ROBOTIC GRIPPER

WITH PRESSURE SENSOR

ENM2104 Instrumentation and Measurement

William, TRAN
10364208
phtran@our.ecu.edu.au
03/06/2016

Executive Summary

For this project, myself, William Tran [10364208] and colleague, Aldo Mado [10415642] have designed a grip-pressure system to provide pressure feedback when actuating our robotic gripper. This gripper is a 3D-printed prototype for our involvement in the 2016 'Autonomous Robotics Competition' (ARC) hosted by National Instruments (NI). The development of our grip-pressure system utilises an Arduino UNO board, a hi-torque servo and an FSR or 'Force Sensitive Resistor' (pressure sensor) as our hardware configuration. For software, the LabVIEW IDE was utilised to program our software configuration, including the PID (Proportional-Integral-Derivative) controller and soft sensor model.

Taking inspiration from the LabVIEW NI ELVIS II hardware and 'QNET Mechatronics Sensor Trainer' software utilised in the instrumentation labs. The design of our DAQ system incorporated industry methods of electrical and measurement analysis to develop formal technique in prototyping an efficient, accurate and robust grip-pressure system. The system was able to produce a maximum of 196.2Pa of pressure. The limitation of applying further pressure was due to how much power the Arduino could provide. This can easily be mitigated by using a relay to power the gripper separately.

The name of 'FSR' is misleading as it is not a 'force' sensor, but a pressure sensor that directly decreases in resistance when applying pressure on its sensing area. This pressure can be interpreted as Pascal (Pa) or pound per square inch (psi). However, for making our program mathematically simple and process efficient we will refer to pressure as the mass in 'grams' with varying signal from the FSR sensor.

The link provided below directs to our YouTube demonstration video, illustrating our completed grip-pressure system in operation.

https://www.youtube.com/watch?v=metdbaOA8Kc

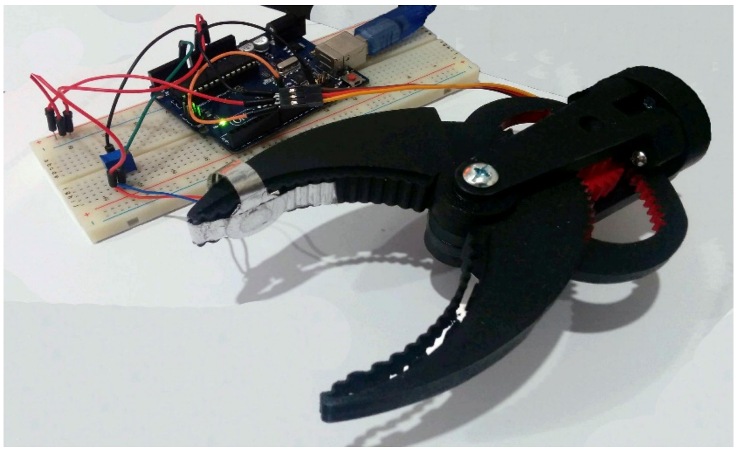


TABLE OF CONTENTS

TABLE OF FIGURES

1. Introduction

1.1 Description of the Project

The goal of our project is to monitor and control the pressure applied to a target object by a robotic gripper. An FSR (pressure sensor) will be utilised to obtain a signal of the applied gripping pressure. This signal will be acquired by an 'Arduino UNO' microcontroller for DAQ processing. The Arduino will filter the signal and actuate the servo in a controlled manner to either increase or decrease the pressure on the object. In essence, the project's goal is to effectively grip and firmly hold an object in a controlled, efficient and robust manner. Refer to **Figure 1**, illustrating the basic structure of our DAQ system.

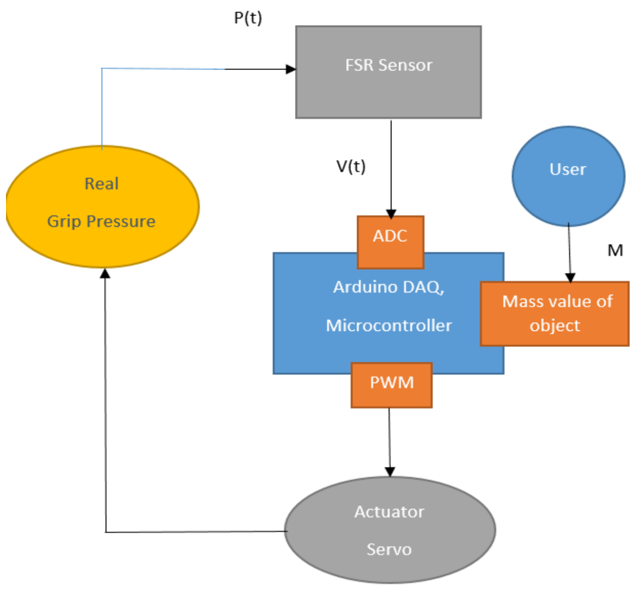


FIGURE 1 DAQ SYSTEM FLOW CHART

2. Implementation

2.1 Abstracting the Problem

The FSR must produce a voltage signal between 0-5V as an analog signal to be used for pressure calculations. From these calculations, the Arduino will produce a digital pulse-width-modulation (PWM) signal for outputting servo controls. A single degree of servo rotation will correspond to ~16.67µs (microseconds) to actuate the gripper from fully-opened (2500µs) to fully-closed (1000µs) (National Instruments, 2016).

For sampling the FSR signal, the Arduino will produce a maximum sampling rate of ~9.6kHz derived from research done for the Arduino's 'Analog to Digital' Converter (ADC). Refer to equations **(1)** and **(2),** illustrating the derived maximum sampling rate of ~9.6KHz.

$$f_{CLK} = \frac{ADC_{CLK}}{Prescale\ Factor} = \frac{16 \times 10^6\ Hz}{128} = 125 \times 10^3\ Hz \qquad (\mathbf{1})$$

The maximum sample rate is then the division of the ADC clock frequency by 13, as the ADC reads the analog pins every 13 cycles.

$$Sample\ rate_{max} = \frac{f_{CLK}}{ADC_{cycle}} = \frac{125 \times 10^3\ Hz}{13} \approx 9600\ Hz \qquad (\mathbf{2})$$

However, only a maximum of 200Hz will be needed for reading the FSR signal as anything higher than 200Hz results in additional noise. The resolution for our Arduino will be 8 bits (255 values) of resolution, determined by 'NI MAX (Measurement & Automation) Hardware' explorer. Therefore, the voltage increments resolvable by the Arduino will be of ~20mV increments. Refer to equation **(3)** used to calculate the voltage increment.

$$\varepsilon_v = \frac{\Delta V_{FS}}{2^n - 1} = \frac{5-(0)\,V}{2^8-1} = \frac{5V}{255} \approx 0.0196V \qquad (\mathbf{3})$$

2.2 Division of Tasks

The project tasks were divided up between myself (William Tran) and my colleague Aldo Mado.

Allocated tasks include:

1. **Design and construct robotic gripper.** – Both Aldo and William.
2. **Research FSR sensors.** – Aldo.
3. **Hardware implementation.** – Both Aldo and William.
4. **Programming.** – William.
5. **Write the report.** – An individual task.

2.3 Problems Faced

The original code had to be redesigned to incorporate a PID controller for sensing and control feedback. The PID mitigated dynamic response errors of pressure readings. The errors in pressure reading would accumulate resulting in an overshooting of the output PWM signal. This caused the servo to oscillate the gripper from applying too much pressure to too little pressure, such that the object would fall off. Refer to **Figure 2,** comparing two versions of our DAQ system; one with and the other without the PID control.

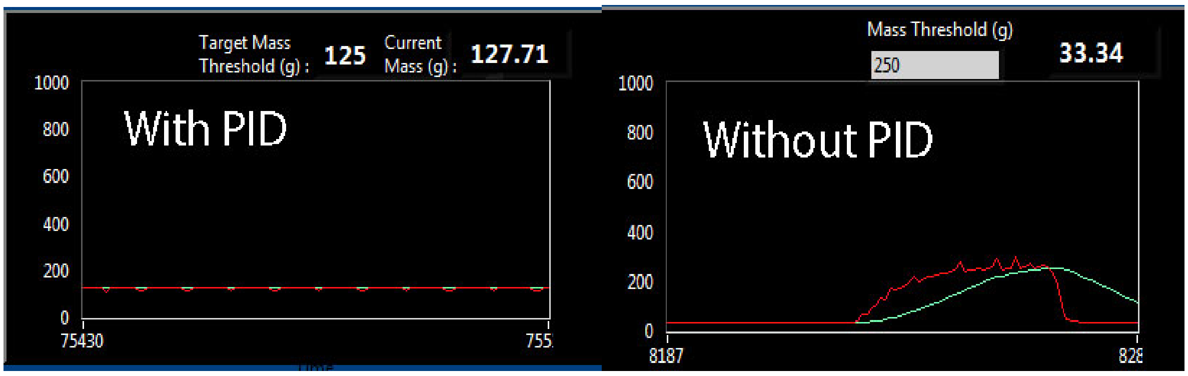


FIGURE 2 PID WITH AND WITHOUT COMPARISON

3. Sensor Selection

3.1 Identification of the Sensor Type

The FSR sensor would best suit our prototype as the gripper does not require precision pressure measurements associated with other load cells, and strain gauges. Our model, 'FSR-PRT402E' requires a 100KΩ pull-down resistor in a voltage divider circuit to attain Arduino's full operating voltage range of 0-5V without further amplification **(Interlink Electronics, 2016).**

'Force Sensitive Resistor'(FSR) does not vary resistance to directly measure force, contrary to the name. Instead, the 'applied pressure' to the sensing area will vary the resistance to produce a signal. With no pressure, the output resistance will be larger than 1MΩ, essentially an open switch. When pressure is applied, the resistance decreases, allowing increased conduction.

Figure 3 and 4 (Interlink Electronics, 2016), from the manufacturer, illustrate this linear response behaviour with a 'resistance vs. mass' graph, as well as a 'conduction vs. mass' graph, respectively. Our particular FSR sensor 'FSR-PRT402E' from 'Interlink Electronics' has a diameter of 12.7mm or ~127mm² sensing area**.** A mass can be measured anywhere in the range of 100g~10kg **(Interlink Electronics, 2016)**. For this project, a range of 100g~ 500g will be sufficient to detect and firmly hold various light-weight objects.

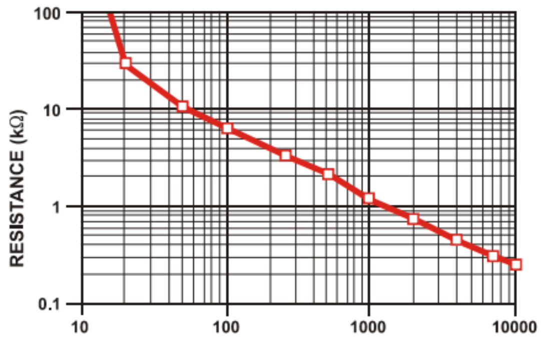


FIGURE 3 FSR RESISTANCE VS. MASS

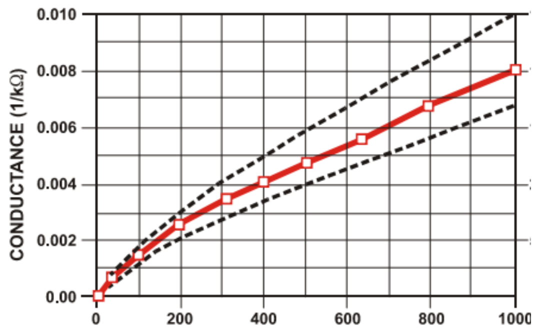


FIGURE 4 FSR CONDUCTANCE VS. MASS

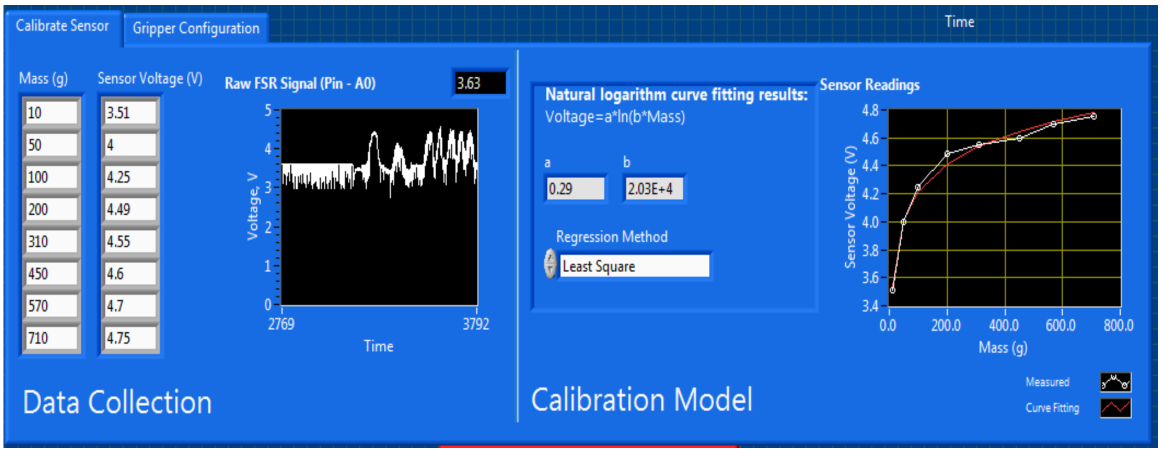
3.2 Performance Test of the Sensor

Table 1 shows a collated table of different mass values used to pressure test and calibrate the FSR sensor. The weight of the object is measured first by a kitchen scale to determine its mass value. Then, it is placed on to the FSR sensing area to read the corresponding voltage.

Mass(m), kg	Force (ma), N a = 9.81 m/s²	Sample Rate, Hz Max (200Hz)	Pull-down Resistance, kΩ	Measured Output Voltage, V	Manufacturer Output Voltage, V	Average Difference, V
0.1	0.98	170	100	2.43	4.25	
0.2	1.96	170	100	3.74	4.4	
0.35	3.43	170	100	4.25	4.6	
0.5	4.91	170	100	4.35	4.7	0.8

TABLE 1 FSR PERFORMANCE TEST AND COMPARSION

The FSR output voltage is entered into our program along with the corresponding mass value for calibration. Refer to **Figure 5,** illustrating our program's FSR calibration panel for the user to enter the values. The entered values in the array are then plotted and a regression line method such as 'least squares' is fitted to aid in the process of finding an equation that best models our FSR sensor mathematically. This sensor model is then used as our 'soft sensor' for pressure feedback.



Mass (g)	Sensor Voltage (V)
10	3.51
50	4
100	4.25
200	4.49
310	4.55
450	4.6
570	4.7
710	4.75

Data Collection

Natural logarithm curve fitting results:
Voltage=a*ln(b*Mass)

a: 0.29
b: 2.03E+4

Regression Method: Least Square

Calibration Model

FIGURE 5 FSR CALIBRATION PANEL

The performance comparison was done using the graph in **Figure 6**. This graph shows the manufacturers' test results as well as demonstrating the logarithmic relationship of the voltage and mass described by equation **(4).**

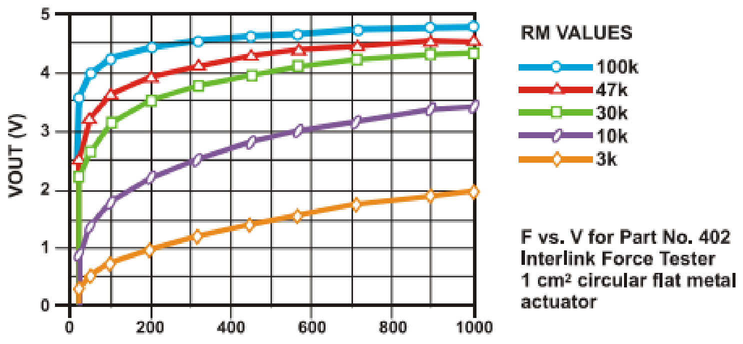


FIGURE 6 VOLTAGE VS. MASS GRAPH FROM MANUFACTURER

$$Voltage = a \times \ln(b \times Mass) \qquad (\mathbf{4})$$

The constants, 'a' and 'b' in **(4)** represent the amplitude and scale, respectively, to correctly model and associate voltage reading to standard mass measurements. The performance of our FSR sensor produced lower values than the manufacturers' results. The reason may be due to adding a rubber-pad in-between the weight and sensing area for better contact with object. The pad has a larger diameter (~15mm) than the FSR sensing area (~12.7mm). Therefore, reducing the pressure, which proportionally reduces the voltage signal. Refer to equation (**5)**, illustrating this relation.

$$Pressure = \frac{Force}{Area} \qquad (\mathbf{5})$$

4. Measurement System Implementation

4.1 Hardware Configuration

The hardware includes a 3D-printed robotic gripper that is actuated by a hi-torque digital servo motor (MG996R). For this project, the servo motor operates at 4.8~5V.

'Analog In' signal pin 'A0' on the Arduino will be attached to the FSR sensor in a voltage divider circuit with a 10KΩ pull-down resistor. For output, the 'Digital (PWM~)' pin '9' will be attached to a servo motor for gripper actuation. Refer to the circuit schematic in **Figure 7,** detailing the various components to be implemented in hardware.

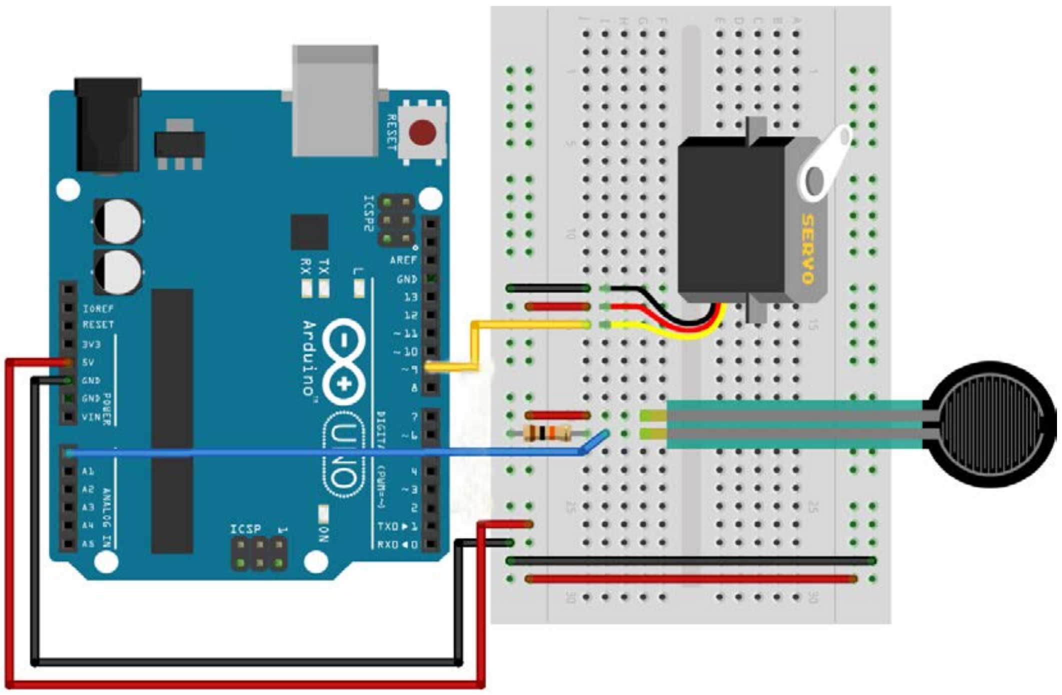
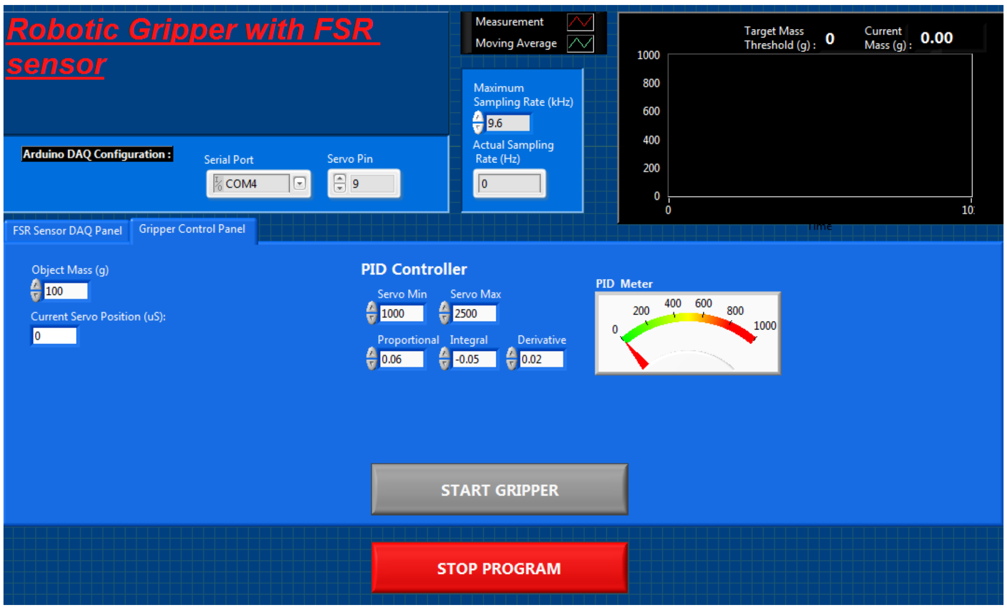


FIGURE 7 HARDWARE DAQ SYSTEM SCHEMATIC

4.2 Software Configuration

The LabVIEW IDE is used to compile and code the Arduino UNO board for DAQ and control operations. **Figure 8**, illustrates the front panel GUI (Graphical User-Interface) for the VI (Virtual Instrument) code. Refer to **Appendix 1** detailing the VI code behind the GUI with an included download link to the VI code.



Robotic Gripper with FSR sensor

Measurement
Moving Average

Maximum Sampling Rate (kHz): 9.6
Actual Sampling Rate (Hz): 0

Target Mass Threshold (g) : 0
Current Mass (g) : 0.00

Arduino DAQ Configuration :
Serial Port: COM4
Servo Pin: 9

FSR Sensor DAQ Panel | Gripper Control Panel

Object Mass (g): 100
Current Servo Position (uS): 0

PID Controller
Servo Min: 1000
Servo Max: 2500
Proportional: 0.06
Integral: -0.05
Derivative: 0.02

PID Meter

START GRIPPER
STOP PROGRAM

FIGURE 8 MAIN VI CODE GUI

For sensor processing, the elements in our VI code are inspired by the ‘QNET Mechatronics Sensor Trainer’ utilised in the instrumentation labs. The VI, ’07_QNET_MECHKIT_Optical.vi’ provided by NI ELVIS II was used as a reference model for our DAQ system’s calibration panel in **Figure 5,** as noted. In addition, the FSR signal will also be filtered by calculating a moving average of the signal to reduce noise.

For gripper feedback and control operations, the PID controller ‘simplepid.vi’ is directly from the NI community site created by DAQTunes (DAQTunes, 2009). The PID controller is utilised to minimise overshoot errors when controlling the gripper to move closer and closer to the calculated pressure threshold. The desired threshold is calculated by using the 'user-entered' mass of the object plus 25% of that mass to firmly hold the target.

5. Conclusion

The outcome of the project was successful in meeting the goals of monitoring and controlling the pressure applied to a target object using our gripper prototype. The robotic gripper was able to apply up to ~300g of mass or 196.2Pa of pressure calculated in **(6).** This is a little lower than our initial design of up to 500g, which may be limited by how much power the Arduino can provide as our servo should be able to produce pressure above 327Pa (~500g).

$$Pressure = \frac{Mass \times g}{\pi r^2} = \frac{0.3 \times 9.81}{\pi \times 0.015^2} = 196.2\text{Pa} \qquad \textbf{(6)}$$

The result of our grip-pressure system can be seen in **Figure 9**. Our robotic gripper in **Figure 9** is regulating the applied pressure on the target object to approach the pressure threshold.

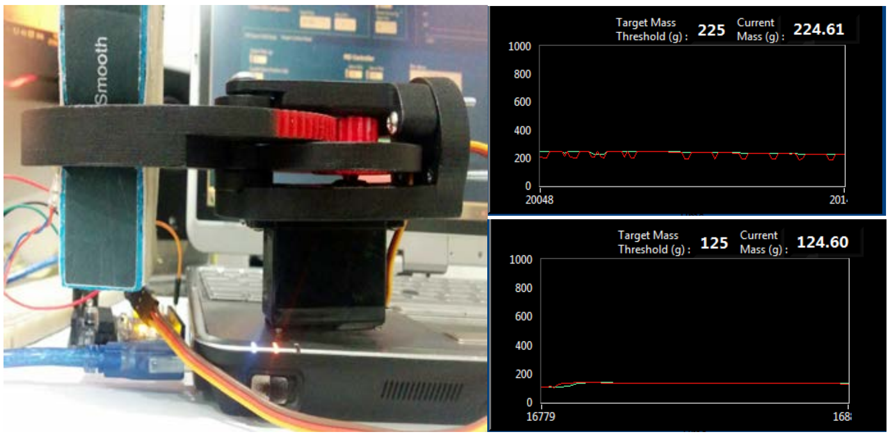


FIGURE 9 PRESSURE REGULATION RESULTS

In summary, through the completion of this project, we have developed formal techniques in electrical and measurement analysis. As well as, knowledge in designing DAQ systems to meet our goals of operating our gripper in an accurate, efficient and robust manner.

6. Appendices

Appendix 1

The secured download link provided below directs to a 'zip.' file that contains all the LabVIEW VI codes needed to run this project.

Download Link: http://www.evernote.com/l/AV7XqIErkZFJUZKnzwI8vliXeN4aI6q4PPQ/

The software dependencies required to run the VI code, includes:

- LabVIEW Service Pack 1 2015.
- LabVIEW Mathscript Node RT 2015 Module.
- LINX 2015 (3rd party) module for serial communication with the Arduino UNO board.
- LabVIEW Control Design and Simulation 2015 Module.

<Mathscript M-Code>

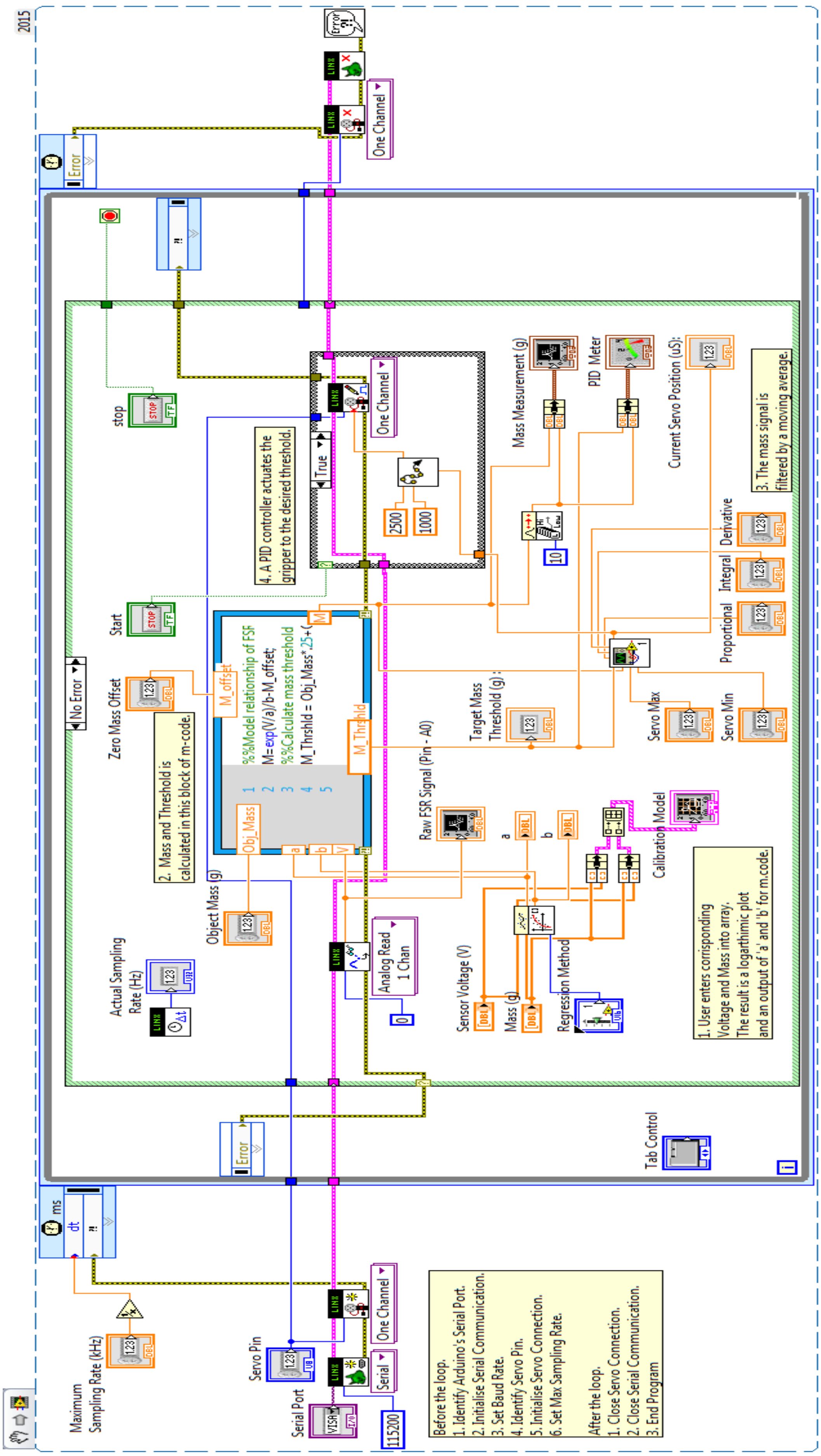
```
%%Model relationship of FSR voltage to mass.

M=exp(V/a)/b-M_offset;

%%Calculate mass threshold to apply on object.

M_Thrshld = Obj_Mass*.25+Obj_Mass;
```

<Arduino Main VI Code>



Before the loop.
1. Identify Arduino's Serial Port.
2. Initialise Serial Communication.
3. Set Baud Rate.
4. Identify Servo Pin.
5. Initialise Servo Connection.
6. Set Max Sampling Rate.

After the loop.
1. Close Servo Connection.
2. Close Serial Communication.
3. End Program

1. User enters corrisponding Voltage and Mass into array. The result is a logarthimic plot and an output of 'a' and 'b' for m.code.

2. Mass and Threshold is calculated in this block of m-code.

```
%%Model relationship of FSF
M=exp(V/a)/b-M_offset;
%%Calculate mass threshold
M_Thrshld = Obj_Mass*.25+(
```

3. The mass signal is filtered by a moving average.

4. A PID controller actuates the gripper to the desired threshold.

5 References

Arduino. (2016, May 14). *Arduino UNO* . Retrieved from Arduino: https://www.arduino.cc/en/Main/ArduinoBoardUno

DAQTunes. (2009, nov 30). *USB 6211 Application Code*. Retrieved from National Instruments - Community: https://decibel.ni.com/content/docs/DOC-8355

Interlink Electronics. (2016). *FSR Force Sensing Resistor.* Camarillo: Interlink Electronics.

National Instruments. (2016, 05 18). *LabVIEW Interface for Arduino* . Retrieved from National Instruments: https://decibel.ni.com/content/groups/labview-interface-for-arduino?view=discussions

National Instruments. (2016, 05 21). *Servo Servo Set Pulse Width N Channels*. Retrieved from LabVIEW MakerHUB: https://www.labviewmakerhub.com/doku.php?id=learn:libraries:linx:reference:labview:servo:servo-set-pulse-width-n-channels

Servo Database. (2016, May 14). *TowerPro MG996R Servo*. Retrieved from ServoDatabase.com: http://www.servodatabase.com/servo/towerpro/mg996r

Toms, L. (2015, 3 5). *The Arduino DAQ Chronicles*. Retrieved from Measuring Stuff Blog: https://sites.google.com/site/measuringstuff/the-arduino