ABSTRACT

Title of Thesis: DEVELOPMENT AND INTEGRATION OF TACTILE SENSING SYSTEM

Rishabh Agarwal
Master of Science, 2018

Thesis directed by: Professor Sarah Bergbreiter
Department of Mechanical Engineering

To grasp and manipulate complex objects, robots require information about the interaction between the end effector and the object. This work describes the integration of a low-cost 3-axis tactile sensing system into two different robotic systems and the measurement of some of these complex interactions. The sensor itself is small, lightweight, and compliant so that it can be integrated within a variety of end effectors and locations on those end effectors (e.g. wrapped around a finger). To improve usability and data collection, a custom interface board and ROS (Robot Operating System) package were developed to read the sensor data and interface with the robots and grippers. Sensor data has been collected from four different tasks: 1. pick and place of non-conductive and conductive objects, 2. wrist-based manipulation, 3. peeling tape, and 4. human interaction with a grasped object. In the last task, a closed loop controller is used to adjust the grip force on the grasped object while the human interacts with it.

DEVELOPMENT AND INTEGRATION OF TACTILE SENSING SYSTEM

by

Rishabh Agarwal

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2018

Advisory Committee:
Associate Professor Sarah Bergbreiter, Advisor
Professor Miao Yu
Professor Elisabeth Smela

Acknowledgments

First and foremost I would like to thank Prof. Sarah Bergbreiter for all her support and guidance over the past two years. I would like to thank all the members of the Mircorobotics lab for their advice and support throughout this project. I would specially like to thank Mr. Ryan St. Pierre and Mr. Ivan Penskiy for guidance and feedback. I would also like to thank Mr. Hee-Sup Shin for helping me getting started with the project and guiding me through the process of circuit designing.

I would like to acknowledge the support of Robotics Realization Lab of the University of Maryland to provide access to the range of robotics equipment.

I would like to thank Dr. John MacCarthy for his continuous guidance over the course of my masters program.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Robotic manipulators have been well-established in the industry for decades, and are exceptionally good at performing complex grasping and manipulation tasks in a carefully engineered environment. In production lines, manipulators are performing tasks ranging from pick & place to automobile assembly with great accuracy. It is only possible due to the closely controlled environment in which these manipulators operate. Any slight change in the environment and we see these manipulators tends to fall apart.

With the introduction of robots to more dynamic environments such as living room or kitchen, manipulators will have to perform complex human-like maneuvers such as grasping everyday objects, passing on drinks from the refrigerator, etc. The manipulators will be operating in constrained and unknown workspaces with limited information on objects. To overcome this lack of information, significant work has been done on integrating force sensors and remote sensors such as cameras, sonars, and laser scanners with manipulators [1, 2]. The remote sensors tend to consume high processing power to provide an estimate of object's pose, and they lack in providing any confirmation on if the object has been gripped or not. Moreover, they are unable to provide some essential object properties such as weight of the

object, slippage etc. Most of the force sensors integrated along with manipulators provide an accurate measure of the normal component of the grasping force, but in general require high forces to be applied on the objects and are incapable of measuring the shear forces.

Humans are much more adaptable to their environment. We can see humans grasping and manipulating new objects in an completely unknown environment. Bin picking which is one of the most common tasks in industries could be performed by humans without even looking at the bin whereas robots tend to fail even with the sense of vision and normal force.

Humans heavily rely on the shear data to perform dexterous manipulation tasks. These shear forces are extremely crucial for grasping and manipulating objects with different attributes. To utilize tactile sensing, humans contact the object first and once in contact with the object, the resulting reaction forces are used to gauge and improve gasping; measured shear forces are influenced by the weight and shape of the object, especially if tilt is involved [3, 4]. More than the global shape of the object, it is the local shape (shape of the object around the contact area) that affects the reaction force and the grasp. The mechanoreceptors in the skin of the human finger tips are highly sensitive to forces tangential to the skin and help provide a clear indication of the grasp [5]. The work in [3] shows how tactile perception from the mechanoreceptors allow humans to perform complex maneuvers.

Due to the potential advantages for grasping and manipulation, researchers have developed numerous sensors capable of detecting these tactile forces in works like [6, 7, 8]. Many of these sensors are small, flexible, and relatively low cost. In-

dustrial sensors like the 3D40 by ME-Meßsysteme Company [9], 3 axis load cells by Interface Company [10], the 3-axis Force Sensor by Tec Gihan Co. [11], BioTac by SynTouch, Inc [12], and OMD by Optoforce [13] provide force data in 3 axes. Although tactile sensors is not something new, the key challenges for proper utilization of the sensors lies with their integration. While researchers provide a number of unique and different tactile sensing technologies, most of them are still not mature enough to be employed on robotic systems. Industry on the other hand does have solutions implemented but they typically cost thousands of dollars and are relatively large and bulky. Along with that most of the industrial solutions require significant changes in the exiting setup thus restricting the environment once again.

With both industry and academia working towards grasping and maneuvering applications, a shear sensing system that can be easily integrated onto existing manipulators is needed now more than ever. A low cost and compatible shear sensing system would not only benefit existing research work but will also lead to new developments in the field of sensitive manipulation [14]. The inclusion of shear sensing will allow making decisions about the manipulator motion and applied grasping forces to avoid slippage or object damage.

The shear sensing system that we are introducing includes a microcontroller, custom PCB, and 3 axis force sensor with circumscribed area of 2.9 mm x 1.6 mm and the total sensor footprint including wiring and connector is 15 mm x 25 mm. The sensor itself is made completely out of silicone elastomer. Its small footprint and ability to withstand high strains makes it easy to integrate with variety of grippers or finger shapes.

In this thesis, we showcase the development and integration of shear sensing system with existing robotics platforms. The work builds up upon the previous works [15, 16] and demonstrates systems integration and applications of the sensors.

1.2 Challenges

For the development of tactile systems it is essential to understand the challenges we will be facing with their integration and usage along with robotic systems. A lot of good tactile sensors have been developed in the past but most of them tend to lack the focus on their application in real world. In past, a lot of focus have been on the performance parameters such as sensitivity of the sensors, dynamic resolution, and noise levels. Although all these parameters are essential this would is more focused towards overcoming the challenges of the sensors application in real world scenarios. The following characteristics were considered essential. The importance of each will be covered in consecutive chapters:

1. **Modularity:** The sensors used in the work were easy to design and fabricate, allowing fast (within 8 hrs) fabrication of sensors with different sensitivity and measurement range. A modular system with an ability to simply replace the sensors will facilitate shear force measurement for a range of applications.

2. **Compact Design:** Minimalistic design will allow easy mounting and will help in avoiding any modification to the existing setup.

3. **Object material and geometry independence:** Responsive to different

materials and shapes of the objects will allow measurement of interaction with range of objects.

4. **Software compatibility:** Compatible with existing software framework to avoid additional step of establishing communication with existing system.

5. **External environment independence:** Responsive in different environments and for range of tasks to ensure operation in unknown environments.

1.3 Our approach

This work talks about design, development, and integration of the shear sensing system. The sensing system includes the tactile sensor, custom interface board, microcontroller, connectors, main operating system (Ubuntu, Mac or Windows) and communication protocols used between them. The sensor used is an all elastomer based capacitive sensor previously developed in the work [15]. For the sake of completion, this works talks about the sensor's working principle and design, but apart from some slight design changes, the key sensing principle is the same as described in [15, 16].

1.3.1 Hardware Design

The sensor hardware was based on an Arduino and custom PCB designed around an Analog Devices AD7746 Capacitance to Digital Converter (CDC) chip. The connections between the circuit and the sensor were kept minimalistic and modular by using FCI clinchers to allow rapid integration of different sensor configurations

(discussed in detail in 2). I2C communication protocol was used to communicate between the AD7746 chip and Arduino. The main system and Arduino communicated using rossserial protocol via USB.

1.3.2 Software Design

For the software it was essential to ensure the compatibility of the sensors with existing robotic systems. ROS is one of the most widely used software frameworks for robots, and was therefore chosen for the sensors software architecture. Writing a custom ROS package ensures that the sensor will be functional with any ROS version, allowing communication between the sensor and both old and new robot systems.

1.4 Contribution

The primary contribution of the current work is the integration of the sensors [15] into a complete sensing system (hardware and software). A custom printed circuit board (PCB) and Arduino provide the hardware interface to the robot, and a ROS package was written as a software interface to the sensor. In addition, the sensors are applied to two different robot manipulators- Robotiqs' 2-finger adaptive gripper [17] and the RightHand Labs Reflex Takktile gripper [18] and data is collected from several experiments to demonstrate how sensor measurements can be used in grasping and force control tasks.

1.5 Road Map

This thesis is organized into five chapters, and the thesis presents an expanded version of a recently submitted conference paper [19]. Chapter 2 presents the sensor working principle, fabrication method and peripheral hardware design and development. The software development and the integration of sensor system with robotic platforms are discussed in chapter 3. In chapter 4, the system performance and experimental results are presented in detail.

Finally, chapter 5 concludes the work performed and presents potential next steps for the project.

Chapter 2

Sensor Design and Hardware Architecture

2.1 Introduction

This chapter presents the sensor's design, working principle, and fabrication technique. The design and working principle are based on the works [15, 16] and the fabrication method is derived from the work [20]. The primary focus of this chapter is to present the work done in developing the hardware peripherals for the sensor.

As mentioned before, the sensor system includes the custom PCB made around the AD7746 chip to measure capacitance, the Arduino microcontroller to configure the PCB and FCI clinchers for the connections with the sensor. This chapter discusses the capacitance measuring principle used for sensor feedback and how that led to the current PCB design and the physical integration of the sensor with the robotic systems.

2.2 Sensor Design and Working Principle

The sensor design is derived from the work of [15, 16] with some minor differences. To measure forces in normal and shear directions, the sensor utilizes a conductive elastomer pillar and pad features as shown in Figs. 2.1 and 2.2.

When an external force is applied to the sensor, the central pillar and sur-

rounding material deform leading to a change in the gaps between the pillar and 3 pads. This deformation results in a change in capacitance as measured between the pillar and each of the pads, correlating to the applied forces along the 3 axes. Under a normal load the gaps between the pillar and pads change together, leading to a common capacitance change in all the three capacitances. Under shear loading the pillar deforms towards one pad and away from the other, leading to a capacitance differential.

As shown in the Fig. 2.1 the sensing area is covered with a layer of PDMS but it is important to note that the upper PDMS layer is optional. The working principle remains the same whether the PDMS layer is present or not. The following parameters are affected by the absence of the upper layer.

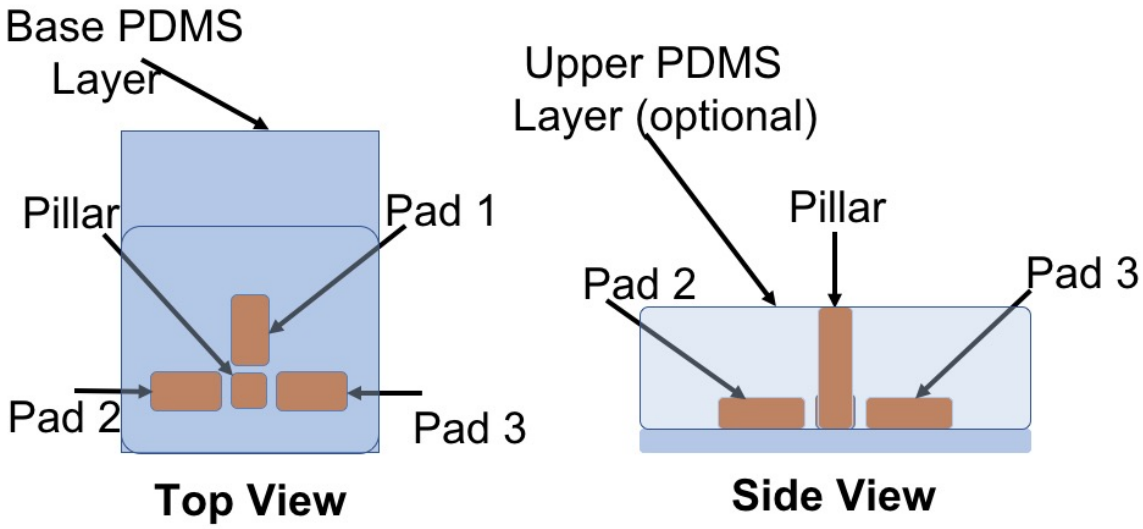


Figure 2.1: Sensor model schematic

1. **Sensing Area:** The upper PDMS layer allows sensing of forces that are not directly applied to the pillar. The absence of this layer restricts the force

measurement to the forces on the pillar.

2. **Sensitivity:** The upper PDMS layer creates some resistance for the bending of the PDMS layer, whereas the absence leads to a much higher magnitude of bending, hence, a more sensitive response from small forces.

3. **Noise:** The absence of PDMS layer exposes the sensor to the external environment leading to higher level of noise.

4. **Material Dependence:** Since the conductive features will be exposed to the environment in the absence of PDMS layer, the sensors will not be able to function with conductive objects.

The difference in the sensitivity between the PDMS enclosed and non-enclosed sensors allowed utilization of the same sensor system for different tasks such as pick & place and tape removal. Chapter 4 talks about the different experiments in detail.

Parameters like the gap between the pillar and the pads, height of the pillar, and height of the pads play an important role for determining the sensitivity. The study in [16] shows how the parameters affect the sensor's performance. The fabrication process, as discussed in section 2.3, showcases the ease of manufacturing sensors with different configurations, which allowed us to choose the optimal sensor configuration based on the task's requirement.

The key difference between the previous work and this work is the way these sensors are used. In past, the absolute capacitances between the pillar and each of the 3 pads were measured, whereas in the current work we measure the difference

in capacitance between pads 2 and 3. This change allowed the use of single AD7746 chip to measure the output from the sensor, making the circuit simple and reducing the overall processing requirement.

This work focuses on utilizing the capacitance difference due to shear loading to measure the tangential reaction force. However, normal and shear forces can be measured with this design as previously demonstrated in [15, 16]. Future sensors can lay out the three pads so that tangential reaction forces in both directions can be measured. Chapter 5 talks more about the force measurement in both directions.

2.3 Sensor Fabrication

A benefit of this sensor architecture is the ease of design and manufacturing; the manufacturing process is similar to the one described in [20]. As discussed in section 2.2, based on the task's requirement the gap and the height of the features are calculated. Once the design is finalized, it is then modeled in Camotics 1.0.0 to generate NC code for milling the features. A reusable mold is milled (Roland MDX-540SA) from an acrylic sheet. The complete milling process takes less than an hour for a batch of 3 sensors. Conductive and non-conductive elastomers are then molded to create the sensor—conductive features are used for the pillars, pads, and wires. Sylgard 184 PDMS (Dow Corning) was chosen as the elastomer for the sensor due its low elastic modulus, high failure strain, and conductive behavior when doped with conductive filler particles. Conductive PDMS (cPDMS) was formed by mixing 10 wt.% carbon nanotubes (CheapTubes) with PDMS. Fig. 2.2 shows the

manufactured sensor after the elastomers are cured and peeled from the mold.

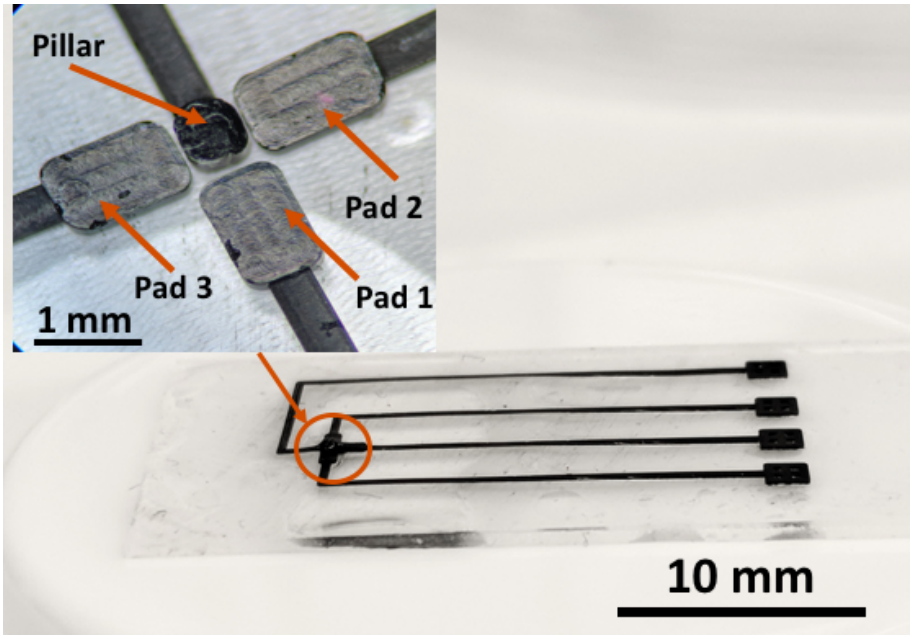


Figure 2.2: Manufactured Sensor

Sensors with several different heights and gaps were manufactured over the period of last year, and based on the experiments and expected forces for the tasks, such as pick & place, wrist manipulation and human interaction with objects, a design with a 1.4 mm tall pillar and 400 µm tall pads was chosen with a gap of 35 µm between the pillar and the pads. This design was used in all the experiments discussed in Chapter 4, but it should be noted that the ease of re-designing and manufacturing these sensors make them adaptable to experiments or tasks requiring different ranges and sensitivities.

2.4 Hardware Integration

The hardware side of the sensors included three main components—custom PCB, microcontroller, and connectors. Ease of integration with the robotic system and communication between the components was the primary objective towards designing the hardware system. The hardware architecture, along with the different communication protocols, is shown in Fig. 2.3.

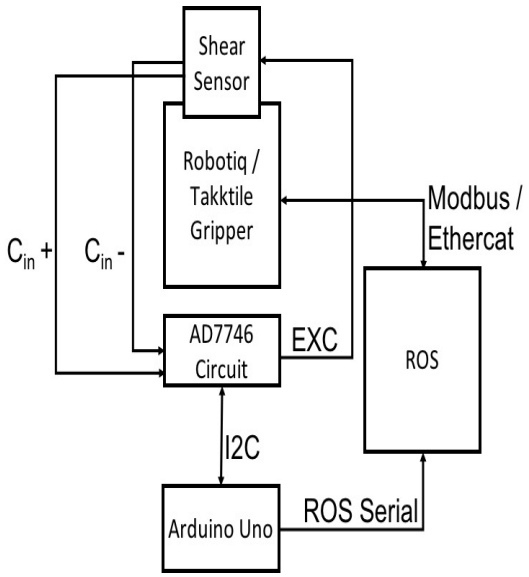


Figure 2.3: Hardware schematic, along with the respective communication protocols

2.4.1 Custom PCB

The custom PCB was designed around an Analog Devices AD7746 Capacitance to Digital Converter (CDC) chip. The PCB designed for this work was focused towards achieving minimal footprint while incorporating multiple layouts to test different configurations of the AD7746 chip.

In the previous work, the sensor output was directly measured from a AD7746

Evaluation Board [21]. The evaluation board communicated with the PC using the evaluation board software, providing an extensive UI to configure the AD7746 chip as per user requirements. The major drawbacks for using the evaluation board were as follows.

1. The evaluation software was only compatible with windows OS (rarely used by any robotics platform).

2. The footprint of the board was unnecessarily large for the required applications.

3. For the connection of the board with the sensors the BNC connectors were used with alligator ends, which lead to large noise (around 89 fF) in the measurements due to their exposed nature as shown in Fig. 2.4.

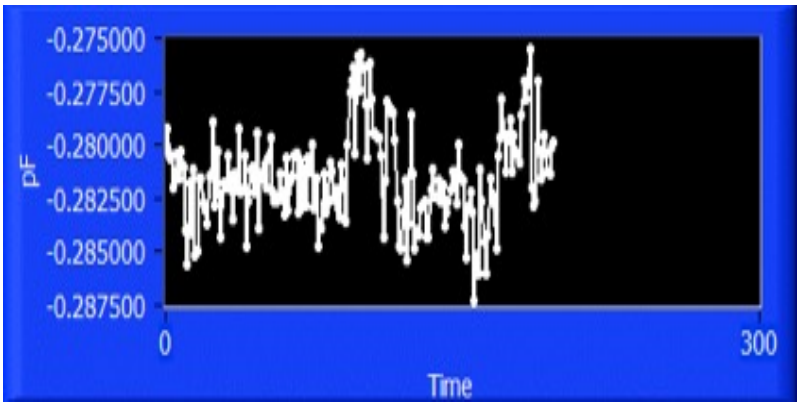


Figure 2.4: Sensor feedback using AD7746 evaluation board

Due to the issues we decided to create our own PCB around the AD7746 chip. For noise reduction, a low pass filter was added and a range extension circuit was included to test sensors with different configurations (the range extension circuit

was not used for any of the experiments). The model diagram for the circuit is shown in Fig. 2.5 along with the final manufactured circuit board in Fig. 2.6. The Arduino was not part of the PCB and was connected externally. The decision to exclude the microcontroller was made to provide flexibility of using inbuilt robotic system controller (if available). The circuit diagrams for the final circuit is attached in Appendix C.

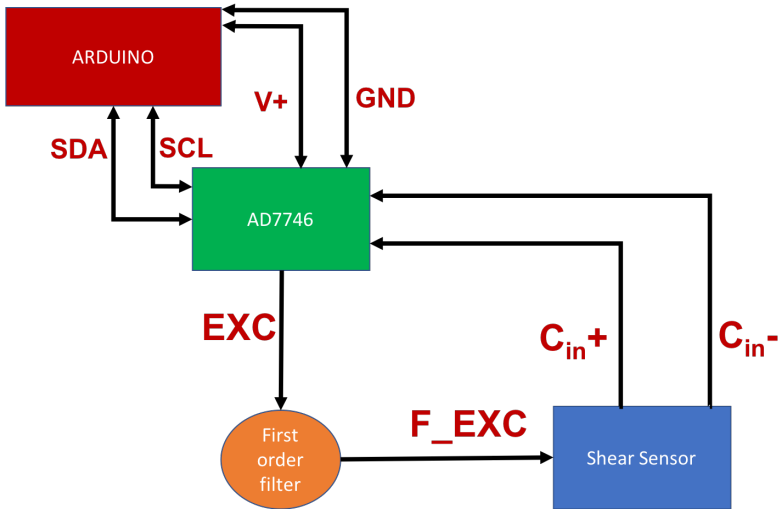


Figure 2.5: Circuit model of the final PCB

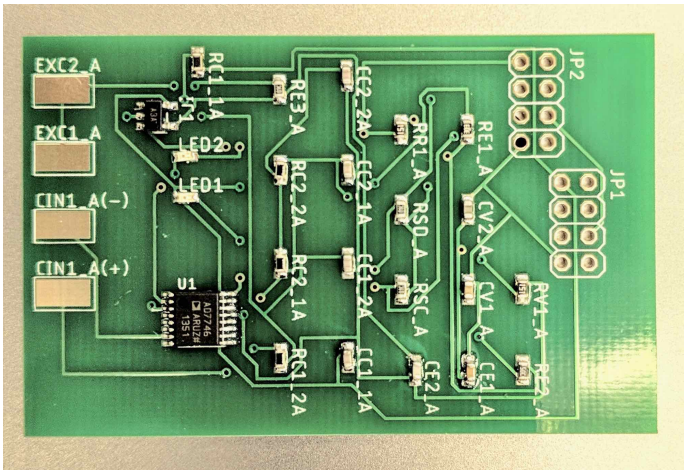


Figure 2.6: Circuit used for the sensor integration

2.4.1.1 AD7746 Chip

The AD7746 is a high resolution, $\Sigma-\Delta$ capacitance to digital converter. It measures the capacitance directly connected to the device. It has two capacitance channels as shown in Fig. 2.8 and each channel can be configured to work in either single-ended or differential mode. Single ended mode measures the absolute capacitance connected to the CIN(+) channel, whereas the differential mode measure the difference in the capacitance of the two capacitors connected to the CIN(+) and CIN(-) channels. The AD7746 supports an I2C-compatible 2-wire serial interface. Two wires on the I2C bus are called SCL (clock) and SDA (data). These two wires carry all addressing, control, and data information one bit at a time over the bus to all connected peripheral devices. The SDA wire carries the data, while the SCL wire synchronizes the sender and receiver during the data transfer. A device that initiates a data transfer message is called a master, while a device that responds to this message is called a slave [22]. In this work, the AD7746 works as a slave whereas the Arduino uno takes the role of the master.

The AD7746 has 20 registers and the Arduino can write to or read from all of the AD7746 registers except the address pointer register, which is a write-only register. The Arduino code to configure the AD7746 chip is included in Appendix A.

As discussed previously, shear forces are measured by examining the differential capacitance between opposite pads. Therefore, measuring the differential capacitance between Pads 2 and 3 can be achieved using single channel on the chip.

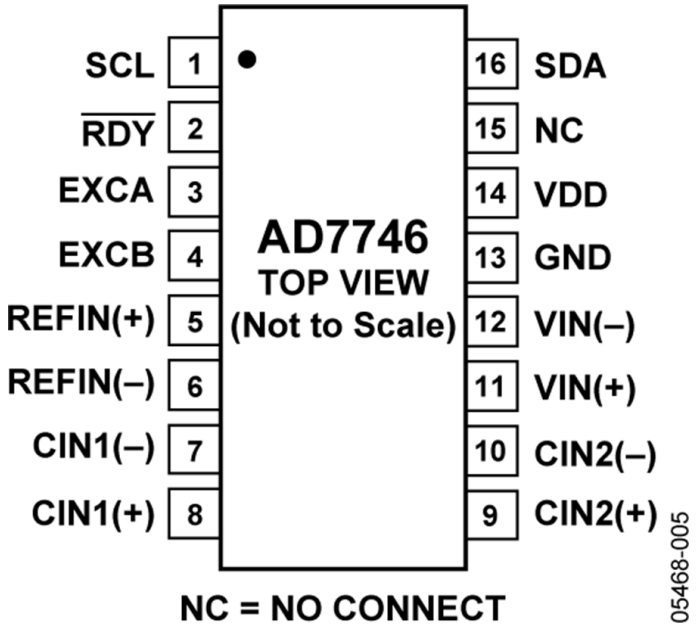


Figure 2.7: Circuit diagram of the AD7746 chip [21]

This makes the system much simpler when compared to [16] where multiple AD7746 chips were used to track each individual pillar-pad capacitance pairs. The tradeoff is that this configuration does not allow for measurement of normal force on its own. However, as demonstrated in chapter 4, most basic gripping contacts result in some shear forces. Therefore, the AD7746 was configured to a single channel differential mode for all the experiments performed in Chapter 4. Chapter 5 talks about the work done to measure forces in all three directions, but none of the experiments in this thesis utilize that ability.

2.4.1.2 Excitation Signal Filter

The board includes an first order RC lowpass filter circuit for the excitation signal. The filter used a resistance of 10 kΩ and a capacitance of 47 pF. The filter prevents

the RF frequencies from entering the AD7746; more details on the external filter circuits are included in [23]

2.4.2 Microcontroller

As mentioned above, the Arduino Uno was used as the master to configure the AD7746 chip. The Arduino supported I2C communication which lead to ease of integration with the AD7746. Although several other micro-controllers with I2C support could have done the same job, the key distinguishing feature for the Arduino was its integration with ROS. As mentioned in Chapter 1, one of the aims for the sensor system was compatibility with ROS; Arduino IDE supported rosserial communication, which allowed data transfer between the Arduino and ROS using a USB connection. The Arduino IDE allows rosserial using a universal ROS library allowing communication with all the ROS versions without affecting the processing speed. Other controllers, on the other hand, would require some form of ROS version running natively in order to communicate with ROS, thereby limiting the speed of communication and complicating the overall process.

Arduino published the sensor feedback using a Float64 type message on a ROS topic that was communicated with the main PC using rosserial. The rosserial protocol is aimed at point-to-point ROS communications over a serial transmission line. Rosserial uses the same serialization/de-serialization as standard ROS messages, simply adding a packet header and tail which allows multiple topics to share a common serial link. The communication of multiple topics allowed communication

of both channel data with the main ROS system [24].

Arduino published the data on the Linux OS using ROS, but for support with Mac OS and Windows OS another script was written to publish the sensor output on the serial monitor. Therefore, using Arduino made the task of compatibility with different OS simpler. It is worth noting that the circuit and the sensor could be used with other controllers as well, but Arduino was the controller used in all of our experiments and is recommended for ROS integration.

2.4.3 Connectors

For connection between the sensor and the PCB it was essential to have a compact and robust connection as the sensor was meant to experience high forces. The ease of sensor redesign and manufacturing allowed a modular design for the sensors and the connectors were supposed to be chosen keeping the modularity in mind. The alligator clips used by the AD7746 evaluation board provided a modular design but they had a large footprint and were sensitive to the external noise due to their exposed nature.

FCI Clincher connectors were used to connect the elastomer sensors with the PCB because to their minimalistic design and low exposure with the surroundings. The only downfall was that once connected with the sensor, it was harder to take them off the sensor. Hence, the sensor and attached clinch connector acted as an unit.

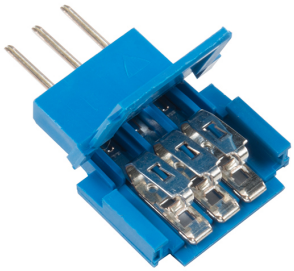


Figure 2.8: FCI clincher connector used in the final setup [25]

Chapter 3

Sensor Software Architecture

The complete software architecture was divided in two parts—Arduino Node and ROS Node. The Arduino node was used to configure the custom board and publish the raw capacitance output from the sensors on a */raw_capacitance* ROS topic. The ROS node initiated the communication with arduino and processed the raw capacitance. The processed data was published on the */capacitance_val* topic, which was used by other packages to read from the sensor.

3.1 Arduino Interface

The Arduino node configured the AD7746 chip by assigning all the AD7746 registers respective values at the start of the node. A simple calibration was performed each time the node was initiated to identify the no-load capacitance value. In case of sensor saturation or wire shorting, the calibration protocol was called again.

After the calibration was completed, the ROS node handler was started on the Arduino to publish raw capacitance data on the */raw_capatiance* topic. The ROS node reads from the same topic to perform further data processing. User can also access this raw data by subscribing to the */raw_capacitance* topic.

For accessing the capacitance feedback on Windows OS or Mac OS, another Arduino script was written that performed the tasks in the same sequence but

instead of starting a ROS node handler it published the sensor feedback on the arduino serial monitor. Both codes are included in Appendix A for further reference.

3.2 ROS Interface

Robot Operating System (ROS) is the most widely used development framework for robotics platforms. Over the years, several robots like Baxter, Sawyer, PR2, and Turtlebots have software that has been developed using ROS [26]. ROS currently supports more than 100 robotic platforms and grippers [27]. Along with the robotic platforms, sensors like Kinect, Real Sense 3D and Velodyne provide support for ROS. The communication structure in ROS makes it one of the most widely accepted platform. The structure of ROS communication is shown in Fig. 3.1.

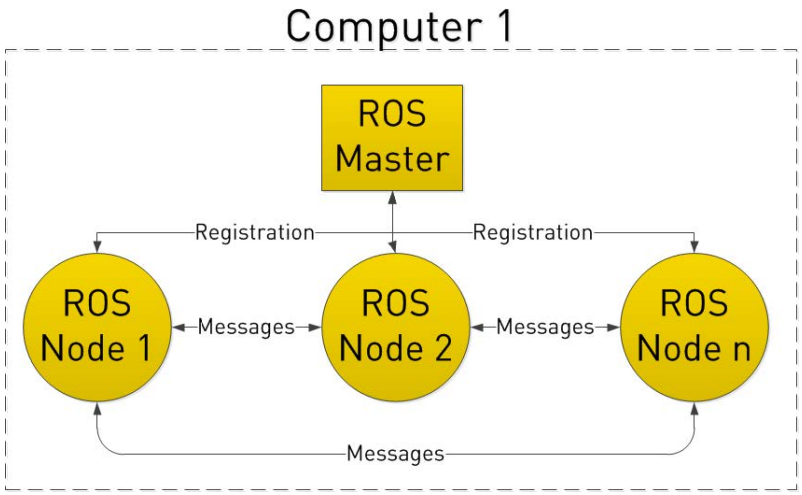


Figure 3.1: ROS communication structure [28]

For all the different components like sensor, manipulator, and gripper, ROS starts a unique node for each component, and all the data between the different components is communicated using these nodes. As long as the node is under the

same master, any other node can access the data published by that node. This makes integration of new sensors and components with the existing system convenient. Therefore, compatibility with ROS allowed the sensors to be integrated with other ROS platforms without making any change to their existing setup.

One of the key issues with ROS is cross compatibility. Each Linux version supports a unique ROS version, and devices running different ROS version cannot communicate with each other. Therefore, it was essential to ensure that the sensor system was version independent by creating custom package and using only universally supported dependencies.

To understand the custom ROS package and the choices made, it is important to know a general structure of ROS. The ROS workspace consists of three folders—src, build, and devel. Src is the folder where all the ROS packages are kept. Build and devel are created by default by using the make function in catkin. Therefore, to use the sensor the user will have to download the package from the GitHub repository and place it in the src. Once the package is present in the src, *catkin_make* will be called to build the sensor dependencies.

3.2.1 Custom ROS Package

The key task of the ROS package was to start communication with the Arduino and process the published raw sensor measurements as described in algorithm 1.

The mean nominal differential capacitance along with the peak-to-peak noise from this measurement is measured when the system is started. If a change in this

differential capacitance (defined as */raw_capacitance*) is detected, a flag is thrown to indicate the state. The continuously published */raw_capacitance* values are plotted as "Capacitance Differential" in the results below.

Algorithm 1 Pseudo code for raw sensor measurements

```
1: procedure SHEAR_SENSOR(/raw_capacitance)
2:     while /raw_capacitance published do
3:         filter /raw_capacitance data
4:         publish filtered capacitance data
5:     for first 200 data points do
6:         calculate mean differential capacitance
7:         publish mean capacitance data
8:         calculate noise
9:     while /raw_capacitance published do
10:        measure /raw_capacitance
11:        if mean − /raw_capacitance > noise then
12:            raise flag
```

A general ROS package includes four key component—src folder, launch folder, CMakeList.txt and package.xml. The src folder contains all the necessary node files responsible for starting the main processing code. The launch folder includes the launch file; running the launch file initiates the communication with the Arduino and executes the node file from the src folder. Although CMakeList.txt and package.xml are created by default, CMakeList.txt required several changes to define

dependencies.

For the sake of compatibility with different ROS versions, custom messages were used. Use of custom messages and their integration is explained in subsequent sections.

3.2.1.1 ROS Message

ROS communicates via publishing messages on different topics. These messages could be either a standard message (using the default ROS message format) or custom message (personalized format based on standard formats). Using custom message allowed us to publish multiple information on a single topic. For example, as shown in algorithm 1 the sensor node has to broadcast the mean of the sensor feedback and the sensor feedback. Using our custom ROS message we were able to publish both the data on a single topic (*/capacitance_val*), making it easier for the user to keep the track of all the information. The custom message used only the standard Float64 format, which was uniform in all the versions of ROS.

To use the custom message, a cap.msg file (as shown in appendix B) was written defining the format of the message and it was kept in the msg folder. The package recognizes the custom message format once the cap.msg file is sourced in the CMakeList.txt file as shown in Appendix B.

3.2.1.2 ROS Node

The *shear_filter* is the main sensor node that receives the raw sensor feedback from the Arduino, performs the rolling window filter over the raw data, and publish the mean and filtered capacitance. The node is written in Python and uses the Savitzky Golay filter from the scipy library [29]. Since all the libraries are available in Python versions above 2.3, the node will function as long as the scipy library is installed. For installing the scipy and other dependencies, a script file was written as discussed in section 3.2.2.

The shear_filter.py spawns the *shear_filter* node that publishes the processed data on the */capacitance_val* topic using cap.msg format. Any other node can receive the processed data by subscribing to the topic. The complete shear_filter.py file is added in Appendix B.

3.2.1.3 Launch File

The primary objective of the launch file is to establish communication with the Arduino and spawn the *shear_filter* node. Without the launch file, the user would have to manually establish a connection the with Arduino using the rosserial_python package and set the parameters like communication baud rate and port number. The launch file shear_sensor.launch in the launch folder starts the rosserial node and the user can manually change the setting like the baud rate and port number in the launch file. Since the launch file starts the master by itself, the user just needs to start the launch file for standalone interaction with the sensor.

To ensure that test data is recorded every time a launch file is running, a rosbag is started that collects the messages published on the */capacitance_val* topic along with the time stamp of the message. Once the launch file is stopped the bag file is saved in the log folder for future access. For more details on the launch file please refer to Appendix B.

3.2.2 Dependencies

Most of the dependencies required for the package are built into Linux versions above 14.04. The package requires the rosserial_python package (to initiate serial communication) and scipy library (to call the Savitzky Golay filter) which are not standard in all the versions. To ensure the user has every dependency installed and the package can access those dependencies, a small script was written to check and install the packages if missing.

Chapter 4

Experiments and Results

4.1 Introduction

In recent years manipulators are breaking conventional boundaries by operating in open environments such as households, and hospitals. Unfortunately, interaction of robots with the open environment is difficult due to the unknown or imperfect model of their surroundings. Without the information of their surroundings manipulators can damage itself or the objects they are manipulating. Most robotic systems use visual feedback which does not provide adequate information for grasping objects. Haptic feedback is either completely absent or limited to normal force detection. As discussed in the Chapter 1, the absence of the shear feedback is due to the fact that most of the available technologies are not operational with the robotic systems or require extensive modifications.

In this chapter, we report a series of experiments that showcase application and integration of the sensor system with grippers like RightHand Labs ReFlex Takktile gripper and a Robotiq 2-finger adaptive gripper to test the ease of installation with multiple grippers along with sensing performance in a variety of grasping and manipulation tasks. The experiments were designed to show how sensor feedback could improve and enhance different manipulation tasks. Results show the sensor response over the course of the experiments. Further analysis of the data allowed

us to detect contact and slippage.

Section 4.2 talks about the experimental setup and other sensor integration details. Section 4.3 showcase system's idle state performance in different environments. Section 4.4 starts the experiment part with the pick and place experiment performed along with KUKA iiwa. Section 4.5 lays down the relation between sensor response and shear detection. This relation is exploited in the section 4.6 for a closed-loop human interaction. Finally, Section 4.7 concludes the experiments.

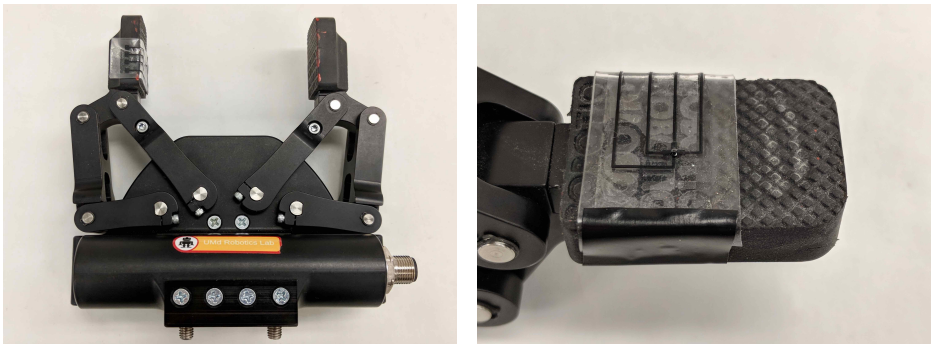
4.2 Experimental Setup

The key focus during the integration of the sensor system was to ensure that it does not affect the exiting setup. The sensor system should act as a plug and play accessory that can be used for different grasping and manipulation tasks without modification. The sensors conformable design facilitated the physical mounting of the sensors on various surfaces. Integration of the sensor with the Takktile gripper and the Robotiq gripper is shown in Fig. 4.1. The connecting pads were kept out of the gripping areas to ensure the connections were not affected over the course of experiments.

The experiments included a basic pick and place task, wrist-based manipulation, peeling tape, and closed-loop interaction between the human and a robot grasped object. For the pick & place, wrist manipulation, and tape removal experiments the Robotiq gripper with KUKA iiwa was used. The experiments were open loop, and raw sensor feedback was recorded for each of them. For the human inter-

(a) ReFlex gripper + sensor

(b) Zoomed in image of sensor installation



(c) Robotiq gripper + sensor

(d) Zoomed in image of sensor installation

Figure 4.1: Single sensor mounted on robotiq and ReFlex gripper at the grasping locations

action experiment, a closed loop controller was written to enable human interaction with grippers using the shear feedback. The Takktile gripper was used for the closed loop experiment and the sensor feedback along with the gripping force applied by the gripper were recorded.

All the tests were performed using the same sensor with a 35-micron gap between the pillar and pads, pillar height of 1400 micron, pad height of 400 microns

and covered with PDMS on both sides. Only for the tape removal experiment, as discussed in section 4.5, the sensor with no upper PDMS layer was used to detect lower magnitudes of shear forces.

The complete sensor system setup with the KUKA iiwa is shown in Fig. 4.2. The setup was within a single link of iiwa and did not constrain the joint motion. A USB connection was the only wire coming out of the setup; hence, risk of a lose connection was minimal.

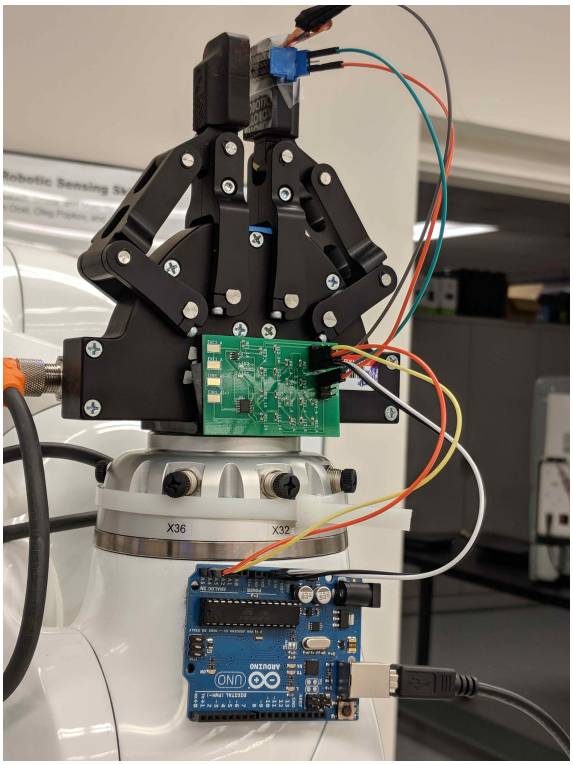


Figure 4.2: Shear sensing system integrated with KUKA iiwa

4.3 Sensor Performance on Robots

For any practical application of the sensing system, it was important to ensure that it is functional alongside high powered electro-mechanical devices such as a KUKA iiwa robot arm. Unshielded capacitive sensors can result in unpredictable performance in noisy environments [30]. To test the sensor performance on a robot, sensor measurements (with no loads applied) were collected for the following cases: 1) without the robot motors running and 2) when mounted on the KUKA robot with all motors running to emulate an electro-mechanically affected environment. Fig. 4.3 shows the raw sensor data for both experiments. The mean of both measurements is approximately 0.22 pF with noise levels of approximately 10 fF. A small change of 1.7 fF was measured between the mean values in both experiments—well within the noise of the sensor.

4.4 Pick and Place Experiment

Pick and place is one of the most common tasks that robots perform, and vision sensors are the most common sensing systems used for this task. This experiment shows how such a task could be completed by using the shear sensor alone, thereby reducing the computing power required and providing feedback in cases where vision might not be easily used (e.g., a dark environment). The sensor feedback prevents damage to the object (from crashing into the table) along with avoiding grasping failures due to slippage.

The entire experiment was divided into 5 stages as illustrated in Fig. 4.4.

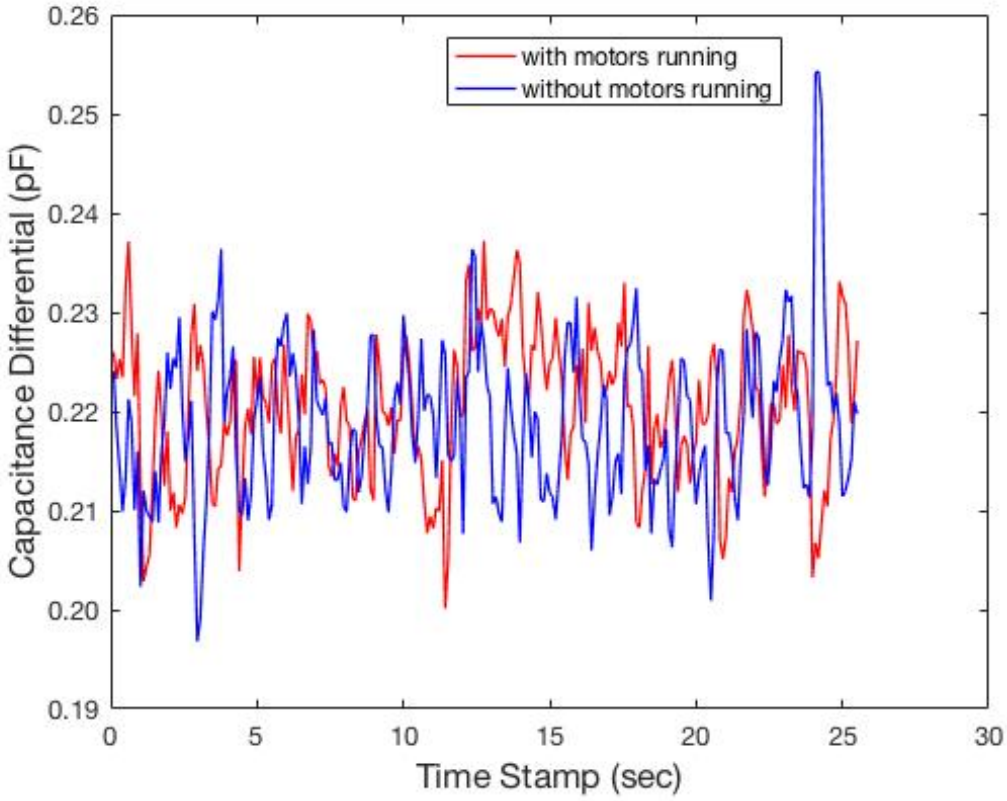
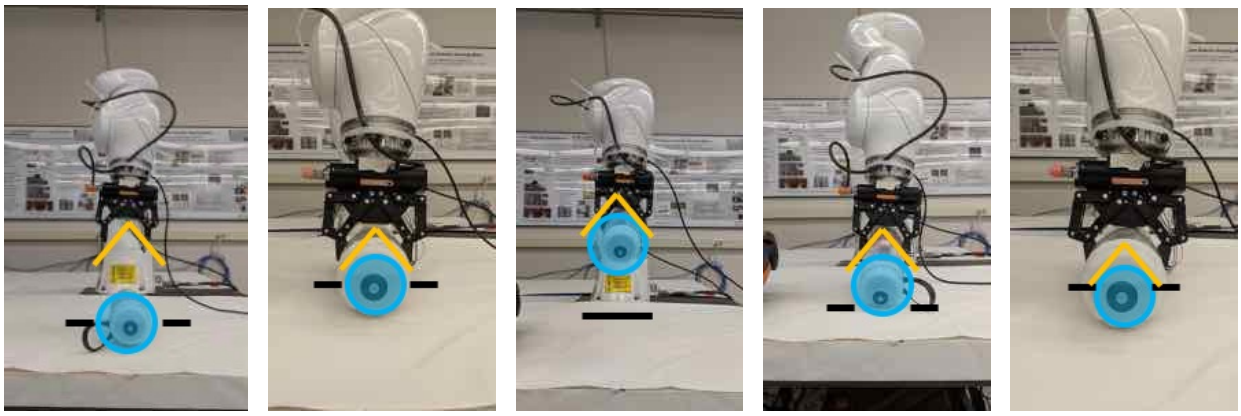


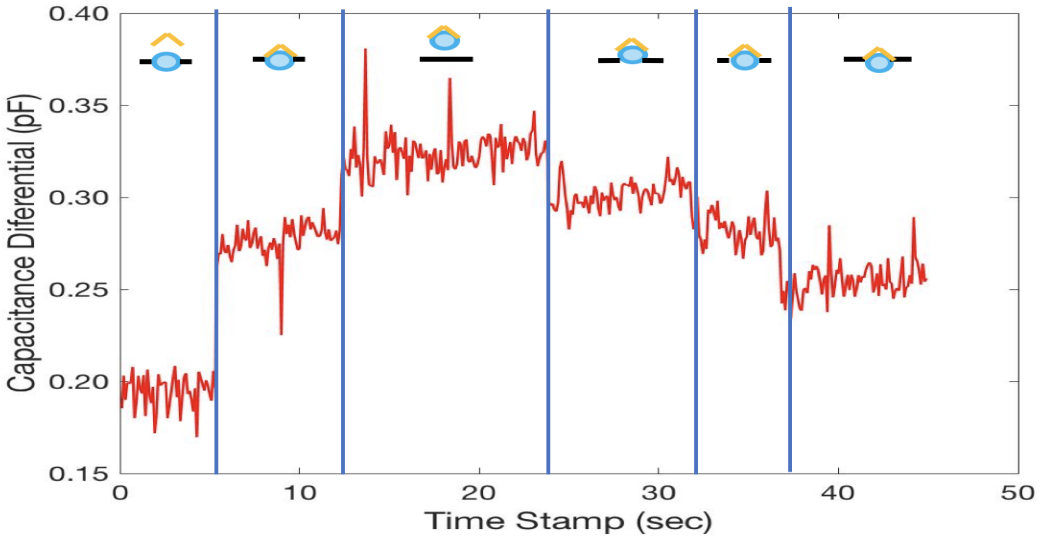
Figure 4.3: Sensor response with no applied load without robot motors running (blue) and with motors running (red)

Stage 1 shows the robot directly above the object (a plastic water bottle filled with water), but not grasping the object—its nominal state. In Stage 2, the robot grips the object but does not lift it off the table. In Stage 3 the object is lifted by the robot. The robot returns the object to the table in Stage 4, barely touching the table surface. Finally, in Stage 5, the robot pushes the object into the surface; this stage could cause damage in some systems and is used to illustrate what should be avoided.

The sensor feedback as shown in Fig. 4.4f provides a clear indication of dif-



(a) State 1 (b) State 2 (c) State 3 (d) State 4 (e) State 5



(f) Sensor response

Figure 4.4: Pick and place experiment demonstrated different reaction forces applied to the gripper based on the interaction between the water bottle, gravity, and the table

ferent stages based on the measured capacitance differential. As mentioned earlier, the geometry and grasp of an object can impart shear forces to the gripper and this is clearly illustrated in these results. As soon as the object is grasped in Stage 2 (with no forces intentionally applied tangential to the gripping axis), a shear force is measured by the sensor. When the robot lifts the water bottle, a positive change in capacitance differential is measured due to the gravitational force on the water bottle. The capacitance differential below this nominal grasped value in Stage 5 is due to the upward force imparted onto the water bottle by the table. Apart from occasional spikes (which could be filtered in hardware or software), the sensor feedback was consistent enough to provide comprehensible distinction.

To test the repeatability of the sensors, the object was picked and placed several times; Fig. 4.5 shows part of the raw sensor response. The mean values for the two states remain consistent with occasional noise spikes. The setup was exactly the same as the one shown in Fig. 4.4.

One of the features of the sensor is that they are functional with objects made from different materials and though the object in the experiments described above was a plastic water bottle (approximately 400 g), the sensor was also able to identify these distinct states when working with objects made of metal. This result demonstrates that the sensor is not affected by conductive materials. Fig. 4.6 shows the response of the sensor while performing the pick and place experiment on a cylindrical 500 g metal mass. In the experiment, the metal cylinder was simply grabbed and lifted. The initial jump in the capacitance was high compared to the previous experiment, this behavior was due to two possible reasons: 1. higher

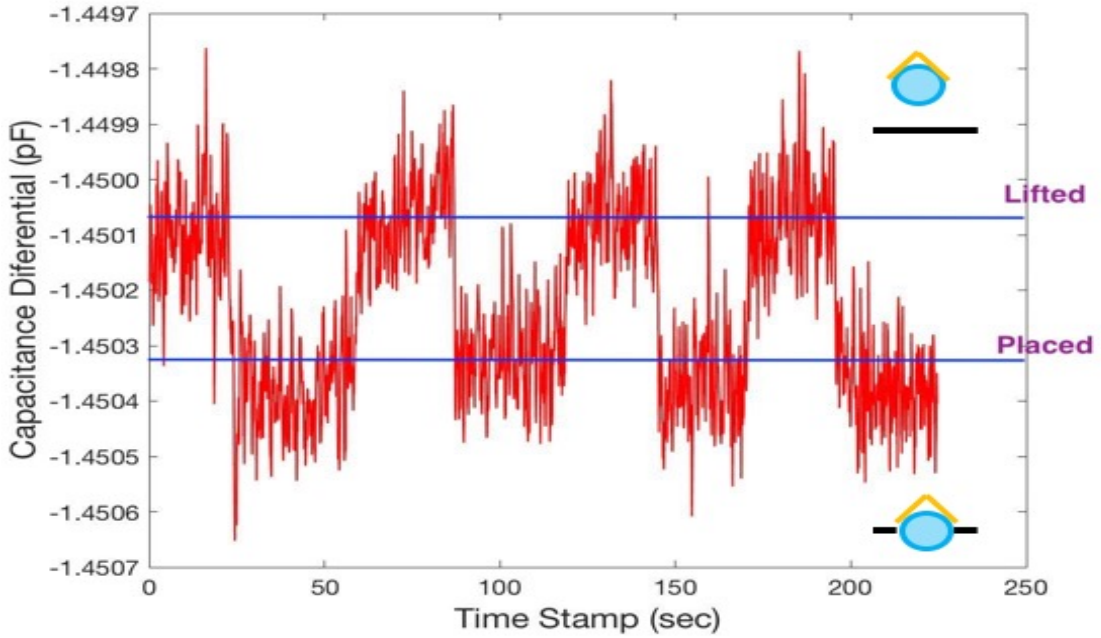


Figure 4.5: Sensor response from repeated picking and placing

grasping force used to lift the heavier weight, and 2. the metallic material of the object.

4.5 Shear Forces during Manipulation

To examine sensor measurements during manipulation tasks, two different tasks were chosen: 1) a relatively simple wrist-based manipulation task, and 2) tape removal. During the wrist-based manipulation experiment, the Robotiq gripper was used to grasp the plastic water bottle at one end. The force applied was strong enough to prevent the water bottle from falling, but light enough to allow slip between the gripper and the object within the grasp. This force was not altered through the course of the experiment. The water bottle was then manipulated relative to the

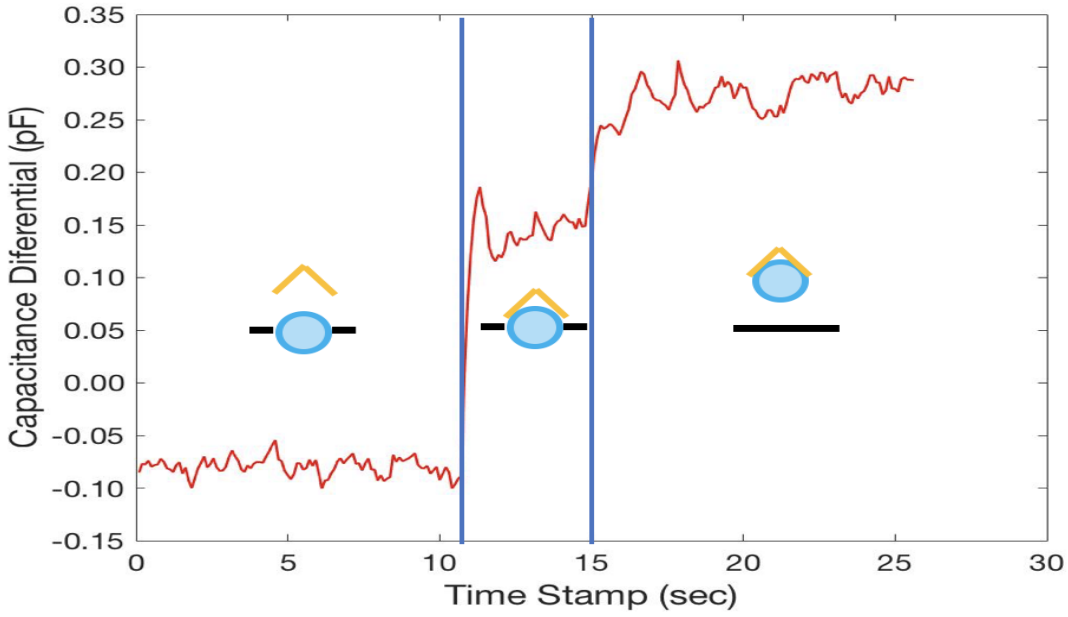
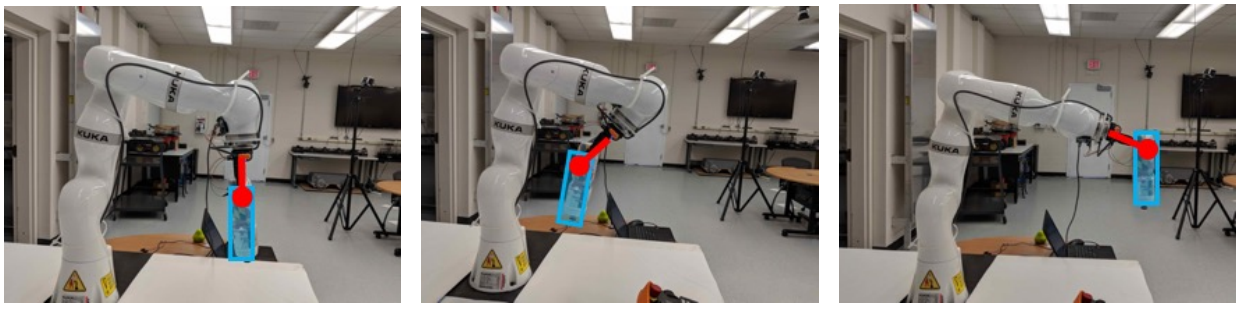


Figure 4.6: Sensor feedback for the grasping and lifting a metallic object

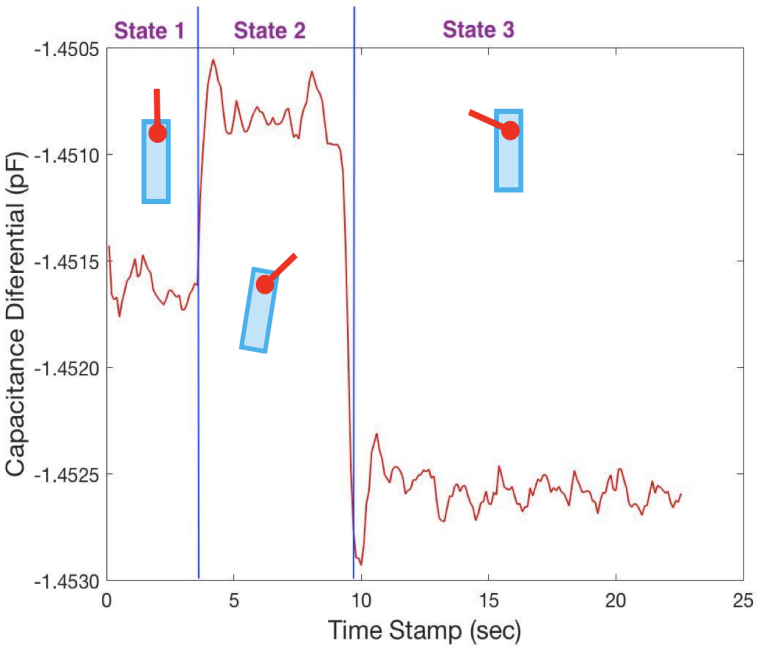
gripper using gravity by swinging the robot's wrist back and forth between States 2 and 3 as shown in Fig. 4.7a-c.

As shown in Fig. 4.7d, the sensor response changes in each of these three states. In each of the states, the weight of the water bottle is imparting a different vector of shear force on the gripper (and sensing system). State 1 shows the capacitance differential from simply gripping the bottle. The capacitance differentials in States 2 and 3 above and below this nominal gripped value show that the shear force is pointing in opposite directions in these states.

In the tape removal experiment, the Robotiq gripper was used to grip one end of the tape and the robot was moved in a vertical direction until the tape was peeled off the surface (Fig. 4.8a). This task was chosen due to its relevance in the packing and unpacking industries and to show off the modularity of the sensing



(a) State 1 (b) State 2 (c) State 3



(d) Sensor response

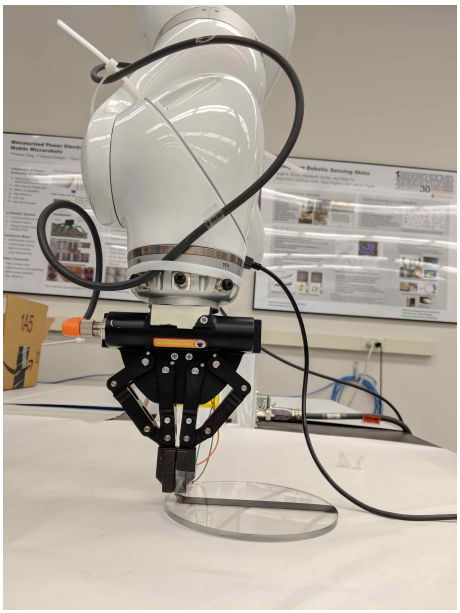
Figure 4.7: Wrist manipulation experiment demonstrating the tangential reaction forces applied to the manipulator by the water bottle

system; a sensor without the 'Upper PDMS' layer is used to increase sensitivity and the sensors were easily swapped out using the Clincher connectors. The increased sensitivity is seen in Fig. 4.8b—instead of a range of tens of femtoFarads, signals here span over a picoFarad. In addition, the signal from the tape removal process is particularly noisy. This is not surprising given the stick/slip nature of the adhesive. Despite this, clear distinctions in the removal process can be seen in this figure. For example, the signal corresponding to the point at which the tape is finally removed from the surface experiences a dramatic change in shear force.

In addition, Fig. 4.8 shows that the sensor without the upper PDMS layer exhibits significant hysteresis. The signal after the tape has been removed is not at the same level as when the tape is first grasped (at approximately 17 sec). This is due to the lower pillar elasticity that results when the pillar is not surrounded with PDMS. With a low restoring force, the pillar remains deformed even after a shear load is removed. However, the mean of the capacitance differential before and after the tape is removed remains relatively stable.

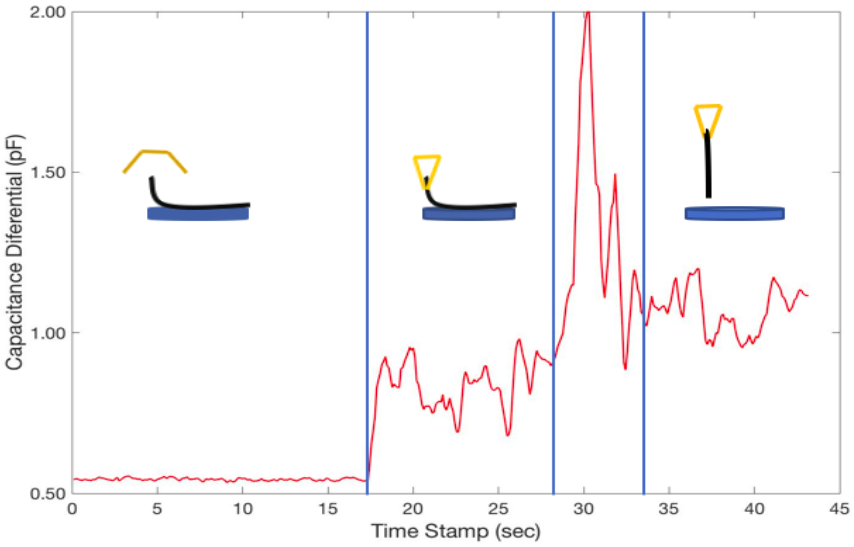
4.6 Human Interaction and Closed Loop Control

With robots being introduced in households and open environments, their chances of interacting with humans is bound to increase. This is a challenge, especially for mobile manipulators as their tasks tend to be relatively dangerous for humans. Even the current work place environments face difficulties with human-robot interaction [31]. The major issue is that the communication between human and robot is often



(a) Experimental setup for removing tape

from an acrylic block



(b) Sensor response

Figure 4.8: Experimental setup and sensor response during tape removal

via rudimentary systems like physical controllers and buttons. Due to which even the cooperative tasks in manufacturing lines are more turn based rather than parallel. One of the applications of the sensing system is to create fluid workflow by providing a more intuitive way for the humans to interact with robots.

In the final experiment, the sensing system was applied to a different gripper – the RightHand Labs ReFlex Takktile gripper (Figs. 4.1a/b). Since the sensor is able to measure tangential shear forces, interactions like pulling and pushing on the object can be distinguished from regular slipping based on the magnitude of variation in capacitance. These interactions could ultimately be used as a more natural way for humans to program manipulators [31, 32].

To validate this approach, the ReFlex hand was used to grasp the plastic water bottle, and the bottle was pulled and pushed in the Y-direction as marked in Fig. 4.9a. A controller was written to increase grasp force if the water bottle was pushed in the negative Y-direction. The measured shear forces were thresholded into four different levels—two in each direction—and four corresponding grasp forces (indicated by motor load) were applied to the water bottle. Based on these thresholded levels, if the water bottle was pulled in the positive Y direction, the hand released the bottle. Releasing the grip on the object increased the sensor response, thereby increasing the grip again so the bottle was never fully released from the hand. The bottle was repeatedly pulled and pushed in succession for this experiment and Figs. 4.9b/c show the capacitance differential from the shear sensing system as well as the motor load as published by the ReFlex hand. As expected based on this controller, the motor load drops when the water bottle is pulled and increases when the water

bottle is pushed.

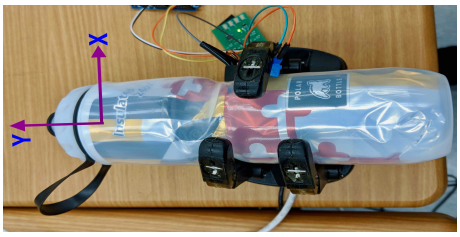
4.7 Summary

The sensor system was tested for different manipulation tasks and sensor feedback was collected. The sensors were able to identify the contact with the object in all the experiments, thereby, providing a rudimentary estimation of the normal contact force. This allowed us to perform simple tasks without any additional normal force sensors or visual feedback. The new setup has significantly less noise (around 10 fF) compared to previous setup (around 89 fF).

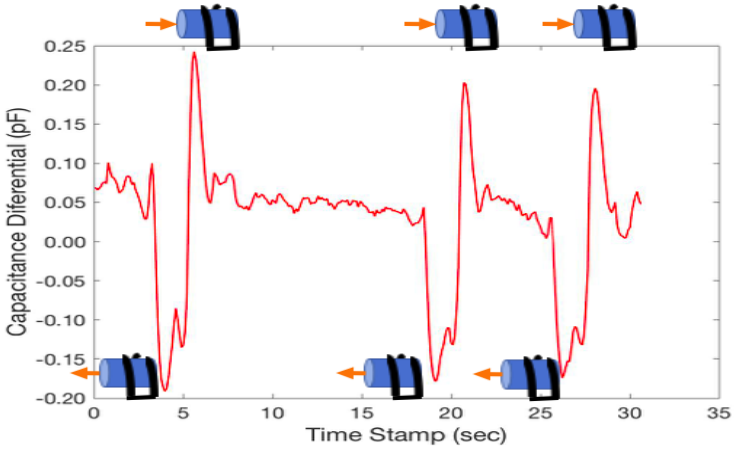
The object material independence was shown in Fig. 4.4 and 4.6 as the response was similar for the plastic and the metal object. It is important to note that though the sensor response was identical, the sensor noise (around 50 fF) and response magnitudes (around 100 fF) were much higher for metallic object.

Wrist manipulation experiment helped us visualize a key sensor behavior which could lead to a better dynamic control of objects over more complex maneuvers. The modularity of the sensor system was displayed by the tape removal experiment as with a simple switch of the sensor we were able to measure low forces occurring during untaping process.

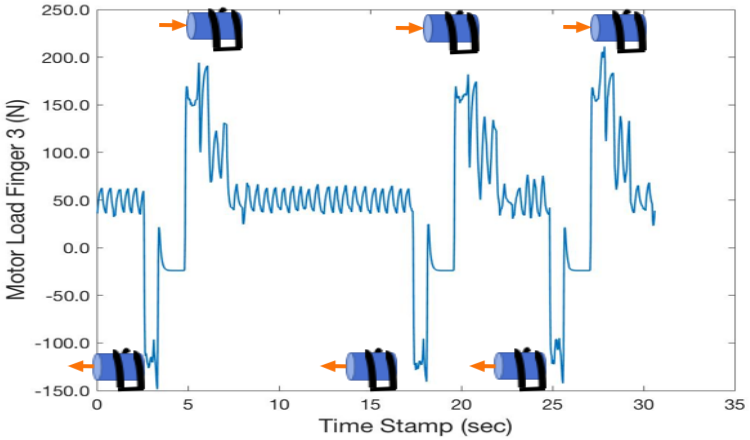
Closed loop experiment exploited the complete benefit of the sensor software architecture by implementing a simple integration of the grippers with the sensor system. Both the systems were controlled using ROS and a simple controller was implemented by including the sensor feedback with the existing gripper script.



(a) Experimental setup using the Right-Hand Reflex Takktile hand with the shear sensor on the isolated finger



(b) Sensor response over the course of interaction with the object



(c) Motor load applied by the isolated finger

Figure 4.9: Experiment on human-robot interaction with a grasped object using the Reflex Takktile gripper and sensor feedback

An interesting observation made during all the experiments was the hysteresis shown by the sensor. Due to the physical behavior of sensor, the sensor response takes some time to reach it's initial value once the force is removed. The time may vary from milli seconds for PDMS covered sensors to couple of seconds for non-PDMS covered sensors.

It is worth noting that the current sensor system suffers from limitations like limited sensing area and only shear force detection. The current system utilizes a singular sensor thereby limiting the net sensing area. Also, the experiments conducted collects the shear feedback using a single capacitance channel and hence forces in only one direction is collected. The Chapter 5 talks in detail how these limitations could be overcome in future works.

Chapter 5

Conclusion

This thesis presents the integration of a low cost, conformal, 3-axis sensing system with multiple off-the-shelf robot end effectors. While only shear in one direction was measured in these experiments, both normal and shear loads are available from the sensor. The hardware is simple to install without any design changes to the grippers and a software interface is provided as a ROS package. The sensor is able to measure tangential reaction forces that occur during grasping and manipulation as demonstrated through a variety of experiments that show the reaction to a grasped object being gripped, lifted, rotated or tugged on. The sensor also shows promise in being able to improve the intuitiveness of human-robot interactions.

5.1 Collaboration with other universities

Collaboration work along with the University of Pennsylvania and Johns Hopkins University is currently undergoing. Due to the small package size and ease of access to sensor feedback, students from those universities were able to test our sensors along with their projects.

In the University of Pennsylvania, a group working on Human-Robot interaction is using our sensors to measure the shear forces experienced by the robot during human interaction. We are currently in the process of creating a grid of

sensors for the group to facilitate them in measuring the tactile forces at the contact locations. In Johns Hopkins, students were looking for alternative sensors for contact detection.

The acceptance of the sensor system and feedback from the groups is helping us to determine the limitations and the next steps required for improving the systems performance.

5.2 Limitations

- The current system only measures shear in one direction, though, the sensor itself can provide feedback along three axis. Incorporating the complete feedback from the sensor will help in achieving better control.

- Only a single sensor unit was used in the system, which limits the sensing area and sensor placement at the accurate grasping locations. The sensors should cover the entire grasping area to ensure detection of contact forces all over the surface.

- The system works on the principle of relative change in the feedback value; no characterizations tests have been performed with the sensing system to provide co-relation between sensor feedback and absolute force applied.

5.3 Future Work

Future work will continue to improve the sensors' effectiveness in aiding grasping and improving the interface for human interaction. The following steps will help in improving the system performance and utilize the sensor to its full potential.

1. One of the steps would be to read the remaining capacitance feedback between the pillar and pad 1 as shown in Fig. 2.1. The ground work for reading this capacitance value is already done and is included in Appendix A. The Arduino script utilizes the other channel of the same AD7746 chip to measure the absolute capacitance between the pillar and pad 1. Since none of the experiments were performed with this setup it is not included in the work completed.

2. Although the sensor only measures shear force along one direction, as we can see from the Chapter 4 that the feedback provides an indication of the initial grasping force. Since the sensor is based on the principle of changing capacitance due to bending of the pillar, sensor characterization tests are required to related feedback along each axis with the normal force and the shear force along that axis.

3. To cover the entire grasping area, sensing skins for the gripper needs to be created. The all elastomer design of the sensors will help in fabricating a conformable sensing skin for the different gripper shapes and sizes. A simpler PCB design as shown in Appendix C could be used for measuring feedback

from a single sensor unit and a single microcontroller using a multiplexer can provide feedback from the entire skin.

Appendix A

Arduino Code

A.1 Arduino with rosserial

The Arduino code used to configure the AD7746 chip and communicate with ROS.

```
#include <Wire.h>

//ROS Libraries

#include <ros.h>

#include <std_msgs/Float64.h>

//AD7746 definitions

#define I2C_ADDRESS 0x48 //0x90 shift one to the rigth

#define WRITE_ADDRESS 0x90

#define READ_ADDRESS 0x91

#define RESET_ADDRESS 0xBF

#define REGISTER_STATUS 0x00

#define REGISTER_CAP_DATA 0x01

#define REGISTER_VT_DATA 0x04

#define REGISTER_CAP_SETUP 0x07

#define REGISTER_VT_SETUP 0x08

#define REGISTER_EXC_SETUP 0x09

#define REGISTER_CONFIGURATION 0x0A

#define REGISTER_CAP_DAC_A 0x0B
```

```
#define REGISTER_CAP_DAC_B 0x0B

#define REGISTER_CAP_OFFSET 0x0D

#define REGISTER_CAP_GAIN 0x0F

#define REGISTER_VOLTAGE_GAIN 0x11

#define RESET_ADDRESS 0xBF

#define VALUE_UPPER_BOUND 16000000L

#define VALUE_LOWER_BOUND 0xFL

#define MAX_OUT_OF_RANGE_COUNT 3

#define CALIBRATION_INCREASE 1

//setup parameters

byte calibration;

byte outOfRangeCount = 0;

unsigned long offset = 0;

unsigned long offsetting;

//initialize the ROS node

ros::NodeHandle nh;

float x;

std_msgs::Float64 cap_msg;

ros::Publisher chatter("chatter", &cap_msg);

void setup()

{

  Wire.begin(); // sets up i2c for operation

  Serial.begin(9600); // set up baud rate for serial

  Wire.beginTransmission(I2C_ADDRESS); // start i2c cycle
```

```
  Wire.write(RESET_ADDRESS); // reset the device
  Wire.endTransmission(); // ends i2c cycle
  //wait a tad for reboot
  delay(1);
 // EXC source A
  writeRegister(REGISTER_EXC_SETUP, _BV(3) | _BV(1) | _BV(0));
// cap setup reg - cap enabled
  writeRegister(REGISTER_CAP_SETUP,_BV(7)| _BV(5));
  offset = ((unsigned long)readInteger(REGISTER_CAP_OFFSET)) << 8;
  writeRegister(0x0A, _BV(7) | _BV(6) | _BV(5) | _BV(4) | _BV(3) | _BV(2)
      | _BV(0));
  //wait for calibration
  delay(10);
  displayStatus();
  //Serial.print("Calibrated offset: ");
  offset = ((unsigned long)readInteger(REGISTER_CAP_OFFSET)) << 8;
  //Serial.println(offset);
 // cap setup reg - cap enabled
  writeRegister(REGISTER_CAP_SETUP,_BV(7)| _BV(5));
  writeRegister(REGISTER_EXC_SETUP, _BV(3)); // EXC source A
   // continuous mode
  writeRegister(REGISTER_CONFIGURATION, _BV(7) | _BV(6) | _BV(5) | _BV(4)
      | _BV(3) | _BV(0));
  displayStatus();
```

```
  calibrate();
 //initialization of ROS Publisher
  nh.getHardware()->setBaud(9600);
  nh.initNode();
  nh.advertise(chatter);
}
void loop() // main program begins
{
  // if we recieve date we print out the status
  if (Serial.available() > 0) {
    // read the incoming byte:
    Serial.read();
    displayStatus();
  }
  long value = readValue();
 if ((value<VALUE_LOWER_BOUND) or (value>VALUE_UPPER_BOUND)) {
    outOfRangeCount++;
  }
  if (outOfRangeCount>MAX_OUT_OF_RANGE_COUNT) {
    if (value < VALUE_LOWER_BOUND) {
      calibrate(-CALIBRATION_INCREASE);
    }
    else {
      calibrate(CALIBRATION_INCREASE);
```

```
    }
    outOfRangeCount=0;
  }
  x=value;
  //ros publisher on capacitance
  cap_msg.data = x;
  chatter.publish( &cap_msg );
  nh.spinOnce();
  delay(1);
}
void calibrate (byte direction) {
  calibration += direction;
  //assure that calibration is in 7 bit range
  calibration &=0x7f;
  writeRegister(REGISTER_CAP_DAC_A, _BV(7) | calibration);
}
void calibrate() {
  calibration = 0;
  long value = readValue();
  while (value>VALUE_UPPER_BOUND && calibration < 128) {
    calibration++;
    writeRegister(REGISTER_CAP_DAC_A, _BV(7) | calibration);
    value = readValue();
  }
```

```
}


long readValue() {
  long ret = 0;
  uint8_t data[3];
  char status = 0;
  //wait until a conversion is done
  while (!(status & (_BV(0) | _BV(2)))) {
    //wait for the next conversion
    status= readRegister(REGISTER_STATUS);
  }
  unsigned long value = readLong(REGISTER_CAP_DATA);
  value >>=8;
  ret =value;
  return ret;
}
```

A.2 Arduino standalone script

The Arduino code used to configure the AD7746 chip and publish feedback on serial monitor for Mac OS and Windows OS.

```
#include <Wire.h>

//AD7746 definitions
```

```
#define I2C_ADDRESS 0x48 //0x90 shift one to the rigth

#define WRITE_ADDRESS 0x90

#define READ_ADDRESS 0x91

#define RESET_ADDRESS 0xBF

#define REGISTER_STATUS 0x00

#define REGISTER_CAP_DATA 0x01

#define REGISTER_VT_DATA 0x04

#define REGISTER_CAP_SETUP 0x07

#define REGISTER_VT_SETUP 0x08

#define REGISTER_EXC_SETUP 0x09

#define REGISTER_CONFIGURATION 0x0A

#define REGISTER_CAP_DAC_A 0x0B

#define REGISTER_CAP_DAC_B 0x0B

#define REGISTER_CAP_OFFSET 0x0D

#define REGISTER_CAP_GAIN 0x0F

#define REGISTER_VOLTAGE_GAIN 0x11

#define RESET_ADDRESS 0xBF

#define VALUE_UPPER_BOUND 16000000L

#define VALUE_LOWER_BOUND 0xFL

#define MAX_OUT_OF_RANGE_COUNT 3

#define CALIBRATION_INCREASE 1

//setup parameters

byte calibration;

byte outOfRangeCount = 0;
```

```
unsigned long offset = 0;

unsigned long offsetting;

void setup()

{

  Wire.begin(); // sets up i2c for operation

  Serial.begin(9600); // set up baud rate for serial

  Wire.beginTransmission(I2C_ADDRESS); // start i2c cycle

  Wire.write(RESET_ADDRESS); // reset the device

  Wire.endTransmission(); // ends i2c cycle

  //wait a tad for reboot

  delay(1);

 // EXC source A

  writeRegister(REGISTER_EXC_SETUP, _BV(3) | _BV(1) | _BV(0));

// cap setup reg - cap enabled

  writeRegister(REGISTER_CAP_SETUP,_BV(7)| _BV(5));

  offset = ((unsigned long)readInteger(REGISTER_CAP_OFFSET)) << 8;

  writeRegister(0x0A, _BV(7) | _BV(6) | _BV(5) | _BV(4) | _BV(3) | _BV(2)

      | _BV(0));

  //wait for calibration

  delay(10);

  displayStatus();

  Serial.print("Calibrated offset: ");

  offset = ((unsigned long)readInteger(REGISTER_CAP_OFFSET)) << 8;

  Serial.println(offset);
```

```
 // cap setup reg - cap enabled
  writeRegister(REGISTER_CAP_SETUP,_BV(7)| _BV(5));
  writeRegister(REGISTER_EXC_SETUP, _BV(3)); // EXC source A
   // continuous mode
  writeRegister(REGISTER_CONFIGURATION, _BV(7) | _BV(6) | _BV(5) | _BV(4)
      | _BV(3) | _BV(0));
  displayStatus();
  calibrate();
void loop() // main program begins
{
  // if we recieve date we print out the status
  if (Serial.available() > 0) {
    // read the incoming byte:
    Serial.read();
    displayStatus();
  }
  long value = readValue();
 if ((value<VALUE_LOWER_BOUND) or (value>VALUE_UPPER_BOUND)) {
    outOfRangeCount++;
  }
  if (outOfRangeCount>MAX_OUT_OF_RANGE_COUNT) {
    if (value < VALUE_LOWER_BOUND) {
      calibrate(-CALIBRATION_INCREASE);
    }
```

```
    else {
      calibrate(CALIBRATION_INCREASE);
    }
    outOfRangeCount=0;
  }
  x=value;
  Serial.print(value);
}
void calibrate (byte direction) {
  calibration += direction;
  //assure that calibration is in 7 bit range
  calibration &=0x7f;
  writeRegister(REGISTER_CAP_DAC_A, _BV(7) | calibration);
}
void calibrate() {
  calibration = 0;
  long value = readValue();
  while (value>VALUE_UPPER_BOUND && calibration < 128) {
    calibration++;
    writeRegister(REGISTER_CAP_DAC_A, _BV(7) | calibration);
    value = readValue();
  }
}
```

```
long readValue() {
  long ret = 0;
  uint8_t data[3];
  char status = 0;
  //wait until a conversion is done
  while (!(status & (_BV(0) | _BV(2)))) {
    //wait for the next conversion
    status= readRegister(REGISTER_STATUS);
  }
  unsigned long value = readLong(REGISTER_CAP_DATA);
  value >>=8;
  ret =value;
  return ret;
}
```

A.3 Arduino dual channel script with ROS

```
#include <Wire.h>
//ROS Libraries
#include <ros.h>
#include <std_msgs/Float64.h>
//AD7746 definitions
#define I2C_ADDRESS 0x48 //0x90 shift one to the rigth
```

```
#define WRITE_ADDRESS 0x90

#define READ_ADDRESS 0x91

#define RESET_ADDRESS 0xBF

#define REGISTER_STATUS 0x00

#define REGISTER_CAP_DATA 0x01

#define REGISTER_VT_DATA 0x04

#define REGISTER_CAP_SETUP 0x07

#define REGISTER_VT_SETUP 0x08

#define REGISTER_EXC_SETUP 0x09

#define REGISTER_CONFIGURATION 0x0A

#define REGISTER_CAP_DAC_A 0x0B

#define REGISTER_CAP_DAC_B 0x0B

#define REGISTER_CAP_OFFSET 0x0D

#define REGISTER_CAP_GAIN 0x0F

#define REGISTER_VOLTAGE_GAIN 0x11

#define RESET_ADDRESS 0xBF

#define VALUE_UPPER_BOUND 16000000L

#define VALUE_LOWER_BOUND 0xFL

#define MAX_OUT_OF_RANGE_COUNT 3

#define CALIBRATION_INCREASE 1

//setup parameters

byte calibration;

byte outOfRangeCount = 0;

unsigned long offset = 0;
```

```
unsigned long offsetting;
//initialize the ROS node
ros::NodeHandle nh;
float x;
std_msgs::Float64 cap_msg;
ros::Publisher chatter("chatter", &cap_msg);
void setup()
{
  Wire.begin(); // sets up i2c for operation
  Serial.begin(9600); // set up baud rate for serial
  //Serial.println("Initializing");
  Wire.beginTransmission(I2C_ADDRESS); // start i2c cycle
  Wire.write(RESET_ADDRESS); // reset the device
  Wire.endTransmission(); // ends i2c cycle
  //wait a tad for reboot
  delay(1);
  // EXC source A
  writeRegister(REGISTER_EXC_SETUP, _BV(3) | _BV(1) | _BV(0));
// cap setup reg - cap enabled
  writeRegister(REGISTER_CAP_SETUP,_BV(7)| _BV(5));
  offset = ((unsigned long)readInteger(REGISTER_CAP_OFFSET)) << 8;
  writeRegister(0x0A, _BV(7) | _BV(6) | _BV(5) | _BV(4) | _BV(3) | _BV(2)
      | _BV(0)); // set configuration to calib. mode, slow sample
  //wait for calibration
```

```
  delay(10);
  displayStatus();
  offset = ((unsigned long)readInteger(REGISTER_CAP_OFFSET)) << 8;
  // cap setup reg - cap enabled
  writeRegister(REGISTER_CAP_SETUP,_BV(7)| _BV(5));
  // EXC source A
  writeRegister(REGISTER_EXC_SETUP, _BV(3));
// continuous mode
  writeRegister(REGISTER_CONFIGURATION, _BV(7) | _BV(6) | _BV(5) | _BV(4)
      | _BV(3) | _BV(0));
  displayStatus();
  calibrate();
  //initialization of ROS Publisher
  nh.getHardware()->setBaud(9600);
  nh.initNode();
  nh.advertise(chatter);
}
void loop() // main program begins
{
  // if we recieve date we print out the status
  if (Serial.available() > 0) {
    // read the incoming byte:
    Serial.read();
    displayStatus();
```

```
}
long value = readValue();
if ((value<VALUE_LOWER_BOUND) or (value>VALUE_UPPER_BOUND)) {
  outOfRangeCount++;
}
if (outOfRangeCount>MAX_OUT_OF_RANGE_COUNT) {
  if (value < VALUE_LOWER_BOUND) {
    calibrate(-CALIBRATION_INCREASE);
  }
  else {
    calibrate(CALIBRATION_INCREASE);
  }
  outOfRangeCount=0;
}
x=value;
//ros publisher on capacitance
cap_msg.data = x;
chatter.publish( &cap_msg );
nh.spinOnce();
delay(1);
selectCIN(2);
value = readValue();
x=value;
//ros publisher on capacitance
```

```
  cap_msg.data = x;
  chatter.publish( &cap_msg );
  nh.spinOnce();
  delay(1);
  selectCIN(1);
}
void calibrate (byte direction) {
  calibration += direction;
  //assure that calibration is in 7 bit range
  calibration &=0x7f;
  writeRegister(REGISTER_CAP_DAC_A, _BV(7) | calibration);
}
void calibrate() {
  calibration = 0;
  long value = readValue();
  while (value>VALUE_UPPER_BOUND && calibration < 128) {
    calibration++;
    writeRegister(REGISTER_CAP_DAC_A, _BV(7) | calibration);
    value = readValue();
  }
}
// -- Channel and sampling frequency selection --
void selectCIN(int CIN){   //, uint8_t CAP_FREQ){
  if (CIN == 1)
```

```
  {
  // cap setup reg - cap 1 enabled
    writeRegister(REGISTER_CAP_SETUP, _BV(7) | _BV(5));
    // Single conversion mode
   // writeRegister(REGISTER_CONFIG, CAP_FREQ | _BV(0));
  }
  else if (CIN == 2)
  {
  // cap setup reg - cap 2 enabled
    writeRegister(REGISTER_CAP_SETUP, _BV(7) | _BV(6));
    // Single conversion mode
   // writeRegister(REGISTER_CONFIG, CAP_FREQ | _BV(0));
  }
  }
long readValue() {
  long ret = 0;
  uint8_t data[3];
  char status = 0;
  //wait until a conversion is done
  while (!(status & (_BV(0) | _BV(2)))) {
    //wait for the next conversion
    status= readRegister(REGISTER_STATUS);
  }
  unsigned long value = readLong(REGISTER_CAP_DATA);
```

```
  value >>=8;

  ret =value;

  return ret;

}
```

Appendix B

ROS Package

B.1 ROS Launch file

The ROS launch file **shear.launch** used to initiate the communication with the Arduino and process the raw data.

```
<?xml version="1.0"?>

<launch>


<!-- record topics listed below -->

<arg name="topic_name" default=" capacitance_val"/>


<!--start the communication with arduino-->

  <node pkg="rosserial_python" type="serial_node.py" name="serial_node">

    <param name="port" value="/dev/ttyACM0"/>

    <param name="baud" value="9600"/>

  </node>


<!--start the filter node to get the capacitance and mean-->

  <node name="filter_cap" pkg="shear_sensor" type="filter_cap.py" />
```

```
<!-- record topics -->

<node name="record_capacitance_data" pkg="rosbag" type="record"

      args="$(arg topic_name)" output="screen"/>


</launch>
```

B.2 ROS Node

The ROS node file **filter_cap.py** used to read from the Arduino, filter the data and publish the processed data on */capacitance_val* topic.

```
#!/usr/bin/env python

import rospy

import builtins

import collections

import numpy as np

import matplotlib.pyplot as plt

from std_msgs.msg import String

from std_msgs.msg import Float64

from shear_sensor.msg import cap

from scipy.signal import savgol_filter


# create the list of capacitance

cap_list=collections.deque(maxlen=20)
```

```
cap_list_mean=collections.deque(maxlen=200)


# float capacitance

capacitance = 100.00

mean_cap = 100.00


# Mean list size

mean_size = 200


# set up publisher

pub_cap = rospy.Publisher('capacitance_val', cap)

msg = cap()


def callback(data):

    # Create the list to filter and calculate mean

    cap_list.append(data.data/100000)

    if len(cap_list) < 20 :

   capacitance = cap_list[-1]

    else :

       # Savitzky Golay filter for continuous stream of data

       y = savgol_filter(cap_list, 15, 2)

       cap_list_mean.append(y[-1])

       capacitance = y[-1]

       if len(cap_list_mean) == mean_size :
```

```
            mean_cap = np.mean(cap_list_mean)
                msg.capacitance = capacitance
                msg.mean = mean_cap
      pub_cap.publish(msg)
                rospy.loginfo("capacitance %s and mean %s" %
                    (msg.capacitance, msg.mean))

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('filter_cap', anonymous=True)
    rospy.Subscriber("chatter", Float64, callback)
    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

B.3 ROS Message

The ROS message format under **cap.msg** used to publish the processed data.

```
float64 capacitance

float64 mean
```

Appendix C

Circuit Diagram

C.1 Circuit diagram for the custom PCB

The circuit diagram for the PCB used in all the experiments.

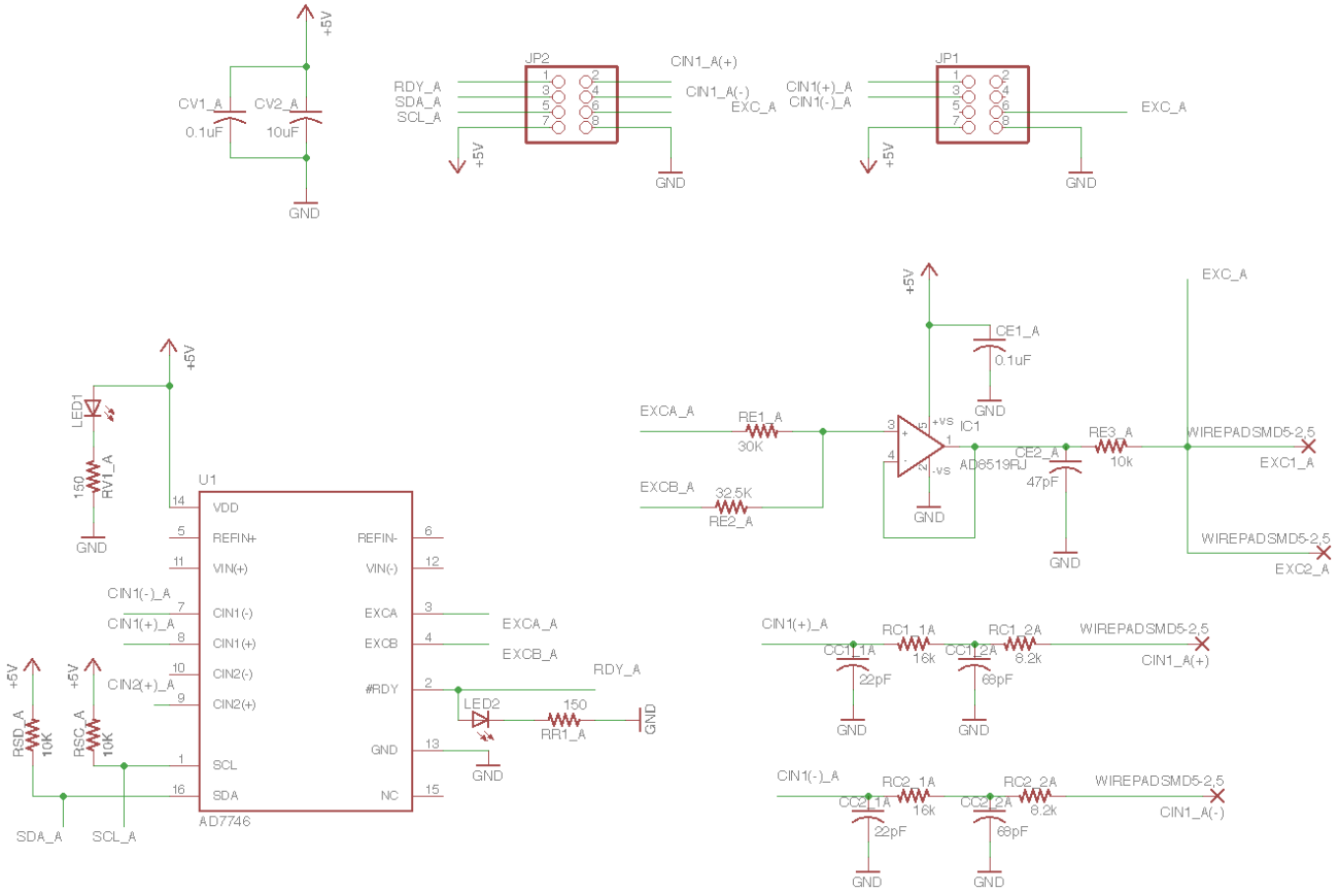


Figure C.1: Circuit diagram for the custom PCB

C.2 Circuit diagram proposed for the optimized PCB

The circuit diagram for the optimized PCB.

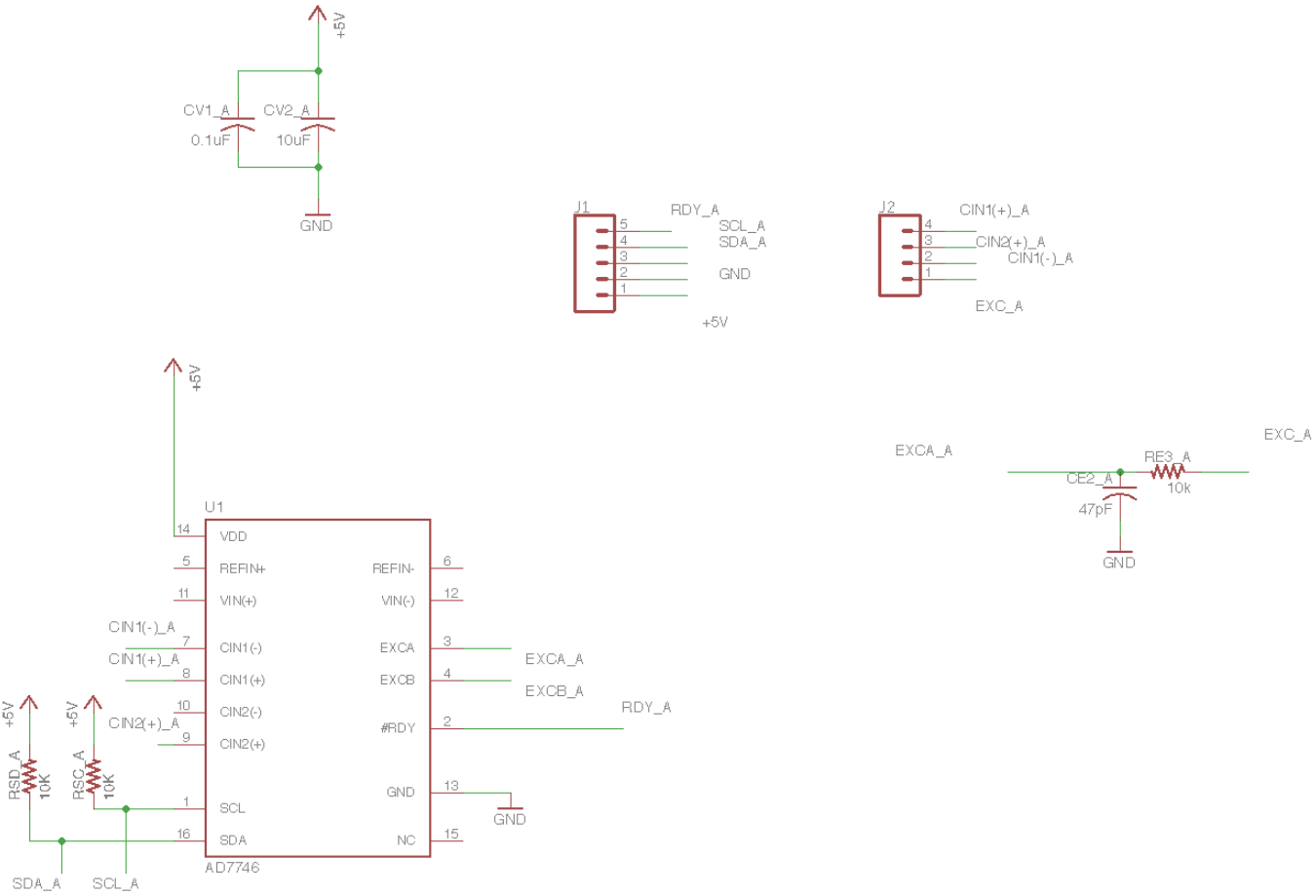


Figure C.2: Optimized circuit for the custom PCB

Bibliography

[1] Y. Hu, G. Lin, C. Yang, Z. Li and C. Y. Su, "Manipulation and grasping control for a hand-eye robot system using sensory-motor fusion," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, 2015, pp. 351-356.

[2] Danica Kragic, Mrten Bjrkman, Henrik I. Christensen, Jan-Olof Eklundh, Vision for robotic object manipulation in domestic settings, Robotics and Autonomous Systems, Volume 52, Issue 1, 2005, Pages 85-100,

[3] Goodwin, A. and Wheat, H. (2004). Sensory signals in neural populations underlying tactile perception and manipulation. Annual Review of Neuroscience, 27(1), pp.53-77.

[4] Goodwin AW, Jenmalm P, Johansson RS. 1998. Control of grip force when tilting objects: effect of curvature of grasped surfaces and applied tangential torque. J. Neurosci. 18:1072434

[5] Birznieks I, Jenmalm P, Goodwin AW, Johansson RS. 2001. Encoding of direction of fingertip forces by human tactile afferents. J. Neurosci. 21:822237

[6] F. L. Hammond, Y. Meng and R. J. Wood, "Toward a modular soft sensor-embedded glove for human hand motion and tactile pressure measurement," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, 2014, pp. 4000-4007.

[7] E. S. Hwang, J. h. Seo and Y. J. Kim, "A Polymer-Based Flexible Tactile Sensor for Both Normal and Shear Load Detections and Its Application for Robotics," in Journal of Microelectromechanical Systems, vol. 16, no. 3, pp. 556-563, June 2007.

[8] Viry, L., Levi, A., Totaro, M., Mondini, A., Mattoli, V., Mazzolai, B. and Beccai, L. (2014), Flexible Three-Axial Force Sensor for Soft and Highly Sensitive Artificial Touch. Adv. Mater., 26: 26592664.

[9] Kabelitz, D. (2018). K3D40 3-Axis force sensor - ME-Systeme. [online] Me-systeme.de. Available at: https://www.me-systeme.de/shop/en/sensors/force-sensors/k3d/k3d40 [Accessed 30 Jan. 2018].

[10] DualM studio - http://www.dualm.com - web site design development, B. (2018). 3AXX 3-Axis Load Cells. [online] Interfaceforce.com. Available at: http://www.interfaceforce.com[Accessed 30 Jan. 2018].

[11] Small 3-axis Force Sensor (Three force components, shear force measurement) - Tec Gihan (Japan). [online] Available at: http://www.tecgihan.co.jp/en/products/3axis-force-sensor/ [Accessed 30 Jan. 2018].

[12] Inc., S. (2018). SynTouch, Inc.. [online] Syntouchinc.com. Available at: https://www.syntouchinc.com/en/ [Accessed 28 Feb. 2018].

[13] Ltd., O. (2018). 3D force sensor applied in field of robotic sensors - OptoForce. [online] Optoforce.com. Available at: https://optoforce.com/3d-force-sensor-omd [Accessed 30 Jan. 2018].

[14] Torres-Jara, Eduardo and Natale, Lorenzo. (2018). Sensitive Manipulation: Manipulation Through Tactile Feedback. International Journal of Humanoid Robotics. 15. 1850012. 10.1142/S0219843618500123.

[15] Charalambides, A. and Bergbreiter, S. (2015). A novel all-elastomer MEMS tactile sensor for high dynamic range shear and normal force sensing. Journal of Micromechanics and Microengineering, 25(9), p.095009.

[16] Charalambides, A. and Bergbreiter, S. (2017). Tactile Sensors: Rapid Manufacturing of Mechanoreceptive Skins for Slip Detection in Robotic Grasping (Adv. Mater. Technol. 1/2017). Advanced Materials Technologies, 2(1).

[17] Robotiq. (2018). 2-Finger Adaptive Robot Gripper - Robotiq. [online] Available at: https://robotiq.com/products/2-finger-adaptive-robot-gripper [Accessed 16 Mar. 2018].

[18] RightHand Labs — Robotic Grippers — Tactile Sensors. (2018). RightHand Labs — Robotic Grippers — Tactile Sensors. [online] Available at: https://www.labs.righthandrobotics.com/reflexhand [Accessed 16 Mar. 2018].

[19] Agarwal and Bergbreiter, Measurement of shear forces during gripping tasks with a low-cost tactile sensing system IROS 2018. Manuscript submitted for publication.

[20] H. S. Shin, A. Charalambides, I. Penskiy and S. Bergbreiter, "A soft microfabricated capacitive sensor for high dynamic range strain sensing," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, 2016, pp. 5572-5578.

[21] Analog.com. (2018). AD7746 Datasheet and Product Info — Analog Devices. [online] Available at: http://www.analog.com/en/products/analog-to-digital-converters/integrated-special-purpose-converters/capacitive-to-digital-and-touch-screen-controllers/ad7746.html product-overview [Accessed 16 Mar. 2018].

[22] Basics, C. and Basics, C. (2018). Basics of the I2C Communication Protocol. [online] Circuit Basics. Available at: http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ [Accessed 20 Mar. 2018].

[23] Analog Devices (2010). AN-1055: EMC Protection of the AD7746. Analog Devices, Inc.

[24] Wiki.ros.org. (2018). rosserial - ROS Wiki. [online] Available at: http://wiki.ros.org/rosserial [Accessed 17 Mar. 2018].

[25] Amphenol FCI Clincher Connector (3 Position, M. (2018). Amphenol FCI Clincher Connector (3 Position, Male) - COM-14197 - SparkFun Electronics. [online] Sparkfun.com. Available at: https://www.sparkfun.com/products/14197 [Accessed 23 Mar. 2018].

[26] robots.ros.org. (2018). robots.ros.org. [online] Available at: http://robots.ros.org/ [Accessed 21 Mar. 2018].

[27] Ros.org. (2018). ROS packages. [online] Available at: http://www.ros.org/browse/list.php [Accessed 13 Apr. 2018].

[28] Wiki.ros.org. (2018). ROS/Concepts - ROS Wiki. [online] Available at: http://wiki.ros.org/ROS/Concepts [Accessed 18 Mar. 2018].

[29] Scipy-cookbook.readthedocs.io. (2018). Savitzky Golay Filtering SciPy Cookbook documentation. [online] Available at: http://scipy-cookbook.readthedocs.io/items/SavitzkyGolay.html [Accessed 17 Mar. 2018].

[30] Yin, J., Santos, V. and Posner, J. (2017). Bioinspired flexible microfluidic shear force sensor skin. Sensors and Actuators A: Physical, 264, pp.289-297.

[31] C. Heyer, "Human-robot interaction and future industrial robotics applications," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, 2010, pp. 4749-4754.

[32] Sadun, A.S. and Jalani, Jamaludin and Sukor, J.A.. (2016). An overview of active compliance control for a robotic hand. 11. 11872-11876.