the changing diameter of the end of the highlighter. To make sure that this side would fit around the highlighter, we designed this end to be around 1mm wider than the widest diameter of the highlighter. We implemented a collet here as well, so that this side could collapse tightly around the highlighter. After collapsing the prongs of the collet, we used zip ties or tape to fasten this side of the joint.

The other 3D-printed part for our system is the holder. The holder provides a glove-like housing for the complete solenoid-connector-highlighter fixture. In addition, the dimensions of the holder are designed for simple mounting to the acrylic mounting plate of the XY table. The holder uses a shaft around 6 inches long to house the highlighter. At the top of the shaft a rectangular-shaped wall concentric to the shaft houses the cube-shaped solenoid. This configuration allows the solenoid to sit safely on the top of the holder in place while the throw and the highlighter move up and down. Full details of this component can be seen in Figure 33.

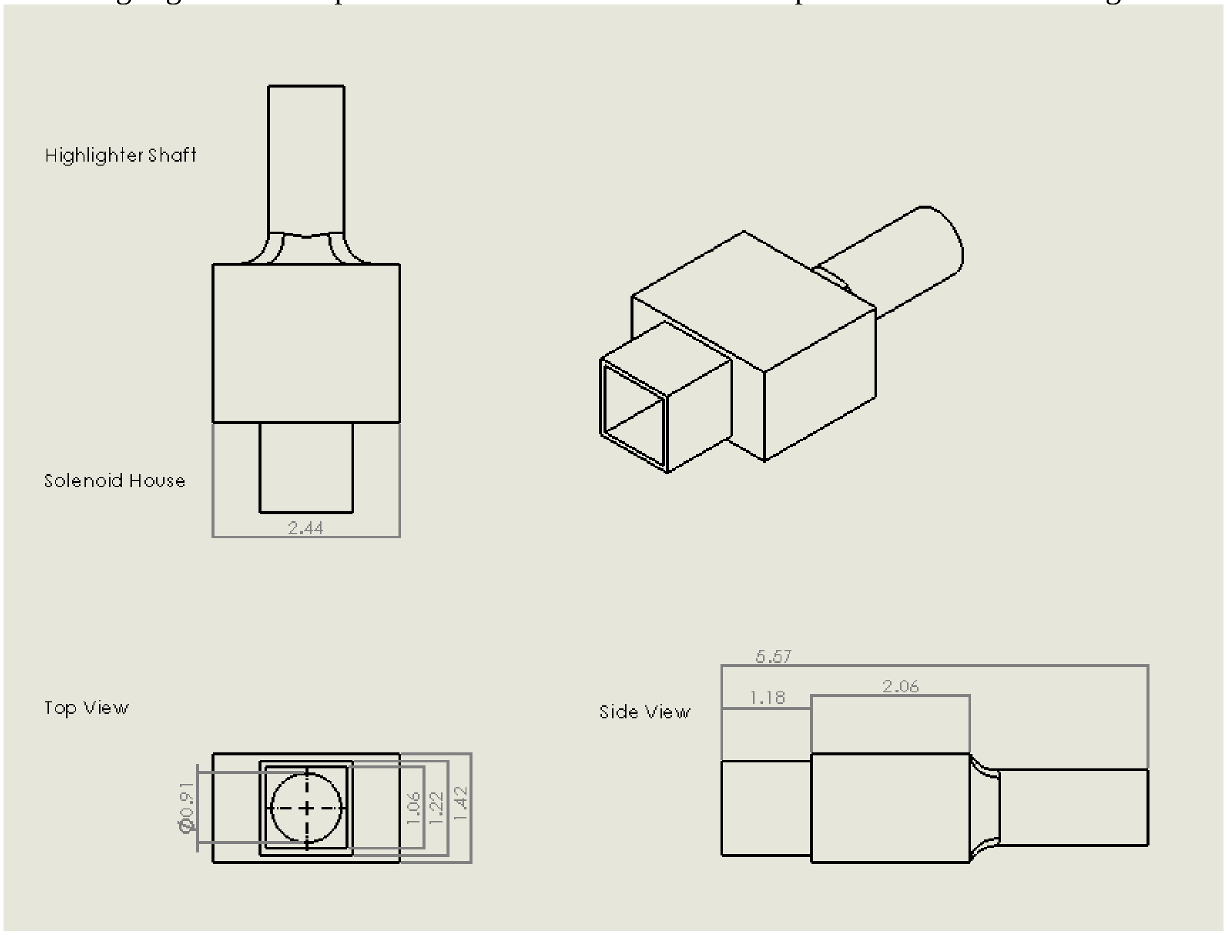
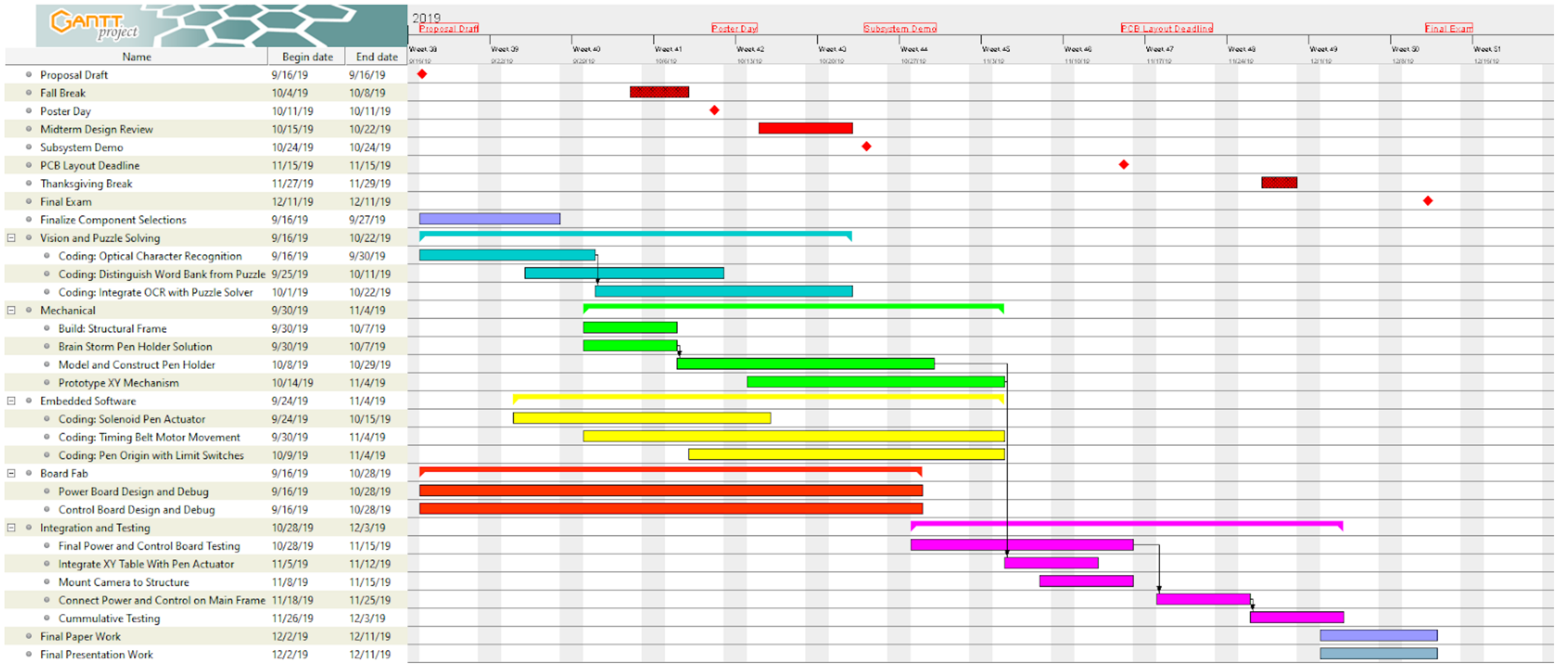


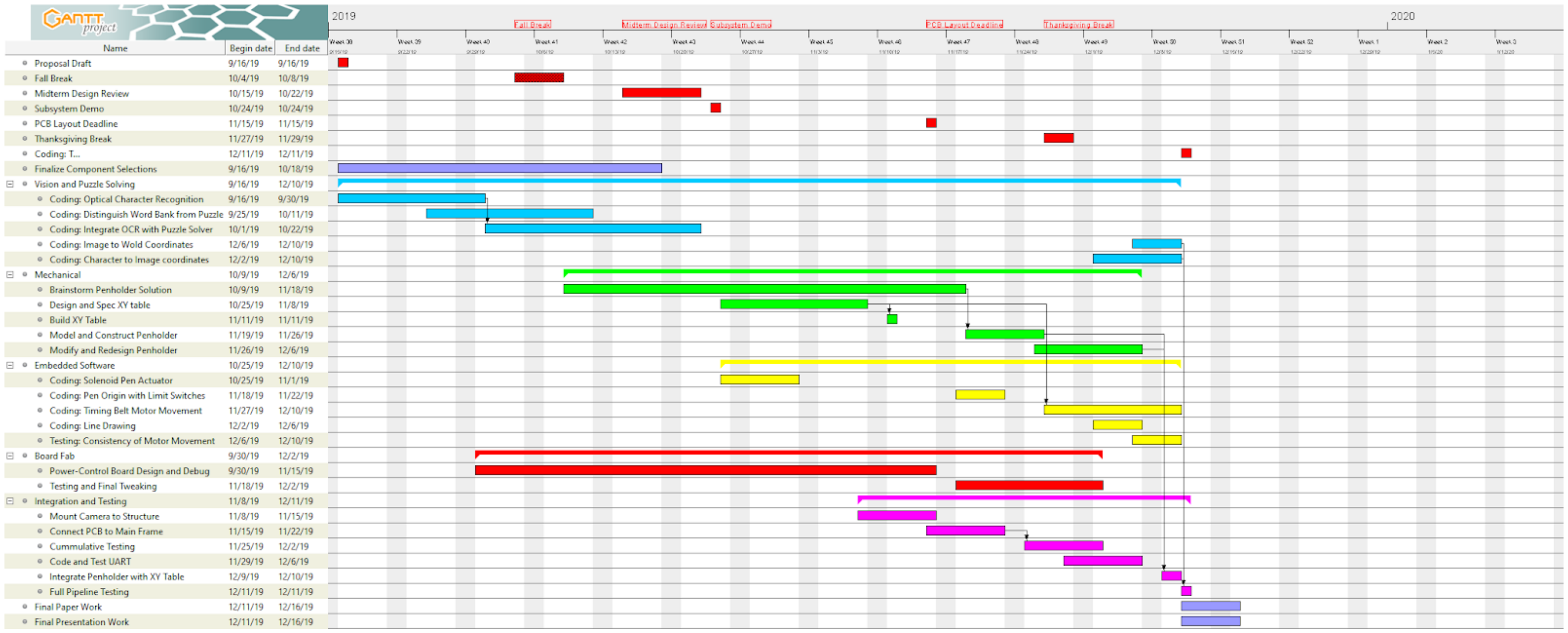
Figure 33 Holder Drawing (Inches)

Project Time Line



Name	Begin date	End date
Proposal Draft	9/16/19	9/16/19
Fall Break	10/4/19	10/8/19
Poster Day	10/11/19	10/11/19
Midterm Design Review	10/15/19	10/22/19
Subsystem Demo	10/24/19	10/24/19
PCB Layout Deadline	11/15/19	11/15/19
Thanksgiving Break	11/27/19	11/29/19
Final Exam	12/11/19	12/11/19
Finalize Component Selections	9/16/19	9/27/19
Vision and Puzzle Solving	9/16/19	10/22/19
Coding: Optical Character Recognition	9/16/19	9/30/19
Coding: Distinguish Word Bank from Puzzle	9/25/19	10/11/19
Coding: Integrate OCR with Puzzle Solver	10/1/19	10/22/19
Mechanical	9/30/19	11/4/19
Build: Structural Frame	9/30/19	10/7/19
Brain Storm Pen Holder Solution	9/30/19	10/7/19
Model and Construct Pen Holder	10/8/19	10/29/19
Prototype XY Mechanism	10/14/19	11/4/19
Embedded Software	9/24/19	11/4/19
Coding: Solenoid Pen Actuator	9/24/19	10/15/19
Coding: Timing Belt Motor Movement	9/30/19	11/4/19
Coding: Pen Origin with Limit Switches	10/9/19	11/4/19
Board Fab	9/16/19	10/28/19
Power Board Design and Debug	9/16/19	10/28/19
Control Board Design and Debug	9/16/19	10/28/19
Integration and Testing	10/28/19	12/3/19
Final Power and Control Board Testing	10/28/19	11/15/19
Integrate XY Table With Pen Actuator	11/5/19	11/12/19
Mount Camera to Structure	11/8/19	11/15/19
Connect Power and Control on Main Frame	11/18/19	11/25/19
Cummulative Testing	11/26/19	12/3/19
Final Paper Work	12/2/19	12/11/19
Final Presentation Work	12/2/19	12/11/19

Figure 34 Gantt Chart from Project Proposal



Name	Begin date	End date
Proposal Draft	9/16/19	9/16/19
Fall Break	10/4/19	10/8/19
Midterm Design Review	10/15/19	10/22/19
Subsystem Demo	10/24/19	10/24/19
PCB Layout Deadline	11/15/19	11/15/19
Thanksgiving Break	11/27/19	11/29/19
Coding: T...	12/11/19	12/11/19
Finalize Component Selections	9/16/19	10/18/19
Vision and Puzzle Solving	9/16/19	12/10/19
Coding: Optical Character Recognition	9/16/19	9/30/19
Coding: Distinguish Word Bank from Puzzle	9/25/19	10/11/19
Coding: Integrate OCR with Puzzle Solver	10/1/19	10/22/19
Coding: Image to Wold Coordinates	12/6/19	12/10/19
Coding: Character to Image coordinates	12/2/19	12/10/19
Mechanical	10/9/19	12/6/19
Brainstorm Penholder Solution	10/9/19	11/18/19
Design and Spec XY table	10/25/19	11/8/19
Build XY Table	11/11/19	11/11/19
Model and Construct Penholder	11/19/19	11/26/19
Modify and Redesign Penholder	11/26/19	12/6/19
Embedded Software	10/25/19	12/10/19
Coding: Solenoid Pen Actuator	10/25/19	11/1/19
Coding: Pen Origin with Limit Switches	11/18/19	11/22/19
Coding: Timing Belt Motor Movement	11/27/19	12/10/19
Coding: Line Drawing	12/2/19	12/6/19
Testing: Consistency of Motor Movement	12/6/19	12/10/19
Board Fab	9/30/19	12/2/19
Power-Control Board Design and Debug	9/30/19	11/15/19
Testing and Final Tweaking	11/18/19	12/2/19
Integration and Testing	11/8/19	12/11/19
Mount Camera to Structure	11/8/19	11/15/19
Connect PCB to Main Frame	11/15/19	11/22/19
Cummulative Testing	11/25/19	12/2/19
Code and Test UART	11/29/19	12/6/19
Integrate Penholder with XY Table	12/9/19	12/10/19
Full Pipeline Testing	12/11/19	12/11/19
Final Paper Work	12/11/19	12/16/19
Final Presentation Work	12/11/19	12/16/19

Figure 35 Final Chart of Actual Work

Key	
Color	Role
🟪	Integration
⬜	Joint Work
🟨	Embedded
🟧	Board Fab
🟥	Deadlines
▦	Holidays
🟦	Software
🟩	Mechanical

Figure 36 Gantt Chart Key

Just from the images it’s clear to see how much our actual timeline deviated from what we originally had set out. At the start of the semester, we were far too optimistic about the speed of parts orders and board sendouts. Having our board arrive near the mid-end of November set us back quite a bit as far as testing went. A lot of the embedded software relied on the PCB for testing, so not having it present for so long pushed a considerable amount of the work to the end of the semester. Furthermore, it also took much longer to learn the 3D modelling software that was required to model our penholder and even to modify and tweak it. When we got our first penholder in the middle of November, we realized we would need to remake it, something we hadn’t accounted for in our initial chart. The combined time between learning the software and redesigning the holder delayed quite a bit of the integration testing, and forced us to work on near the project deadline.

Now for a serial task breakdown. We realized that within each subsection (mechanical, software, …) the work would have to be more or less serial, as it was typically one of us on each with the exception of mechanical and occasionally software. As can be seen from the updated chart, nearly all the work done in each phase was more-or-less serial because of this, each person could only work on one thing at a time. We did eventually go to diffuse into other sections as areas of the project were starting to be completed. As an example, everyone was working on the python and embedded code by the end of the semester, since they were the only things that needed to be completed. When looking at the overall project schedule, though, each section of the project was more or less serial due to the fact that we were spread quite thin on manpower - five people, four subcategories.

Due to the fact that we had at least one person dedicated to each subcategory - Nick and Sean on mechanical, Chris on computer vision, Wade on embedded, and Josh on electronics - parallelization between disciplines was high. By the middle of the semester when parts began to come in and circuits were being designed, everyone was working on their own part of the project and preparing for the final integration testing. For example, by early November, Nick was working on the XY table, Sean was developing the pen holder, Josh was finalizing the PCB, wade had nearly all of the embedded code finished, and Chris was working on camera calibration and character read-in. Without this high parallelization between each subsection of the project, we likely would not have finished the project by the deadline.

The following is the breakdown for each member's primary and secondary tasks for the project:

- Chris - Software, primarily in CV and camera calibration
- Josh - Electronics
- Nick - Mechanical and Software
- Sean - Mechanical
- Wade - Embedded

Test Plan

From our Midterm Design Review, the testing plan was as follows:

Mechanical tests

- Full range of motion about the XY frame
- Quick, repeated dropping/pulling of highlighter by solenoid

PCB tests

- Build and check power system first (12V, two separate 5V, 3.3V)
- Add the MSP and test programming a test pin to go high
- Incrementally add and test remaining systems using test points
- Quick engagement of LEDs

Computer Vision subsystem tests

- Incrementally test each stage of the pipeline and compare to expected
- A standard page that contains a suite of tests: upper/lowercase alphabets, a few word search grids, and a small set of whole words.
- Test at various angles and in different lighting conditions

The test plan described earlier in the year during the Midterm Design Review was adhered to for the most part. The actual testing conducted was more thorough as there was uncertainty on what all needed to be done when the first one was created.

The test plan for the PCB was tested almost exactly as described. It was incrementally built and tested starting with the power system. Testing for this step consisted of checking the voltages and ground pins with an oscilloscope. Next, the MSP and JTAG section was connected to the board so that the ability to program the MSP could be tested by setting a single pin high. Once this worked, the remaining systems such as the go button, solenoid, and motors were added and tested one-at-a-time. For the motor drivers, 150 ohm resistors were connected to the outputs and checked with the VirtualBench before testing with the actual motors connected. When the solenoid was being tested, it was also checked whether or not it would be able to lift the highlighter as it could according to our previous calculations. It was discovered that it could do it when they were both on their sides, but it had trouble lifting the highlighter against gravity. This presented a challenge because this was an integral part of system functionality. However, this was able to be fixed by removing the attached spring since the solenoid was also working against the spring's force when trying to pull the highlighter up. Performance of the solenoid was then further improved by positioning its distance from the page such that the arm of the solenoid did not have to full extend to have the highlighter touch the page. This allowed the magnet of the solenoid to be closer to the center of the coil, requiring less magnetic force to pull the highlighter up.

The first version of the pen holder had some errors and had to be redesigned. The mounting holes proved too deep for any normal length screw, so instead of attaching the holder with screws 3M Command Strip Velcro was used instead. This had the added benefit of allowing for micro adjustments of the holders distance off the ground. The inner diameter of the tube portion of the holder had to be increased to accommodate for the width of the zip ties that attached the highlighter to the solenoid. The original tongue on the solenoid side of highlighter-solenoid interconnect was too thick to fit into the slot on the solenoid arm, so a new, thinner version was printed. While the pen holder successfully held the highlighter, and allowed for the highlighter to picked up and placed back down, initial tests showed that highlighter had a tendency to skew off axis while being dragged. To correct for this, a metal washer was placed at the bottom of the pen holder tube to restrict the inner diameter of the tube to a distance closer to the diameter of the highlighter.

For the MSP, the testing procedure was done on a case-by-case basis. For example, after the various pins were initialized, the register window in Code Composer Studio debug mode was used to check that they were properly configured as inputs or outputs and that the outputs were expected. The go button was tested on a breadboard and was connected to the Launchpad to verify that the MSP was properly registering button presses and that the go button's LED was able to be turned on. The other major components like the motor drivers and solenoid could not easily be tested until they were placed on the PCB. The system clocking was verified by toggling a pin using a timer sourced from the master clock and measuring the frequency using a VirtualBench. The PWM frequency for the motors was also checked in a similar manner. In order to test the UART, the Rx and Tx pins were connected together to check if the MSP was receiving back what it was sending. Next, a VirtualBench was used to manually verify the Tx output and check that the baud rate was correct.

Once all of the main components were tested individually and the UART was working, the larger system began to be tested. This consisted of pressing the go button to initiate it, the Jetson sending hard-coded coordinate locations, the MSP acknowledging when it finished a "word" so it could be sent another one, and the Jetson letting the MSP know when it ran out of points. The other system that needed to be tested was the mapping of the real-world locations to a coordinate system in the MSP and the Jetson. For the MSP, the first step involved defining a coordinate system based on the distance travelled per step and definition of how many steps should be in a point. Once this was done, the consistency was tested by having the highlighter move to various systems around the plane and see if it could return to the starting location based only on coordinates. Originally, there was some error with this, which was assumed to be because of inconsistent stepping perhaps due to the very fast starting and stopping of the motors. By slowing down the motors and slightly increasing the number of steps between coordinate points, this issue was fixed and the motors became remarkably consistent. On the Jetson's side, there were some initial difficulties with properly getting the real world mappings but this was solved using calibration techniques described in more detail in the technical section. Once both sides had this functionality working, the accuracy of the real-world mappings to the MSP's coordinate points had to be tested by sending it locations of objects on the word search and seeing it the motors could move there. Unfortunately, this process was not originally as accurate as desired and it involved quite a bit of tweaking with the offsets and different coordinate system precisions to get down. The coordinate system was changed by decreasing the number of steps between points, so that more specific values could be reached. This process was repeated and the design was tweaked until it was working as best as it seemed we could get it, which was not perfect but still pretty good.

Final Results

The criteria for success defined in the proposal were that the system could:

1. Move the highlighter up and down
2. Have precise control of XY location of pen
3. Direct the highlighter to correct spots to draw a line through only the words
4. Recognize the words and grids to solve the word search in software

The system meets the first criteria of being able to pick the highlighter up and down. This part actually works very well and seems to work every time as long as the highlighter is oriented correctly in its holder. If it is not, it still works most of the time but will occasionally get stuck. During the entire Capstone demo, the robot solved 17 puzzles without failing to properly move the highlighter up or down even once. This is actually the most consistent of the criteria.

The system meets the second criteria of having precise control over the XY location of the pen. In order to test this out before the demo, several different test runs were conducted that had hard-coded locations to highlight spread all throughout the coordinate plane. However, several of these locations were designed to end up at the beginning or end of words that had already been drawn. Throughout all of these runs, it was able to land on these specified spots without any noticeable error. Overall, it was very consistent at being able to replicate lines and go to specific spots so this criteria was a success.

The system mostly meets the third criteria of being able to direct the highlighter to correct spots to draw a line through only the words. When the system was drawing, there were some noticeable imperfections in this area but it was reasonably close. The system sometimes missed the beginning or end of words or would go slightly over the end. Additionally, it would sometimes be slightly offset from a word. However, the system was never off by more than 0.5 centimeters from where it should have been and was typically off by less than this. When looking at the solved word searches, the highlighted spots have always been close enough that you can tell what word it was meant to be and sometimes the lines looked perfect. For example, in Figure 34 below, the word “paper” (right to left) is highlighted pretty well while “steel” (diagonal) is slightly off, but it is still clear to see. The reason that this happened is due most likely to imprecisions adding up from the detection of real-world locations from the image, rounding when translating it to MSP coordinates, discrete coordinate points with a set number of steps in-between, and measurements of the distance per step. While it is not perfect and does sometimes barely draw through other words, this criteria is considered a success because it is always close enough to tell the actual word.

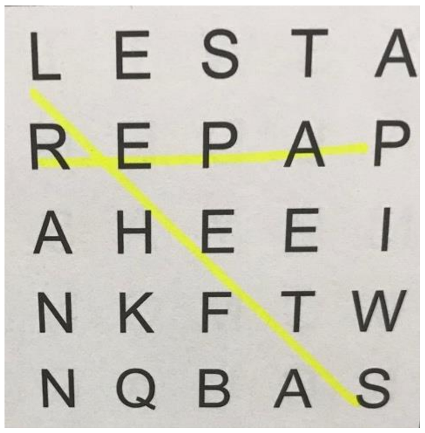


Figure 37 Small Section of Solved Puzzle. Words "Paper" and "Steel" Highlighted

The system mostly meets the fourth criteria of being able to recognize the words and grids to solve the word search in software. Without calculating the actual numbers, the system is able to find over 90% of the words in the grid as it has never missed more than two of the twenty-four words on any single run. As described previously, several steps are in place to correct missed character detection in the word bank and grid, such as trying different letters for words that were not found after the first attempt. The detection could still use some improvement, but getting almost all of the words with a correctness rate of over 90% meets expectations.

Costs

As shown in the spreadsheet "Component Costs" included in the Appendix, the total cost for all of the electrical and mechanical components used in the project was $697.91. However, this includes the MSP430FR2476 Launchpad which costs $15.57 but was only used for production purposes. Additionally, this did not include several components. For example, it does not take the camera into account, as one was available from a previous semester's project. The camera used was a Microsoft LifeCam Studio that retails for around $100 [59] Additionally, the costs for ordering the PCB, which was approximately $33, or the cost of having an independent company, WWW Electronics, solder the motor drivers and MSP onto the board were not considered. Lastly, it does not include the cost of 3D printing two highlighter holders as they were printed for free at the University of Virginia. If this is all taken into account with some approximations as shown in the spreadsheet, the total cost for one unit is approximately $835.

If 10,000 units were manufactured at once, then the production costs would drop. The spreadsheet shows the new costs when the various components are ordered in bulk. However, some components only had one cost listed with no decrease when more were ordered. In these cases, this price was used with the understanding that discounts would most likely actually be available if the components were ordered in big batches. Furthermore, when the discounts were listed, most of them were listed for orders of 1,000 or less. In these cases, the highest order amount was used, but it could possibly be reduced further when ordering 10,000. Using these values, the total calculated for all of the components was $597.96 when not including the MSP Launchpad. With the other factors considered such as the PCBs and cameras, the cost would increase to $750 per unit, a reduction of $85. However, as already stated, this is an over-approximation and the actual costs would be lower.

None of these cost evaluations included labor costs to assemble the device. When considering this, the costs would increase even further. However, if automation were used for some of the assembly, the initial costs would be large, but spreading the costs out through the production of many units would cause it to actually be cheaper in the long-run. Additionally, owning a 3D printer and a reflow oven could save some costs by allowing everything to be manufactured in-house. The parts that could be automated might be the assembly of the chassis, printing the highlighter holder, and the production of the PCB.

Future Work

In the future, one of the main goals is to make the robot be able to highlight all of the words faster. To accomplish this, one of the changes would be to optimize the routing path. Right now it highlights the words in the order they were found which is inefficient and means the highlighter has to travel greater distances than necessary. It is possible to find a more minimal total required distance to be travelled by representing this as a travelling salesman problem (TSP). The problem is not exactly TSP as each word has a start and end location which must be considered. One simplifying approach would be to only consider each word’s beginning location on where to go to next from the previous word’s end location. However this approach would have a very low chance of producing the absolute shortest path, but would definitely be shorter. A second approach would be to consider all possible start and end locations when determining where to go next and then removing the chosen point’s other endpoint from future consideration. This would produce better results but is more algorithmically complex and would add significantly more computation time. This problem is NP-hard, simplistically meaning that the computation time grows significantly with the number of words and quickly becomes too slow to be used reasonably. However, some approximation and acceleration techniques can be used to speed it up to a reasonable level.

A second approach to make the machine faster would be to speed up the motor movement. In order to do so, a different method of travelling to locations would most likely need to be used. Currently, Bresenham’s line algorithm is used to determine where to go next because it is a simple, but effective way to draw diagonal lines with two motors. However, this requires travelling at most a couple of coordinate points at a time, which means there is rapid starting and stopping of the motors. Doing this complicates the use of acceleration and deceleration techniques because this method produced more error over time since the motors only took a couple steps at a time before stopping. Since ramping is not used for acceleration, these motors would currently stall if their step speed increased. This issue could be fixed by removing Bresenham’s and calculating new stepping frequencies for each motor to achieve different angled paths, so that the motors would run longer and ramping could be used. A potential pitfall of this is that a slightly off angle could make a significant difference, especially for longer words. Another fix would be to introduce a technique for calculating and correcting for error, which would require further research and learning before it could be implemented on our end.

Moving away from speed, another desired improvement is to increase the reliability when capturing and reading in the word bank from the page. Currently, around 90% of words are reliably scanned in, and with the existing permutative solving algorithm, that number can reach approximately 95%. However, there are still occasional words that slip through the algorithm and are thus not going to be highlighted by the robot. These words are primarily complex proper nouns that dictionaries and spell checkers are not aware of or words with spaces and special characters that are hard for the spell checker to parse through. A possible remedy for these words is to train the JamSpell checker on a more detailed data set, as it currently has only been trained on 600,000 lines from Wikipedia and news articles. While the size of the training data was sufficient to capture and correct most words coming it, there is still room for improvement in special case scenarios. Another addition that would help in the correction of the solution space would be to incorporate pytesseract confidence levels into the bank read-in. Currently, the robot uses a hard-coded custom swap out table based on similar characters, and the robot exhaustively checks each combination against the word puzzle. Instead of doing this, confidence levels from Tesseract could be used that would allow for smarter permutative checking that is far more reliable than the current guess and check method.

Another main goal for the future is increasing the highlighting precision. At this point, the robot highlights the words very close to the actual spot, but sometimes has an offset or misses a letter. A potential improvement might come from increasing the coordinate system's resolution by reducing the number of steps. As discussed previously, making this change would probably require switching to a different line algorithm, such as changing the motor periods, as reducing the steps per point in its current state causes some problems. Another fix could come on the Jetson's side by using more calibration points for more accurate real world distances. Solving the Perspective-n-Point problem only requires 4 points at a minimum, but we could have used a more dense calibration scheme of 9 points to further increase accuracy. Furthermore, a more accurate measure of the actual distance traveled per step could be taken. By further increasing all of these precisions, it is hoped that the word search solver would be capable of much better highlightings as the errors are most likely caused by compounding imprecisions.

References

[1]D. M. Threlkel, “NEMA Enclosure Types,” no. 17, p. 9.

[2]“NEMA connector,” Wikipedia. 25-Nov-2019.

[3]“IPC-2221 Generifc Standard on Printed Board Design,” p. 10, 1998.

[4] “Code Composer Studio User’s Guide — Code Composer Studio 9.2.0 Documentation.” [Online]. Available: https://software-dl.ti.com/ccs/esd/documents/users_guide/index.html. [Accessed: 16-Dec-2019].

[5] “What Is VirtualBench? - National Instruments.” [Online]. Available: https://www.ni.com/en-us/shop/electronic-test-instrumentation/virtualbench/what-is-virtualbench.html. [Accessed: 16-Dec-2019].

[6]“MSP430FR2476 16 MHz ultra-low-power microcontroller with 64 KB FRAM, 8 KB RAM, 12-bit ADC, 43 IO, 5 16-bit timers | TI.com.” [Online]. Available: https://www.ti.com/product/MSP430FR2476?utm_source=google&utm_medium=cpc&utm_campaign=epd-null-null-gpn_en-cpc-pf-google-wwe&utm_content=msp430fr2476&ds_k=%257b_dssearchterm%257d&DCM=yes&gclid=EAIaIQobChMIvuHa4q275gIVi4CfCh3MQAmlEAAYASAAEgK6G_D_BwE&gclsrc=aw.ds. [Accessed: 16-Dec-2019].

[7]“What is Multisim™? - National Instruments.” [Online]. Available: https://www.ni.com/en-us/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-multisim.html. [Accessed: 16-Dec-2019].

[8]“Ultiboard - National Instruments.” [Online]. Available: http://www.ni.com/en-us/shop/select/ultiboard. [Accessed: 16-Dec-2019].

[9]“OpenCV 4.1.1.” [Online]. Available: https://opencv.org/opencv-4-1-1/. [Accessed: 16-Dec-2019].

[10]tesseract-ocr/tesseract. tesseract-ocr, 2019.

[11]“SOLIDWORKS™ 3D Design Software - Dassault Systèmes®.” [Online]. Available: https://www.3ds.com/products-services/solidworks/. [Accessed: 16-Dec-2019].

[12]“SOLIDWORKS™ 3D Design Software - Dassault Systèmes®.” [Online]. Available: https://www.3ds.com/products-services/solidworks/. [Accessed: 16-Dec-2019].

[13]“The Environmental Impacts of Using Paper | Environmental Professionals Network.” .

[15]“1926.55 App A - Gases, vapors, fumes, dusts, and mists. | Occupational Safety and Health Administration.” [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1926/1926.55AppA. [Accessed: 16-Dec-2019].

[16]“1910.211 - Definitions. | Occupational Safety and Health Administration.” [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.211. [Accessed: 16-Dec-2019].

[17]“OpenBuilds Part Store.” [Online]. Available: https://openbuildspartstore.com/. [Accessed: 16-Dec-2019].

[18]“NVIDIA Jetson Nano Developer Kit,” NVIDIA. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano-developer-kit/. [Accessed: 16-Dec-2019].

[19]C. W. Cox and P. R. Lantz, “Calibration system for vision-based automatic writing implement,” US6885759B2, 26-Apr-2005.

[20]T. Morita, “Apparatus and a method of mounting electronic components,” US5992013A, 30-Nov-1999.

[21]S. F. Rutkowski, C. U. Hardwicke, M. F. X. Gigliotti, and M. R. Jackson, “Robotic pen,” US7351290B2, 01-Apr-2008.

[22]“VES150PS12 XP Power | Power Supplies - External/Internal (Off-Board) | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/xp-power/VES150PS12/1470-4464-ND/7897384. [Accessed: 16-Dec-2019].

[23]“KPJX-4S-S Kycon, Inc. | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/kycon-inc/KPJX-4S-S/2092-KPJX-4S-S-ND/9990088. [Accessed: 16-Dec-2019].

[24]“0685H9150-01 Bel Fuse Inc. | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/bel-fuse-inc/0685H9150-01/507-1944-1-ND/5466716. [Accessed: 16-Dec-2019].

[25]“LTST-C171GKT Lite-On Inc. | Optoelectronics | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/lite-on-inc/LTST-C171GKT/160-1423-1-ND/386792. [Accessed: 16-Dec-2019].

[26]“12V-SB-CW-12M Inspired LED, LLC | Optoelectronics | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/inspired-led-llc/12V-SB-CW-12M/1647-1021-1-ND/5866500. [Accessed: 16-Dec-2019].

[27]“691311400102 Würth Elektronik | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/wurth-electronics-inc/691311400102/732-2823-ND/2508592. [Accessed: 16-Dec-2019].

[28]“691351400002 Würth Elektronik | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/w-rth-elektronik/691351400002/732-2811-ND/2508580. [Accessed: 16-Dec-2019].

[29]“SMAJ12A Littelfuse Inc. | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/littelfuse-inc/SMAJ12A/SMAJ12ALFCT-ND/762482. [Accessed: 16-Dec-2019].

[30]“OKX-T/5-D12N-C Murata Power Solutions | Mouser,” Mouser Electronics. [Online]. Available: https://www.mouser.com/ProductDetail/580-OKX-T-5-D12N-C. [Accessed: 16-Dec-2019].

[31]“10-02931 Tensility International Corp | Cable Assemblies | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/tensility-international-corp/10-02931/839-1518-ND/9686414. [Accessed: 16-Dec-2019].

[32]“2920L500/16MR Littelfuse Inc. | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/littelfuse-inc/2920L500-16MR/F6136CT-ND/4147146. [Accessed: 16-Dec-2019].

[33]“ESD9X5.0ST5G ON Semiconductor | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/on-semiconductor/ESD9X5.0ST5G/ESD9X5.0ST5GOSCT-ND/2120624. [Accessed: 16-Dec-2019].

[34]“MEZD71202A-G Monolithic Power Systems Inc. | Power Supplies - Board Mount | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/monolithic-power-systems-inc/MEZD71202A-G/1589-1465-ND/6823828. [Accessed: 16-Dec-2019].

[35]“0ZCG0200AF2B Bel Fuse Inc. | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/bel-fuse-inc/0ZCG0200AF2B/507-2074-1-ND/6210551. [Accessed: 16-Dec-2019].

[36]“MEZD71202A-F Monolithic Power Systems Inc. | Power Supplies - Board Mount | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/monolithic-power-systems-inc/MEZD71202A-F/1589-1464-ND/6823827. [Accessed: 16-Dec-2019].

[37]“ESD9X3.3ST5G ON Semiconductor | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/on-semiconductor/ESD9X3.3ST5G/ESD9X3.3ST5GOSCT-ND/1967048. [Accessed: 16-Dec-2019].

[38]“MSP430FR2476TPTR Texas Instruments | Mouser,” Mouser Electronics. [Online]. Available: https://www.mouser.com/ProductDetail/595-MSP430FR2476TPTR. [Accessed: 16-Dec-2019].

[39]“ABS25-32.768KHZ-T Abracon LLC | Crystals, Oscillators, Resonators | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/abracon-llc/ABS25-32.768KHZ-T/535-9166-1-ND/675683. [Accessed: 16-Dec-2019].

[40]“PRG18BB471MB1RB Murata Electronics | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/murata-electronics/PRG18BB471MB1RB/490-2473-1-ND/588687. [Accessed: 16-Dec-2019].

[41]“0702461404 Molex | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/molex/0702461404/WM4079-ND/2421597. [Accessed: 16-Dec-2019].

[42]“61300311121 Würth Elektronik | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/wurth-electronics-inc/61300311121/732-5316-ND/4846825. [Accessed: 16-Dec-2019].

[43]“Xtension Limit Switch Kit - OpenBuilds Part Store.” [Online]. Available: https://openbuildspartstore.com/xtension-limit-switch-kit/. [Accessed: 16-Dec-2019].

[44]“JB15HBPC-JB NKK Switches | Switches | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/nkk-switches/JB15HBPC-JB/360-2560-ND/2104205. [Accessed: 16-Dec-2019].

[45]“3992 Adafruit Industries LLC | Motors, Solenoids, Driver Boards/Modules | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/adafruit-industries-llc/3992/1528-2797-ND/9826285. [Accessed: 16-Dec-2019].

[46]“TIP102 STMicroelectronics | Discrete Semiconductor Products | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/stmicroelectronics/TIP102/497-2546-5-ND/603571. [Accessed: 16-Dec-2019].

[47]“PMEG4030EP,115 Nexperia USA Inc. | Discrete Semiconductor Products | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/nexperia-usa-inc/PMEG4030EP115/1727-5841-1-ND/2697505. [Accessed: 16-Dec-2019].

[48]“577002B00000G Aavid, Thermal Division of Boyd Corporation | Fans, Thermal Management | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/aavid-thermal-division-of-boyd-corporation/577002B00000G/HS105-ND/102507. [Accessed: 16-Dec-2019].

[49]“NEMA 17 Stepper Motor,” OpenBuilds Part Store. [Online]. Available: https://openbuildspartstore.com/nema-17-stepper-motor/. [Accessed: 16-Dec-2019].

[50]“A3982SLBTR-T Allegro MicroSystems | Integrated Circuits (ICs) | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/allegro-microsystems-llc/A3982SLBTR-T/620-1299-1-ND/2000030. [Accessed: 16-Dec-2019].

[51]“691311400104 Würth Elektronik | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/products/en?keywords=691311400104. [Accessed: 16-Dec-2019].

[52]“ESDA14V2SC5Y STMicroelectronics | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/stmicroelectronics/ESDA14V2SC5Y/497-13408-1-ND/3737644. [Accessed: 16-Dec-2019].

[53]Texas Instruments, MSP430FR4xx and MSP430FR2xx family. .

[54]F. Ozinov, bakwc/JamSpell. 2019.

[55]“OpenCV: Camera Calibration and 3D Reconstruction.” [Online]. Available: https://docs.opencv.org/4.1.1/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a. [Accessed: 16-Dec-2019].

[56]“tesseract-ocr/tesseract,” GitHub. [Online]. Available: https://github.com/tesseract-ocr/tesseract. [Accessed: 16-Dec-2019].

[57]“OpenCV: Structural Analysis and Shape Descriptors.” [Online]. Available: https://docs.opencv.org/4.1.1/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0. [Accessed: 16-Dec-2019].

[58]“tesseract-ocr/tesseract,” GitHub. [Online]. Available: https://github.com/tesseract-ocr/tesseract. [Accessed: 16-Dec-2019].

[59]“Microsoft Webcam: LifeCam Studio | Microsoft Accessories.” [Online]. Available: https://www.microsoft.com/accessories/en-us/products/webcams/lifecam-studio/q2f-00013. [Accessed: 16-Dec-2019].

Appendix

Component Costs:

The following table shows the various electrical and mechanical components used and their costs. One important thing to note is that the green boxes in the “Price per 10000” column indicate that the price shown is for quantities less than 10,000. Blank entries indicate that the only price found were per unit. Furthermore, at the end of the components list, the first set of orange cells show the total costs per unit and per 10,000 units only considering these components. The second set of orange cells are for the whole cost including things like the PCB.

Quantity	Manufacturer	MFG Part Number	Vendor Part Number	Price	Price per 10000	Hyperlink
1	Jetson Nano	DEV-15297	474-DEV-15297	$123.75		https://www.mouser.com/ProductDetail/SparkFun/DEV-15297?qs=vLWxofP3U2w1e%252BbCN1adVQ%3D%3D
1	Texas Instruments	LP-MSP430FR2476	296-53618-ND	$15.57		https://www.digikey.com/product-detail/en/texas-instruments/LP-MSP430FR2476/296-53618-ND/10314032
1	Adafruit	3992	1528-2797-ND	$7.50		https://www.digikey.com/product-detail/en/adafruit-industries-llc/3992/1528-2797-ND/9826285
20	Inspired LED	12V-SB-CW-12M	1647-1021-1-ND	$1.07	$0.66	https://www.digikey.com/product-detail/en/inspired-led-llc/12V-SB-CW-12M/1647-1021-1-ND/5866500

1	XP Power	VES150PS12	1470-4464-ND	$68	$57.12	https://www.digikey.com/product-detail/en/xp-power/VES150PS12/1470-4464-ND/7897384
1	CNC Tech	800-18-2-2-SVT0-BL-00200	1175-1298-ND	$8.97	$5.34	https://www.digikey.com/product-detail/en/cnc-tech/800-18-2-2-SVT0-BL-00200/1175-1298-ND/3528263
1	Tensility International Corp	10-02931	839-1518-ND	$4.70	$2.54	https://www.digikey.com/product-detail/en/tensility-international-corp/10-02931/839-1518-ND/9686414
1	OpenBuilds	2365-Bundle(Sleek Silver)		$281.99		Here
2	OpenBuilds	623	819368021264	$17.99		Here
2	OpenBuilds	490	819368021653	$1.49		Here
1	OpenBuilds	110-pack(8mm)	819368021820	$1.29		Here
1	OpenBuilds	536-Pack	819368021646	$2.99		Here
1	OpenBuilds	280-LP(250mm)	819368020403	$2.63		Here
1	OpenBuilds	150-LP(500mm)	819368020137	$2.63		Here
2	Murata Electronics	PRG18BB471MB1RB	490-2473-1-ND	$0.54	$0.21	https://www.digikey.com/product-detail/en/murata-electronics/PRG18BB471MB1RB/490-2473-1-ND/588687
5	KEMET	C0805C220J5GACTU	399-1113-1-ND	$0.10	$0.02	https://www.digikey.com/product-detail/en/kemet/C0805C220J5GACTU/399-1113-1-ND/411388
1	Abracon	ABS25-32.768KHZ-T	535-9166-1-ND	$0.63	$0.33	https://www.digikey.com/product-detail/en/abracon-llc/ABS25-3

						2.768KHZ-T/535-9166-1-ND/675683
1	Wurth	61300311121	732-5316-ND	$0.13	$0.06	https://www.digikey.com/product-detail/en/wurth-electronics-inc/61300311121/732-5316-ND/4846825
1	Molex	0702461404	WM4079-ND	$1.86	$0.90	https://www.digikey.com/product-detail/en/molex/0702461404/WM4079-ND/2421597
5	Yageo	CC0805JRNPO9BN102	311-1122-1-ND	$0.13	$0.03	https://www.digikey.com/product-detail/en/yageo/CC0805JRNPO9BN102/311-1122-1-ND/303032
20	Samsung	CL21B104KBCNNNC	1276-1003-1-ND	$0.10	$0.01	https://www.digikey.com/product-detail/en/samsung-electro-mechanics/CL21B104KBCNNNC/1276-1003-1-ND/3889089
1	ON Semiconductor	TIP102	497-2546-5-ND	$0.70	$0.25	https://www.digikey.com/product-detail/en/stmicroelectronics/TIP102/497-2546-5-ND/603571
5	Stackpole	RNCP0805FTD1K00	RNCP0805FTD1K00CT-ND	$0.10	$0.01	https://www.digikey.com/product-detail/en/stackpole-electronics-inc/RNCP0805FTD1K00/RNCP0805FTD1K00CT-ND/2240568
2	Allegro	A3982SLBTR-T	620-1299-1-ND	$4.53	$2.84	https://www.digikey.com/product-detail/en/allegro-microsystems-llc/A3982SLBTR-T/620-1299-1-ND/2000030
10	TE Connectivity	RL73H2HR22FTE	A138600CT-ND	$0.53	$0.17	https://www.digikey.com/product-detail/en/te-connectivity-passive-product/RL73H2HR22FTE/A138600CT-ND/9372001
5	Samsung	CL21B224KBFNNNE	1276-1093-1-ND	$0.11	$0.02	https://www.digikey.com/product-detail/en/samsung-electro-me

						chanics/CL21B224KBFNNNE/1276-1093-1-ND/3889179
15	Samsung	CL21A106KQCLRNC	1276-2405-1-ND	$0.09	$0.02	https://www.digikey.com/product-detail/en/samsung-electro-mechanics/CL21A106KQCLRNC/1276-2405-1-ND/3890491
5	Panasonic	EEU-FR1V680	P14435-ND	$0.10	$0.06	https://www.digikey.com/product-detail/en/panasonic-electronic-components/EEU-FR1V680/P14435-ND/2433569
2	NXP Semiconductors	PMEG4030EP,115	1727-5841-1-ND	$0.43	$0.13	https://www.digikey.com/product-detail/en/nexperia-usa-inc/PMEG4030EP115/1727-5841-1-ND/2697505
5	Samsung	CL21A226KQCLRNC	1276-6687-1-ND	$0.17	$0.03	https://www.digikey.com/product-detail/en/samsung-electro-mechanics/CL21A226KQCLRNC/1276-6687-1-ND/5961546
5	STMicroelectronics	ESDA14V2SC5Y	497-13408-1-ND	$0.45	$0.13	https://www.digikey.com/product-detail/en/stmicroelectronics/ESDA14V2SC5Y/497-13408-1-ND/3737644
5	Samsung	CL21B105KAFNNNE	1276-1066-1-ND	$0.11	$0.02	https://www.digikey.com/product-detail/en/samsung-electro-mechanics/CL21B105KAFNNNE/1276-1066-1-ND/3889152
5	Yageo	RC0805FR-0747KL	311-47.0KCRCT-ND	$0.10	$0.01	https://www.digikey.com/product-detail/en/yageo/RC0805FR-0747KL/311-47.0KCRCT-ND/730920
5	Panasonic	ERJ-PB6D1471V	P21036CT-ND	$0.25	$0.03	https://www.digikey.com/product-detail/en/panasonic-electronic-components/ERJ-PB6D1471V/P21036CT-ND/6215291

1	NKK Switches	JB15HBPC-JB	360-2560-ND	$4.83	$3.16	https://www.digikey.com/product-detail/en/nkk-switches/JB15HBPC-JB/360-2560-ND/2104205
1	Nexperia USA	BC817-25,235	1727-6184-1-ND	$0.14	$0.03	https://www.digikey.com/product-detail/en/nexperia-usa-inc/BC817-25235/1727-6184-1-ND/2762681
5	Yageo	RC0805FR-0732K4L	311-32.4KCRCT-ND	$0.10	$0.01	https://www.digikey.com/product-detail/en/yageo/RC0805FR-0732K4L/311-32.4KCRCT-ND/730806
6	Stackpole	RNCP0805FTD150R	RNCP0805FTD150RCT-ND	$0.10	$0.01	https://www.digikey.com/product-detail/en/stackpole-electronics-inc/RNCP0805FTD150R/RNCP0805FTD150RCT-ND/2240550
20	ON Semiconductor	ESD9X3.3ST5G	ESD9X3.3ST5GOSCT-ND	$0.19	$0.04	https://www.digikey.com/product-detail/en/on-semiconductor/ESD9X3.3ST5G/ESD9X3.3ST5GOSCT-ND/1967048
6	Wurth	691311400102	732-2823-ND	$0.41	$0.25	https://www.digikey.com/product-detail/en/wurth-electronics-inc/691311400102/732-2823-ND/2508592
10	ON Semiconductor	ESD9X5.0ST5G	ESD9X5.0ST5GOSCT-ND	$0.14	$0.03	https://www.digikey.com/product-detail/en/on-semiconductor/ESD9X5.0ST5G/ESD9X5.0ST5GOSCT-ND/2120624
6	Yageo	RC0805FR-072KL	311-2.00KCRCT-ND	$0.10	$0.01	https://www.digikey.com/product-detail/en/yageo/RC0805FR-072KL/311-2.00KCRCT-ND/730611
2	Littlefuse	SMAJ12A	SMAJ12ALFCT-ND	$0.40	$0.09	https://www.digikey.com/product-detail/en/littelfuse-inc/SMAJ12A/SMAJ12ALFCT-ND/762482

5	Panasonic	ERJ-P06J511V	P510ADCT-ND	$0.11	$0.02	https://www.digikey.com/product-detail/en/panasonic-electronic-components/ERJ-P06J511V/P510ADCT-ND/525521
10	Lite-On	LTST-C171GKT	160-1423-1-ND	$0.28	$0.05	https://www.digikey.com/product-detail/en/lite-on-inc/LTST-C171GKT/160-1423-1-ND/386792
2	Littlefuse	NANOSMD500LR-2	NANOSMD500LR-2CT-ND	$0.81	$0.36	https://www.digikey.com/product-detail/en/littelfuse-inc/NANOSMD500LR-2/NANOSMD500LR-2CT-ND/7321653
10	Keystone	5011	36-5011-ND	$0.34	$0.14	https://www.digikey.com/product-detail/en/keystone-electronics/5011/36-5011-ND/255333
10	Keystone	5010	36-5010-ND	$0.34	$0.14	https://www.digikey.com/product-detail/en/keystone-electronics/5010/36-5010-ND/255332
1	Monolithic Power Systems	MEZD71202A-F	1589-1464-ND	$5.22	$4.17	https://www.digikey.com/product-detail/en/monolithic-power-systems-inc/MEZD71202A-F/1589-1464-ND/6823827
5	Samsung	CL21B225KPFNNNE	1276-1188-1-ND	$0.13	$0.03	https://www.digikey.com/product-detail/en/samsung-electro-mechanics/CL21B225KPFNNNE/1276-1188-1-ND/3889274
5	Stackpole	RMCF0805FT68R0	RMCF0805FT68R0CT-ND	$0.10	$0.00	https://www.digikey.com/product-detail/en/stackpole-electronics-inc/RMCF0805FT68R0/RMCF0805FT68R0CT-ND/2418450
5	Littlefuse	0805L200SLTHYR	F5780CT-ND	$1.91	$0.86	https://www.digikey.com/product-detail/en/littelfuse-inc/0805L200SLTHYR/F5780CT-ND/3661922

1	Monolithic Power Systems	MEZD71202A-G	1589-1465-ND	$5.22	$4.17	https://www.digikey.com/product-detail/en/monolithic-power-systems-inc/MEZD71202A-G/1589-1465-ND/6823828
5	Bel Fuse	0685H9150-01	507-1944-1-ND	$0.34	$0.19	https://www.digikey.com/product-detail/en/bel-fuse-inc/0685H9150-01/507-1944-1-ND/5466716
20	Keystone	5012	36-5012-ND	$0.34	$0.14	http://www.digikey.com/product-detail/en/5014/36-5014-ND/255336
2	Wurth	691311400104	732-2825-ND	$0.83	$0.50	https://www.digikey.com/products/en?keywords=691311400104
1	Kycon	KPJX-4S-S	2092-KPJX-4S-S-ND	$2.23	$1.16	https://www.digikey.com/product-detail/en/kycon-inc/KPJX-4S-S/2092-KPJX-4S-S-ND/9990088
1	Aavid	577002B00000G	HS105-ND	$0.24	$0.16	https://www.digikey.com/product-detail/en/aavid-thermal-division-of-boyd-corporation/577002B00000G/HS105-ND/102507
1	Aavid	4880G	HS417-ND	$2.30	$1.54	https://www.digikey.com/product-detail/en/aavid-thermal-division-of-boyd-corporation/4880G/HS417-ND/1625654
2	Wurth	691351400004	732-2813-ND	$2.81	$1.80	https://www.digikey.com/product-detail/en/w-rth-elektronik/691351400004/732-2813-ND/2508582
6	Wurth	691351400002	732-2811-ND	$1.24	$0.76	https://www.digikey.com/product-detail/en/w-rth-elektronik/691351400002/732-2811-ND/2508580
1	Texas Instruments	MSP430FR2476TPTR	595-MSP430FR2476TPTR	$4.66	$2.25	https://www.mouser.com/ProductDetail/Texas-Instruments/MSP430FR2476TPTR?qs=sGAEpiMZZ

						Mve4%2FbfQkoj%252BOEyn8p2BPKaS4cts2BXbo4%3D
1	Murata Power Solutions	OKX-T/5-D12N-C	580-OKX-T/5-D12N-C	$6.20	$4.87	https://www.mouser.com/ProductDetail/Murata-Power-Solutions/OKX-T-5-D12N-C?qs=%2Fha2pyFadui3oegDspIAZ2JIneSKfMI5Hkx%252B35klhqZMdhvS2LH6fw%3D%3D
Cost/unit	**697.905**					
Cost/10000	**$597.96**				* less than 10000 units	
Additional Costs						
PCB	$33					
WWW Electronics	$10	approximation				
LifeCam Studio Camera	$100					
3D Printing	$10	approximation				
Total Cost/unit	**$835**					
Total Cost/unit	**$750.91**					