ECE 445

Spring 2024

Senior Project Design Document

Automatic Puzzle Solver

Team 71
Eric Chen
Conor Devlin
Alex Kim

TA: Angquan Yu

Abstract

This document presents the development of an automatic puzzle-solving robot designed to autonomously assemble 3x3 jigsaw puzzles. The robot integrates a precision-controlled robotic arm equipped with a suction device, a belt and pulley system powered by stepper motors for precise movement, and an OpenCV-based computer vision system to identify puzzle pieces through color recognition. The project aims to provide an innovative solution for puzzle enthusiasts and individuals facing challenges in manually solving jigsaw puzzles. The document outlines the problem statement, proposed solution, high-level requirements, and the design process, including subsystem overviews, software design, and tolerance analysis. It also presents a cost analysis, project schedule, and addresses ethical and safety concerns related to the development and operation of the automatic puzzle solver. The successful implementation of the project demonstrates the potential for automation in recreational activities and highlights the possibilities for future enhancements and alternative applications.

Contents

1 Introduction

1.1 Problem

In the realm of recreational activities assembling jigsaw puzzles remains a popular pastime. There are many times when we are in the middle of solving a jigsaw puzzle and get stuck. For some people like children, manually solving jigsaw puzzles can be a daunting task. Many pieces look similar to the human eye, and this can be frustrating when trying to do what is deemed a relaxing activity.

1.2 Solution

Our project aims to develop an innovative solution for puzzle enthusiasts and those facing challenges in manually assembling jigsaw puzzles. The project involves designing and implementing an Automatic Jigsaw Puzzle Solver equipped with a precision-controlled robotic arm attached to a suction device. The objective is to create a user-friendly device capable of autonomously solving puzzles of varying complexities. Our goal will be to be able to solve a 3x3 puzzle.

The setup will be as follows, a robot machine overlay will be placed over the jigsaw puzzle. We will have a series of pulleys and belts to move a robotic arm capable of extending in z direction via a linear actuator to pick up puzzle pieces. This robotic arm will be moved in the x and y direction using the belt and pulley system which is powered by stepper motors (reference visual aid). The robot will start by scanning each puzzle piece using an OpenCV-compatible camera. Our circuit board which is attached to a computer will then calculate exactly where each puzzle piece should be connected. Using computer vision, we will then grab each puzzle piece and move it to the desired location to complete the puzzle.

The robotic arm uses a linear actuator, suction cup, and camera to pick up the puzzle pieces. The linear actuator is needed so we can move puzzle pieces over each other and not drag the pieces along the table. The suction cup will be small enough to form a seal on the puzzle piece and it will be powered by a vacuum pump through a pipe. The camera will be used to identify the pieces and precisely place them together.

1.3 Visual Aid

Refer to Fig B.11 in appendix B. The green circles around the webcam and the solenoid indicate purposeful design alternatives. The red X over the PCB indicates an unplanned design alternative.

1.4 High-Level Requirements

The following are the high-level requirements we hoped to accomplish:

- The mechanical movement will be precise to 1 mm of movement

The 1 mm precision of movement requirement allowed us to consistently place the puzzle pieces in the correct spot on the finished grid. We originally made this requirement because we wanted to solve a real puzzle with small pieces that had tabs. If the pieces had tabs that had to fit into each other then we would have needed that 1mm precision to complete the puzzle. We moved to colored squares to simplify the project but the 1mm precision helped us maintain consistency with our open-loop control structure.

- We can correctly identify where individual puzzle pieces are located on the puzzle

This is an essential component of the puzzle-solving machine. If it could not identify where the individual pieces were located on the completed puzzle then the machine would not be able to solve the puzzle. The identification of the puzzle pieces was carried out with OpenCV with color and contour recognition which was always the plan.

- We can complete the whole 3x3 piece puzzle in 7 minutes

Solving the puzzle in a reasonable amount of time was a practical requirement that was not essential to the design. We wanted to show the whole operation in the demo and we wanted to debug the puzzle efficiently which would be difficult if the puzzle took a long time to solve.

2 Design

2.1 Block Diagram

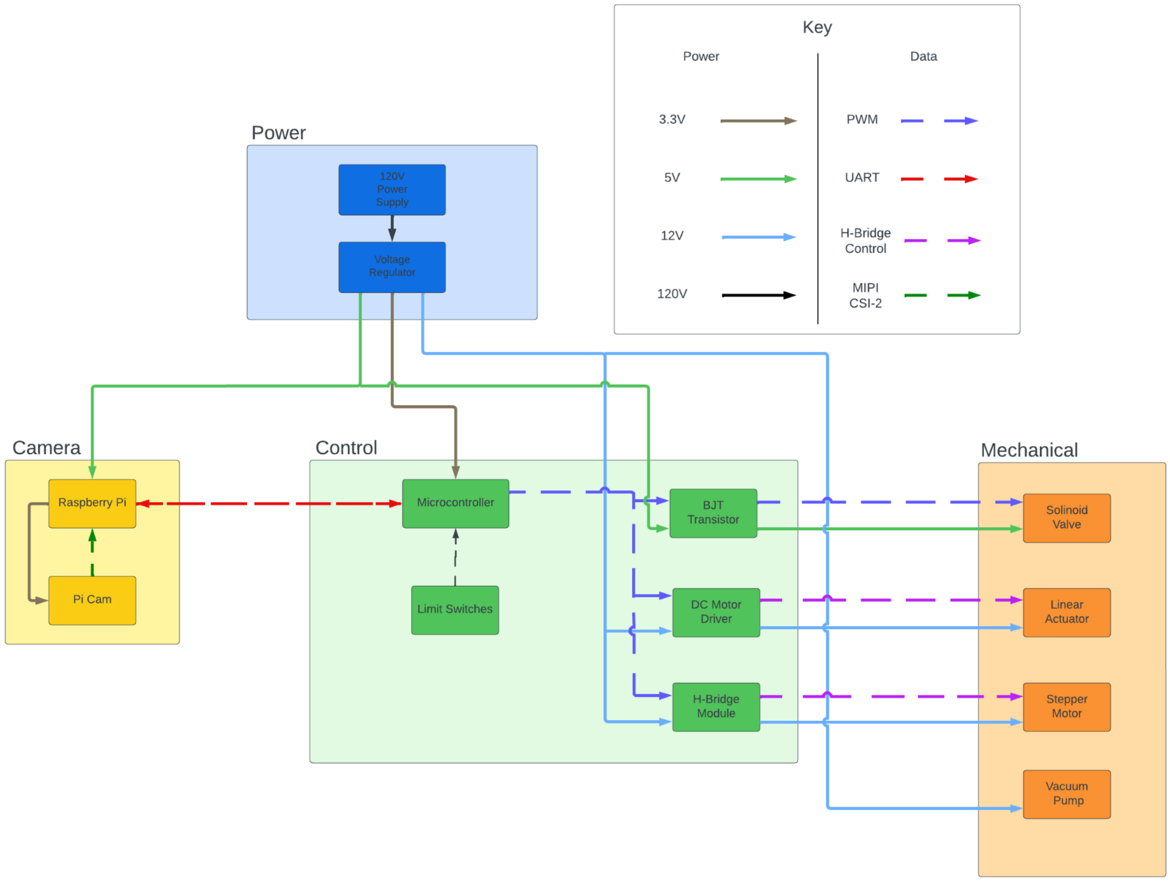


Figure 2.3.1

2.2 Physical Design

Refer to figures B.12 and B.13 in appendix B. Figure B.12 is the robotic movement scheme from an overhead view. Figure B.13 is the robotic arm or head from a side view.

2.3 Design Alternatives

Referencing figure 2.3.1 in section 2.3 you can see the changes to our design. These changes happened as we moved forward with the project and changed our needs. The three main changes to the project are the change from the pi-cam to the webcam for the openCV algorithm, the addition of the solenoid in between the pump and the suction cup, and the Arduino instead of our PCB.

Solenoid: Our original plan was to use a BJT transistor as a switch to allow our microcontroller to turn on and off the suction to pick up and drop the puzzle piece. During our design review, we presented our visual aid to Professor Gruev and he told us that we will need a solenoid. He was correct. When we got the pump we realized that the air only went one way and without a solenoid the tube would always have a vacuum in it. The solenoid was controlled with a BJT transistor and the pump was always on. When we would want the suction cup to pick up a piece we would turn on the solenoid to connect the suction cup with the pump. When we would want the suction cup to drop the piece we would turn off the solenoid which would connect the suction cup to regular air pressure.

Arduino: Our original design was to use a microcontroller on our PCB to control our project. We started using a breadboard and Arduino to test out the functionality of our components to learn how to correctly use them before we started using the pcb. We started late on the PCB and tried to design the entire PCB at once. The pcb order arrived later than we expected and we did not solder it until a few days before the demo. We tried to program it and found that there was a short between the power and ground of the programming header. This is why we were not able to use the PCB as originally planned. We just used the Arduino and breadboard design that we were testing with for our final demo.

Webcam: In our original design, our team decided to use a Pi Camera v3 to interface with the Raspberry Pi 4 that we were using as our main processing unit in order to run OpenCV to detect features and colors. However, due to many complications and compatibility issues with the OS of the Pi, we decided that it was best to stick with a normal video webcam as it would interface more smoothly with the legacy camera functionality of the 32-bit Legacy Bullsye operating system that we had downgraded to at the time. Overall, this switch made for less roadblocks during development by allowing us to test features on our own laptops instead of needing to use the Raspberry Pi everytime. The web camera that we used also happened to have a higher resolution than the Pi Camera which was also a plus when developing an HSV algorithm with OpenCV.

2.4 Subsystem Overview

2.4.1 Power Subsystem

The subsystem will contain the circuitry to convert power to the needed voltage levels to power our puzzle-solving robot machine and all of its parts. This system will distribute power to the appropriate parts of the robot such as the stepper motors (powering the belts and pulleys to move the arm), the linear actuator to extend the robot arm in the z-direction, and all the components of the PCB and our camera used for computer vision.

The kit provided to us by the department can plug into the wall and supply 5V and 12V of DC power in relation to common. We will need 3.3V to power the logic in the motor controllers. We will make a power bus with 3.3V by dividing the 5V with resistors. The logic in the motor controllers will have a current from 1 to 20 microamperes.

2.4.2 Control Subsystem

The control subsystem provides the microcontroller, an ATMEGA 328 PB, necessary to efficiently operate all mechanical components including the motor drivers, linear actuator, and vacuum pump. The limit switches will be used to zero our stepper motors. We will use the start button to start and stop the machine.

Upon start-up, the Arduino will spin the stepper motors to move the car, which is the mechanical part that will be attached to the belt to physically move things, to the limit switch. The microcontroller will zero the stepper motors, which means the microcontroller will start recording the position by counting the steps.

2.4.3 Camera Control Subsystem

Our camera subsystem will consist of the Raspberry Pi Camera Module 3 along with its connections to our off-board module, which will be a Raspberry Pi. For the off-board controller specifically, we will use a Raspberry Pi 4 (4GB version). Our camera will be 25 × 24 × 12.4 mm in size and around 4 grams in weight. It has a 12MP, 120 Degree Wide pre-attached lens. We will use the highest resolution available which is 2304 × 1296p56 for video and still image capture. While it is stated that this camera comes with a 120-degree wide-angle lens, the true horizontal field of view is more towards 102 degrees. This camera will be directly attached to the Raspberry Pi 4 (RP4) and will stream/save video and still images directly to the RP4 to be used with OpenCV for our puzzle image detection software stack. We will be using UART protocol communication, a type of serial communication, between the RP4 and the Arduino Uno (on-board microcontroller). Specifically, the RP4 GPIO 14 (TX) pin will be directly connected to the D0 (PD0, RX) pin on the Arduino. The baud rate of the RP4 will be configured to be at 115200. The special bootloader configuration file on the RP4 will ensure BOOT_UART flags are enabled on launch. The RP4 will be sending commands to the Arduino such as next coordinates to move the robot arm to and when to pick up or put down a puzzle piece. To ensure that the Camera Subsystem is fulfilling its responsibilities for sending transmissions to the onboard control system, a requirements & verification table can be found below.

2.4.4 Mechanical Subsystem

The mechanical subsystem executes physical actions based on instructions from the control system to manipulate the arm. This subsystem contains stepper motors controlled by the motor driver to precisely control the movement of the belt system. The linear actuator provides vertical movement for the arm, which in turn allows a flat suction cup to pick up the puzzle pieces. The suction will be provided by an air hose connected to the vacuum pump.

2.4.5 Circuits

Diagrams and schematics of our circuits for the machine are all in Appendix B. Most of our circuits correspond to the control subsystem because we included all of the control for the mechanical components in the control subsystem. The mechanical subsystem is directly connected to the outputs of the control subsystem. The diagrams in Appendix B are for the pcb which we did not finish but we replicated the schematics on the breadboard that we used to control the machine in the final demo.

Microcontroller

Figure B.1 is the microcontroller circuit. It includes the microcontroller, a usb to program and power it, and the programming header. The microcontroller has pins with labels on it that connect it to the other circuits. We chose to use the ATMEGA32U4 because it has 5V outputs which are the logic inputs to the drivers and the power for the solenoid which will make the rest of the circuit easier.

Power

Figure B.2 is the power circuit. This circuit is what we will use to power our machine. The circuit in Figure B.2 was designed for a PCB that we did not use but the principle is the same. In the circuit, we stepped down 12V from a barrel jack to 5V with a buck converter. We use the 12V to power the actuator, steppers, and vacuum pump. We use the 5V to power the solenoid, the microcontroller, and the switches.

Stepper Drivers

Figure B.3 is the circuit for the H-Bridge drivers that control the stepper motors. We are not using microstepping so we leave the M0, M1, and M2 pins unconnected. Sleep and reset pins are both active low so we power those with 5V. The VMOT pin is connected to 12V and VGND and GND are common. The outputs A0, A1, B0, and B1 are directly connected to the stepper motors.

Linear Actuator

Figure B.4 is the linear actuator circuit. This circuit is built around the L298 dual H-Bridge IC. We modeled our circuit after the L298N separate module.

Solenoid

Figure B.5 is the circuit for the solenoid. This circuit was a complicated last-minute addition to our machine. We originally powered the solenoid with the 5V pin on the Arduino which could output up to 800mA and later when we were integrating everything we found that the GPIO pin could not power it. This is because the GPIO pins on the Arduino uno can only output up to

40mA of current. We used a BJT transistor as a switch to use a GPIO pin to turn on and off 5V power from the kit. We needed to used the correct base and collector resistors to allow the correct current.
We used a TIP29C. First, we needed to find Vce(sat) to know the voltage drop across the collector and emitter during saturation which we wanted to be in. This was in the datasheet as 0.7V. The current that is needed to power the solenoid is 380mA at 6V. I am powering it at 5V so I will calculate the current to keep the power the same.

$$I * V = P -> 6V * 0.38A = 2.28W$$

$$2.28W/5V = 0.456A$$

From these equations we attempted to run the solenoid at 0.456A to give it the same power as it would at 6V. The next step is to calculate the collector resistor. This is using Ohm's law with the extra voltage drop over Vce(sat) that we found earlier.

$$Rc = (Vcc - Vce)/I$$

$$Rc = (5V - 0.7V)/0.456A = 9.43\ Ohms$$

We have found that the resistor between the 5V source and the collector pin needs to be 9.43 ohms but we rounded that to 10 ohms because that is a common resistor value. The next step in the process is to calculate the base current Ib at the edge of saturation(eos).

$$Ib(eos) = Icmax/\beta min -> Ib(eos) = Icmax/hEF$$

$$Ib(eos) = 0.456/50 = 0.00912A$$

Now we have determined that the current needed at the base, to give us a collector current of 456mA, is 9.12mA. This current will just take us to the edge of saturation and we want to be deep into saturation and will not accidentally fall out of it due to small current or temperature changes. I will use an overdrive factor of 2 and I will multiply our value by 2 to get 18.24mA which is about 20mA. To find the resistor at the base I will again use Ohm's law. I will need to know what voltage is across the bass and emitter which is 1.3V from the equation sheet.

$$V/I = R -> (Vcc - Vbe)/Ib = Rb$$

$$(5 - 1.3)V/0.02A = 185\ ohms$$

Now we know our base and collector resistances which are 185 ohms and 9.43 ohms respectively. We used two 330 ohm resistors in parallel because the resistance would be 165 ohms which is close to 185 ohms. It is also less resistant than 185 ohms which will allow more current which would just push the BJT farther into the saturation region which is desired.

Outputs

Figure B.6 is the miscellaneous outputs section. The three outputs with the resistors are for the three limit switches which allow us to zero the machine. We added resistors to them to limit the current that they would draw since the microcontroller can sense the output as low currents. The other outputs are extra outputs on the microcontroller that could be used for things we forgot about.

2.5 Software Design

One core part of our project involves the use of OpenCV to detect puzzle pieces and analyze them in order to determine the proper orientation and position in which each individual piece should be rotated and moved. Our inputs to the camera system are simply the live photo and video feed that will be fed into our software stack.

An HSV algorithm is used in combination with OpenCV to process the color images fed into the computer from our camera subsystem. A simple flow chart of the software stack can be seen below (Figure B.10).

Once the camera is initialized, it will capture frame-by-frame data in an executive loop while in our "Capture Image" phase. I will be using a standard capture rate which will cap out at around 30 frames per second, any more and there are risks of unnecessary performance drops in exchange for only a slightly smoother system. After capturing a frame from our live video feed, the computer will be using cv2's cvtColor functionality to convert the entire frame into the HSV color space while also creating a mask for each color space that we are trying to identify. The mask (cv2.inRange) contains the upper and lower bounds for each individual color on the HSV scale for use of identification. A local database of color masks will be stored on the computer (see Figure A.5) with all nine corresponding colors that will be on our finished puzzle grid. Once our color is located, the system will then find the contours of the detection areas (cv2.findCountours) to identify our puzzle piece. To account for false positives and have better filtering, only the contour piece with the largest area will be selected as the "puzzle piece". This should cover most cases as the view of our camera is not significantly wide and will be quite close to the table/viewing area. There would be a minimal chance that the largest (red, blue, green, etc.) piece in the view of the camera is not a puzzle piece. The detected puzzle piece will then be highlighted in the frame and labeled according to the color that is determined by the HSV algorithm. This process will then continue running in an executive loop until a user interrupt occurs to send a robot move command to the ESP32 microcontroller which will then operate the motor drivers of our robot arm accordingly.

Figure 3: HSV Color Mask Database

2.6 Tolerance Analysis

Mechanical Precision: One of our high-level requirements is that we will be able to move the suction cup with 1mm precision in any direction. We will be moving the arm with the suction cup attached to it with a belt and pulley system powered by stepper motors. We have chosen a stepper motor with 200 steps per revolution. The steps are the base unit of movement of the stepper motor. The distance of one revolution is determined by the radius of the pulley the belt is attached to.

$$2\boldsymbol{\pi} r = C$$

$$C/steps = distance\ of\ step$$

$$2\pi r/200 = 1mm$$

To get this level of precision we will need the radius of the pulley to be less than 31.83mm. We plan to have pulleys with a radius of only 10-20mm.

Vacuum Suction: We will be using a vacuum suction cup to pick up the puzzle pieces. The vacuum holding the puzzle piece will need to be strong enough to hold the puzzle piece when we are lifting it off the table since that is when the vertical acceleration will be greatest. The force needed to lift the piece will be

$$F = m * (g + a)$$

where m is the mass of the puzzle piece, g is the acceleration of gravity, and a is the acceleration of the puzzle piece being lifted. The large 3x3 puzzle we are going to use is listed on Amazon with a total weight of 4.6oz or 0.5111oz per piece. This is equivalent to about 14.49 grams which we will round to 15 grams or 0.015kg. The acceleration of gravity is 9.81 m/s^2. We approximate that we will pick up the piece with a max acceleration of 0.5 m/s^2.

$$F = 0.015 * (9.81 + 0.5) = 0.15465$$

We will need 0.15465 N to pick up the piece. The pressure of the vacuum needed is determined through this equation

$$P = F/A$$

The pressure we need is based on the size of the suction cup we choose. We are thinking of using a suction cup with a .2 inch diameter which is a 5.08mm diameter and 2.54mm radius. The area of the suction cup is

$$A = \pi r^2$$

so the area of the suction cup will be 0.00002026829m^2. With this force and area, the pressure of the vacuum pump will need to be 7630.14 Pa. The vacuum pump we plan to use will have a max vacuum pressure of 420 mmHg or 56 kPa. The parts we plan to use will be able to lift the puzzle pieces.

3 Requirements and Verification

3.1 Power Subsystem

Please refer to Fig A.1 for our requirements and verifications for the power subsystem. This subsystem is responsible for providing the appropriate voltage levels to power the various components of the robot. To ensure proper operation, one critical requirement was maintaining the correct 5V and 12V of power on the bus. We verified this by using a voltmeter to measure the difference between power and ground, confirming it was in the acceptable range of ± 0.2V. To protect the board from destructive voltage spikes, we specified placing a large electrolytic capacitor of at least 47µF across the motor power (VMOT) and ground near the board. During testing, we installed a 47µF capacitor across these points and monitored the voltage spikes across the motor power, ensuring they remained under the 45V maximum rating. By implementing these requirements and verifying them, we ensured the power subsystem would operate reliably and provide clean power without risk of damage from voltage spikes or improper logic levels.

3.2 Control Subsystem

Please refer to Fig A.2 for our requirements and verifications for the control subsystem.
Many of the control subsystem verifications tie directly into the transmission portion of the camera subsystem's verifications. The microcontroller (Arduino Uno) was able to successfully receive communications from the Raspberry Pi 4 (reference Fig B.9) through terminal response. Through the completion of the project, we were also able to visually verify the Arduino is able to take note of a specific coordinate on the board, zero itself, and then move accurately to said coordinate within +/- 5 millimeters of precision. Our puzzle pieces were all placed accurately in their final destination points as displayed during our live demonstration previously.

3.3 Camera Subsystem

Please refer to Fig A.3 for our requirements and verifications for the camera subsystem.
The main camera faults that we wanted to test and verify were any sudden disconnections, memory errors, and data corruption between the control block. Besides this, during the idle state of the camera, we aimed to have 95% accuracy in color detection and to be able to stream consistent feed to the Raspberry Pi 4 from the webcam at a passable FPS (frames per second). Through visual testing, we were able to verify that a sudden disconnection of the camera would cause a verbose terminal error and force a shutdown of the program after ten seconds (please see Fig B.7) along with the proper handling and transferring of data from the camera subsystem to the control subsystem for processing (see Fig B.9). An example of our verification for the camera subsystem's idle state along with our real-time 95% video color detection accuracy can be found in Fig B.8 as well which was verified through visual verification. All requirements and verifications were met for this subsystem as seen in our working demonstration of the autonomous robot.

3.4 Mechanical Subsystem

The mechanical subsystem is responsible for executing physical actions based on instructions from the control system to manipulate the robot arm. To ensure proper functionality and safety, we have established a set of requirements and verifications as outlined in Fig A.4.

One of the key requirements is the ability of the vacuum pump and suction cup to pick up puzzle pieces effectively. We verified this by holding a puzzle piece up to the suction cup and turning on the vacuum pump, confirming that the piece was securely held by suction alone. Additionally, we tested the linear actuator's ability to reach puzzle pieces on the table when fully extended, by placing a piece directly below the actuator and lowering it until the suction cup made contact with the piece, extending approximately 2 inches.

We specified the stepper motors should receive an amperage between 0.85A and 1.36A from the drivers. When we measured all three motors simultaneously using an ammeter, we discovered that the performance was optimal at 480mA.

Proper cooling of the motor driver IC is crucial for reliable operation. We addressed this by attaching a heat sink to the IC and conducting a full 7-minute test run. Using a thermocouple, we monitored the temperature of the IC throughout the test to ensure it remained below 60°C.

Through these processes, we have ensured that the mechanical subsystem operates reliably, effectively, and in a safe manner.

4 Cost and Schedule

4.1 Cost Analysis

The total cost for parts as seen below before shipping is $115.44. Estimating 5% shipping cost adds another $5.77 and a 10% sales tax adds $11.54. We are all in Computer Engineering and the expected starting yearly salary of a CompE graduate from UIUC is $109,176. This means we can expect a salary of $52.49/hr per team member. $52.49 x 2.5 x 60 hours x 3 team members = $23,620.50 in labor cost. The machine shop quotes a price of $50/hr for 40 hours a week for 2 weeks = $4,000. This comes out to be a total cost of $27,753.25.

Description (Part #)	Manufacturer	Qty	Price	Links
Raspberry Pi Camera Module 3 Wide Lens (5658)	Raspberry Pi (Adafruit)	1	35.00	Link
Raspberry Pi 4 Model B - 4 GB (4296)	Raspberry Pi (Adafruit)	1	55.00*	Link
Arduino Uno Rev3 (A000066)	Arduino	1	27.60***	Link
Limit Switch 10-pack (3-01-1546)	HiLetgo	1	5.99	Link
Micro Air Pump - DC 12V Micro Vacuum Pump (B07FGFPKNS)	DEWIN	1	10.49	Link
2" Stroke Micro Electric Linear Actuator (UYG2206LAS50MM-1)	UYGALAXY	1	35.99	Link
Stepper Motor Driver 3 Pack (DRV8825)	WWZMDiB	1	7.99	Link
Nema 17 Stepper Motor (17HS4401S)	Usongshine	2	19.98	Link
Standard Servo Motor (HS-311)	Hitec	1	13.49***	Link
Suction Cup	McMaster-Carr	1	9.89**	Link
3m Silicone Tube 2mm ID x 4mm OD Flexible Silicone Rubber Tubing Water Air Hose Pipe Transparent (B08H1ZD5VZ)	Gikfun	1	8.18**	Link
DC DC CONVERTER 0.8-6V (LMZ12008TZ/NOPB)	Texas Instruments	1	17.94	Link
XL Series Lightweight Timing Belt Pulleys	Mc-Master Carr	4	45.70	
PCB parts	Mouser		28.42	
			207.50	

* = Purchased individually ** = Purchased by the machine shop *** = From lab or shop

4.2 Schedule

Schedule:
Include a timetable showing when each step in the expected sequence of design and construction work will be completed (general, by week), and how the tasks will be shared between the team members. (i.e. Select architecture, Design this, Design that, Buy parts, Assemble this, Assemble that, Prepare mock-up, Integrate prototype, Refine prototype, Test integrated system).

Week	Class Task	Conor	Eric	Alex
2/26	Design/Peer review PCB planning and review Order parts	Finish Design and Peer review	Code outline and libraries we will use	
3/04	PCBway orders (pass audit) Teamwork evaluation, Final machine shop design	Finalize machine shop design, Linear actuator to work. Start designing the PCB	Can recognize color and edges with openCV	
3/18		Get solenoid to work. Get the steppers to work. Finalize design for first PCB	Work on CV algorithm for puzzle piece recognition	
3/25	Individual progress reports	Make circuit for linear actuator. Start Second round PCB off of Uno design.	Finish puzzle piece finding algorithm	
4/01		Machine shop communication. Control steppers with Arduino and limit switch	Arduino and Pi can communicate	Get limit switch to work, buy some parts
4/08		Make all control and mechanical parts work	Work out bugs in preparation for mock demo	

		together for mock demo		
4/15	Mock demo Team contract fulfillment	Worked out jittering problem and power problem	Prepare for mock demo/ make puzzle more complicated	Prepare for mock demo/ make puzzle more complicated
4/22	Final demo Mock presentation	Tried to program pcb. Put finishing touches on project	Prepare for demo and finishing touches to project	Prepare for demo and solder PCB
4/29	Final presentation Final paper Lab checkout Lab notebook	Finish paper and presentation	Finish paper and presentation	Finish paper and presentation

5 Conclusions

5.1 Accomplishments

In this project, we successfully designed and built an automatic puzzle-solving robot capable of autonomously solving 3x3 jigsaw puzzles. Our machine demonstrated precise mechanical movement within 1 mm, accurately identified individual puzzle pieces using computer vision techniques, and completed the entire puzzle-solving process in under 7 minutes. By integrating a robotic arm with a suction device, a belt and pulley system driven by stepper motors, and an OpenCV-based vision system, we created a user-friendly device that could be used to efficiently solve puzzles of varying complexities.

Throughout the development process, we overcame challenges related to mechanical precision, vacuum suction, and computer vision algorithms. Our design decisions, such as selecting appropriate stepper motors, pulleys, and suction cups, ensured that the robot could accurately move and manipulate puzzle pieces. The successful implementation of the HSV color detection algorithm allowed for reliable identification and localization of individual pieces.

5.2 Uncertainties

While our automatic puzzle solver performed well in controlled environments, there are some uncertainties that may affect its performance in real-world scenarios. Factors such as variations in lighting conditions, puzzle piece sizes, and surface textures could impact the accuracy of the computer vision system. Additionally, the current design is limited to solving 3x3 puzzles, and its scalability to larger and more complex puzzles remains uncertain.

To address these uncertainties, we propose the following alternatives:
Implementing a source of lighting to ensure consistent illumination of the puzzle area.
Incorporating machine learning algorithms to improve the robustness of the piece recognition system.
Conducting extensive testing with a diverse range of puzzle types and sizes to evaluate the system's performance and identify areas for improvement.

5.3 Future Work/Alternatives

The successful development of our automatic puzzle solver opens up exciting possibilities for future work and alternative applications. One potential avenue is to expand the system's capabilities to handle larger and more intricate puzzles, such as 1000-piece jigsaw puzzles or 3D puzzles. This would require enhancements to the mechanical design, computer vision algorithms, and control systems.

Another promising application of our technology is in the field of document reconstruction. The machine could be adapted to piece together torn or shredded documents, which could be valuable in forensic investigations or historical document restoration. By leveraging the precise mechanical control and computer vision techniques developed for puzzle-solving, the system could accurately align and reassemble document fragments.

Additionally, the automatic puzzle solver could be utilized in the realm of cartography and geographic information systems (GIS). The machine could assist in piecing together patches of maps or aerial photographs, enabling the creation of seamless, high-resolution maps for various applications, such as urban planning, environmental monitoring, and disaster response.

In terms of broader impacts, our automatic puzzle solver has the potential to enhance accessibility and inclusivity in the realm of recreational activities. By automating the puzzle-solving process, our machine can enable individuals with physical limitations or cognitive challenges to enjoy the benefits of puzzle-solving, such as improved problem-solving skills and mental stimulation. Moreover, the technology developed in this project could contribute to advancements in robotics, computer vision, and automation, with potential applications in various industries, including manufacturing, healthcare, and education.

5.4 Ethical Concerns

From an ethical standpoint, our project adheres to the principles outlined in the IEEE Code of Ethics.

Privacy (IEEE I.1, ACM 1.2): Our automatic puzzle solver utilizes a camera system that may unintentionally capture sensitive information. To address this concern, we have implemented measures to minimize data collection and adhere to data privacy regulations, upholding the principle of respecting the privacy of others.

Transparency and Explainability (IEEE I.2, ACM 2.7): As an autonomous system, it is crucial that the decision-making process of our puzzle solver is transparent and understandable. By developing explainable algorithms, providing visualizations of the robot's reasoning, and allowing for human intervention when needed, we aim to avoid real or perceived conflicts of interest and ensure that the technology is transparent and accountable.

5.5 Safety Concerns

From a safety standpoint, we have taken proactive measures to ensure the well-being of users and the general public. Our design process has prioritized the identification and mitigation of potential hazards associated with the moving components and operational aspects of the automatic puzzle solver.

Physical Safety (IEEE II.9, ACM 2.1): The moving parts and potential pinch points of our automatic puzzle solver pose risks to people and property. To prioritize the safety, health, and welfare of the public, we have designed the machine to ensure safe operation around people by adhering to safe distances and clear communication protocols.

System Reliability (IEEE II.9, ACM 2.3): Malfunctions or errors in the puzzle solver could lead to safety hazards. In accordance with the principle of accepting responsibility for our decisions, we have conducted rigorous testing, implemented fail-safe mechanisms, and established regular maintenance procedures to ensure the reliable and safe operation of the machine.

References

ACM. (2018). ACM Code of Ethics and Professional Conduct. Association for Computing Machinery. https://www.acm.org/code-of-ethics

Amazon.com, Inc. (n.d.). Air Pump Electric Treatment Instrument. Amazon. https://www.amazon.com/Air-Pump-Electric-Treatment-Instrument/dp/B07FGFPKNS

Amazon.com, Inc. (n.d.). UYGALAXY Stroke Electric Linear Actuator. Amazon. https://www.amazon.com/UYGALAXY-Stroke-Electric-Linear-Actuator/dp/B0B4BJD4HS/ref=sr_1_4?keywords=mini%2Blinear%2Bactuator&qid=1707237976&sr=8-4&th=1tuator/dp/B0B4BJ5GLL/ref=sr_1_4

ArduCam. (n.d.). Raspberry Pi Pinout. https://www.arducam.com/raspberry-pi-camera-pinout/

Handson. (n.d.). 17HS4401S Datasheet. DatasheetsPDF.com. https://datasheetspdf.com/pdf-file/1310364/Handson/17HS4401S/1

Hitec RCD USA. (n.d.). HS-311 Servo Motor Datasheet. University of Texas at Austin. https://users.ece.utexas.edu/~valvano/Datasheets/ServoHS311.pdf

IEEE. (n.d.). IEEE Code of Ethics. https://www.ieee.org/about/corporate/governance/p7-8.html

Pololu Corporation. (n.d.). Pololu - A4988 Stepper Motor Driver Carrier. Pololu. https://www.pololu.com/product/2133

Texas Instruments. (n.d.). LMZ12008 SIMPLE SWITCHER® Power Module Datasheet. Texas Instruments. https://www.ti.com/lit/ds/slvsa73f/slvsa73f.pdf?ts=1708583617637&ref_url=https%253A%252F%252Fwww.google.com%252F

Texas Instruments. (n.d.). LMZ12008 SIMPLE SWITCHER® Power Module Product Information. Texas Instruments. https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Flmz12008

University of Illinois Urbana-Champaign. (n.d.). Salary Averages. Electrical and Computer Engineering at Illinois.
https://ece.illinois.edu/admissions/why-ece/salary-averages

Appendix A Verification Tables and Other Tables/Diagrams

Requirements	Verification
• The 5V and 12V power bus will have the correct voltage	• Use a voltmeter to measure the difference between the 5V bus/12V bus and common. The voltage should be +/- 0.2 V.
• To protect the board from destructive LC voltage spikes we will put a large (at least 47 µF) electrolytic capacitor across motor power (VMOT) and ground somewhere close to the board.	• Place >47µF electrolytic capacitor across VMOT and ground • Turn on the machine and monitor voltage spikes across motor power and ensure it is under the 45V maximum

Fig A.1

Requirements	Verification
• The microcontroller will be able to record the position of the suction cup, above the table	• Turn on the machine so the stepper motors get zeroed • Move to a designated point on the board, our point will be 100 steps in the x direction and 100 steps in the y direction or (100,100), mark this position of the suction cup on the table • Move the stepper motors in a manner that is similar to how the puzzle solver will operate. Move 123 steps in the x direction and 112 in the y direction or (+123,+112), then (-50,+50), then (+73,-41), then (-7,-19). • Move the suction cups to the original position by moving (-139, -102) and mark the position of the suction cup over on the table • Verify that the marks are in the same place +/- 5 millimeters
• The microcontroller should be able to receive a signal from the Raspberry Pi and react accordingly	• Have the Raspberry Pi send a signal to lower the linear actuator • Within 3 seconds of the pi signal being sent the microcontroller should lower the linear actuator

Fig A.2

Requirements	Verification
• Simulate a sudden camera disconnection, when no communication is received for 10 seconds, display error.	• Ensure that the entire camera subsystem (RP4 + Pi Camera) and currently running in an idle state (reference software flow chart). • Cut power/turn off the Pi Camera source. Confirm that the RP4 enters a fatal communication state.
• The camera system's idle state should receive consistent feed from the Pi Camera 3.	• Verify that the camera has been initialized and is properly connected to the RP4. • Verify that the correct resolution and frame rate are still running after 5 seconds in the idle state.
• The camera system should send an error message and stop camera recording when any memory or storage errors are encountered.	• Fill up current RP4 storage with test files and verify that streaming is cut from the Pi Camera when storage is full. • Verify that a correct corresponding error message is sent to the user through the verbose terminal.
• Camera subsystem should be able to handle data corruption during transfer to control system block.	• Remove receiver to transmission wire between RP4 and Arduino mid-way through control operation. • Verify that a correct corresponding error message is sent to the user through the verbose terminal. • Verify that the operation is shut off and handled accordingly.
• The camera subsystem should be able to perform real-time video processing. • We want to aim for 95% accuracy real-time video processing object detection.	• After ensuring that the video capture is available and working, verify that the processed real-time video stream meets processing benchmarks. (Edge detection, object tracking, etc.) • Verify that we have 95% accuracy in puzzle recognition compared to real truth.
• The camera system should be able to send basic commands and coordinates to the control subsystem	• Verify that the UART connection is stable and the baud rate is correct between both subsystems. • Execute user tests and send subsystem data to the motor control subsystem; verify that the motor control subsystem reacts accordingly ○ Example: new pose/coordinate move,

Requirements	Verification
	emergency stop control, etc.

Fig A.3

Requirements	Verification
• The vacuum pump and suction cup will be able to pick up puzzle pieces.	• A puzzle piece will be held up to the button of the suction cup. • The vacuum pump will be turned on and the puzzle piece should be held to the suction cup with just suction.
• The linear actuator should be able to reach puzzle pieces on the table, with the suction cup, when fully extended.	• A puzzle piece will be placed on the table directly below the linear actuator and the linear actuator will be lowered. • The suction cup connected to the linear actuator will touch the puzzle piece
• The stepper motors should receive an amperage between 0.85A-1.36A from the drivers to ensure proper usage.	• Turn on the machine and run all three motors • Use an ammeter to measure the current to the motors from the drivers. • Ensure it is within range and adjust voltage accordingly
• A heat sink will be placed on the motor driver in order to cool the IC.	• Attach heat sink to IC • Turn on the machine and run 7 minutes worth of movement with the motors. • Use a thermocouple to measure the heat of the IC throughout the run to keep temperatures under 60°C

Fig A.4

Color	Upper Boundary	Lower Boundary
Red	[10, 255, 255]	[0, 120, 70]
Blue	[130, 255, 255]	[110, 50, 50]
Light Green	[80, 255, 255]	[40, 40, 40]
Orange	[15, 255, 255]	[5, 50, 50]
Yellow	[30, 255, 255]	[20, 100, 100]
Violet	[160, 255, 255]	[130, 50, 50]
Magenta	[170, 255, 255]	[140, 50, 50]
Dark Green	[90, 255, 255]	[60, 50, 50]
White	[180, 30, 255]	[0, 0, 200]

Fig A.5

Appendix B All Other Figures

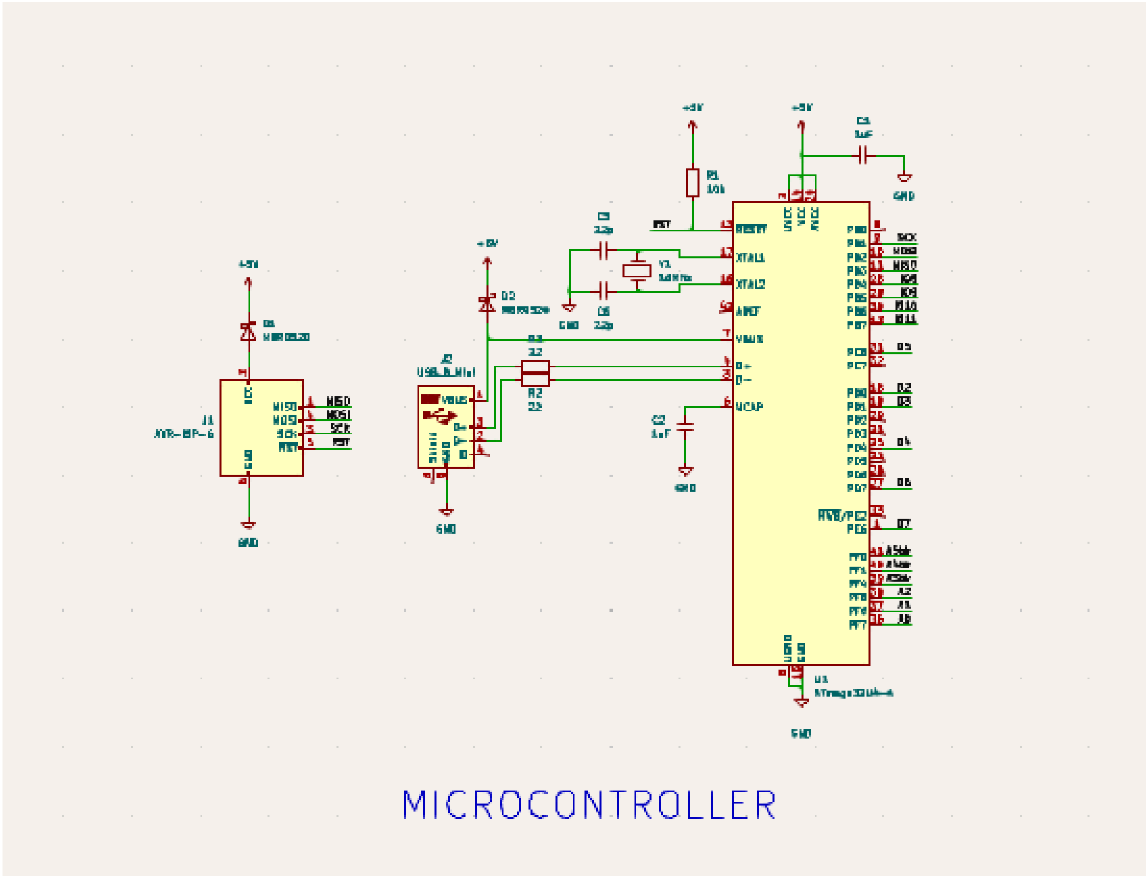


Fig B.1

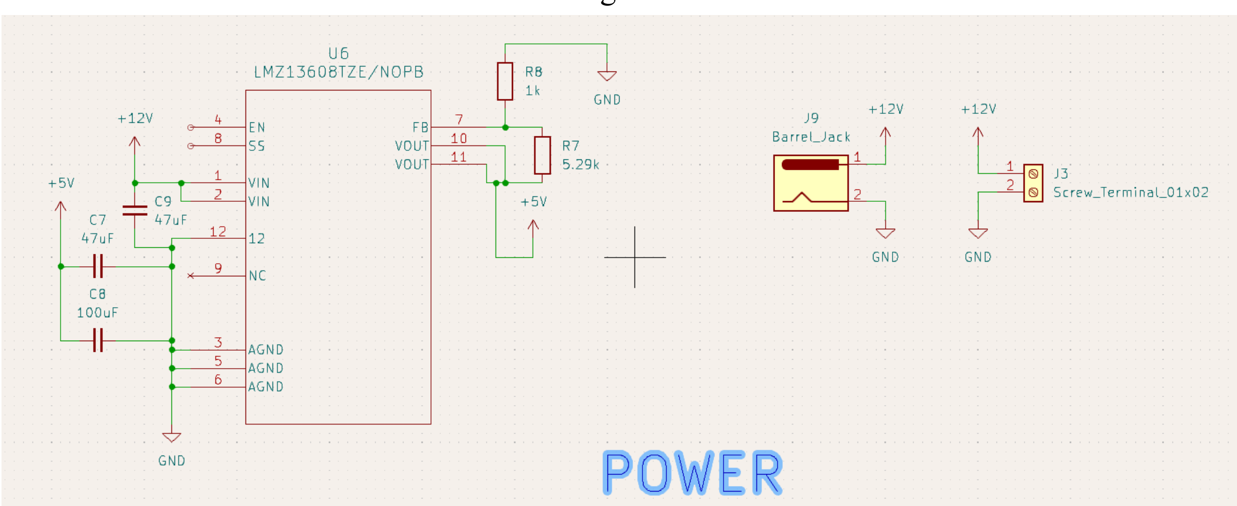
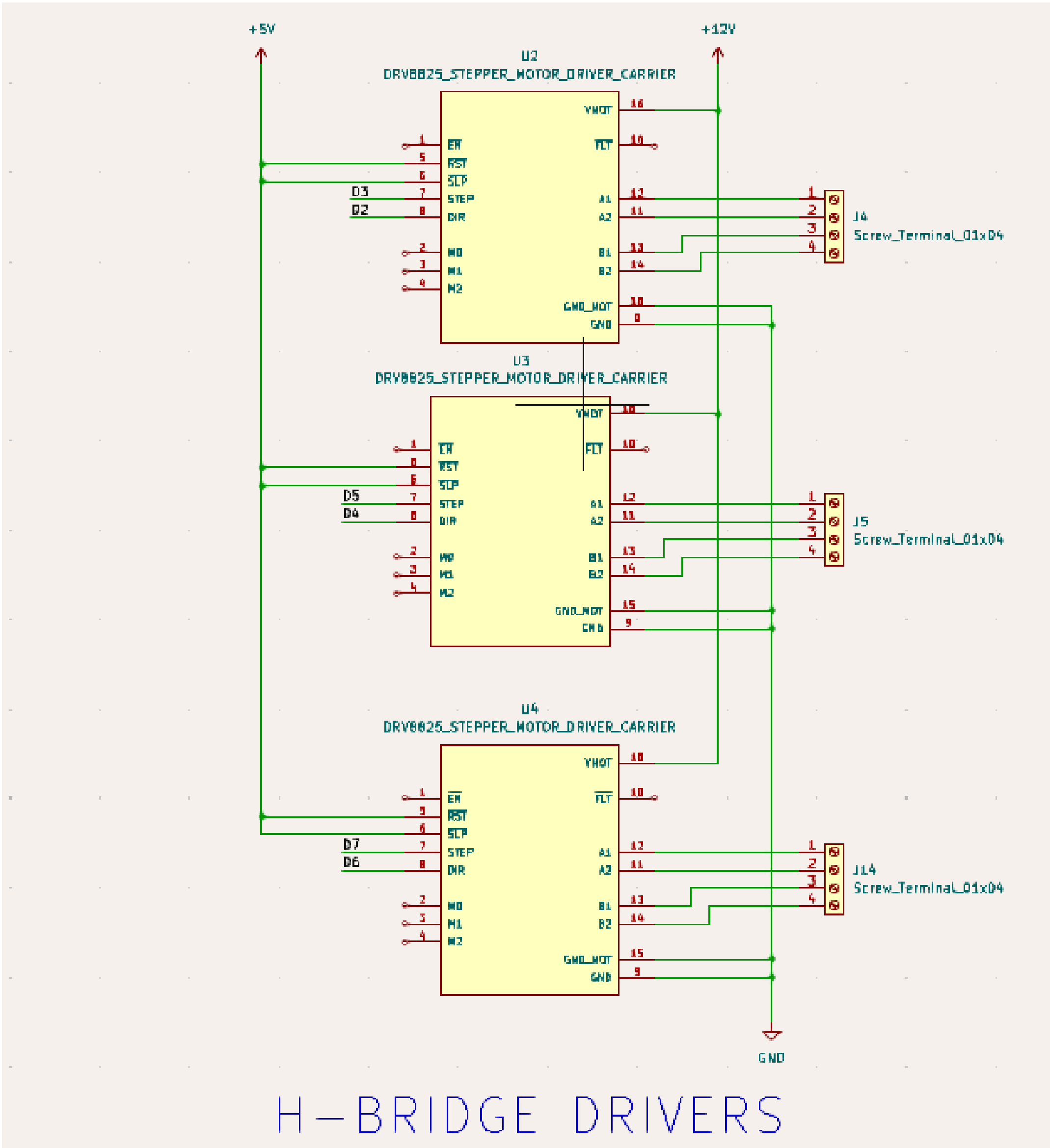


Fig B.2



+5V
+12V
U2
DRV8825_STEPPER_MOTOR_DRIVER_CARRIER
U3
DRV8825_STEPPER_MOTOR_DRIVER_CARRIER
U4
DRV8825_STEPPER_MOTOR_DRIVER_CARRIER
J4
Screw_Terminal_01x04
J5
Screw_Terminal_01x04
J14
Screw_Terminal_01x04
D3
D2
D5
D4
D7
D6
GND
H-BRIDGE DRIVERS

Fig B.3

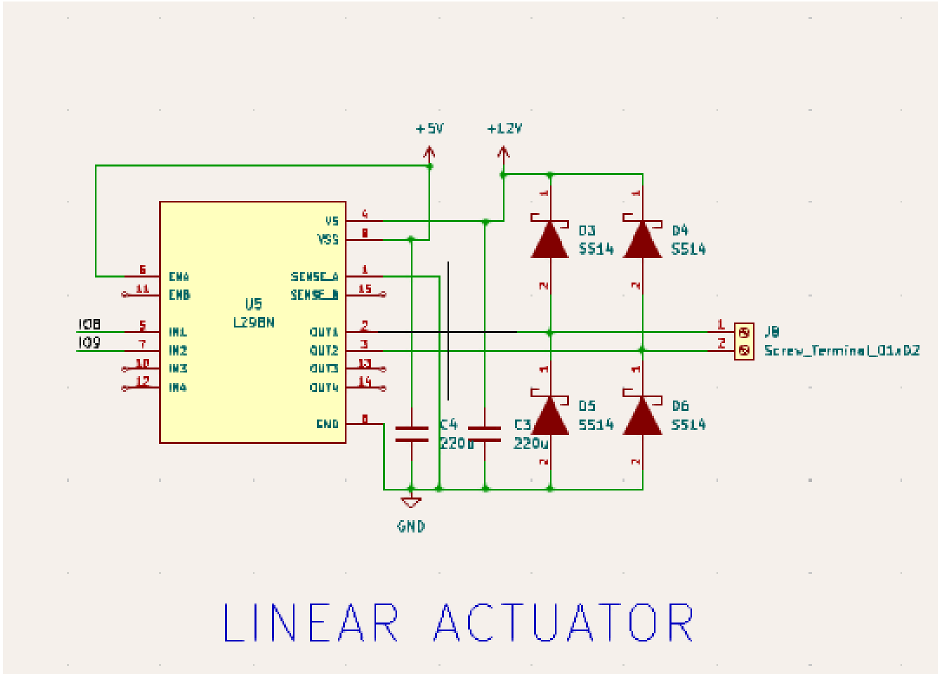

+5V
+12V
VS
VSS
ENA
ENB
U5
L298N
SENSE_A
SENSE_B
IO8
IO9
IN1
IN2
IN3
IN4
OUT1
OUT2
OUT3
OUT4
GND
D3
SS14
D4
SS14
D5
SS14
D6
SS14
J8
Screw_Terminal_01x02
C4
C3
220u
GND
LINEAR ACTUATOR

Fig B.4

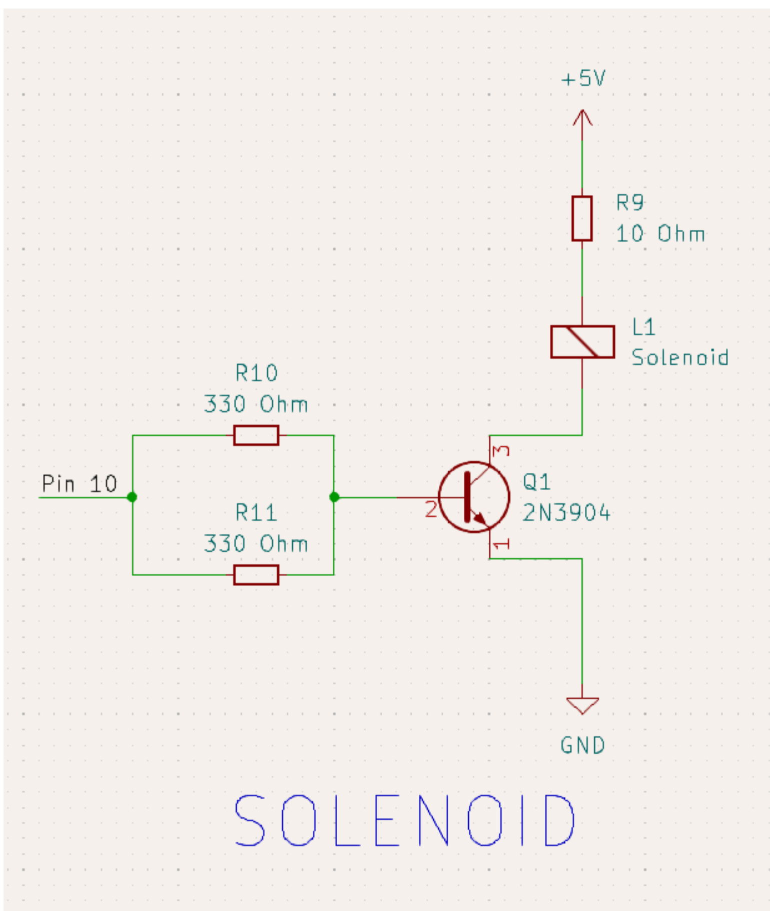

+5V
R9
10 Ohm
L1
Solenoid
R10
330 Ohm
Pin 10
R11
330 Ohm
Q1
2N3904
GND
SOLENOID

Fig B.5

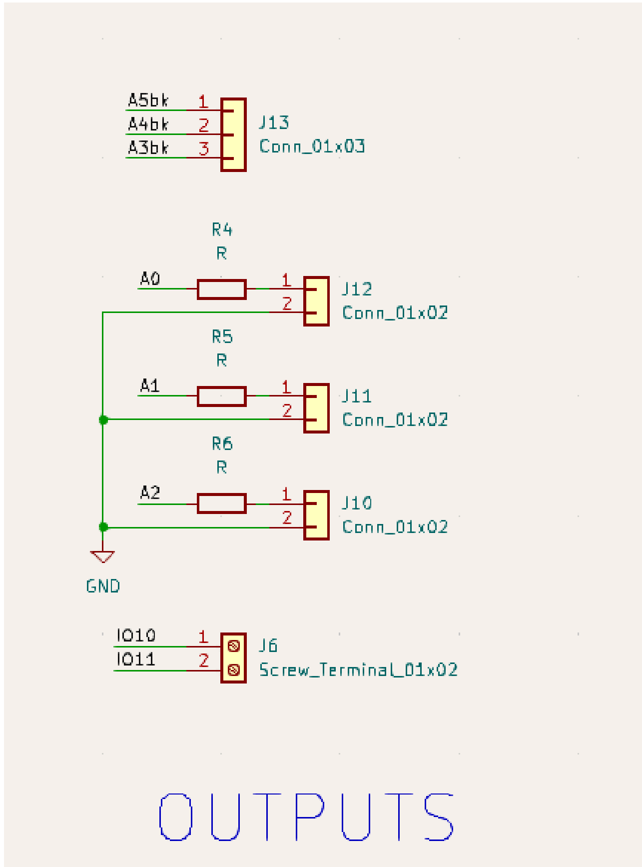


Fig B.6

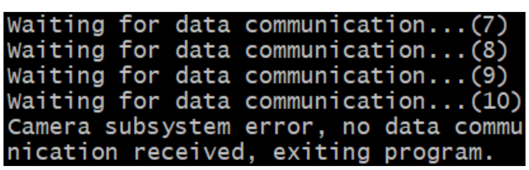
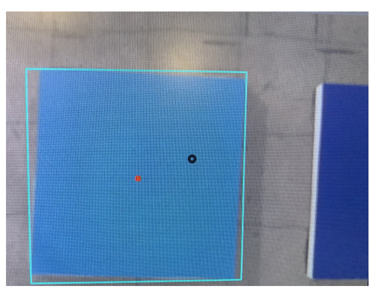


Fig B.7



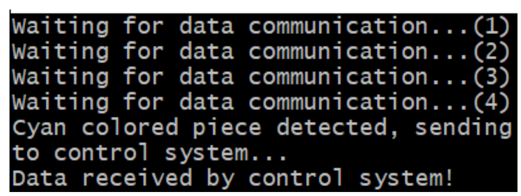


Fig B.9

Fig B.8

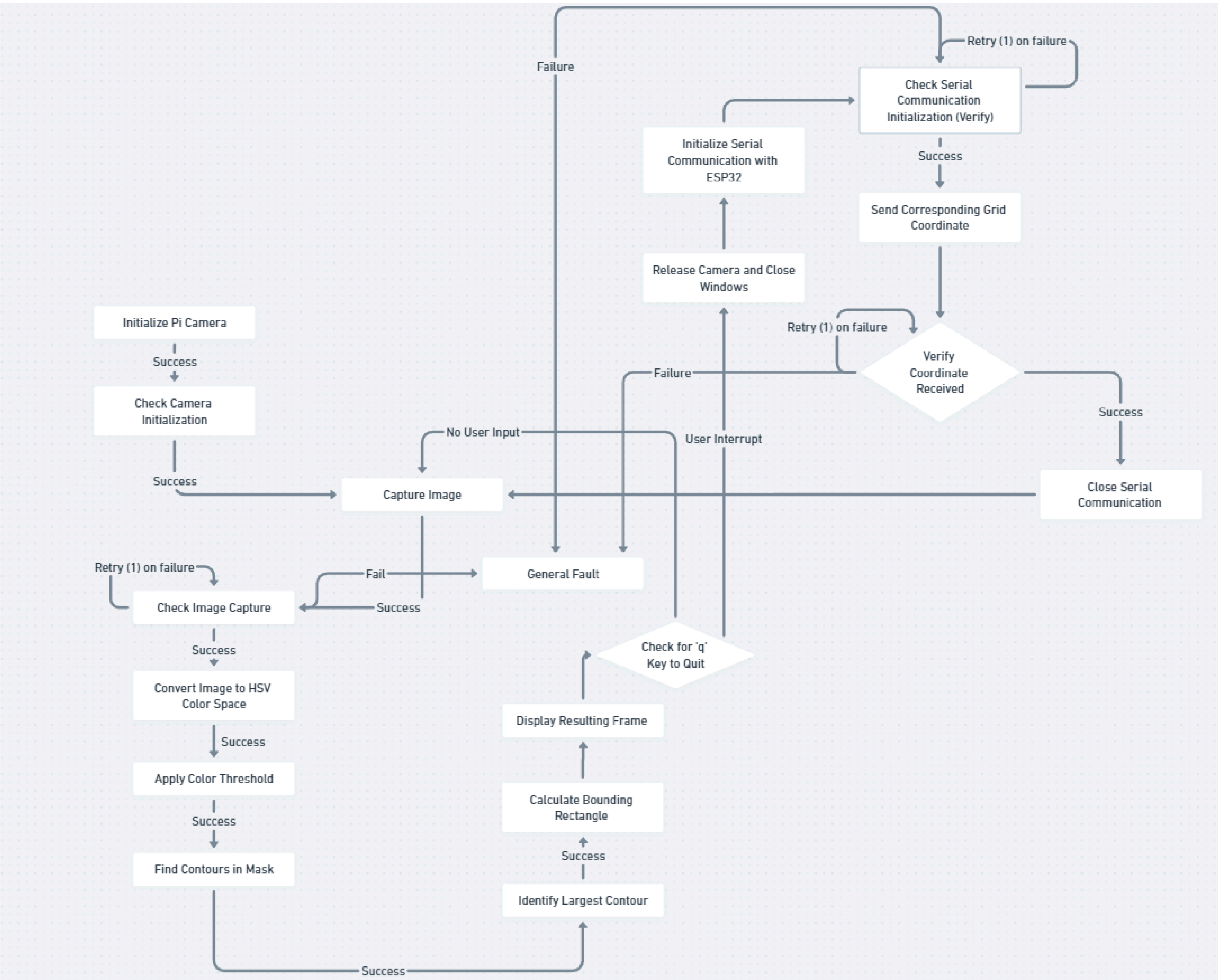

Initialize Pi Camera
Success
Check Camera Initialization
Success
Capture Image
No User Input
Fail
Success
General Fault
Retry (1) on failure
Check Image Capture
Success
Convert Image to HSV Color Space
Success
Apply Color Threshold
Success
Find Contours in Mask
Success
Identify Largest Contour
Success
Calculate Bounding Rectangle
Display Resulting Frame
Check for 'q' Key to Quit
User Interrupt
Release Camera and Close Windows
Initialize Serial Communication with ESP32
Check Serial Communication Initialization (Verify)
Retry (1) on failure
Failure
Success
Send Corresponding Grid Coordinate
Verify Coordinate Received
Retry (1) on failure
Failure
Success
Close Serial Communication

Fig B.10

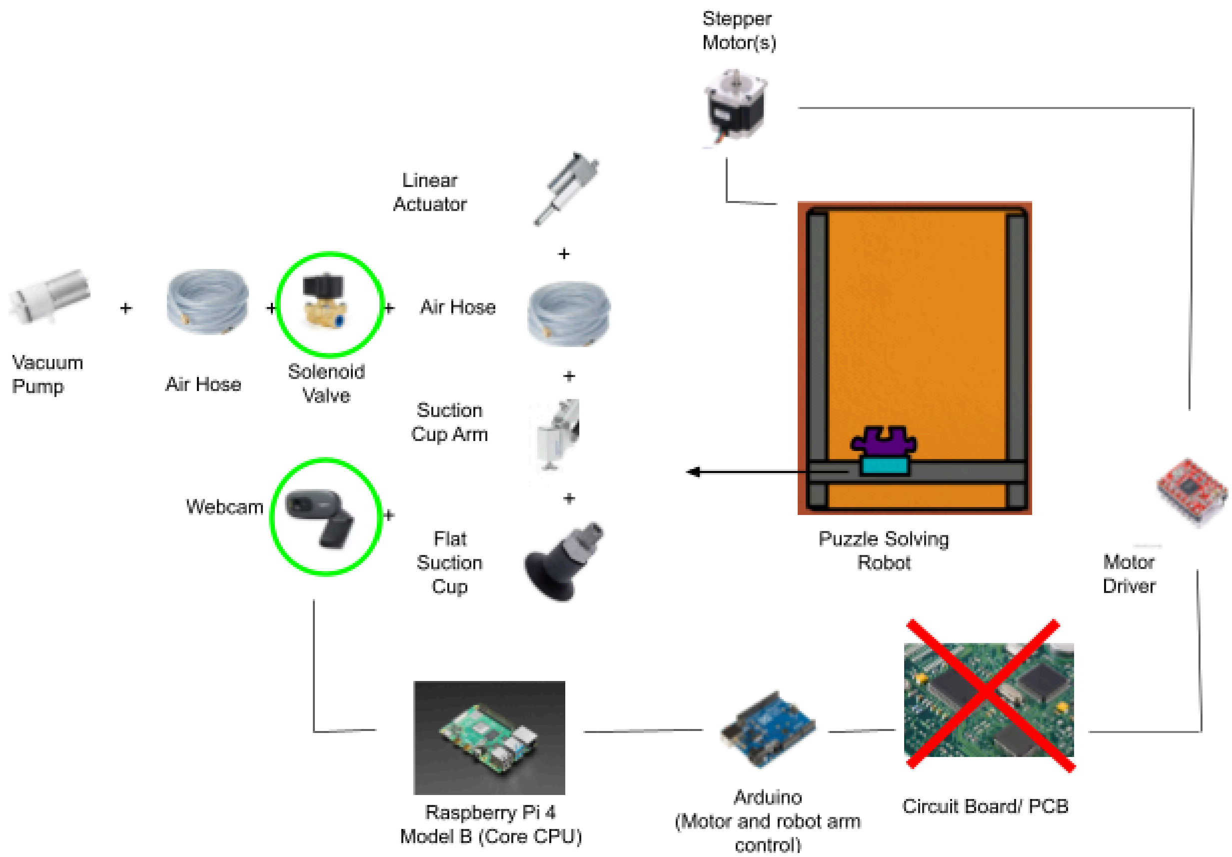


Fig B.11

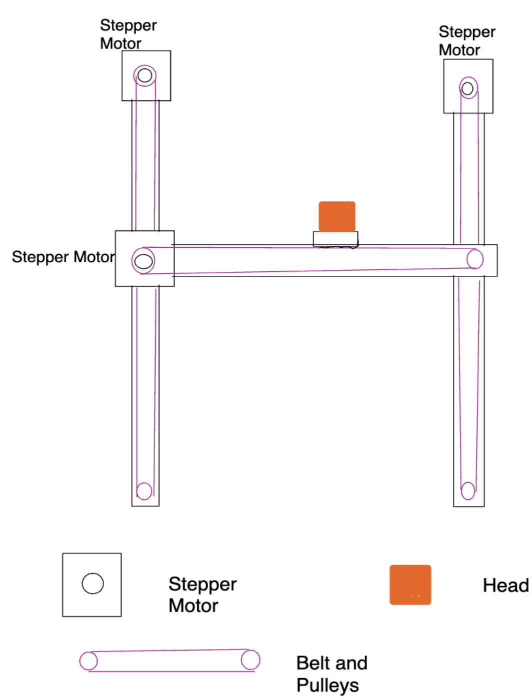


Fig B.12

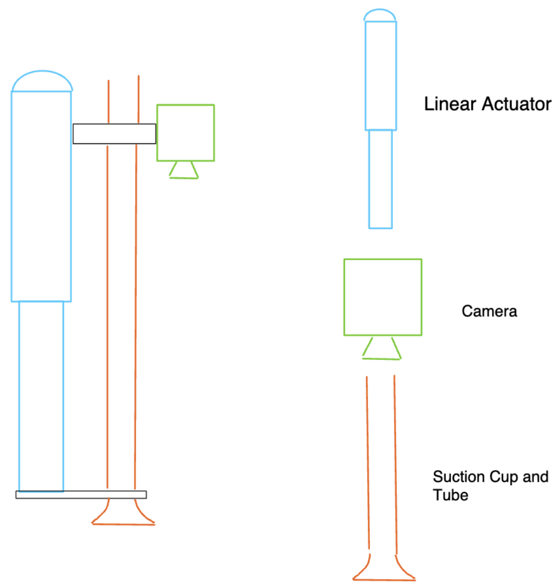


Fig B.13