Simplifying Part Access

By
Matheu Fletcher (matheuf2)
Aidan Yaklin (ayaklin2)
Tejas Aditya (taditya2)

ECE 445 Final Report - Fall 2023
TA: Gregory Jun

6th December, 2023
Project No. 26

Abstract:

The goal of this report is to discuss the creation of our project Simplifying Part Access, an automatic tileable dispenser for tape and reel surface mount device components. Our project allows a user to set information per each reel and request a quantity of components off of said reel. More specifically, we discuss how the subsystems and how we achieved the required precision for such a device to work. Additionally, while we know the device works, there are a few improvements to be made to improve precision and consistency, and to reduce the project's physical footprint. Further improvements to the software could also improve ease-of-use and add functionality, such as wireless connectivity or simultaneously dispensing from several reels.

Table of Contents:

1.Introduction

1.1 Problem

Printed circuit board designs, even at the prototype/hobbyist level, can use a significant number of small components. Typically, these components are packaged in a reel of paper/plastic “tape” with cutouts for each component and a removable film to hold the components inside the tape. The film can be peeled back to remove a few, but it is very easy to accidentally peel it too far and lose a lot of parts. Thus, the preferred way of working with tape-and-reel packaging is to cut the tape to the required length for the project, leaving the film intact on the rest of the reel.

This in itself poses a problem, though. If a project needs more than a few of the same part, manually counting to find the cut point becomes very tedious, with a higher likelihood of making a mistake, and ending up with the wrong number of components.

1.2 Solution

A way to approach this problem would be designing a modular system where the end user can request a specific number of parts from a reel, and it feeds the required length of reeled components out. This can be accomplished using a tileable design where each tile is a box that holds a reel of the desired components. The tape on the reel is pulled by a stepper motor, and the sections are cut by a motorized blade. To confirm that the correct number of components are dispensed, an IR sensor would have been used to measure the movement of the tape. But due to constraints that occurred regarding figuring out the sensor's placement with respect to the mechanical components of the design, it did not get implemented. As such, the lack of the IR sensor is the one deviation from the block diagram.

1.3 Initial Visual Aid

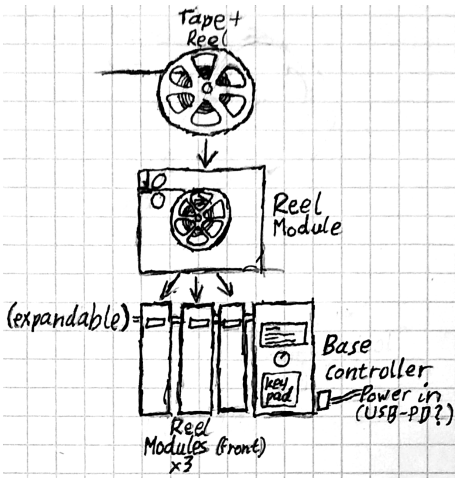


Figure 1: Illustration of proposed overall solution

1.4 High-Level Requirements

- Movement/Sensing - The system should be able to accurately measure out the correct number of components when given the correct component spacing and clocked properly. Movements should be accurate to +/- 1 mm of the requested position.
- The system should have a successful yield rate of 90% at the minimum to ensure that in large quantity orders, parts are not wasted. At least 90% of the time the cutting process dispenses the right quantity of the components with clean cuts and doesn't damage any of the components.
- The system should accept the most common reel diameters (7 inch/178 mm and 13 inch/330 mm), and it should support tape widths up to at least 25 mm, and tape depths up to at most 4 mm.
- The base controller should be able to support at least 3 reel modules. It should supply enough power to drive 3 reel modules simultaneously, and it should disable additional modules so that power limits are not exceeded. (We are currently planning on building 3 reel modules, so the power-limiting behavior can be tested by reducing the limit to 1 or 2).

These requirements act to ensure that we minimize how often the components being dispensed are damaged, while also making sure that it is usable for a variety of reels, both in terms of size and quantity. As such, these requirements act as guidelines for us to ensure that our end goal is something that serves a tangible purpose.

2. Design

2.1 Block Diagram

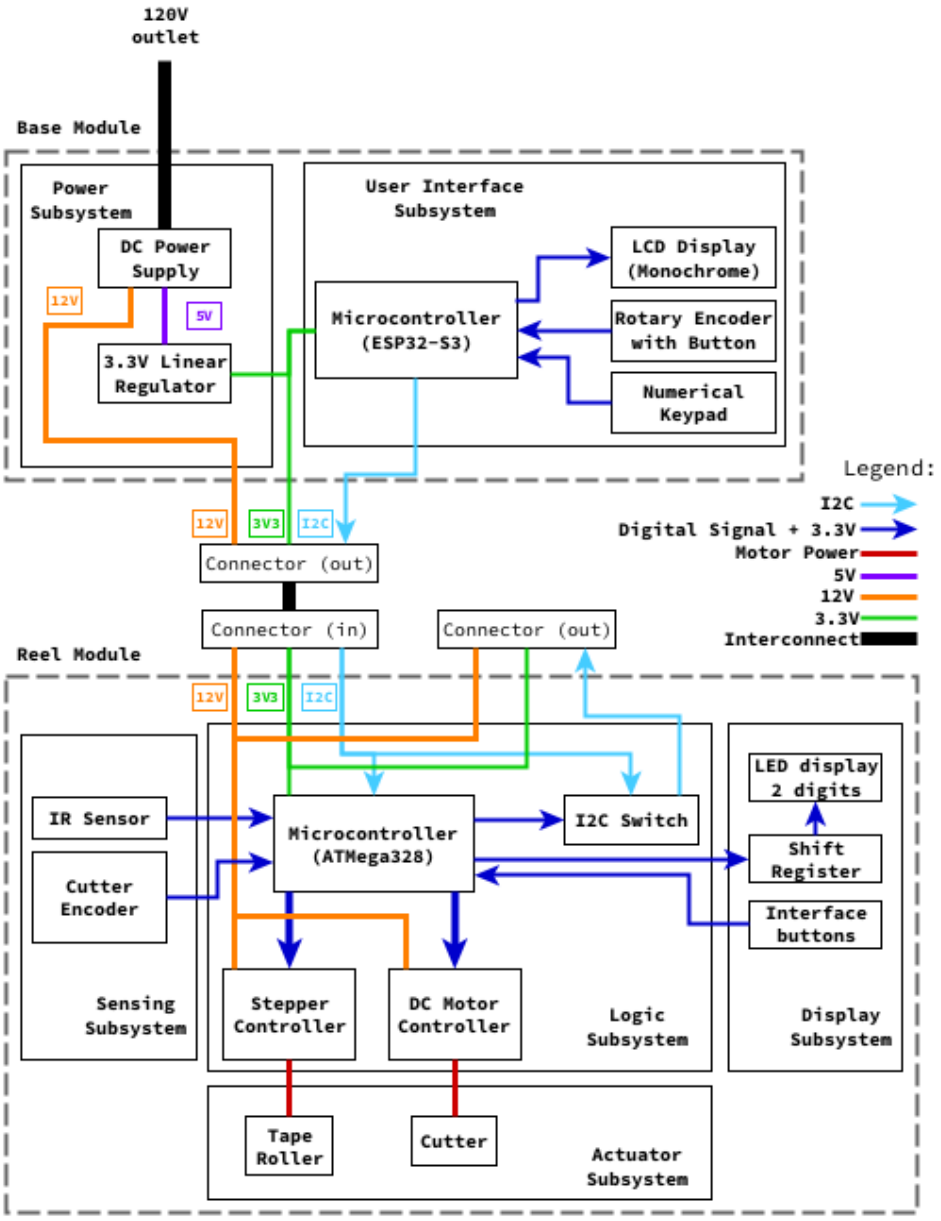


Figure 2: Diagram showcasing individual circuit components of proposed design

The device has two distinct subsystems that can be distinguished as the Base Controller and the Reel Module. The Base controller has all the components that a user will interact with. It has input for power and regulates it to ensure all the components get the right voltage. The LCD screen and keypad allow the user to send commands, select components and quantity to be dispensed. All of this will be parsed by an ESP32 onboard the subsystem and the specific commands sent to the ATMega328's for each Reel Module. Inside each reel module, a stepper motor will pull the reel a predetermined distance to push out the right quantity of components which will be verified by an IR sensor using the sprockets in the reel as reference. Once it is verified, the cutter powered by a motor will slice the reel and dispense the components.

2.2 Subsystems

2.2.1 Power Subsystem

2.2.1.1 Power Supply

We are using a Mean Well RD-85A supply [9] to provide DC power to the device from a power outlet. This provides 85 watts of power as 12 V and 5 V outputs and connects directly to the base module for further regulation and distribution to the reel modules. Since the system runs on a 3.3 V logic supply, we have a standard LM1117 linear regulator on the base module to provide this voltage. The raw 5 V supply voltage is still used to power the LCD display on the base module.

2.2.2 User Interface Subsystem

This subsystem allows the user to operate the reel modules and view their current run status. It consists of an alphanumeric (character) LCD, a numeric keypad to enter spacing and component count values, and a rotary encoder to scroll through menus.

2.2.2.1 LCD

The original design called for using a Newhaven display with specs of 128x32 graphical pixels, monochrome, and backlit [1]. The current implementation uses a 16x2 alphanumeric LCD referenced in the parts table in section 4.1 below, which runs using HD44780U driver and is also monochrome and backlit. This change was done due to the small size of the original screen and the much simpler interface of the 16x2 LCD display.
This will display summary information for all of the modules, as well as menus to configure and control the machine. The details of the menu operation will be detailed in the algorithms section.

2.2.2.2 Rotary encoder

For this, we needed a generic clicky knob encoder with an integrated switch and a D-shaft; we can 3D print a knob for it. [2] This will allow the user to scroll on the summary page and select a reel module to configure.

2.2.2.3 Numerical keypad

Standard 12-button keypad with 0-9, *, and #. [3] This will allow the user to enter numbers for component spacing and other parameters. The "*" key is used for deselecting a reel and also as the kill switch if a cut command has already been sent for that reel module. The "#" key is used as the enter command after a reel is selected and the user is inputting the spacing, repeat quantity or component quantity. Since a repeat count or quantity per cut is not expected to be zero, it is used as indication that the user wants to reset the spacing for that module.

2.2.2.4 Main Processor

We opted to use the ESP32-S3 module because of its low cost, I/O pin count, micro-USB integration, and Wi-Fi option (although this is not a requirement for the project). Microcontrollers with similar performance and pin count typically cost significantly more. [4]

2.2.3 Reel Module - Logic Subsystem

This subsystem controls reel feeding, cutting, and component counting.

2.2.3.1 Microcontroller

We are using an ATmega328PB microcontroller to drive the reel modules. These combine reasonable pin count and processing power at a very low price point. They can be run at 3.3 V if the clock frequency is reduced from 20 MHz to 12 MHz [10]. The pin count ended up being reasonable for our use case, and the processing power was more than sufficient for the very low-level tasks that it had to perform. We were not constrained by the small amount of available memory either–the final software uses 13% of the available program memory and only about 5% of the RAM.

Processing-speed-wise, the processor was able to control all of the subsystems (especially the stepper) with a precision on the order of microseconds. It handled the operation of two simultaneous hardware timers, one at 1 kHz and the other at 133 Hz, along with a rapid I2C status interrupt and an encoder counter running somewhere in the order of a few kHz at full speed, without noticeable lag.

There were a few challenges with using this setup–first, the 328PB revision has additional hardware registers compared to the 328P (famously used on the Arduino Uno), so the base Arduino libraries were incompatible with this chip. Since we were not aware of an Arduino core for the ATmega328PB, we decided instead to write all the software using the low-level register interface provided by the AVR compilers.

Another difficulty was with the board design: we had overlooked a mistake on the schematic, and the ISP programming pins were swapped in the layout. Fortunately, this only required some minor rewiring of our ISP programmer.

2.2.3.2 I2C switch

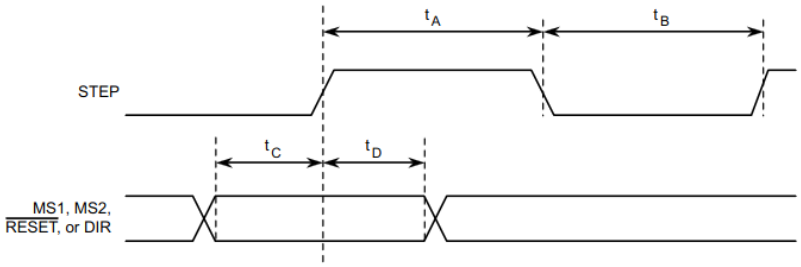
To enumerate the modules on the I2C bus, we employ an I2C bus buffer/switch so that additional modules do not respond to enumeration requests until it is their turn. More information will be provided in the algorithms section.

2.2.3.3 Stepper and DC Motor Drivers

For the stepper motor driver, we are using the Allegro A4982 chip, as it has sufficient current output for our NEMA-17 stepper and supports up to 16x microstepping (if additional resolution is required). This chip has STEP and DIR inputs to command the stepper, an ENABLE input to start/stop commanding the stepper, and a RESET input, which can be connected to the microcontroller's reset pin. [11]

In the final design, we enabled four-level microstepping on the stepper driver, which significantly improved the precision of the stepper drive and reduced the audible noise of the stepping. It also eliminated instances of the stepper slipping or missing steps while moving.

The stepper controller has specific timing requirements for its inputs, which we had to account for in the software, as can be seen in Figure 3:



Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t_A	1	µs
STEP minimum, LOW pulse width	t_B	1	µs
Setup time, input change to STEP	t_C	200	ns
Hold time, input change to STEP	t_D	200	ns

Figure 3: Stepper controller interface timing [11] (The table is part of the figure)

The control software has two-microsecond delays between changing direction and generating step pulses to ensure that these timing requirements are met. It also uses a hardware timer to ensure that the interval between step pulses is extremely consistent. Figure 4 shows the waveform generated by this software. The green line is the step signal, and the magenta line is the direction signal. The top half of

the image shows the full capture at 500 microseconds per division, and the bottom waveform shows a zoomed-in view of the first step pulse and its timing relative to the edge of the direction pulse (at a time scale of 5 microseconds per division). Also note the spacing of the step pulses on the zoomed-out waveform with respect to the division lines.

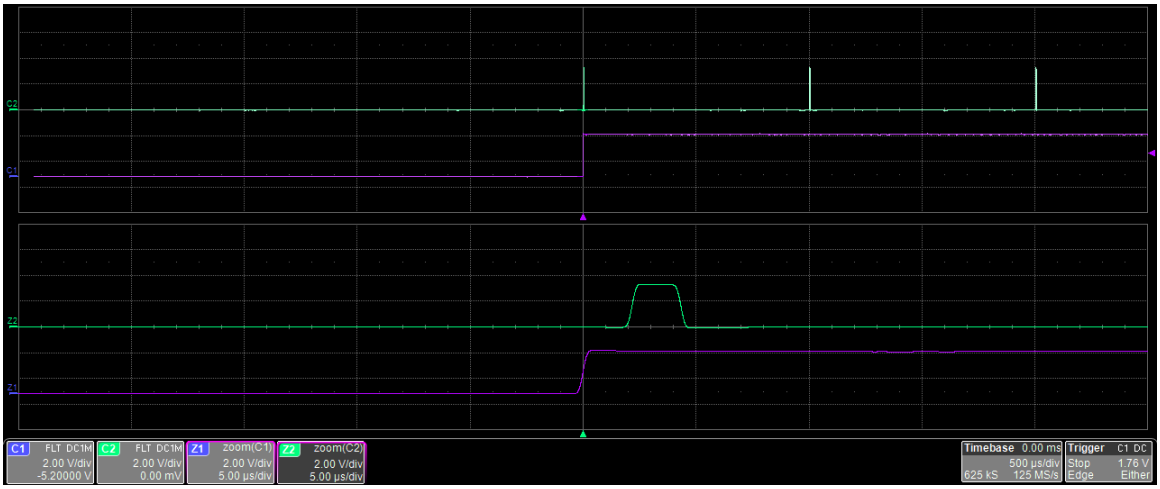


Figure 4: Stepper control waveform measurements.

For the DC motor driver, we are using the DRV8231 H-bridge driver for its low cost and current capabilities. This chip has a basic digital signal interface to control each side of the H-bridge:

Table 2. H-Bridge driver signal meanings

IN1	IN2	OUT1	OUT2	DESCRIPTION
0	0	High-Z	High-Z	Coast; H-bridge disabled to High-Z (sleep entered after 1 ms)
0	1	L	H	Reverse (Current OUT2 → OUT1)
1	0	H	L	Forward (Current OUT1 → OUT2)
1	1	L	L	Brake; low-side slow decay

To control the speed, the inputs can be toggled between a directional command and braking; using PWM, the active duty cycle can be adjusted continuously. [12]

2.2.4 Reel Module - Actuator Subsystem

This subsystem consists of the cutter and feeder actuators.

2.2.4.1 Feed Stepper Motor

After some deliberation running calculations on a different motor to determine the precision needed, as discussed in the later tolerance analysis, we decided to settle on a smaller NEMA17 motor that should provide the necessary precision. [8]

2.2.4.2 Cutter

We decided upon a gear motor with a rotary encoder for this [13]. The gear motor will rotate a standard utility blade attached to it, acting as a pivot and hinge, similar to that of a paper guillotine. Due to the risk of a blade and motor together, we plan to have it mostly enclosed so fingers cannot get in and risk getting injured, while also providing a kill switch to immediately cut power to the entire system.

When we went to assemble the final project, we realized that we selected such a high RPM motor that when we tested, we did not have enough torque to actually cut through the tape. As such, we quickly ordered a 10 RPM motor from Fafeicy [14], and tested it, confirming it had the extra torque that we needed to cut through the tape.

2.2.5 Reel Module - Sensing Subsystem

This subsystem contains the sensors to measure tape motion and cutter position. Since the IR sensor did not get implemented, it is no longer found here.

2.2.5.1 Cutter Motor Encoder

The cutter motor has a built-in magnetic quadrature rotary encoder to report changes in position to the microcontroller. The encoder produces two square-wave pulses 90 degrees out of phase from each other. Depending on the direction of rotation, the phase of the first signal will be ahead of or behind the second signal, which allows the microcontroller to keep track of the position even if the direction changes. Since the position measurement is relative to the position of the motor when the system boots, a calibration operation is run at startup to move the motor at very low power until it hits a physical end stop and stalls (again, at very low power).

2.2.6 Reel Module - Display Subsystem

The reel module will have an integrated 2-digit numerical display and a pair of buttons to allow the user to send basic commands to the reel module without needing a full base controller. Additionally, the LED displays will provide the immediate status of the reel module itself if it has lost communication with the base controller. Due to limited pin count on the microcontroller, we are using a standard 74HC595 shift register to drive the 7-segment displays. This halves the number of signals to control each digit, and it can supply more current (70 mA [15]), allowing for a much brighter display.

2.3 Algorithm Description

This device has several important algorithms that will be described below.

2.3.1 Display Menus

The main screen consists of individual lines providing information about the attached reel modules. This can be navigated by turning the encoder knob. Clicking the button when selected on a reel module will enter the settings for that module. The settings screen allows the user to change the component spacing and set up a cutting job (component count, repeat count) for the module. The user is also shown spacing if it is already saved on the reel board, and the max component quantity per cut based on the physical space limitations is shown on the quantity screen. This screen will have an exit button to return to the main menu. During a cutting operation, the user has the option to cancel the operation using the software implemented kill switch.

2.3.2 Enumeration Algorithm

When several reel modules are attached to the base, they would be virtually indistinguishable from each other, as they are all on the same I2C bus and have the same address. We resolve this by adding an I2C switch and an enumeration system so that when the system boots, only one module is visible on the bus. From here, the base module can assign a unique I2C address to this module and move on to the next one. The communication protocol is as follows:

- The reel module boots in “enumeration” mode, waiting for an ID from the base controller. There should be a mechanism (analog switch or I2C buffer) to disable communication passthrough to the next connected module until this device has been enumerated.
- The base controller sends an ID to the first connected module, and then the module re-enables I2C passthrough so that the next module (if any) can be enumerated. Already-enumerated modules ignore the ID assignment command.
- The base controller can then address both reel modules individually.
- The base controller checks for each reel if spacing has been stored previously for it and reads it from the reel board to store it on the ESP32.

2.3.3 I2C State Machine and Registers

The reel module acts as an I2C slave device, so it has to respond to I2C commands from the control module as quickly as possible. The full flowchart can be found in Appendix F. The software implements a “register” interface to set and read data–each register holds one byte; writing is performed by sending the register index followed by a data byte, and reading is performed by sending the register index and then reading the value. The software responds to the following register indices:

Table 3: I2C registers

Addr	Name	7	6	5	4	3	2	1	0	Description
0x00	REG_STATUS	Error code				x	Cal	Run	Err	Various status/control flags
0x01	REG_CUT	Cut count								Number of components to cut
0x02	REG_SPACING_L	Spacing [7:0]								Low byte of spacing value
0x03	REG_SPACING_H	Spacing [15:8]								High byte of spacing
0x04	REG_ENC_L	Encoder [7:0]								Encoder count low byte
0x05	REG_ENC_1	Encoder [15:8]								Encoder count second byte
0x06	REG_ENC_2	Encoder [23:16]								Encoder count third byte
0x07	REG_ENC_H	Encoder [31:24]								Encoder count high byte

Some registers have special functionality:

- Writing a 1 to the Run or Err bits of the status register will cancel a run operation or clear an error state, respectively. Writing a 1 to the Cal bit initiates a calibration operation. Reading from these bits returns whether such an operation is in progress or if an error has occurred.
- Writing to the cut count register starts a cut operation immediately.
- Writing to the high byte of the spacing register updates the spacing value for the next cut operation and stores it to the EEPROM.
- The encoder registers are read-only and used to measure the position as a 32-bit signed integer number of encoder state changes.

2.3.4 Step Calculation

To minimize accumulated error without an IR sensor (assuming perfect calibration), we designed the stepper code to keep track of the remainder when converting from spacing values to stepper steps:

$$\Delta = (S \times N \times MUL \times MS) \,/\, DIV$$

$$R = (S \times N \times MUL \times MS) \bmod DIV$$

R is then added to an accumulator, and a loop removes DIV from the accumulator and adds 1 to the delta until the total is less than DIV/2 (the ½ factor causes the algorithm to perform rounding instead of truncation). S = the spacing, N = the number of components, MUL/DIV = the conversion factor, and MS is the microstepping level.

2.4 Tolerance Analysis

This device needs to align the cut position within +/- 1 mm of the expected position. In order to do this, we need the resolution of the stepper motor to be small enough to achieve this accuracy. Our current choice of stepper motor has a specified 7.5 degree stride[5], and the stepper driver has support for 8 levels of microstepping[6]. We can calculate the angular resolution θ_M with N levels of microstepping and stride θ_S as:

$$\theta_M = \frac{\theta_S}{N} = \frac{7.5°}{8} = 0.9375°$$

From this, we can determine the maximum wheel radius that will allow our angular resolution:

$$s = r(\pi \times \theta_M)/180 = 0.5\,mm$$

$$r(0.016) = 0.5\,mm$$

$$r = 30.55\,mm$$

Having a larger wheel radius gives us additional pulling force from our stepper torque. We can use the upper bound from this result to determine the trade-off between torque and resolution when choosing exact parts and dimensions.

After doing these calculations, we realized our initial stepper motor choice was of convenience without enough research behind it, so we did more work and found one [8] that has a specified 0.9 degree by default, which would allow us to reach our desired level of precision without microstepping as further research has suggested that microstepping can be rather finicky. That said, we can always do a very minor amount of microstepping if we need to refine it further, but while these calculations told us that the precision was originally possible, they also helped us find a better option for even better expected results.

These calculations became less applicable later, as we were given a specific roller by the machine shop with fairly different dimensions, from which we took the radius of it and reversed the above calculations to instead get the current precision. The precision we got was ~0.43mm per step. While that was better than our requirement of 0.5mm per step, we found that it still was not precise enough to keep us from cutting through chips, so we enabled microstepping in order to move at a quarter of the current step size, which appeared to help reduce that behavior.

4. Costs

4.1 Parts

Table 4. Parts Ordered

Description	Manufacturer	Quantity	Price	Link
Nema 17 0.9 degree stepper	StepperOnline	3	$10.66	Link
LM1117IMPX-3.3/NOPB	Texas Instruments	1	$1.14	Link
RD-85 12V/5V power supply	MEAN WELL USA Inc.	1	$32.28	Link
Standard LCD 16x2	Adafruit	1	$9.95	Link
Rotary Encoder	Bourns Inc.	1	$1.92	Link
12 Button Keypad	SparkFun Electronics	1	$4.95	Link
ESP32 board	Espressif Systems	1	$3.48	Link
IR Sensor	SHARP/Socle Technology	3	$0.68	Link
Stepper Motor Driver	Allegro Microsystems	3	$3.65	Link
PLA filament to print misc. parts	Overture	1	$18.99	Link
ATMEGA328PB-ANR	Microchip Technology	3	$1.85	Link
Razor blade for cutter	Blades	1	$6.85	Link
DC Motor Drive	Texas Instruments	3	$1.52	Link
Worm Gear Motor	Fafeicy	3	$18.30	Link
7-segment display	Lite-On Inc	6	$1.75	Link
Schmitt Inverter Chip	Texas Instruments	3	$0.29	Link
Mosfet Array	Toshiba Semiconductor	3	$0.24	Link
Shift register	Nexperia USA Inc.	3	$0.46	Link
IC Driver	Texas Instruments	3	$1.27	Link

4.2 Labor

The total cost for parts is listed in the figure below before shipping comes out to $207.50. With 5% shipping and tax as 10% as seen in previously submitted projects, it would be $238.63 for parts. Using the 2.5 rule for lab with an estimated $35/hr for 150 hours we expect the project to take across team members, that leaves us with 2.5 * $35/hr * 150 hrs = $13,125 labor cost. Together the total cost of this project is $13,125 + 238.63 = $13363.63.

5. Schedule

Table 5. Team schedule throughout the project

Week	Task	Person
October 2nd - October 8th	Order and gather parts for prototyping	Everyone
	Start PCB Design	Aidan & Tejas
	Research Stepper motor control	Matheu
October 9th - October 15th	Finish PCB design and pass audit	Aidan & Tejas
	CAD initial design for machine shop & coordinate with them	Matheu
	PCB order October 10	Everyone
October 16th - October 22nd	Revise PCB design	Everyone
	Initial code thought process	Aidan & Tejas
	Continue main box CAD design and alternative design if needed	Matheu
October 23rd - October 29th	Start Assembling PCBs	Aidan & Tejas
	Parts ordered and CAD work continued	Matheu
October 30th - November 5th	Working with machine shop on design and alternatives still	Tejas & Matheu
	Continued board cleanup and code start	Aidan
November 6th - November 12th	Getting motors to spin	Aidan
	Codework	Aidan & Tejas
	Designing and printing control box	Matheu
November 13th - November 19th	Revising fitment of printed part with boards and interface devices.	Matheu
	Designed demo module box and mounts, along with mountable standoff for running module. Shortened wires for screen and troubleshooted a short.	Matheu
	Checked in on machine shop status	Matheu & Tejas
	Continued Code writing	Aidan & Tejas
November 20th - November 26th	Fall break, then at end Assembly grind, modifying code, testing and debugging	Everyone
	Extra coding	Tejas & Aidan
November 27th - December 3	Even more debugging, cleanup, and refinement. Then finally demo	Everyone

6. Conclusions

6.1 Accomplishments

Our team is rather proud of the work we contributed together to come up with the design that we did. At a functional level, all the pieces work and come together, we can specify reel spacing (which is saved to EEPROM), can request how many times to repeat a cut, and how many components to cut, and the I2C connection to the reel module works. From that, we know the feed mechanism for the tape works reliably, and that we can actually cut the tape. There are certainly issues regarding the exact precision, but in the end, we produced a finished product that simply needed an extra degree of refinement.

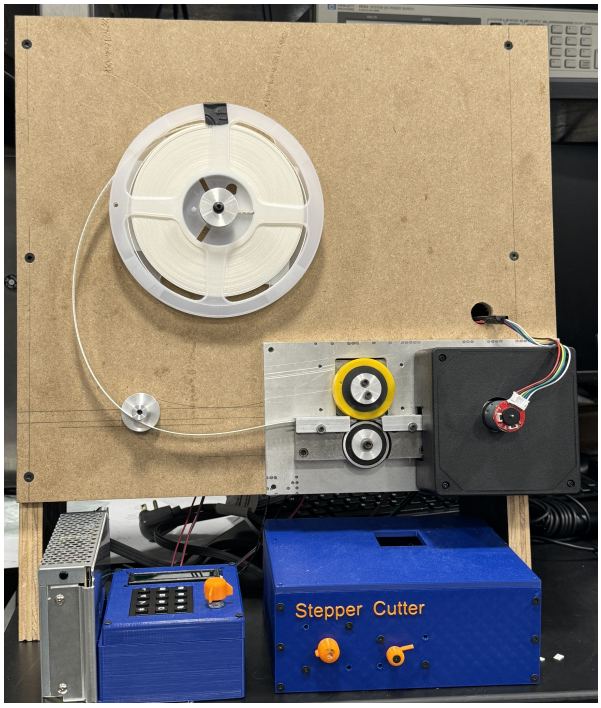


Figure 5. Completed project, with Base module, Reel Module, and prototype Reel Module

6.2 Uncertainties

The largest uncertainties surrounding this project is about how we can improve it, mostly in terms of reliability. While there are no exact numbers in this concern, the general concept is that any slight error that occurs in the feed is additively shifting its position in the tape, making it more likely to cut a chip, and when it cuts a single it is more likely to cut more. As a result, we know that any slight issues will grow over multiple runs.

Additionally, we were not really satisfied with the overall size of the project, and as a result are uncertain about the usefulness of the tileable design in its present state.

6.3 Ethical Considerations

When it came to core ethical concerns for this project, we did not see any clear risks or violations we were worried about, due to the lack of private information or conflicts of interest involved in this project. The notes we did make were about seeking honest criticism as specified by IEEE 7.8 I.5 [7], and to treat others fairly (Specified by IEEE 7.8 II) [7].

The only true concern we had was about the safety risk associated with the cutting blade mounted on the motor and reducing the risk of injury. Our approach to fixing this was primarily to enclose the motor such that it would be harder for an individual to get their fingers into that area. Additionally, we planned on having a hardware kill switch to help mediate that risk in case of any jams or other issues. In the end, we did implement a software kill switch rather than hardware, and we had a pretty developed procedure of making sure we unplugged the entire system and the cutting motor before touching anything near the blade. While an imperfect system this prevented any of us from getting injured. The most extensive damage done was to the top layer of skin on a figure when we checked to confirm that the blade had been dulled after some bad tests.

6.4 Future Work

There are many possible paths to how this project could be continued, the core of which would be a high degree of calibration on the feed distance, and modifying the design to better hold the smaller reels. The latter prevents the tape from curling up, which affects the placement of the blade when cutting. Similarly, the former could be implemented by adding the originally planned IR sensor and using that to recalibrate the feed rate.

Additionally, there are other optimizations that could certainly be made, such as trying to slim down the designs with smaller motors, that would allow the project to be more easily tileable.

Other final considerations consist of other such improvements, better aligning the cutting blade to increase precision, making sure the cuts fall out in the collection area cleaning, improving the cut speed, and adding verification of distance traveled with sensors like was previously discussed.

Another further improvement that could be interesting is the idea of using an SD card or USB port to upload BOM files for component counts to automatically handle scheduling jobs of the necessary cuts. This project overall showcased more proof of concept of what is truly possible, and there are many ways this can be improved upon.

7. Citations:

[1] “Newhaven Display Intl NHD-C12832A1Z-NSW-BBW-3V3,” Digi-Key Electronics. https://www.digikey.com/en/products/detail/newhaven-display-intl/NHD-C12832A1Z-NSW-BBW-3V3/2059235 (accessed Sep. 14, 2023).

[2] “Bourns Inc. PEC16-4220F-S0024,” Digi-Key Electronics. https://www.digikey.com/en/products/detail/bourns-inc/PEC16-4220F-S0024/3534239 (accessed Sep. 14, 2023).

[3] “SparkFun Electronics COM-14662,” Digi-Key Electronics. https://www.digikey.com/en/products/detail/sparkfun-electronics/COM-14662/8702491 (accessed Sep. 14, 2023).

[4] “Espressif Systems ESP32-S3-WROOM-1-N16,” Digi-Key Electronics. https://www.digikey.com/en/products/detail/espressif-systems/ESP32-S3-WROOM-1-N16/16162647 (accessed Sep. 14, 2023).

[5] “Small Stepper Motor - ROB-10551,” SparkFun Electronics. https://www.sparkfun.com/products/10551 (accessed Sep. 14, 2023).

[6] “Allegro MicroSystems A3967SLBTR-T,” Allegro Microsystems. https://www.digikey.com/en/products/detail/allegro-microsystems/A3967SLBTR-T/1006301 (accessed Sep. 14, 2023).

[7] “IEEE Code of Ethics,” *IEEE Code of Ethics*. https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Sep. 14, 2023).

[8] “Nema 17 Bipolar 0.9” StepperOnline. https://www.omc-stepperonline.com/nema-17-bipolar-0-9deg-11ncm-15-6oz-in-1-2a-3-6v-42x42x21mm-4-wires-17hm08-1204s (accessed Sep. 28, 2023)

[9] “Mean Well Power Supply.” https://www.digikey.com/en/products/detail/mean-well-usa-inc/RD-85A/7705988 (accessed Sep 28,2023).

[10] “ATMega328PB,” Microchip Technology. https://ww1.microchip.com/downloads/en/DeviceDoc/40001906A.pdf (accessed Sep 28, 2023).

[11] “A4982 Datasheet,” Allegro Microsystems. https://www.allegromicro.com/~/media/Files/Datasheets/A4982-Datasheet.ashx (accessed Sep 28, 2023).

[12] “DRV8231 3.7-A Brushed DC Motor Driver,” Texas Instruments. https://www.ti.com/lit/ds/symlink/drv8231.pdf (accessed Sep 28, 2023).

[13] “Uxcell gear motor” Uxcell. https://www.amazon.com/gp/product/B0792SBB5T/ (accessed Sep 28, 2023).

[14] “Fafeicy gear motor” Fafeicy. https://www.amazon.com/Reduction-Encoder-Self-locking-Reliable-Performance/dp/B08BL9D6BH/(accessed Nov 10,2023).

[15] “74HC595; 74HCT595” Nexperia. https://assets.nexperia.com/documents/data-sheet/74HC_HCT595.pdf (accessed Dec 6, 2023).

Appendix A Requirements and Verification Tables:

Table 6. UI System Requirements and Verifications		
Requirements	**Verification**	**Verified (Y or N)**
• The ESP32 can enumerate the attached reel modules on the I2C bus • The ESP32 can communicate with the attached reel modules on the I2C bus • The ESP32 can detect communication errors and fail safely	• Connect a reel module or microcontroller with reel module simulation software to the I2C bus • Connect the system to 3.3V power. • Verify that the ESP32 enumerates the connected module • Verify that the ESP32 correctly reads the module's configured component type. • Disconnect the module and verify that an error is reported	**Y**

• The LCD shows a summary of the status of each reel module, specifically:
  ○ the component type
  ○ the number of components to cut
  ○ the number of times to repeat
  ○ status (active/idle/error)
• position in the daisy chain.
• If an error status is present, the component information should be replaced with an error message.
• If no modules are attached, the display should show a "No modules attached" message.
• Each summary should be one line of text, and up to 4 can be displayed on screen without scrolling.
	• Connect the system to 3.3V power with no reel modules.
	• Verify that the LCD displays "No modules attached".
	• Disconnect power, add a microcontroller with a reel module simulation program to the I2C bus, and reconnect power.
	• Verify that the LCD shows the appropriate information when:
	  ○ A job is set up with 10 components repeated 5 times
	  ○ The job is started
	  ○ The job finishes
	  ○ An error occurs
		Y

• The ESP32 can read the rotary encoder and keypad	• Connect the system to 3.3V • Upload an input test program • Verify that the controller can differentiate between left and right turns of the encoder • Verify that the controller can detect when the encoder's button	**Y**

	is pressed ● Verify that the controller correctly reads the button when a single button on the keypad is pressed.	

Table 7. Reel Module Microcontroller Requirements and Verifications		
Requirement	**Verification**	**Verified (Y or N)**
● Can communicate with the base module over I2C at 400 kHz	● Attach a reel module to a working base module ● Apply 3.3V power to the system ● The device should be enumerated over the I2C bus and should be shown on the base module display.	**Y**
● Can send pulses to the stepper driver at a consistent rate (frequency should not deviate more than +/- 5% when running at a constant speed).	● With the stepper motor attached, probe the stepper coil voltages with an oscilloscope. ● Attach a reel module to 3.3V and 12V power. ● Use the on-board interface to command the module to dispense 100 components. ● Verify that the frequency read by the oscilloscope does not fall outside the specified tolerance.	**Y (see Figure 4)**
● Can save reel configuration and calibration to the chip when powered off	● Attach the reel module to 3.3V power ● Set the component spacing to a non-default value (5mm for example) using the on-board interface ● Cycle power to the reel module ● Use the on-board interface to verify that the spacing is the same	**Y**

Table 8. Feed Motor Requirements and Verifications		
Requirement	**Verification**	**Verification status (Y or N)**
• Maximum feed rate of at least 5 mm/s	• Apply 3.3V and 12V power to the reel module • Command the module to cut 100 components • Measure the distance moved and time to calculate the feed rate	**Y**
• Tape feed positional accuracy of at most +/- 0.5 mm	• Cut 10 individual components • Measure the width of each cut component and verify that it is within tolerance	**Y**
• Sufficient torque to pull tape from the reel, especially at low speeds	• Apply 3.3V and 12V power to the module • Command the module to cut 1 component • Verify that 1 component is cut from the tape, the stepper motor does not stick, and the tape does not slip.	**Y**

Table 9. Cutter Motor Requirements and Verifications		
Requirements	**Verification**	**Verification Status (Y or N)**
• Motor can cut through the the tape reliably within the span of a couple seconds	• Apply power, use the interface to command 10 cut operations of consistently length, and verify that the cutter does not jam, and cuts cleanly. • To provide efficiency, time cuts to ensure that once the motor is initiated, each takes a maximum of 2 seconds to cut.	**N**
• Confirm that the blade safety mechanism is working	• Make the opening only slightly larger than our tape, only 10mm in height, and confirm that someone can not reach their finger inside. • Ensure that the kill switch immediately cuts power to the motor, and that it slows down. Can confirm by taking slow motion video on a phone and frame peeking to compare the time required for a complete rotation before and after cutting power to the motor.	**N**

The reasons why these failed are due to specifics. Our current code implementation had the cuts within three seconds rather than two, just to be able to keep a better eye on things. It could have been sped up to meet the requirement. On the other hand, the safety mechanism is working, but our verification defines it more as a hardware implementation rather than the software implementation of a kill switch we ended up with. The hardware one should have also been added and would have been easy to add on the reel module, just adding a switch that is in between the power lines to the motor.

Appendix B Base PCB:

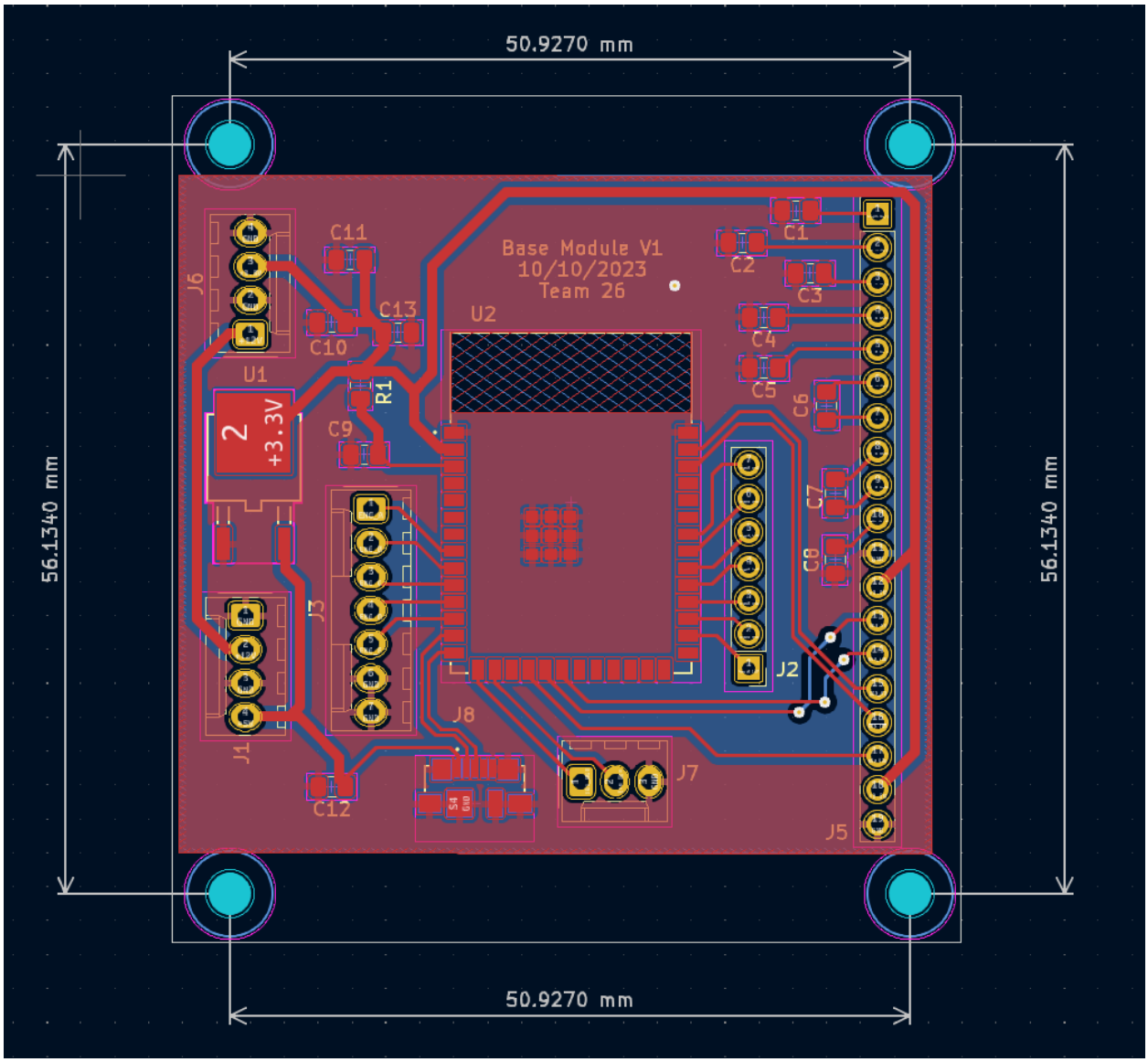


Figure 6: KiCad PCB design for base module

Appendix C Base Schematic:

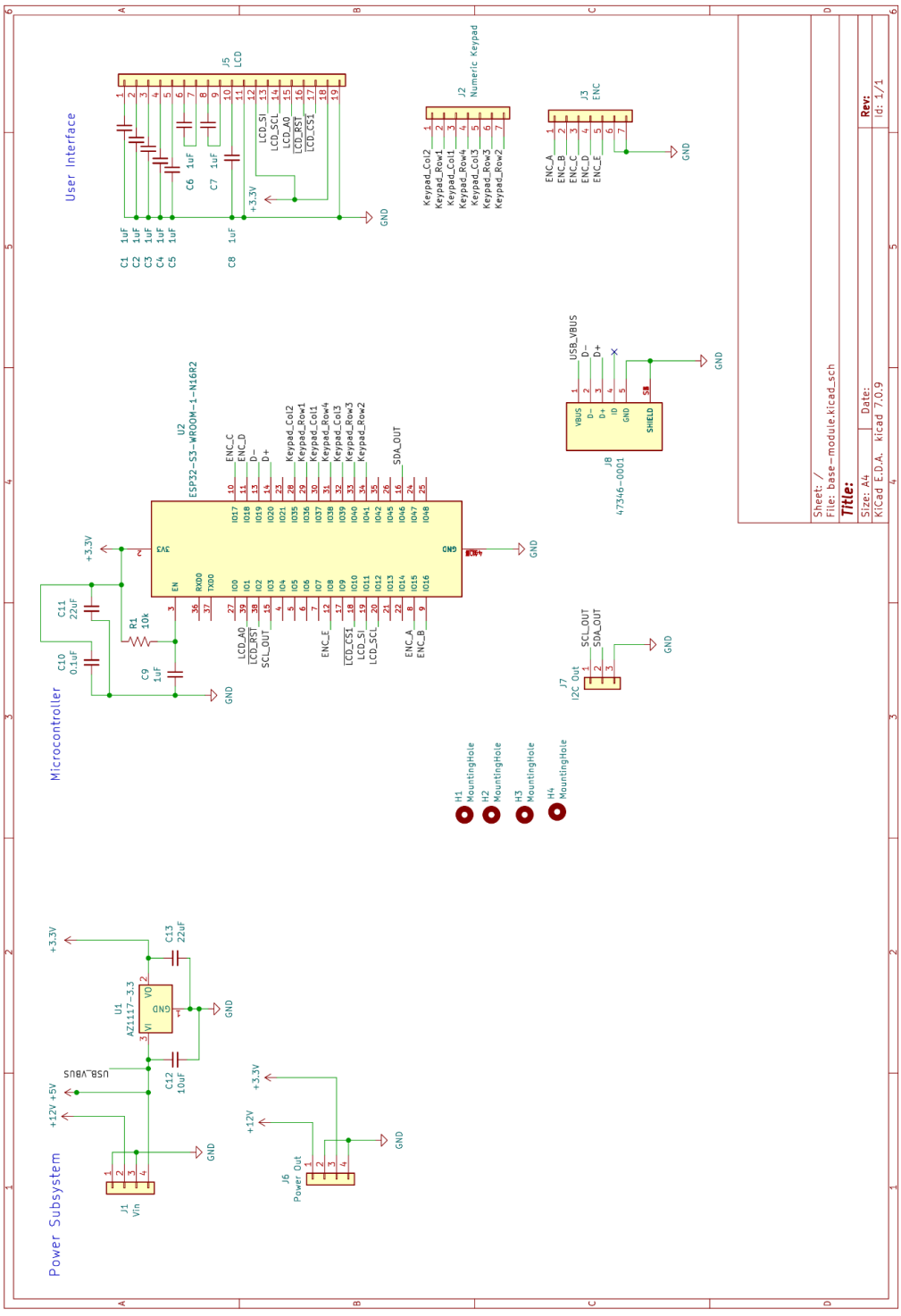


Figure 7: KiCad Schematic design for base module

Appendix D Reel Module PCB:

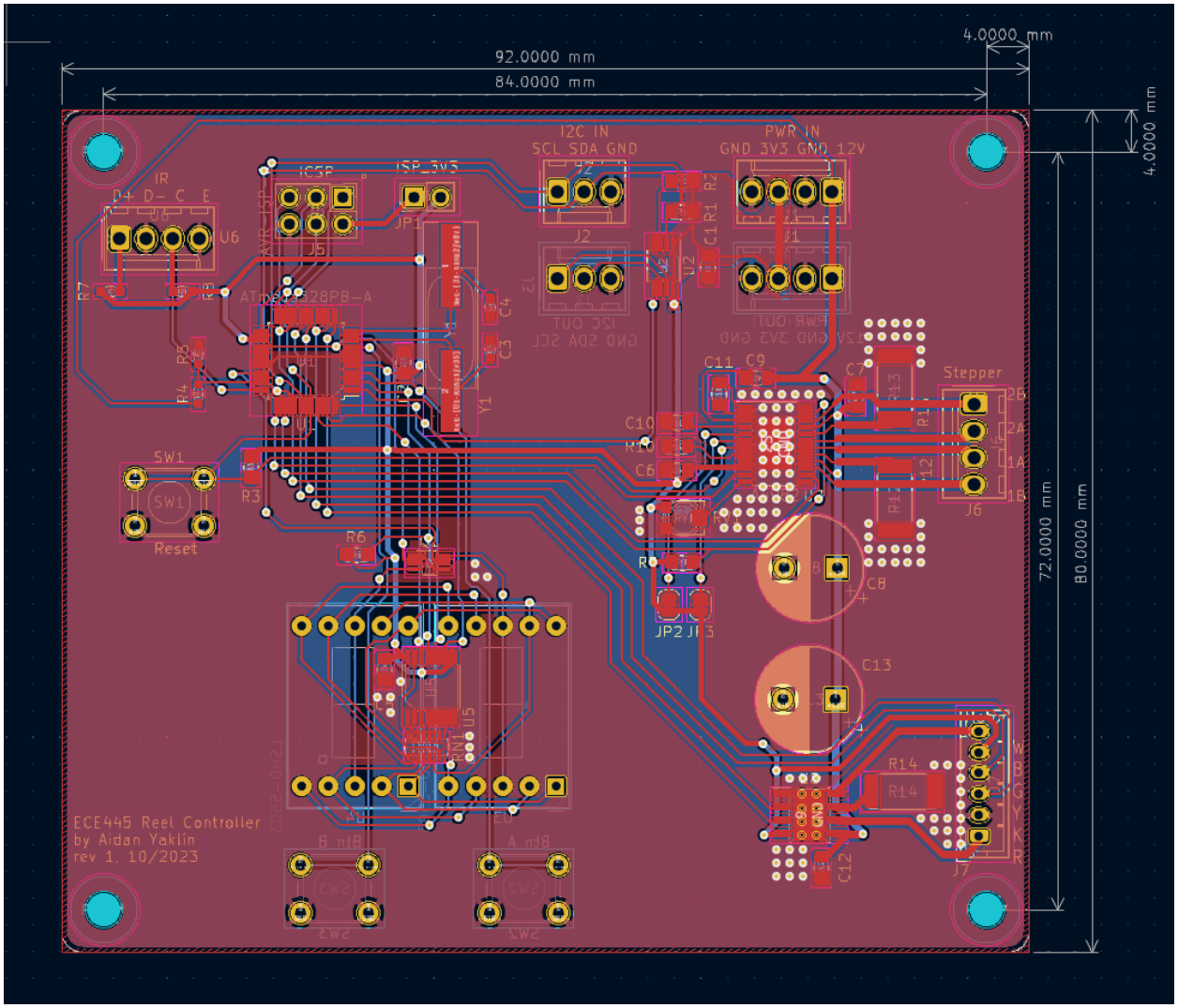


Figure 8: KiCad PCB design for reel module

Appendix E Reel Schematic:

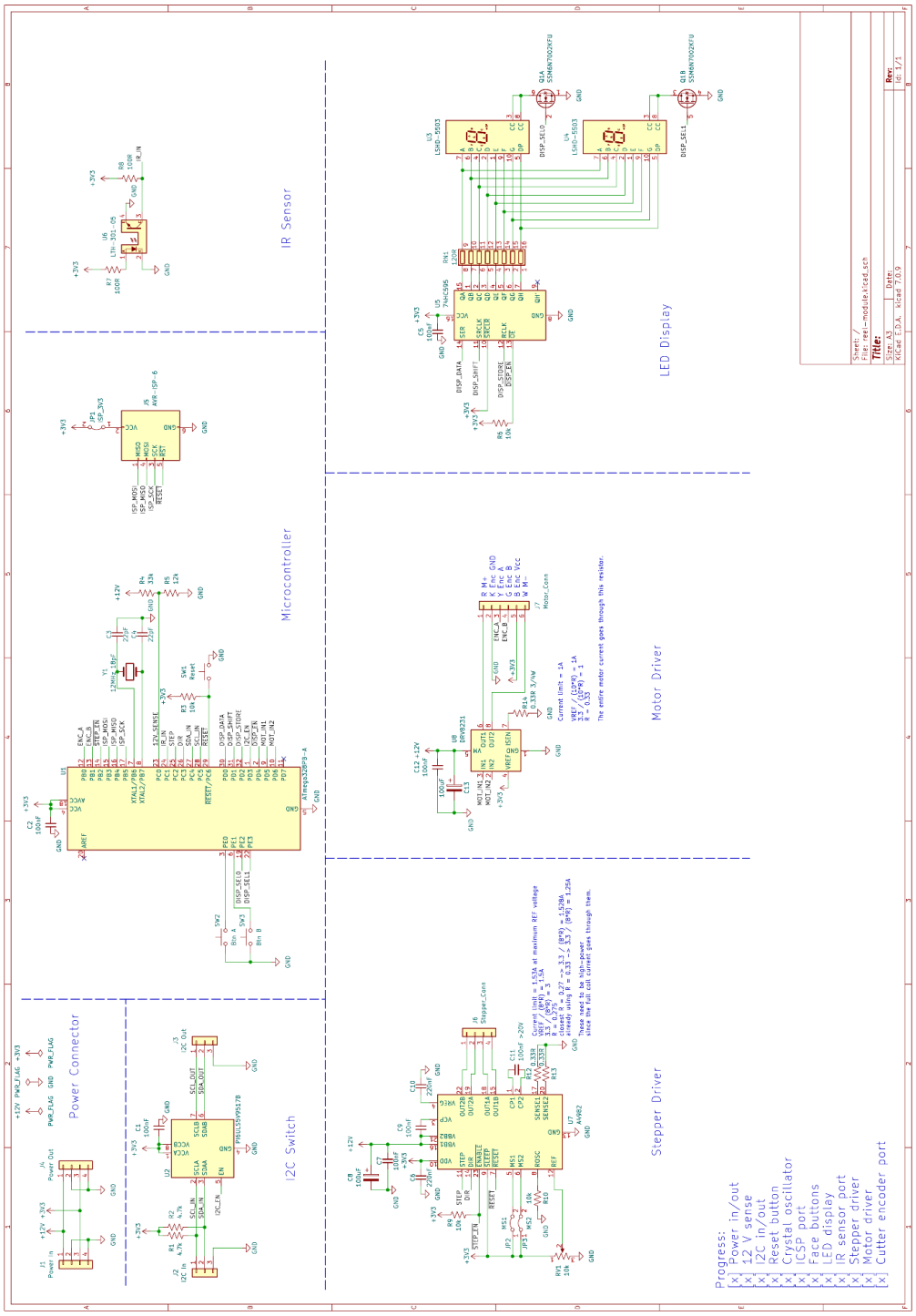


Figure 9: KiCad PCB schematic design for reel module

Appendix F Software Flowcharts:

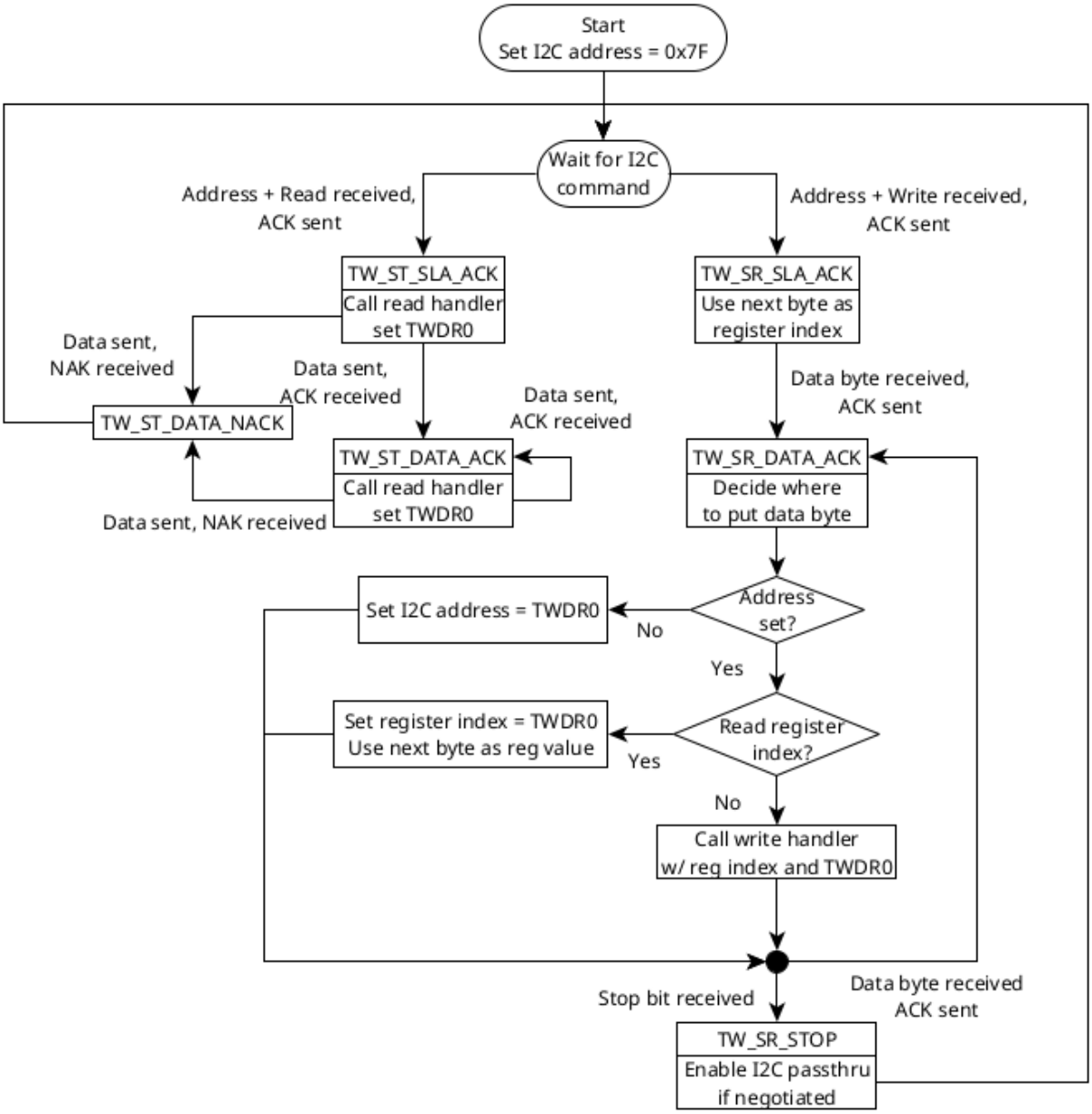


Figure 10: Reel module I2C state machine

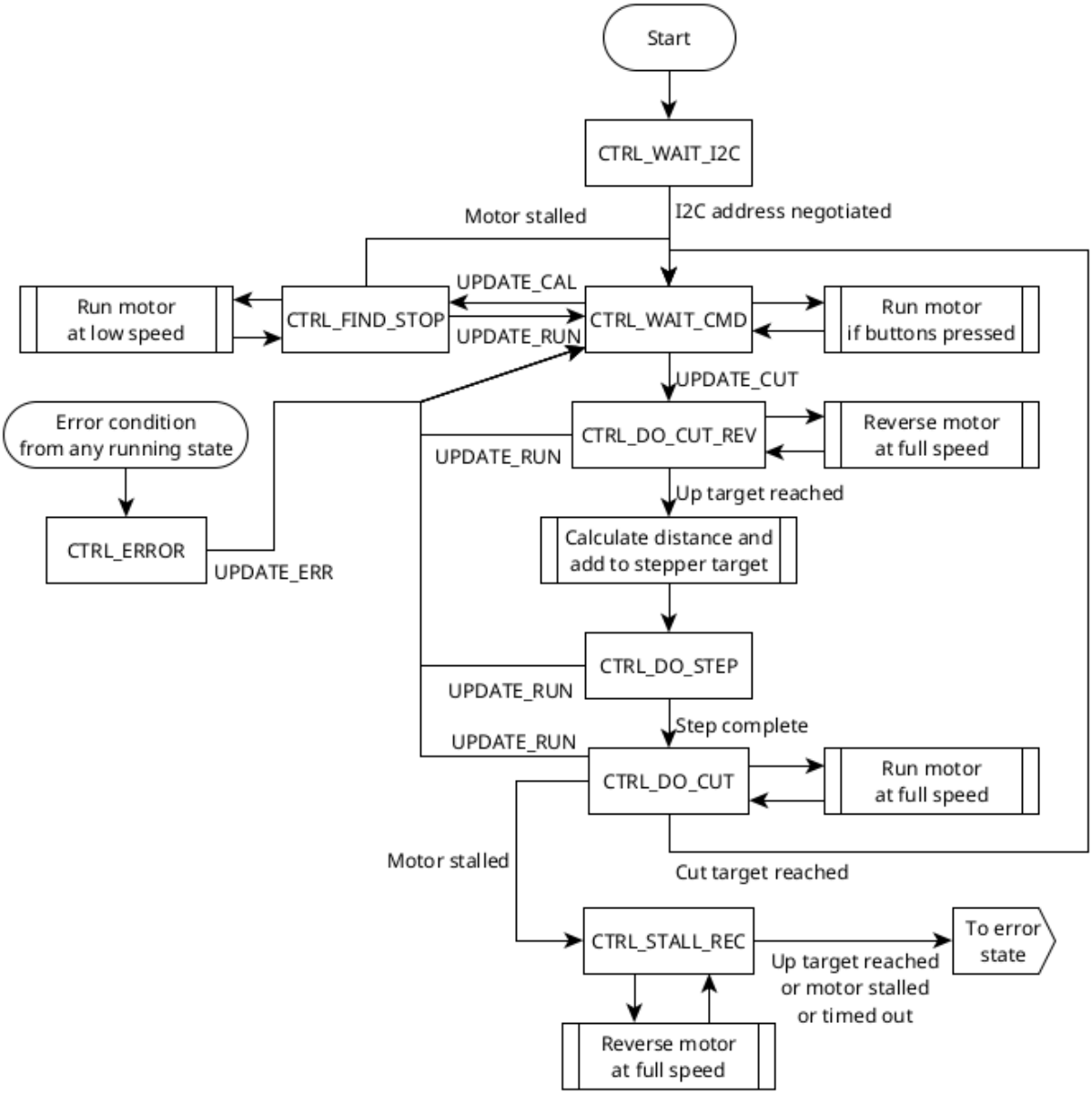


Figure 11: Reel module control state machine

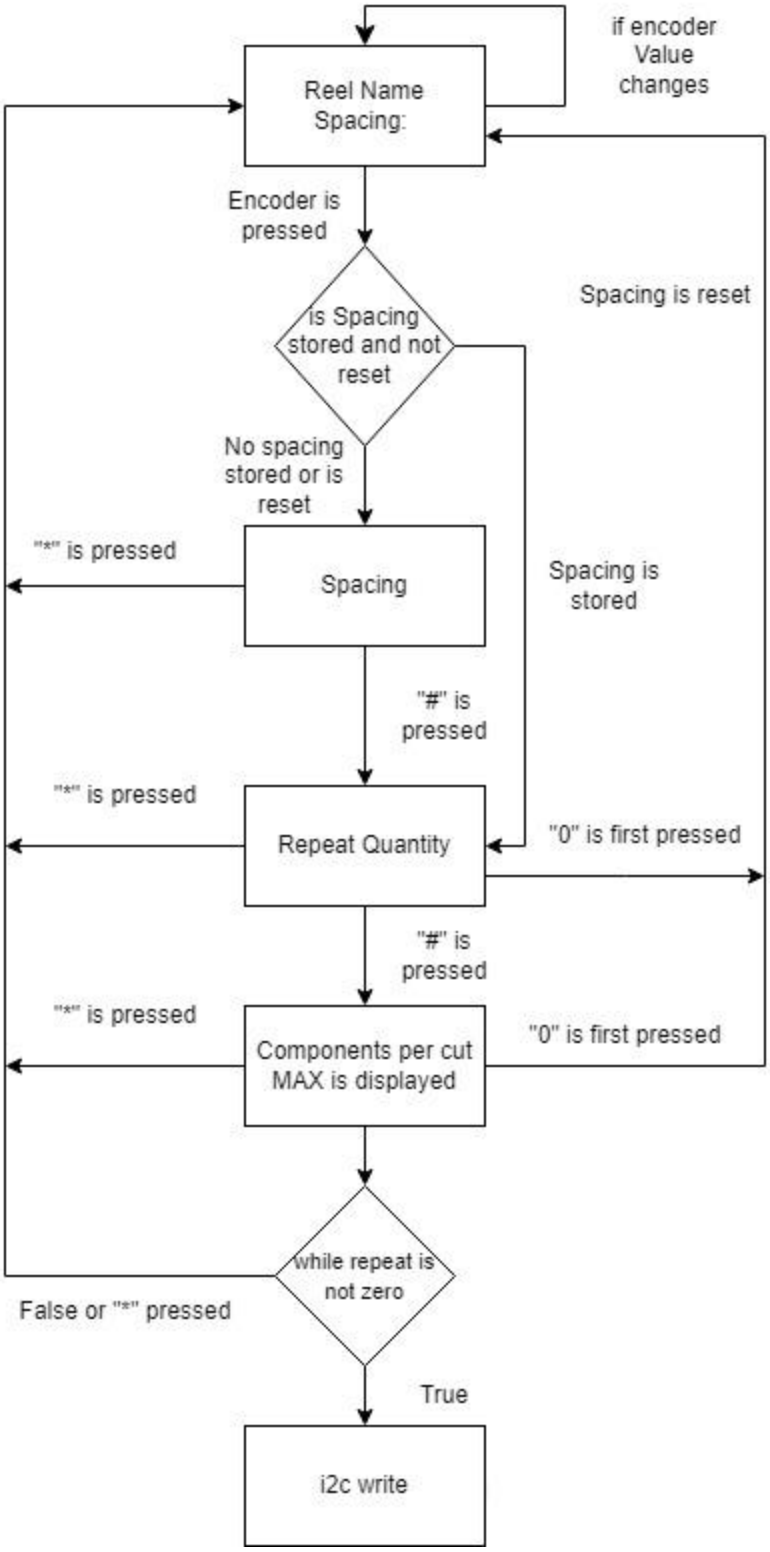


Figure 12: Base module flowchart

Appendix G I2C communication waveform

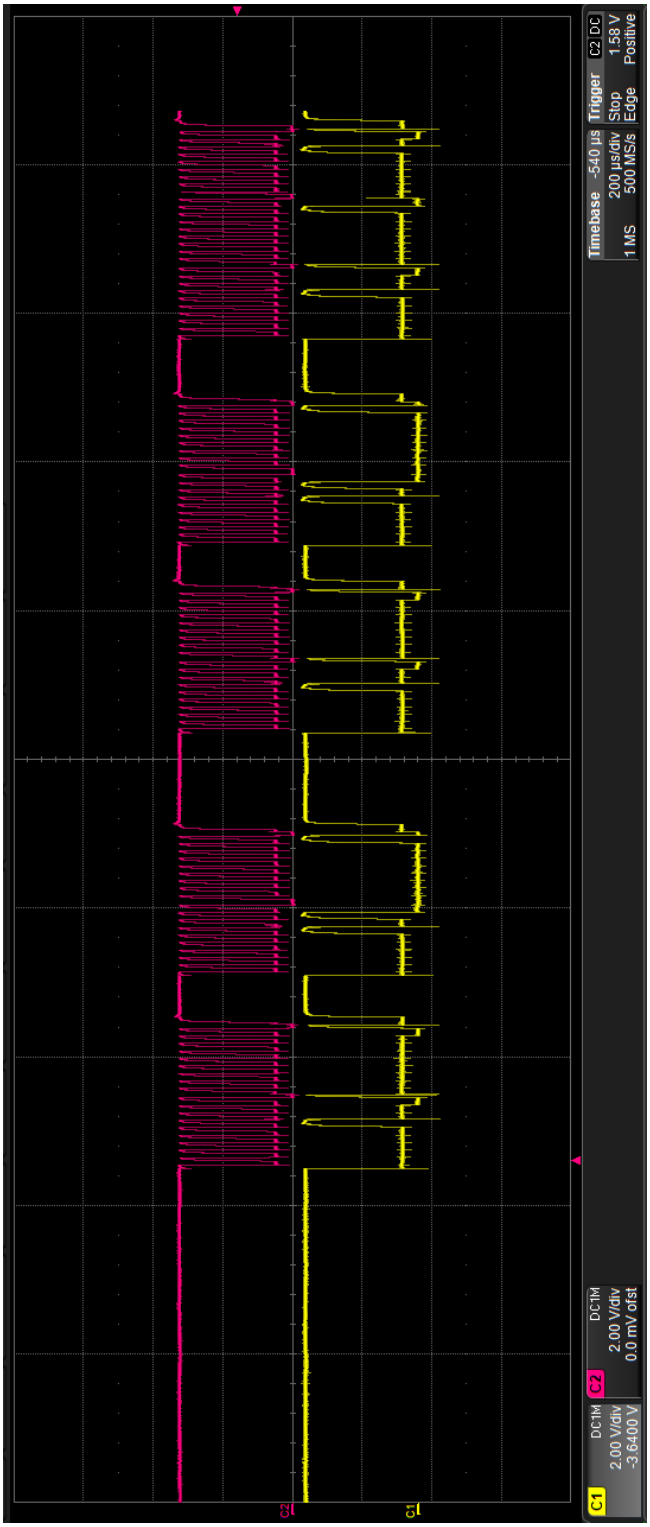


Figure 13: I2C communication when starting a cut operation