Rubik's Cube Solving Machine

# Product Design Specification

**Designers:** Richard Groves & Miguel Mancias

## 

## **INTRODUCTION**

This document outlines the design specifications for a Rubik's Cube Solving Machine. This machine offers three methods of control. It allows manual manipulation through an LCD menu and navigation buttons, enabling users to move the cube pieces according to their input. For automated solving, users can initiate the sequence directly from the LCD menu using the navigation buttons (Note that a PC must be connected to solve the cube automatically). Additionally, the machine can receive UART characters as commands from a PC, providing users with another control layer.

## **SUMMARY**

The project encompasses various engineering aspects, including mechanical design, electronics, software programming, and user interface development. The machine features a 3D-printed structure, an advanced camera detection system, and intuitive user controls to provide a reliable and efficient cube-solving experience.

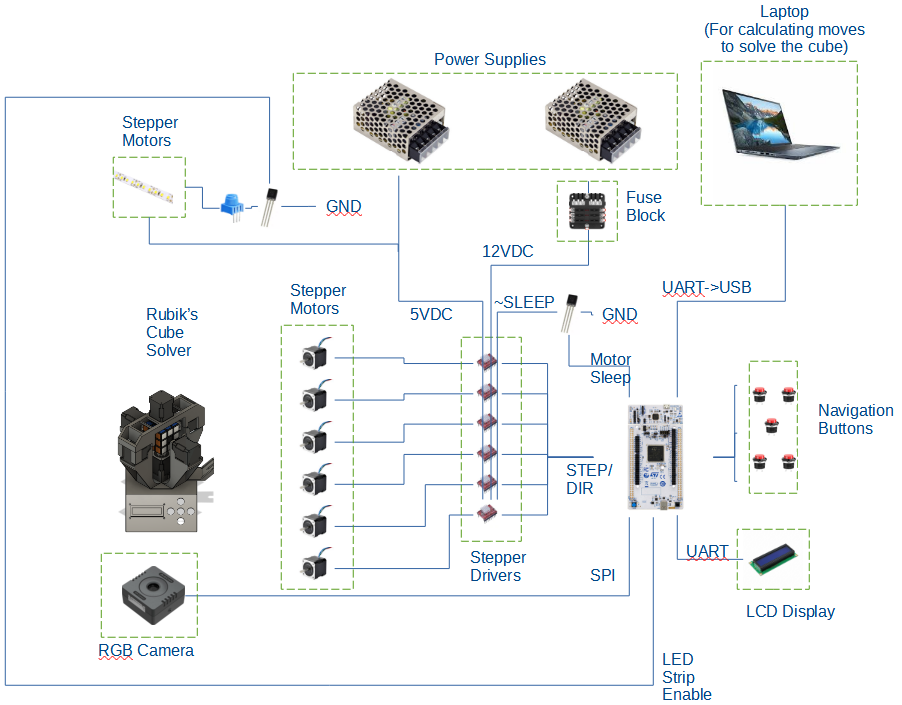
## **SYSTEM DESIGN**

The Rubik's Cube Solving Machine scans the cube using an Arducam Mega-5MP camera, processes the images to determine its state, and executes the solving algorithm through stepper motors. A 16x2 LCD screen and menu navigation buttons facilitate user interactions.

## 

## **HARDWARE ARCHITECTURE**

* **Structure:** 3D-printed using PLA filament to house all components and provide a stable base for operations.
* **Vision and Detection:** An Arducam Mega-5MP camera connected via SPI captures high-resolution images of the Rubik's cube.
* **User Interface:** This device features a 16x2 LCD screen connected through a UART backpack board from Adafruit, enabling user interaction and system status display.
* **Lighting System:** Includes 5VDC LED strips controlled by a potentiometer, allowing adjustment for optimal visibility.
* **Power Setup:** There are dual power supplies. A 5 Volt system powers the LCD screen, navigation buttons, and LED strips, while a 12 Volt system drives the stepper motors.
* **Motors and Drives:** Employs STEPPERONLINE 17HS19-2004S1 stepper motors, managed by A4988 and DRV8825 motor drivers for precise movement and rotation of the cube.

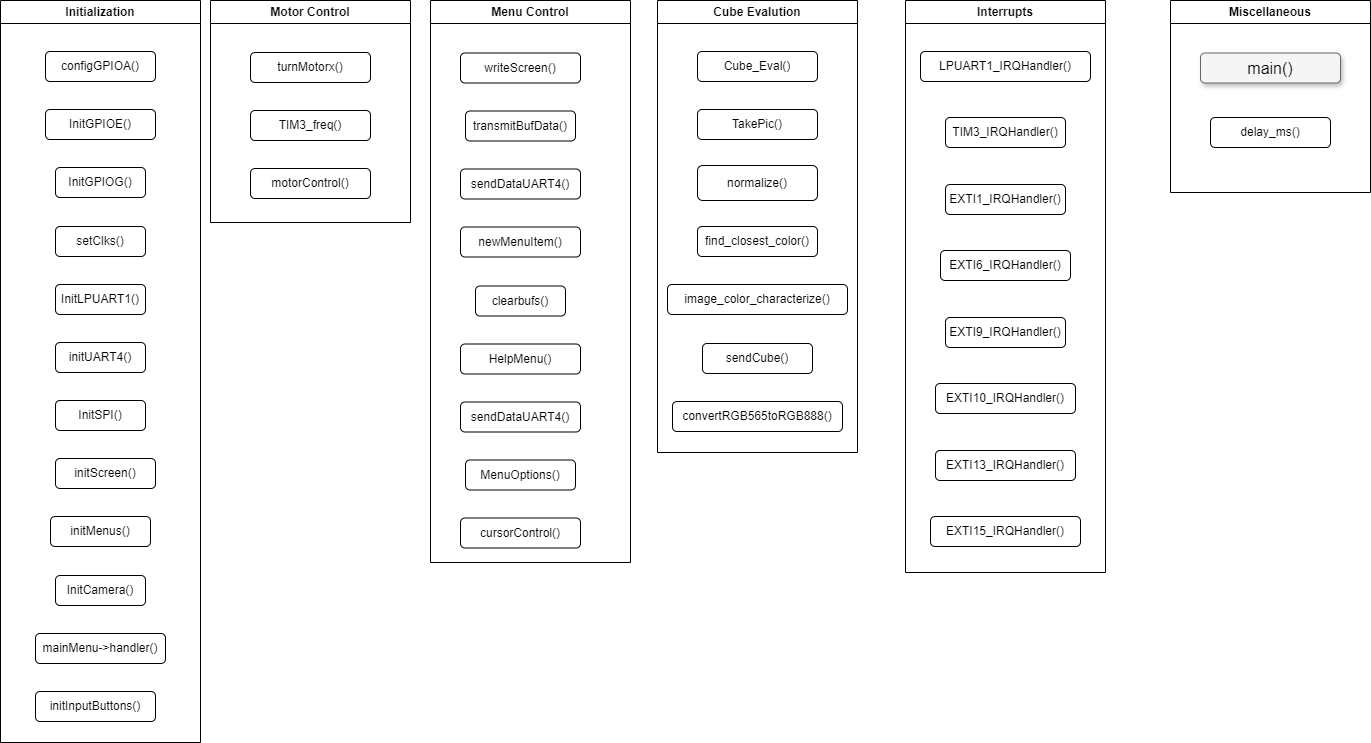


**Figure 1: Hardware**

## 

## **SOFTWARE ARCHITECTURE**

The software system for the Rubik's Cube Solving Machine is composed of an intricate network of functions and structures designed to operate on the STM32L552xx microcontroller. This section outlines the software design, including initialization, control, and communication procedures.



**Figure 2: Top-Level Software Architecture**

### Initialization and Configuration:

The system initializes GPIOs, clocks, UART, SPI interfaces, and peripherals upon startup. The Arducam is a crucial component with specific register settings to capture images in RGB format at a 96x96 resolution. GPIO pins control the LED lighting, which can be turned on or off to ensure optimal imaging conditions.

### Menu Navigation:

A hierarchical menu system is constructed using a series of linked *menuitem*, a custom struct developed for this system, each potentially leading to a submenu and associated with a handler function. This design provides a multi-level, iterative user interface that is navigable via input buttons and displayed on the LCD screen. There are three main menu options: help menu, motor options for motor control adjustments, and automated cube solving.

### Motor Control:

The motor control is achieved through a series of functions capable of turning them in clockwise and counterclockwise directions by specific degrees. Motor operations are controlled with timer interrupts (TIM3), ensuring the precise movement patterns necessary for manipulating the cube.

### Camera Operation and Image Processing:

The system utilizes an Arducam for image capture and an HSV-based color recognition algorithm for more robust and accurate color classification under varied lighting conditions. Unlike RGB, the HSV model separates color information (hue) from lighting (value), making it particularly effective for distinguishing the colors of the Rubik's Cube.

The software processes the captured image, converts RGB values to HSV, and matches them against predefined HSV ranges representing the cube's colors. This method efficiently handles complex lighting scenarios and ensures that the cube's state is accurately determined, facilitating the subsequent solving process.

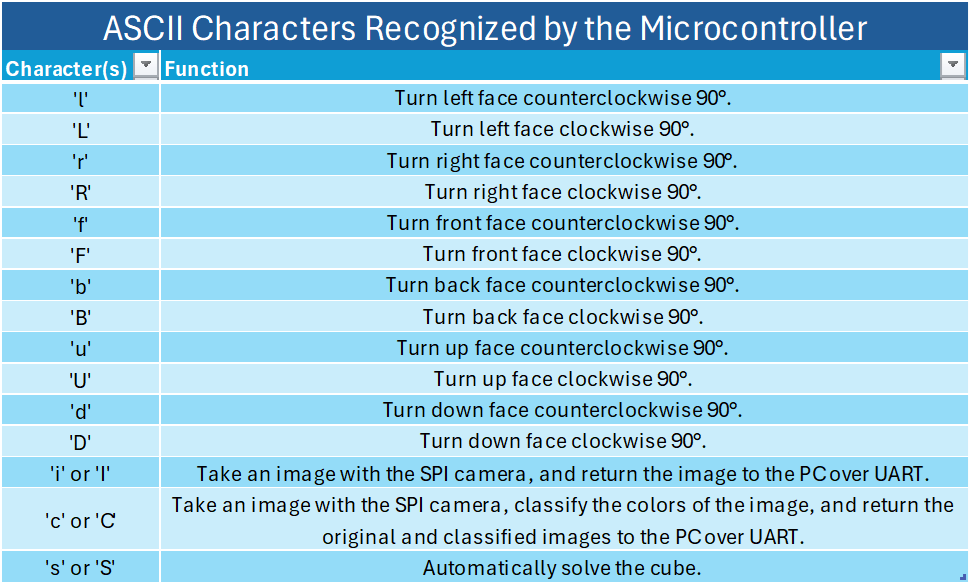
### Cube Evaluation:

A function Cube\_Eval translates the classified image into the Rubik's Cube state. This process involves recognizing color patterns and orienting them into a structured RubiksCube data type, effectively digitizing the cube's physical state.

### Communication Protocol:

The system incorporates two distinct UART interfaces: a bi-directional UART connection facilitates interaction with a PC, transmitting commands to the machine and retrieving the cube's state data for external processing. This interface ensures users can directly manipulate or monitor the machine's status through a PC application. The second UART channel is configured for data transmission only. It is dedicated to displaying status messages, navigation prompts, and other user interface elements on the LCD screen, enhancing the user experience by providing precise and immediate feedback on machine operations.

The SPI protocol is utilized exclusively for interfacing with the Arducam, ensuring rapid and efficient image data transfer necessary for color recognition and cube state analysis. This dedicated communication channel allows for the high throughput required for image processing, maintaining the machine's swift response times and ensuring accurate cube evaluation in preparation for solving algorithms.



### Interrupt Handling:

The software employs an interrupt-driven design to respond to UART data reception, motor timing control, and button inputs for user interface navigation. The LPUART1\_IRQHandler function handles received UART data, allowing dynamic PC communication. Meanwhile, TIM3\_IRQHandler ensures that motor steps are accurately timed, crucial for precise cube manipulation. Additionally, button press events are managed through interrupts, enhancing the responsiveness of the user interface by allowing users to efficiently navigate menus and trigger machine operations directly from the LCD interface. This comprehensive use of interrupts streamlines the interaction between the user commands, machine actions, and feedback loops, ensuring smooth and efficient operation.

### System Loop:

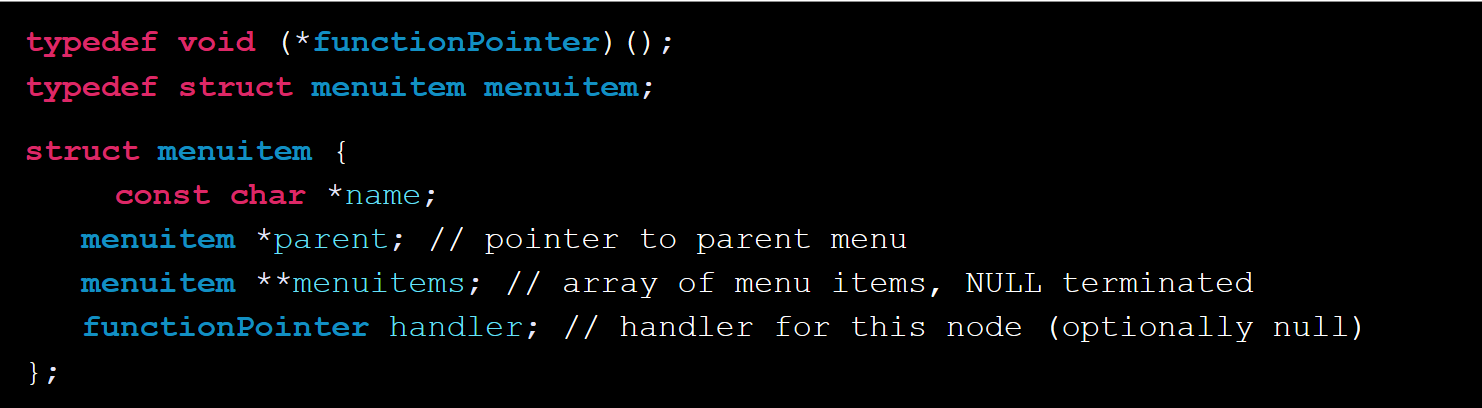
Since this system is interrupt-driven, the main loop of the software just ensures we are continually checking for input from the UART buffer or button presses. The interrupt handlers can handle images, categorize them, evaluate the cube state, and execute motor movements based on user interactions or received commands.

### Power Management:

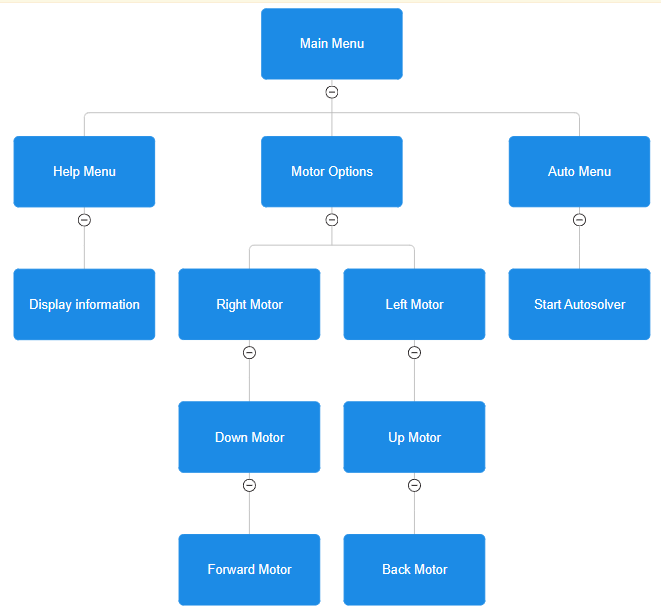
The system includes power management protocols, including handling motor power states and controlling LED lighting to conserve power when not in active use.

## **Menu Architecture**

As for the menu architecture, the *menuitem* structure was developed to allow the creation of menus and submenus. The *menuitem* structure contains essential features for easy “tree-like” implementation. In the figures below, we can see the structure definition code and the menu tree graph.



**Figure 3: Structure Definition**



**Figure 4: Menu and submenus**

It is important to mention that only the Main Menu and Motor Options menus have iterative submenus. This is important for software implementation since both Help and Auto Menu automatically return to the Main Menu after their handler is done.

As mentioned, the buttons are used to interface the user’s commands to the system, and the LCD is used to display these commands or serve as a guide to them. Since the buttons are interrupt-drive, we needed an efficient solution without computationally expensive code run on the interrupt handlers. Hence, a one-hot encoding variable was used to allow a switch statement to be called on every button interrupt, and based on this variable’s value, the correct message would be displayed on the LCD and the corresponding action (if necessary).

The table below shows the one-hot encoding used to identify the menus and submenus.

| State | 0 | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Main | Help | Motor | Auto | MotorR | MotorL | MotorU | MotorD | MotorF | MotorB |

The following figure shows an example interrupt handler (middle button) and how the *state* variable is used to determine the necessary actions.

**void** **EXTI9\_IRQHandler**(){

/\*\*

\* Control for middle button (MB)

\* GPIOG9

\*/

**delay\_ms**(100);

**switch**(state){

**case** 0: // MainMenu start

**if**(menuStart){ // start condition

**MenuOptions**(mainMenu, 0, 1);

menuStart = false;

}

**else**{ //

**HelpMenu**(); // open help menu

**MenuOptions**(mainMenu, menuCount, menuCount+1);

//state = 0; // stay in main menu

}

**break**;

**case** 1:

**MenuOptions**(mainMenu, 0, 1);

state = 0; // return to main menu

**break**;

**case** 2:

**MenuOptions**(motorOptions, motorCount, motorCount+1);

state = 8; // right motor is first motor option

**break**;

**case** 4: // automenu

/\*

\* Do automatic solver stuff

\*/

**break**;

**case** 8: // Motor Right

**cursorControl**(NULL);

motorOptions->menuitems[motorCount]->handler(motorOptions->menuitems[motorCount]);

**cursorControl**(motorOptions->menuitems[motorCount]->name);

**break**;

**case** 16: // Motor Left

**cursorControl**(NULL);

motorOptions->menuitems[motorCount]->handler(motorOptions->menuitems[motorCount]);

**cursorControl**(motorOptions->menuitems[motorCount]->name);

**break**;

**case** 32: // Motor Up

**cursorControl**(NULL);

motorOptions->menuitems[motorCount]->handler(motorOptions->menuitems[motorCount]);

**cursorControl**(motorOptions->menuitems[motorCount]->name);

**break**;

**case** 64: // Motor Down

**cursorControl**(NULL);

motorOptions->menuitems[motorCount]->handler(motorOptions->menuitems[motorCount]);

**cursorControl**(motorOptions->menuitems[motorCount]->name);

**break**;

**case** 128: // Motor Front

**cursorControl**(NULL);

motorOptions->menuitems[motorCount]->handler(motorOptions->menuitems[motorCount]);

**cursorControl**(motorOptions->menuitems[motorCount]->name);

**break**;

**case** 256: // Motor Back

**cursorControl**(NULL);

motorOptions->menuitems[motorCount]->handler(motorOptions->menuitems[motorCount]);

**cursorControl**(motorOptions->menuitems[motorCount]->name);

**break**;

**default**:

}

// clear interrupt

EXTI->RPR1 |= (1<<9);

}

**Figure 5: Middle button switch statement**

Note that the middle button is only used as a select, so this code snip does not change the state variable. However, going up and down does (since we want to point at a specific submenu), so the switch statement actions differ, but the logic is very similar. Ensuring that only one variable was used greatly improved the system's performance and readability of the code.

As mentioned, the middle, up, and down buttons had specific purposes on the LCD; however, we wanted the right and left buttons to control the motor direction if a motor was selected (states >= 8). To perform this, the switch statement remained very similar, except that the motor control functions were called instead of writing to the screen. Since every state represented a motor, integrating motor control was very straightforward. The figures below show how the manual motor control looks after a motor is selected from the motor options menu.



**Figure 6: Motor Options Menu**



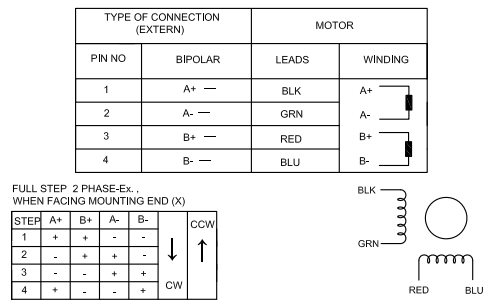
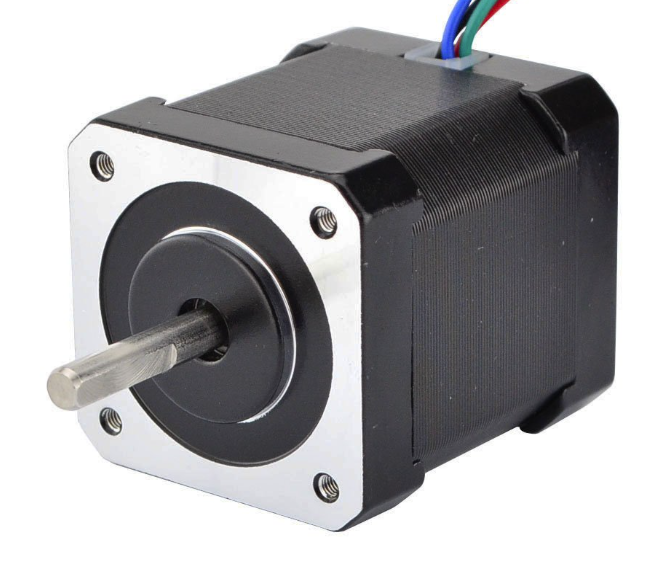
**Figure 7: Manual motor control message**

## 

## **COMPONENTS LIST**

* STEPPERONLINE 17HS19-2004S1 stepper motors

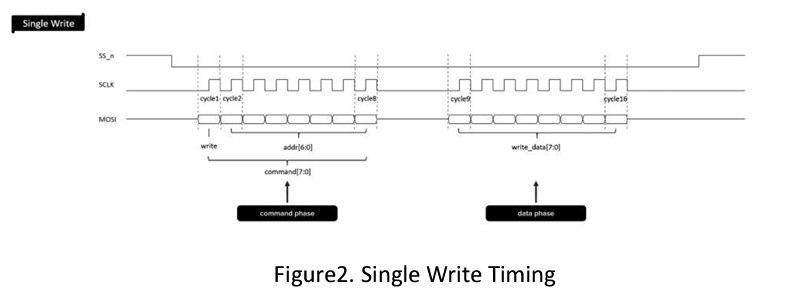
The machine uses six STEPPERONLINE 17HS19-2004S stepper motors. These are Nema 17 bipolar motors known for their high torque and stability. They offer a holding torque of 59Ncm (83.55oz. in) with a rated current of 2A per phase. The motors feature a 1.8° step angle, 1.4 ohms phase resistance, and 3.0 mH inductance. They measure 42x42mm in frame size, with a body length of 48mm and a 5mm shaft diameter.

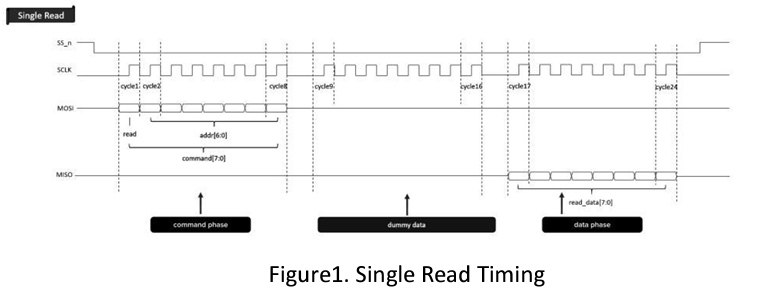


* Arducam Mega-5MP

The Arducam Mega 5MP SPI Camera Module is a sophisticated imaging component that works seamlessly with microcontrollers through a standard SPI interface. It offers a 5-megapixel resolution and autofocus functionality, which ensures clear and sharp images suitable for various applications, including IoT and machine learning environments. The camera operates on a rolling shutter mechanism and supports a 1/4" optical size with an 8MHz interface speed. Additionally, it is characterized by low power consumption, with operational metrics ranging between 182mW and 650mW.





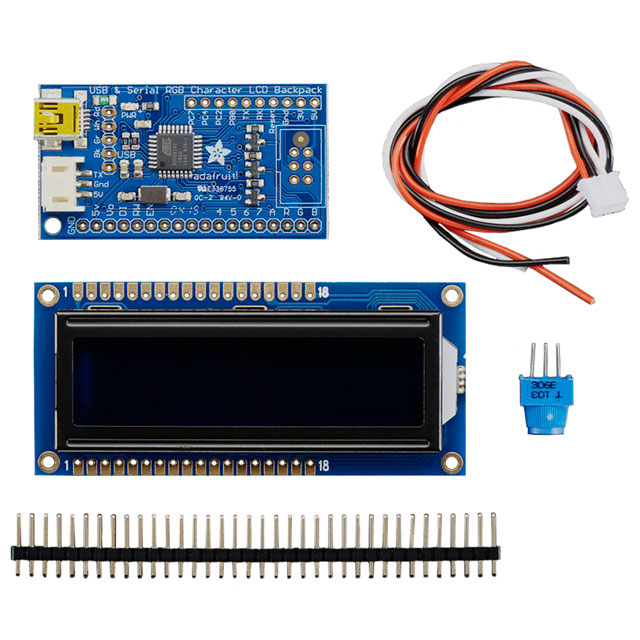


* 16x2 LCD screen with UART backpack (Adafruit)

The Adafruit 784 is a USB + Serial Backpack Kit with a 16x2 RGB negative LCD, making it an excellent addition for projects requiring a character display. The kit is designed to integrate quickly and supports USB and TTL serial inputs. The USB interface functions as a COM/serial port on various operating systems, and the TTL serial input operates at adjustable baud rates from 2400 to 57600.

This product has an AT90USB162 microcontroller chip, which handles all the commands for display operations, such as adjusting the backlight, setting the contrast, and creating custom characters. Users can modify and store settings like baud rate and backlight color in EEPROM, ensuring that customizations persist through power cycles.

The kit also supports a range of commands for display control compatible with Matrix Orbital systems, making it versatile for various applications, particularly those that require an easy-to-use LCD interface. The Adafruit 784 is ideal for users who need a customizable display interface with minimal setup and versatile control features.



* MDee COB LED Strip Lights

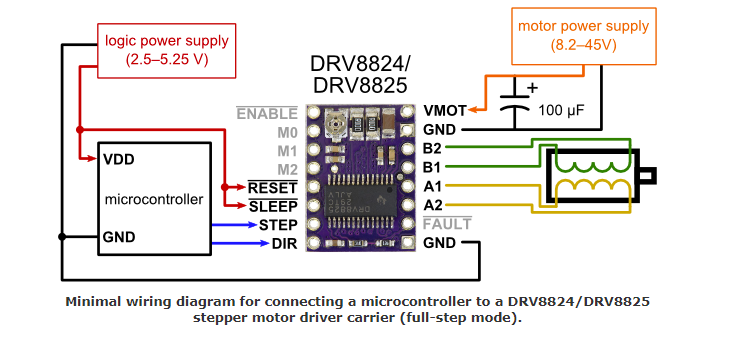
The MDee COB LED Strip Lights were utilized in the Rubik's Cube Solving Machine as a critical component for illuminating the cube during the color evaluation phase. These USB-powered, flexible LED tape lights emit a cold white light at 6000K, providing clear and bright illumination with an impressive CRI of 90+, essential for accurately detecting the cube's colors. The high density of 320 LEDs per meter ensures a uniform light distribution, eliminating shadows and hotspots that could interfere with the color recognition process.

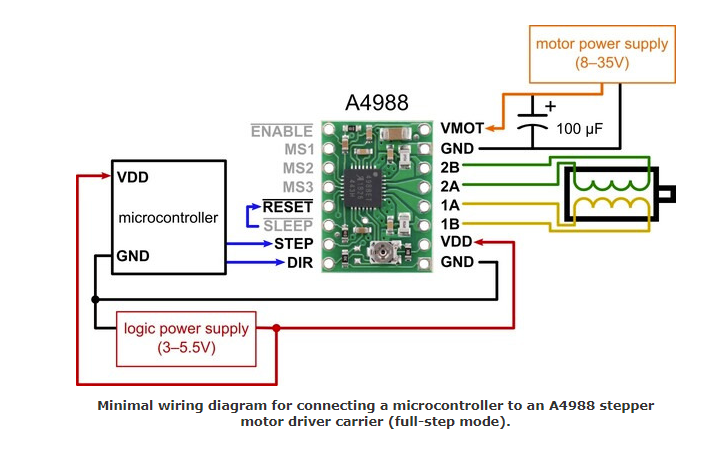


* A4988 and DRV8825 motor drivers

The Pololu's A4988 and DRV8825 stepper motor driver modules are designed to drive micro-stepping bipolar stepper motors. The A4988 can operate stepper motors at up to 35V and ±2A, with microstepping resolutions down to 1/16-step. It features adjustable current limiting, over-current, and over-temperature protection, and a simple step-and-direction interface. The module also includes a potentiometer to adjust the current output, which can be set by measuring the voltage on the VREF pin or changing the potentiometer while measuring the motor current.

The DRV8825 offers enhancements over the A4988, including handling higher motor supply voltages up to 45V and increased current up to 2.5A per coil with sufficient additional cooling. It also supports finer micro-stepping resolutions up to 1/32 step. Like the A4988, it includes over-current protection and thermal shutdown features but also introduces a FAULT output that indicates over-current and thermal shutdown states. Its current limit setting is adjusted similarly by measuring the voltage at the VREF pin or setting it through the potentiometer​.

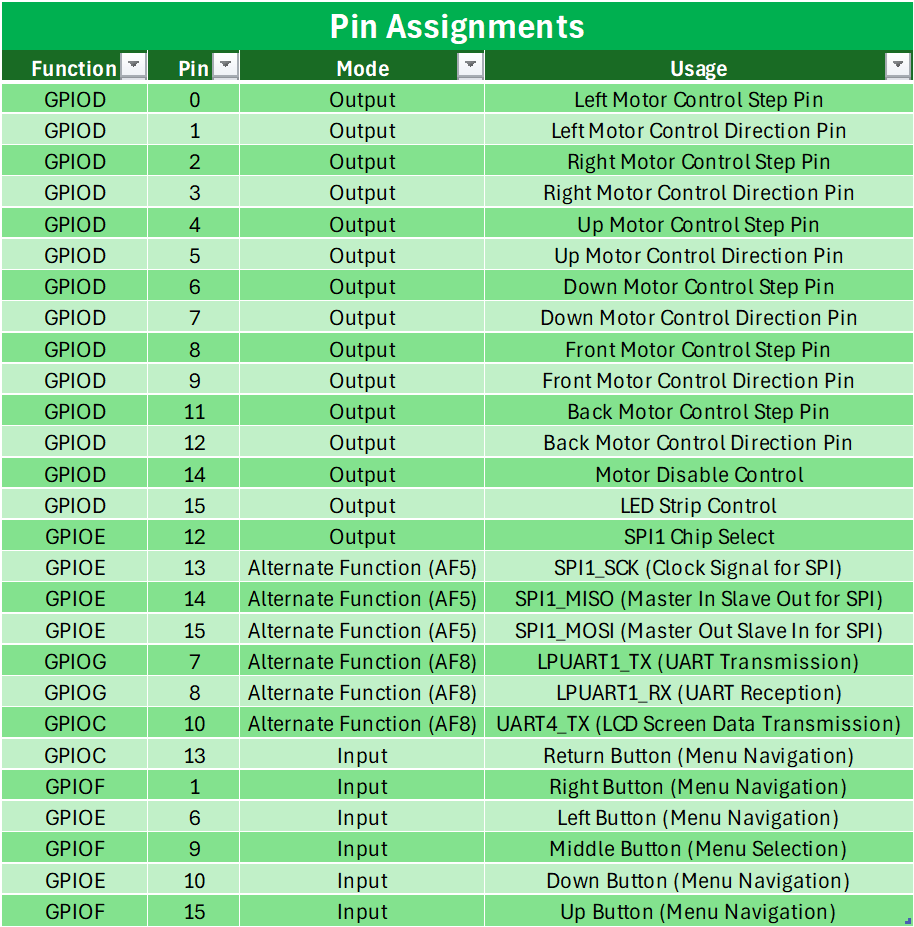




## 

## **Pin Assignment**

Motor control is managed through pins GPIOD0 to GPIOD9, GPIOD11, and GPIOD12, configured as outputs to drive stepper motors precisely. For high-speed data communications, GPIOE13, GPIOE14, and GPIOE15 are assigned alternate function pins for SPI1 (SCK, MISO, and MOSI, respectively). UART communication is enabled through GPIOG7 and GPIOG8 for LPUART1, and GPIOC.10 is utilized for UART4 TX to facilitate data transmission to an LCD screen. Additionally, interface buttons for user interaction are connected to GPIOC13, GPIOF1, GPIOE6, GPIOF9, GPIOE10, and GPIOF15, set as inputs to navigate the system's user interface.



## **POWER CONSIDERATIONS**

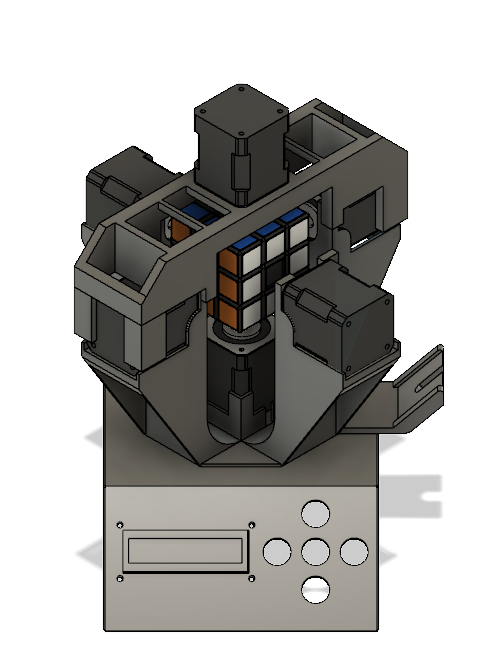
The project is designed with dual power supply systems to handle the power requirements of different components efficiently. A 12VDC power supply is dedicated to driving the stepper motors. This voltage level is necessary to ensure adequate torque and speed from the stepper motors, which are critical for the precise movements required in the system's operation. The 12VDC supply provides a stable and sufficient current to drive the motors without the risk of voltage drops that could disrupt performance.

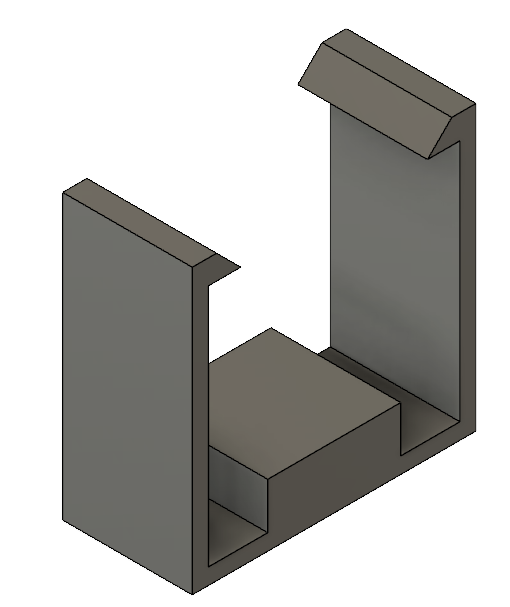
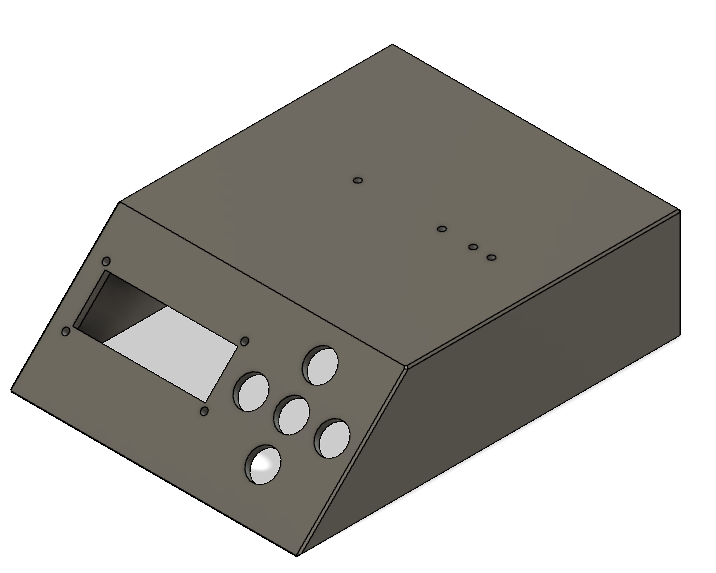
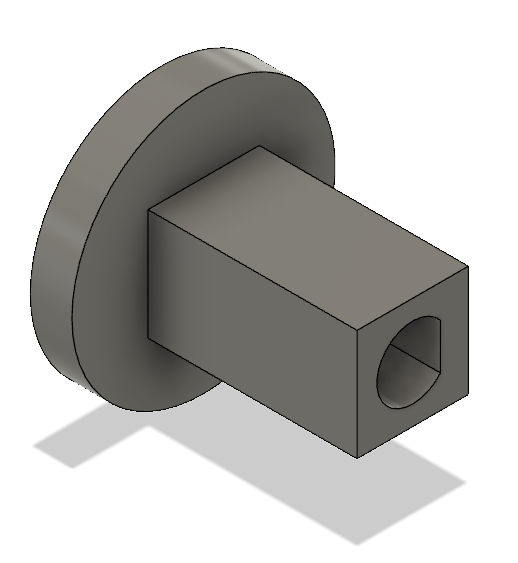
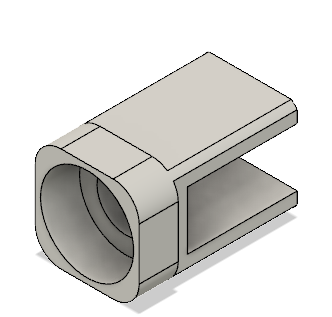
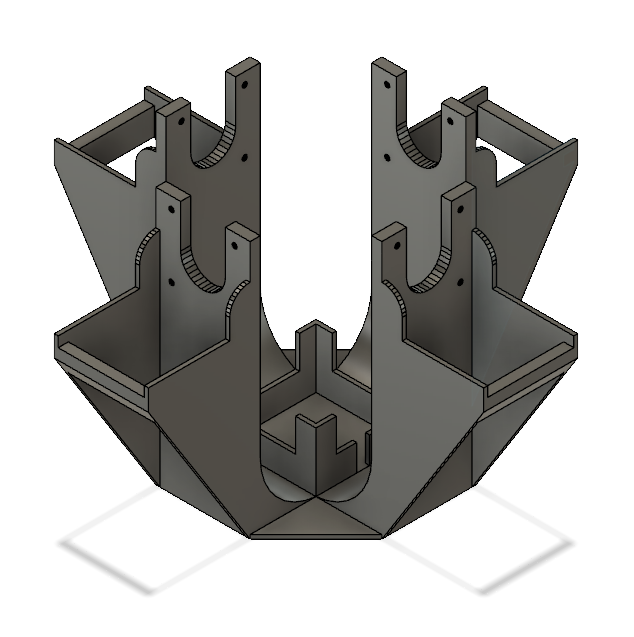
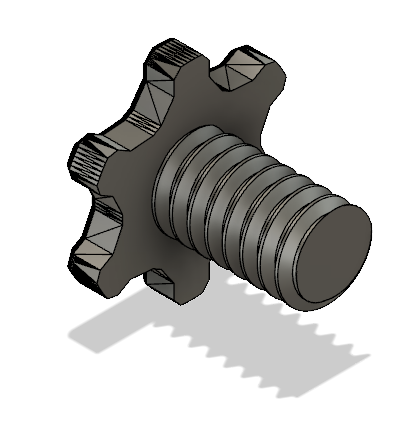
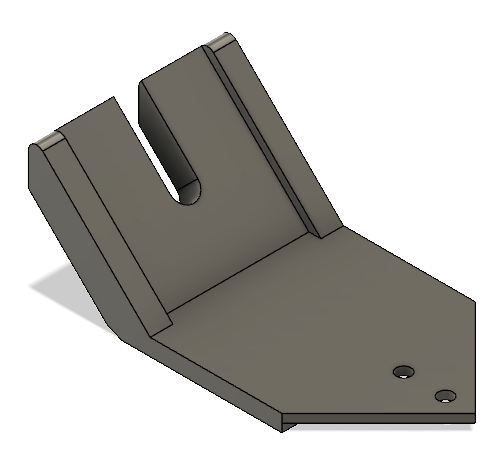
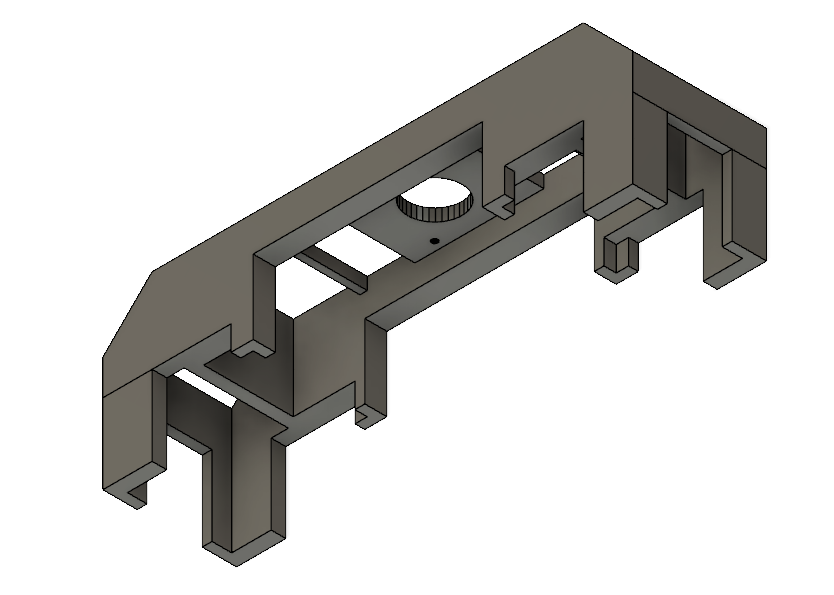
In addition to the 12VDC system, a separate 5VDC power supply is used for powering the LED strips, the LCD, and the navigation buttons. This lower voltage suits these components, which require less power and are sensitive to higher voltages. The 5VDC supply helps to isolate the sensitive electronics from the high-current demands of the stepper motors, thereby protecting them from potential electrical noise or interference. This bifurcation of power supplies optimizes electrical efficiency and enhances the system's overall reliability and safety.

## **3D MODELING AND MECHANICAL DESIGN**

The Rubik's Cube Solving Machine's structural components were designed using Fusion360, a 3D modeling software that allowed for precise and tailored construction of each part. Key elements such as the motor mounts, machine base, and camera mounts were engineered to ensure stability, durability, and functionality. This design process was crucial for creating a compact yet accessible layout that supports easy maintenance and seamless integration of all components.

The parts include the Motor Mount-Top and Motor Mount-Main, which secure the stepper motors to the machine frame, ensuring precise movement and alignment. The Machine Base provides a foundation for the apparatus, and a housing for the LCD screen and navigation buttons. The Camera Mount is tailored to hold the vision system securely. Couplers for the cube and motors and a Coupler-Retainer Clip were also developed to connect moving parts smoothly and reliably. Each component was 3D printed, allowing for rapid prototyping.





## **CONCLUSION**

The design and implementation of this project demonstrate a robust integration of mechanical, electronic, and software components to achieve a high-functioning system. The system ensures efficient power distribution and operational reliability by employing a 12VDC power supply for the stepper motors and a separate 5VDC supply for the LED strips, LCD, and navigation buttons. Utilizing different communication protocols, such as UART and SPI, enhances the versatility and scalability of the system, allowing for future upgrades and modifications.

Overall, the project meets its intended functional requirements and offers a platform for further development in automated and precision-controlled environments. The careful consideration of power management, communication protocols, and interrupt handling underscore a comprehensive approach to system design that could be applied to various robotics, automation, and beyond applications. This project is a valuable model for integrating multiple technologies into a cohesive system capable of performing highly efficient and reliable complex tasks.

## **REFERENCES**

1. STEPPERONLINE. (2023). 17HS19-2004S1 Stepper Motor Datasheet. Available at: [https://www.omc-stepperonline.com/download/17HS19-2004S1.pdf.](https://www.omc-stepperonline.com/download/17HS19-2004S1.pdf)
2. Amazon. (2024). Image of STEPPERONLINE 17HS19-2004S1 Stepper Motor. Retrieved from [https://m.media-amazon.com/images/I/617QPT7+sIL.SL1000.jpg.](https://m.media-amazon.com/images/I/617QPT7+sIL.SL1000.jpg)
3. Amazon. (2024). Image of Arducam Mega 5MP SPI Camera Module. Retrieved from <https://m.media-amazon.com/images/I/61RKn8DyhVL._AC_SX300_SY300_QL70_FMwebp.jpg>.
4. Arducam. (2023). Arducam Mega SPI Camera Application Note. Retrieved from <https://www.arducam.com/downloads/datasheet/Arducam_MEGA_SPI_Camera_Application_Note.pdf>.
5. Digi-Key Electronics. (2024). Image of LCD Display. Retrieved from <https://mm.digikey.com/Volume0/opasdata/d220001/medias/images/2555/MFG_784.jpg>
6. Amazon. (2024). Image of MDee COB LED Strip Lights. Retrieved from <https://m.media-amazon.com/images/I/61+nD+uVGTL.AC_SL1500.jpg>.
7. Pololu Corporation. (2024). DRV8825 Stepper Motor Driver Carrier. Retrieved from <https://www.pololu.com/product/2133>.
8. Pololu Corporation. (2024). A4988 Stepper Motor Driver Carrier. Retrieved from <https://www.pololu.com/product/1182>.