ECE 445

Senior Design Laboratory

Final Report

Human-Robot Interaction for Object Grasping with Visual Reality and Robotic Arms

---

Team #42

Ziming Yan (zimingy3@illinois.edu)

Jiayu Zhou (jiayu9@illinois.edu)

Yuchen Yang (yucheny8@illinois.edu)

Jingxing Hu (hu80@illinois.edu)

TA: Tielong Cai & Tianci Tang

May 19, 2025

Abstract

This document provides an outline and LaTeX template for report formatting in Senior Design. This document does not teach you what to include, or how to use LaTeX. Assumes a workable level of LaTeX proficiency.

Contents

1 Introduction

A general section looks like this. There is usually a blurb introducing the top-level section here.

2 System Description

Generally, we developed the system which we used to achieve our senior design project, and we drew the block diagram to delineate the subsystems and individual tasks, which is shown as Figure 1.The system is composed of four main modules: Control Module, Actuator Module, User Interaction Module, and Power Module, collaboratively enabling a VR-guided robotic claw system with closed-loop force control and CAN communication.

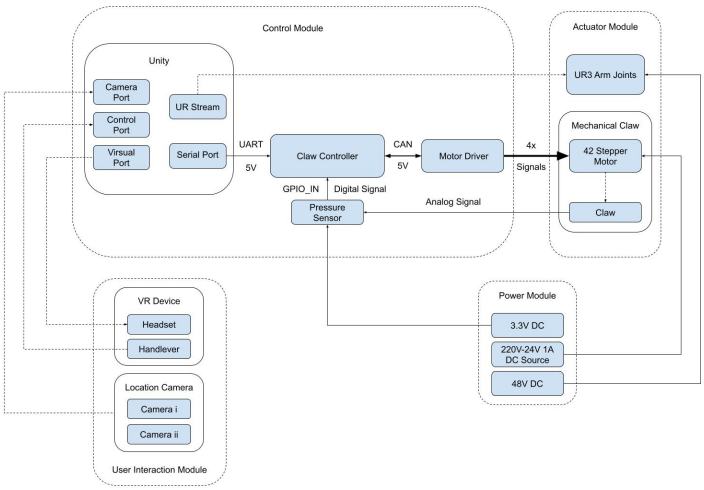

Figure 1: Block Diagram

Control Module

At the core of the system lies the Control Module, where Unity software runs and interfaces with the physical system via multiple communication ports. The UR Stream and Serial Port in Unity handle command and signal exchange through UART with the Claw Controller (based on an STM32 microcontroller). The controller processes incoming signals and sends control commands via CAN bus to the Motor Driver, which in turn drives the 42 Stepper Motor. Additionally, a Pressure Sensor provides real-time feedback to the Claw Controller via analog and GPIO inputs, forming a basic closed-loop control system for grasping force regulation.

Actuator Module

The Actuator Module consists of a 42 Stepper Motor and a 3D-printed claw. The motor is controlled by the Motor Driver, which receives commands from the Claw Controller via CAN bus. The claw's design allows for precise control of the grasping force, enabling the system to adapt to various objects and scenarios.

User Interaction Module

The User Interaction Module is designed to facilitate user engagement with the system. It includes a VR headset and controllers, which provide an immersive experience for the user. The Unity software processes input from the VR controllers and translates it into commands for the Claw Controller, allowing users to interact with virtual objects in a realistic manner.

Power Module

The Power Module supplies the necessary power to the entire system. It ensures that all components, including the Control Module, Actuator Module, and User Interaction Module, receive stable and sufficient power for optimal performance.

Initially, we planned to use a STM32F103 microcontroller and SG-90 motor to control the claw. However, we found the accuracy was low and reaction time was far from satisfactory. Therefore, we changed to use more powerful STM32F407 with M4 core and 42 stepper motor. In addition, we wanted to use unity to control arm through ROS system at first, but we finally found a online source which allowed us to control arm derectly by unity hub, which reduces the complexity and optimizes the performance of the system.

3 Design

3.1 Control Module

3.1.1 Design Procedure

The Control Module is the brain of the system, where all the processing and decision-making occurs. It is responsible for receiving input from the user, processing that input, and sending commands to the Actuator Module. The core of the control module is the Unity, it received the control signal from VR handler, like pressing the buttons to move the UR3 arm and the grip process of claw, then, it translate command signal to the parameters of motor and sent it to subcontrollers.

In addition to Unity, we added a STM32 board to control the stepper motor to finish the closed-loop control of claw, inorder to ensure the stability of grisping. After confirming the main controllers in module, I need to think about how to connect them with each other or other modules. Through refering to the ST company manual of STM407ZG chip, I found that it supports the USART communication protocol, which means it can receive the command from PC Unity with UART line. Moreover, it also has pins with CAN_RX and CAN_TX, which also supports me to control the stepper motor with CAN protocol [1].

After designing the flow path, I wanted to add a sensor on claw to monitor the grip power, so I added the FSR402 pressure sensor and connect it to STM board. Overall, I got the main data flow idea.

3.1.2 Design Details

Aftering determining the main data flow theorically, I design to set up hardware circuit, therefore, I drew the design graph, as shown in the Figure 2. This introduces the main connection of control module.

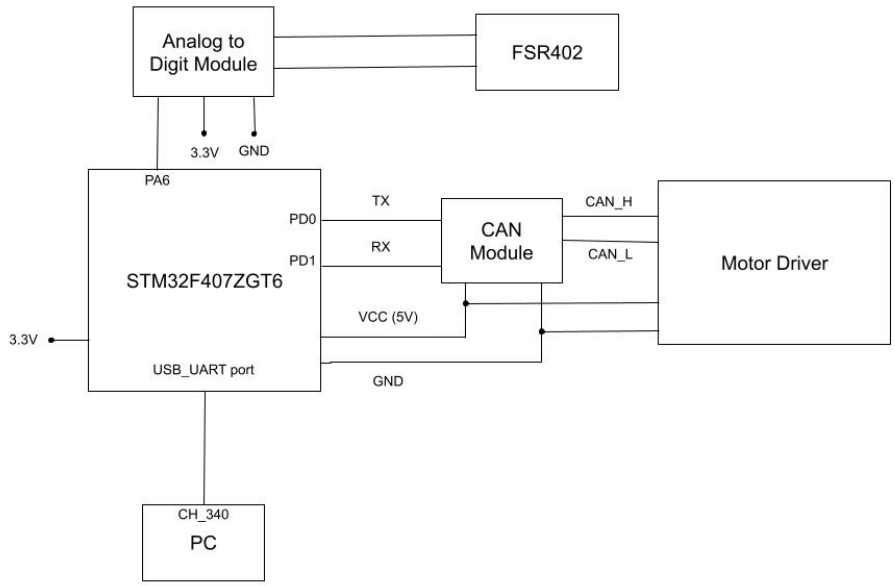

Figure 2: Design Circuit

CAN Communication:

Through referring to the motor driver board X42_V1.3 instruction manual, it allows us to use intergrated CAN command to control the stepper [2]. However, the STM32F407ZG only have RX and TX pins, the transmition of CAN signal should be CAN_H and CAN_L two logical voltage [3], which means I need to design a CAN receive and send message module. After searching online, I decided to use the TJA1050 chip to convert the digital CAN signals from the microcontroller to the differential voltage signals required by the CAN bus and vice versa, as shown in the figure 3. The TJA1050 is a CAN transceiver IC that acts as a bridge between the CAN controller (STM32) and the physical CAN bus. It converts single-ended digital signals from the microcontroller into differential signals suitable for transmission over the CAN network, ensuring noise immunity and reliable data communication. At the receiving end, it converts the differential bus signals back into digital signals that can be interpreted by the microcontroller. And to ensure that the bus signals are stable, two 120 Ω resistors should be connected in parallel at both ends of the CAN bus [3].

Pressure Sensor:

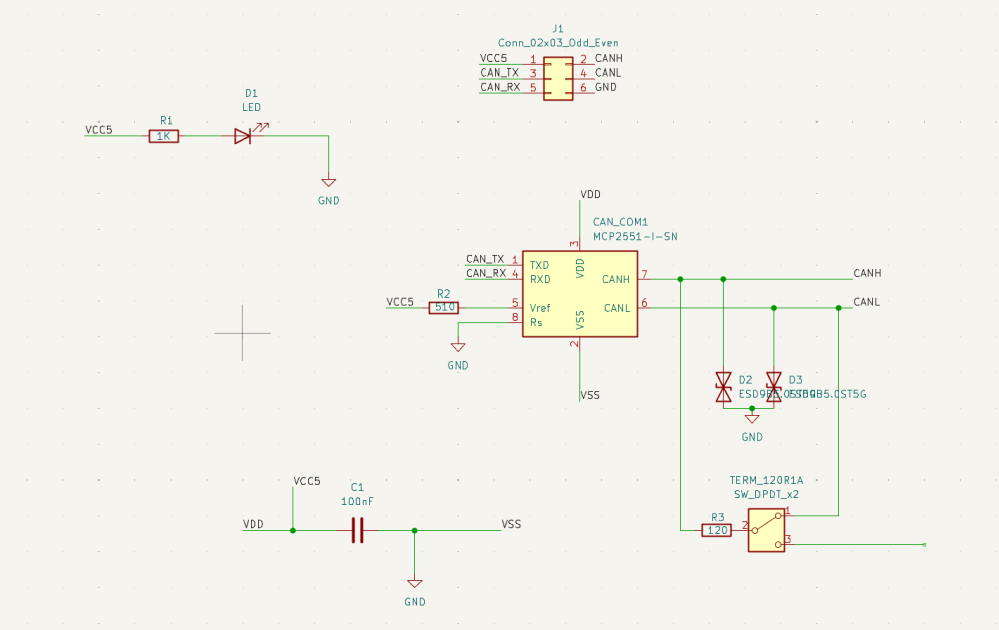

Figure 3: CAN Module

The FSR402 pressure sensor is a force-sensitive resistor that changes its resistance based on the amount of force applied to it. It is commonly used in applications where measuring pressure or force is required. The FSR402 sensor works by changing its resistance when a force is applied to its surface. The more force applied, the lower the resistance. By measuring the voltage across the sensor, we can determine the amount of force being applied. I develop two approaches to read the sensor data:

1. Using the default ADC (Analog-to-Digital) port of STM32F407ZG, which is a 12-bit ADC. The ADC converts the analog voltage signal from the FSR402 into a digital value that can be processed by the microcontroller. The ADC can be configured to read the voltage across the sensor and convert it into a corresponding digital value.

2. Using the LM393 chip to compare the voltage on pressure sensor with referrence voltage (Variable Resistance). When the voltage on FSR402 is larger than referrence, the output will be high voltage. When the pressure is larger than desired, it will change voltage, which achieves simple Analog to Digita transform.

For convinient, I design a circuit intergrated on a small PCB 6 and soldering it. The circuit is shown as Figure 4. In circuit, I connect the FSR402 with 10kΩ resistor. AO port will output the voltage on FSR402 under 3.3V input, using for the 1*st* method. DO port with output the compare consequence of FSR voltage and 10k Ω variable resistance voltage, using

as GPIO input for 2ed method. Finally, I decide to use the 2^{ed} method and the reson will be explained in Verification and Test part.

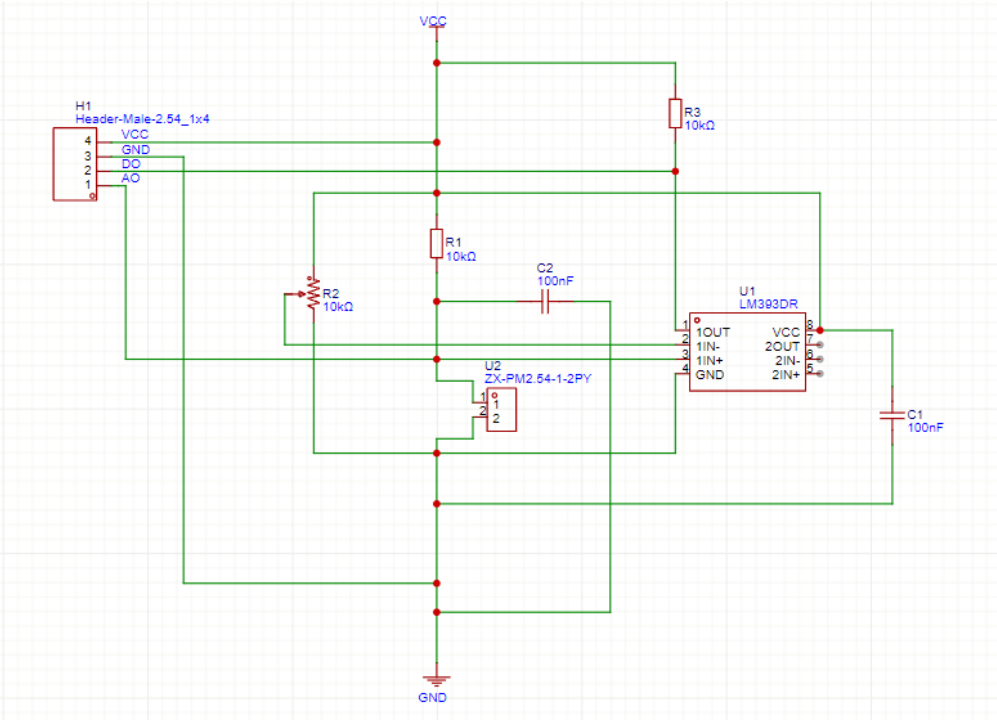

Figure 4: FSR402 Circuit

Coding Explain:

Coding in control module is a huge and most complex part, I will introduce it with two aspects: STM32 and Unity.

In STM32, I develop the codes based on the HAL library which is configured by STM32CubeMX and implemented them in KDM5 (Keil Development Environment).

- can.c: The HAL (Hardware Abstraction Layer) library provides a high-level interface for configuring and using the CAN peripheral. What I did is initializing the handler and change the can_send() function to send standard CAN message to tell the 42 stepper to run its position circle, like move to which position, how fast, when stop, and so on. In addition, I also need to write the receive function to receive the position feedback of motor to control its stop time. For example, if I send 01 36 6B to request the position of motor, it will response 01 36 01 00 01 00 00 6B, which means the postion is 0x00010000 (16), direction is positive. The send function runs smoothly, but the

feedback function costs me plenty of time to debug.

- usart.c: The UART (Universal Asynchronous Receiver-Transmitter) peripheral is used for serial communication with the Unity application. The HAL library provides functions to initialize the UART peripheral and rewrite the HAL_UART_RxCpltCallback() function to achieve the communication between STM32 and Unity.

- sensor.c: AsI mentioned before, I choose to use the DO as output of the pressure sensor. To achieve this, I just initialized the GPIN_IN and open the clock of corresponding pin RCC to receive the digital signal from DO.

- main.c: The main function initializes the system clock, can port, uart port, sensor, and so on. After initialization, I develop the logic of gripping and releasing an infinite loop. In the loop, it continuously checks for incoming data from Unity via UART, the Unity will send "S" for start, "R" for release and "H" for stop. After receiving the start, CAN message will be send to tell motor to begin until the reflected pressure sensor data shows the claw have griped the item. When pressure is over the threshold (input low voltage change), STM32 will send CAN message to stop motor. While Unity sending release signal, the motor will return according to the position feedback minitoring. Then, the STM32 will wait for next command from Unity. The following Moore FSM(Figure 5) shows the overall logics.

Unity:

- VR Handler:

- Motor Control:

- Serial Port: The serial port communication in Unity is implemented using the System.IO.Ports namespace. This allows Unity to open a serial port, send data to the STM32, and receive data back. The serial port settings, such as baud rate and parity, are configured to match those of the STM32. For this project, I use the UART line to connect them and my baud rate was set in STM to 115200, therefore, the serial port in unity setting is corresponding to baud rate and use COM4 CH340 PC port.

3.1.3 Verification and Testing

After going through all the procedures above, I successfully set up the control module with hardware as shown in Figure . However, I encountered plenty of problems during the testing

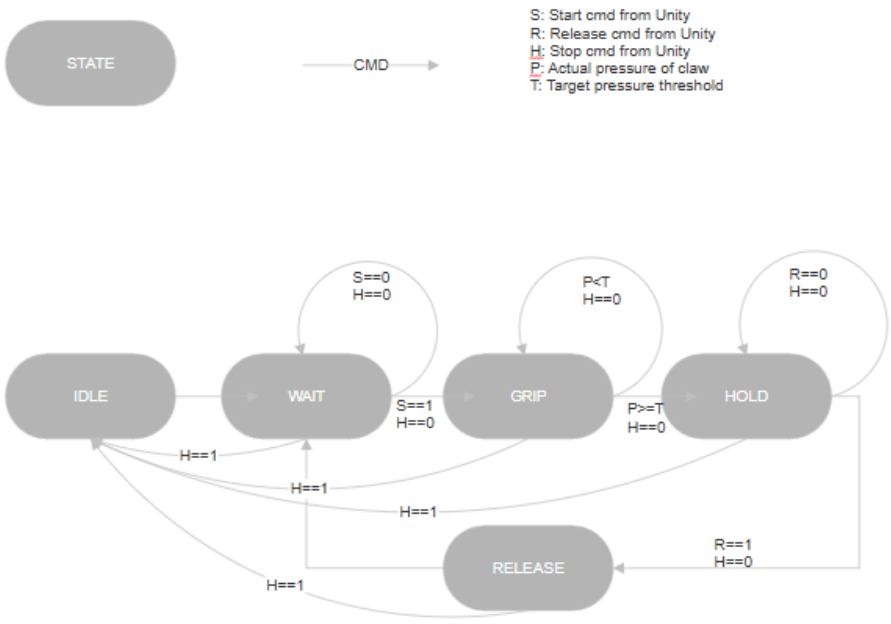

Figure 5: Motor FSM

and debugging process.

- PCB Testing:Before the circuit above, I have designed a failed circuit, which is shown as Figure 7. However, I found errors after the soldering. While applying 3.3V on VCC port, the voltage on AO didn't change with the pressure on FSR402. Therefoe, I used the multimeter to check the circuit. There are errors, like the voltage on AO before FSR insertion should be 0.3V, but I measured 3.17V. In addition, resistance between VCC and DO should be 909.1Ω, but actually wired $9.792k\Omega$. Finally, I found there is open circuit in PCB design. Aftering designing new circuit, I succeed. The following Table 1 shows my measure history.

- Baud Rate Mismatch: While sending the CAN message (position circuit control) to the stepper motor, the motor cannot run correctly as desired. After checking the CAN related codes, I didn't find the bug. Then, I use oscillograph to check the CAN_H port. Normally, if the CAN message send correctly, the CAN_H will have voltage changed, the low voltage should be about 0 and high should be 5V. However, I found that the CAN_H wave always 4.176mV, which means the CAN initialization failed. So I check the default CAN HAL init, the baud rate is set to 1M, which is mismatched

from my motor's CAN receive rate. My designed baud rate should be 50000. The CAN bus baud rate on STM32 is determined by configuring the CAN timing registers, primarily the Prescaler, BS1, BS2, and SJW parameters, and the rate computation should use the equation:

$$\text{Baud Rate} = \frac{\text{CAN Clock Frequency}}{\text{Prescaler} \times (1 + \text{BS1} + \text{BS2})} \tag{1}$$

For my project, system clock is 168Mhz, the CLK OF CAN is APB1 CLK $= \frac{\text{System Clock}}{4} = 42Mhz$ I changed the Prescaler to 14, BS1 to 4, BS2 to 1. Therefore my designed baud rate should be $\frac{42Mhz}{14\times(1+4+1)} = 50000$, which matched the default rate. After changing the baud rate, I successfully finished the communication with motor.

- STM32 ADC Delay:After comparing my two methods prepared to monitor the pressure by FSR, I found that the DO port responses the pressuree threshold faster compared with AO after ADC. I thought the DO will enable my claw more sensitive, but actually, the ADC delay is about 1ms, which is not acceptable for my project. Therefore, I choose to use the DO port as input.

- CAN Feedback:At first, I didn't think the position feedback of motor is necessary. However, I found that there are errors in phase of motor D axle, various from $0.9°$ to $1.2°$. With accumulation, the origin position of claw will be changed, causing the claw cannot close completely. Therefore, I added the CAN feedback function to monitor the position of axle, ensuring the motor run back until the position is 0. Through testing, the average position error decreased, within $0.08°$.

4 Cost & Schedule

4.1 Cost Analysis

Description	Quantity	Price
STM32F407ZGT6 development board	1	¥174.08
Emm42_V5.0 motor control board	1	¥54.99
42mm Stepper motor	1	¥22.99
220V to 24V power supply	1	¥19.80
FSR 402 Pressure Sensor	2	¥20.00
PCB Circuit Board	10	¥43.20
3362P-1-501	5	¥0.48
LM393DR	10	¥0.47
PZ254V-11-04P	20	¥0.29
ZX-PM2.54-1-2PY	5	¥0.27
CGA0603X7R104K500JT	100	¥0.10
0603WAF1001T5E	100	¥0.07
0603WAF1002T5E	100	¥0.07
3362P-1-103	5	¥4.55
Component Courier Fee	2	¥22.00
Total		¥363.37

5 Conclusion

5.1 Summary of Completion

5.2 Further Improvement

5.3 IEEE Ethical Standards Compliance

5.4 Broader Impacts

References

[1] STMicroelectronics, STM32F407ZG - High-performance foundation line, Arm Cortex-M4 core with DSP and FPU, 1 Mbyte of Flash memory, 168 MHz CPU, ART Accelerator, Ethernet, FSMC, https://www.st.com/en/microcontrollers-microprocessors/stm32f407zg.html, Accessed: 2025-05-17.

[2] 张大头, 步进闭环驱动说明书 rev1.3, https://blog.csdn.net/zhangdatou666/article/details/132644047, Accessed: 2025-05-18, 2023.

[3] 艾格北峰, Can 总线协议, https://blog.csdn.net/qq_35057766/article/details/135580884, Accessed: 2025-05-18, 2024.

Appendix A Example Appendix

A.1 Codes

A.2 Figures

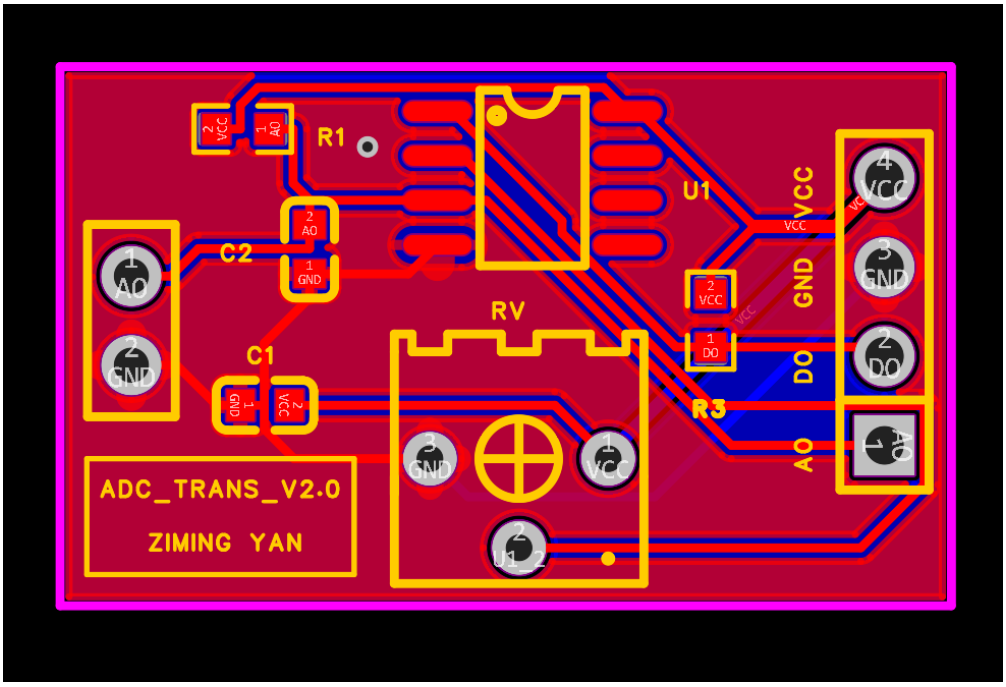

Figure 6: FSR402 PCB

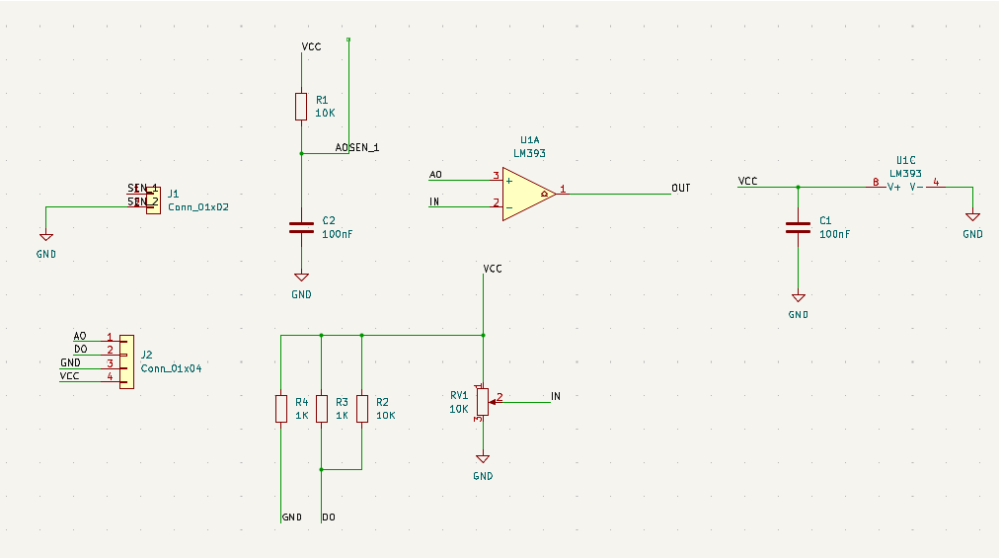

Figure 7: FSR402 Failed Circuit

PCB Type	Test Item	Expected Result	Actual Result
Failed	Voltage on AO	$\frac{1k\Omega}{11k\Omega} \times 3.3V = 0.3V$	3.17V
Failed	R between sensor port 2 and VCC	$10k\Omega$	$9.792k\Omega$
Failed	R between VCC and DO	$\frac{1k\Omega \times 10k\Omega}{1k\Omega + 10k\Omega} \approx 909.1\Omega$	$6.147k\Omega$
Success	Voltage on AO	3.3V	3.197V
Success	R between sensor port 1 and VCC	$10k\Omega$	$9.807k\Omega$
Success	R between VCC and DO	$10k\Omega$	$9.076k\Omega$

Table 1: PCB Testing