

Prototype Development Using LSM303, Stepper Motor and ADC Module on Raspberry Pi 4

Amiraj Nigam, MS.

Electrical Engineering Department, College of Engineering

San Jose State University, San Jose, CA 94303

E-mail: amiraj.nigam@sjsu.edu

Abstract

To build a prototype system using Raspberry Pi, PWM motor drive, NEMA stepper motor, LSM303 sensor and an ADC module with a potentiometer. Additionally, to operate this prototype system by changing the output to the ADC input through potentiometer. The stepper motor will operate at different duty cycle and the stepper motor rotation is recognized and recorded by LSM303 accelerometer and Magnetometer. Implementation of FFT to perform validation of data and efficiency is calculated through power spectrum analysis. Verification is done through Nyquist criteria $F_s \geq 2F_m$.

1. Introduction

The project should drive the stepper motor depending on the input from the potentiometer and the ADC module. It simulates a sensor that provides an input to the open loop system, which is stepper motor in our case. We can see the overall implementation as a closed loop system as the LSM303, three axis accelerometer + magnetometer + temperature sensor provides a feedback for the position of the stepper motor rotation.

The data from the ADC need to be validated. So, compensation function is calculated for the validation and power spectrum is plotted.

$$f(x) + g(x) = ax + b \dots (1)$$

Where $ax + b$ is the ideal linear characteristics of the ADC, $f(x)$ is the actual ADC function (piece wise linear), and $g(x)$ is the compensation function to be designed. Note $b = 0$, and a is a slope for the ADC, for 12 bits ADC, $a = 4096/3.3$, $f(x)$ is from the ADC data characteristic curve and $g(x)$ is the compensation function to be designed.

So $f(x) + g(x) = ax$ and Based on equation (2), we can write $f(x)$ as $f(x) = px + q \dots (3)$

p and q can be found from the actual experiment.

Substitute eqn(3) back to (1),

we have

$$px + q + g(x) = ax$$

$$px + q + a_g x + b_g = ax$$

We shall keep the sampling frequency in accordance to the Nyquist theory as follows:

$$f_{\text{sampling}} \geq 2 f_{\text{max}}$$

2. Methodology

This section shows the systematic implementation of the prototype.

2.1. Objectives and Technical Challenges

The main objective is to drive the NEMA stepper motor using the variable inputs from the ADC module, which is connected to the potentiometer. The duty cycle is changed according to change from the value of potentiometer.

The challenges that I have been through this project are mainly how to interface the sensor module and the ADC module together on the same I2C interface, validation of data from the ADC and taking a corrective action by increasing the sampling rate was also a tedious job. Driving the stepper motor because the driver module is prone to EMI and so it performs worse when in the proximity of the controller board or the power supply. Continuously monitoring ADC output and thus making the motor spin in desired direction was a challenging job but this overall experience enhanced my knowledge about the sensors and the master and slave communication used in embedded systems.

2.2. Problem Formulation and Design

The PWM motor driver and ADC are used to drive the NEMA stepper motor and LSM303 is used to measure the direction and acceleration of the stepper motor respectively. The LSM sensor is mounted on the motor itself so that the movement of the motor can be noted. The direction and the speed (acceleration) are the two main concerns in the project.

We need a stepper motor drive to drive the motor. The PWM output actuates the drive and then drives the motor. The ADC generally does not have one to one mapping of voltage and digital data practically. To have one to one mapping we use potentiometer. By rotating the meter, we can bring down or bring up the value to zero and note the corresponding digital value.

3. Implementation

The hardware and software designs are shown in this section.

3.1. Hardware Design

List of components:

- 1) Raspberry Pi 3B+
- 2) LSM 303
- 3) ADS 1015
- 4) NEMA 17 Stepper motor
- 5) Stepper motor drive module
- 6) Connecting Wires
- 7) PCB

System Block Diagram:

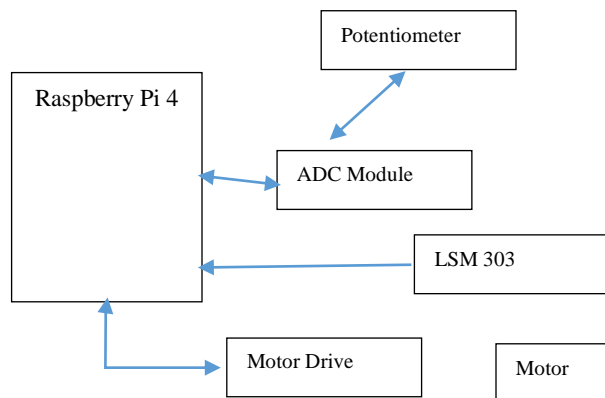


Fig1. Block Diagram of LSM303, ADC, Stepper Motor and driver integrated with Raspberry Pi 4

Prototype Integration Picture:

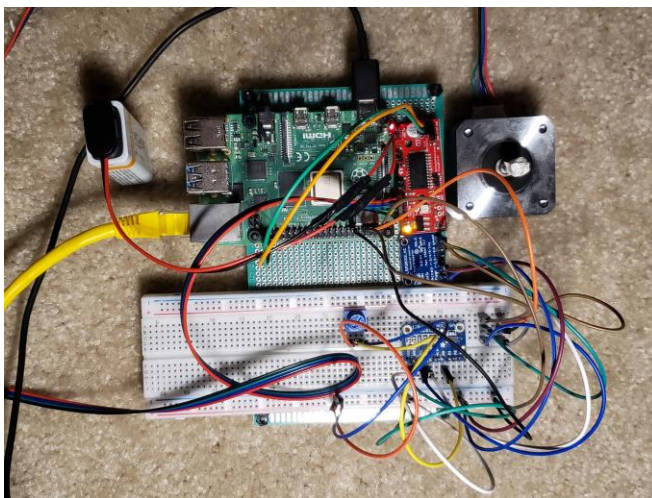


Fig2. LSM303, ADC, Stepper Motor and driver integrated with Raspberry Pi 4

Schematic Diagram:

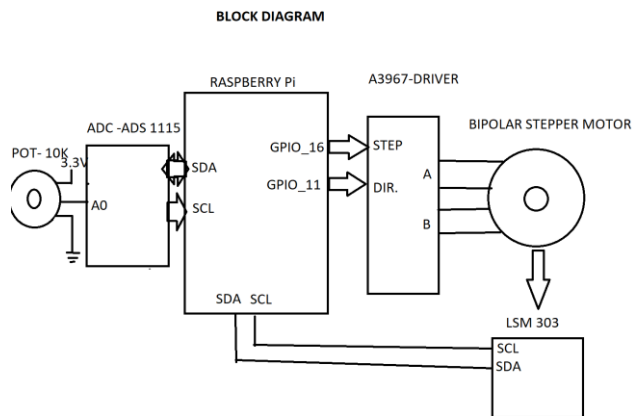


Fig3. Schematic Pin Diagram

3.2 Interfacing and Device detail

3.2.1 ADC ADS 1015

For microcontrollers without an analog-to-digital converter or when you want a higher-precision ADC, the ADS1015 provides 12-bit precision at 3300 samples/second over I2C. The chip can be configured as 4 single-ended input channels, or two differential channels. As a nice bonus, it even includes a programmable gain amplifier, up to x16, to help boost up smaller single/differential signals to the full range. We like this ADC because it can run from 2V to 5V power/logic, can measure a large range of signals and its super easy to use. It is a great general purpose 12 bit converter.

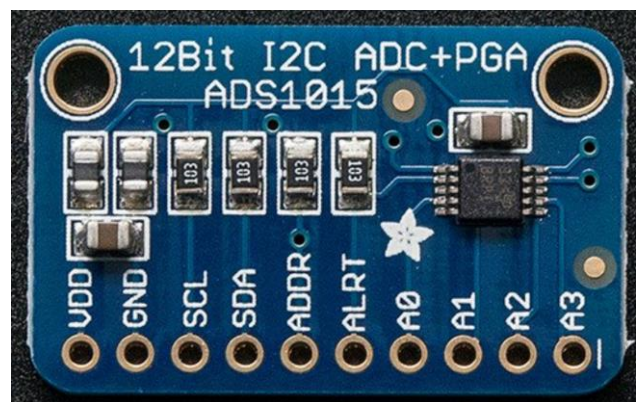


Fig4. 12 bit ADC ADS 1015

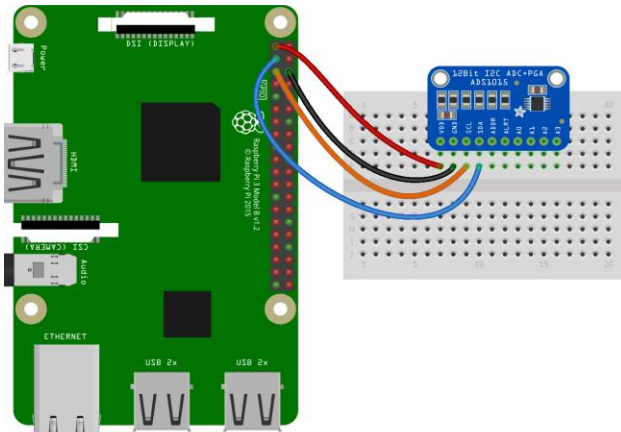


Fig 5. ADC Interfacing with Raspberry Pi 4

Connect the ADC to the Pi as follows:

ADS1x15 VDD to Raspberry Pi 3.3V

ADS1x15 GND to Raspberry Pi GND

ADS1x15 SCL to Raspberry Pi SCL

ADS1x15 SDA to Raspberry Pi SDA

3.2.2 Easy Driver

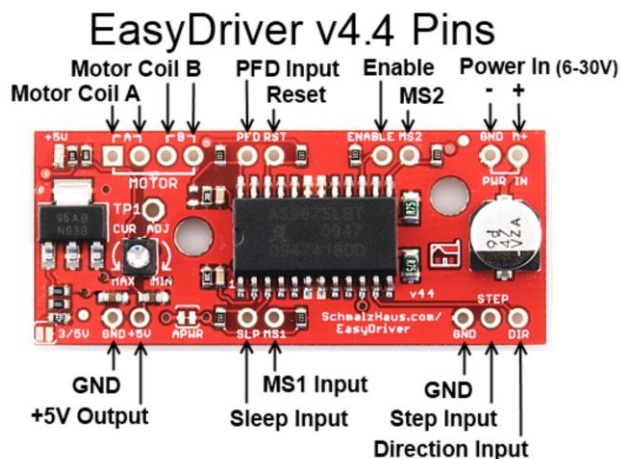


Fig6. Easy driver for NEMA 17 stepper motor

Each Easy Driver can drive up to about 750mA per phase of a bi-polar stepper motor. It defaults to 8 step micro stepping mode. (So if your motor is 200 full steps per revolution, you would get 1600 steps/rev using Easy Driver.) This setting can be easily overridden by tying the MS1 and/or MS2 pin to ground to set the driver to use 1/8, 1/4 or 1/2 micro step mode (See the datasheet for the table of values). It is a chopper micro stepping driver based on the Allegro A3967 driver chip. For the complete specs of the design, read the A3967 datasheet. It has a variable max current from about 150mA/phase to 750mA/phase. It can take a maximum motor drive voltage of around 30V, and includes on-board 5V regulation, so

only one supply is necessary.

3.3.3 LSM 303 3 Axis Accelerometer



Fig 7. LSM303 Accelerometer and Magnetometer.

The LSM303DLHC. This compact sensor uses I2C to communicate and its very easy to use. Since it's a 3.3V max chip, we added circuitry to make it 5V-safe logic and power, for easy use with either 3 or 5V microcontrollers. Simply connect VCC to +3-5V and ground to ground. Then read data from the I2C clock and data pins. There's also a Data Ready and two Interrupt pins you can use (check the LSM303 datasheet for details)

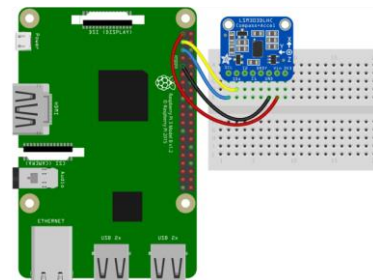


Fig 8. LSM 303 interfacing with Raspberry Pi 4

- Pi 3V3 to sensor VIN
- Pi GND to sensor GND
- Pi SCL to sensor SCL
- Pi SDA to sensor SDA

3.3.4 Voltage Potentiometer

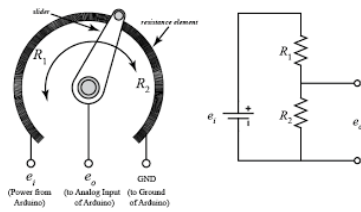


Fig 9. Potentiometer

The potentiometer is a manually adjustable variable resistor with 3 terminals. Two terminals are connected to both ends of a resistive element, and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element. The position of the wiper determines the output voltage of the potentiometer

3.3.5 Stepper motor



Fig 10. NEMA 17 Stepper Motor

This hybrid bipolar stepping motor has a 1.8° step angle (200 steps/revolution). Each phase draws 1.7 A at 2.8 V, allowing for a holding torque of 3.7 kg-cm (51 oz-in). The motor has four color-coded wires terminated with bare leads: black and green connect to one coil; red and blue connect to the other. It can be controlled by a pair of suitable H-bridges.

3.3. Software Design

The software design is divided into two sections viz. User application program and Kernel space program.

The user application program first sets the GPIO port P0.4 as the output port which is used to decide the direction in which the motor should rotate. Secondly, the Potentiometer output is given to the analog input pin of the ADC module. There is a function which checks for the interrupt. Whenever there is an interrupt the buffer is ready and the digital value is stored into a temporary variable. The output of the ADC is then given as the frequency to the PWM. The PWM output is then given to

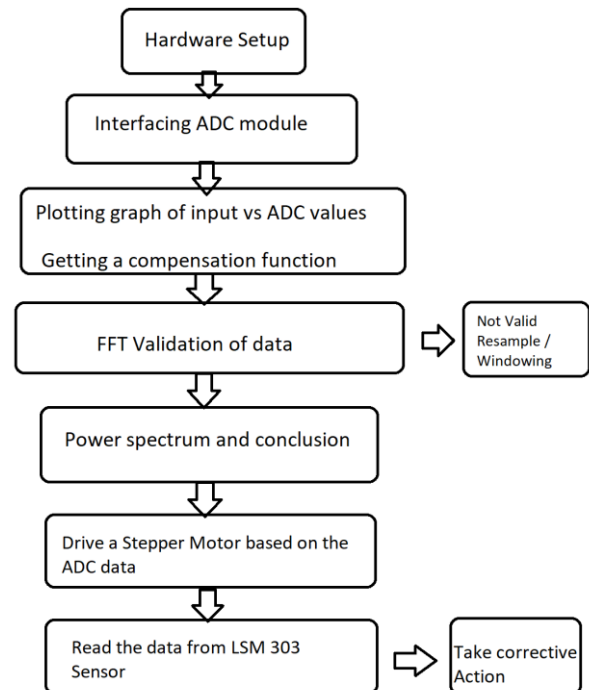
the motor driver STEP pin which controls the duty cycle of the motor.

The software development process was divided into two sections. User application program and Kernel space program.

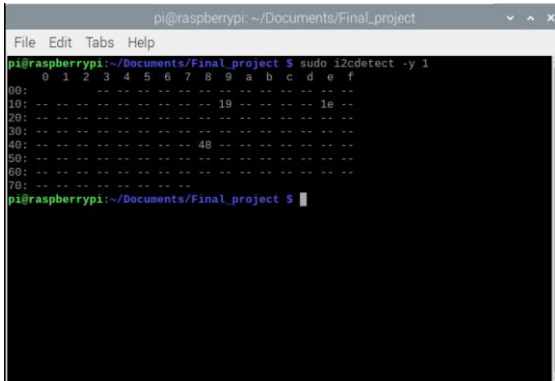
Algorithm:

1. Monitor the value in the ADC.
2. Change the value using potentiometer.
3. According to the change in the values, change the speed of the motor attached with the motor drive.
4. Change the direction of the motor.
5. Measure the speed and acceleration using LSM303.

Design Flowchart:



Detection of Slaves connected using I2C:



```
pi@raspberrypi: ~/Documents/Final_project
File Edit Tabs Help
pi@raspberrypi:~/Documents/Final_project $ sudo i2cdetect -y 1
0 1 2 3 4 5 6 7 8 9 a b c d e f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- 19 -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- 48 -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~/Documents/Final_project $
```

4. Testing and Verification

1)Input Voltage -Potentiometer

The potentiometer is a manually adjustable variable resistor with 3 terminals. Two terminals are connected to both ends of a resistive element, and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element. The position of the wiper determines the output voltage of the potentiometer.

Ethernet cable(LAN CAT5) is used as a serial communication between laptop and Raspberry pi via VNC on putty.

Code is deployed on the terminal and we simultaneously change the value of potentiometer value to vary the ADC digital output from minimum to maximum.

For verification and validation of data, we calculate power spectrum density. DFT equation:

$$X(m) = \frac{1}{N} \sum_{n=0}^{N-1} X(n) e^{-j^2 \pi \frac{mn}{N}}$$

In the above equation:

X(n) : ADC output over time n

X(m) : Discrete Fourier Transform.

m : Frequency index - varies from 0 to N-1

N : Total number of ADC outputs

From the above equation (1), m = 0 is the D.C component and m = (N/2)-1 is the highest frequency index.

Power spectrum density is calculated using the following equation

Power Spectrum $P(m) = \text{Re}^2[X(m)] + \text{Im}^2[X(m)]$ (2)
where

Re : Real part of X(m)

Im : Imaginary part of X(m)

P(m) : Power spectrum for a given frequency index m.

We will calculate the power spectrum through FFT and calculate the efficiency as per the above equation. If, efficiency is below the 80%, it means the data we have sampled is aliasing. So, to correct this we need to increase the frequency as per Nyquist criteria $F_s \geq 2F_{\text{max}}$ to get the correct sampled data. In this experiment we are getting 82.7% efficiency. This indicate, we have satisfied Nyquist criteria.

5.Code

```
import time
import board
import busio
import adafruit_lsm303_accel
import adafruit_lsm303dlh_mag
import Adafruit_ADS1x15
import pigpio
import numpy as np
import math
from math import atan2, degrees
```

```
DIR = 20 # Direction GPIO Pin
STEP = 21 # Step GPIO Pin
CW = 1 # Clockwise Rotation
CCW = 0 # Counterclockwise Rotation
MS0=14#Mode GPIO Pin
MS1=15#Mode GPIO Pin
```

```
pi = pigpio.pi()
```

```
# Set up pins as an output
pi.set_mode(DIR, pigpio.OUTPUT)
pi.set_mode(STEP, pigpio.OUTPUT)
pi.write(DIR,CCW) #set clockwise or anticlockwise
```

```
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_lsm303_accel.LSM303_Accel(i2c)
sensor1=
adafruit_lsm303dlh_mag.LSM303DLH_Mag(i2c)
adc = Adafruit_ADS1x15.ADS1015()
GAIN = 1
voltage=[]
```

```
def vector_2_degrees(x, y):
    angle = degrees(atan2(y, x))
    if angle < 0:
        angle += 360
    return angle
```

```
def get_heading(_sensor):
```

```

magnet_x, magnet_y, _ = _sensor.magnetic
return vector_2_degrees(magnet_x, magnet_y)

def get_inclination(_sensor):
    x, y, z = _sensor.acceleration
    return vector_2_degrees(x, z), vector_2_degrees(y, z)
MODE = (MS0,MS1) # Microstep Resolution GPIO
Pins
RESOLUTION = {'Full': (0, 0),
              'Half': (1,0),
              '1/4': (0,1),
              '1/8': (1, 1)
              }
for i in range(2):
    pi.write(MODE[i], RESOLUTION['Full'][i])
while True:
    for y in range(1648):
        print ('ADC channel 0 value:')
        print('| {0:>6} | {1:>6} | {2:>6} | {3:>6} |'.format(*range(4)))
        print('-' * 37)
        values = [0]*8
        for i in range(4):
            values[i] = adc.read_adc(i, gain=GAIN)
            if i>0:
                values[i]=0
        print('| {0:>6} | {1:>6} | {2:>6} | {3:>6} |'.format(*values))
        print("")
        time.sleep(0.5)
        v=(values[0]*3.3)/1647 #FOR GAIN=1, 2047 adc
counts=4.096v
        voltage.append(v)
        x= values[0]/1647; #1645=Maximum ADC counts
for 3.3v
        x=x*255;# 255=
        pi.set_PWM_dutycycle(STEP, x)
        pi.set_PWM_frequency(STEP, 500)
        acc_x, acc_y, acc_z = sensor.acceleration
        mag_x, mag_y, mag_z = sensor1.magnetic
        print('Acceleration
(m/s^2):({0:10.3f},{1:10.3f},{2:10.3f})'.format(acc_x,
acc_y, acc_z))
        angle_xz, angle_yz = get_inclination(sensor)
        print("XZ angle = {6.2f}deg YZ angle =
{:6.2f}deg".format(angle_xz, angle_yz))

        print("")
        time.sleep(1.0)
        print('Magnetometer
(gauss):({0:10.3f},{1:10.3f},{2:10.3f})'.format(mag_x,
mag_y, mag_z))
        print("heading:
{:2f}
degrees".format(get_heading(sensor1)))
        print("")
        time.sleep(1.0)

```

```

f = np.fft.fft(voltage)
power_spectrum = []
entire_energy = 0
highest_frequency_energy = 0
for i in range(0,1648):#1647 adc counts=3.3v
    power_spectrum.append(math.sqrt((f.real[i]
f.real[i])+(f.imag[i] * f.imag[i]))) *
    total_energy += power_spectrum[i]
    if(i==1023):#(N/2)-1
        highest_frequency_energy=power_spectrum[i]
eta = (highest_frequency_energy/(total_energy))*100
print("ADC Validation Efficiency:" + str(eta) + "%")
if eta>80 and eta <100:
    print("ADC data is valid")
else:
    print("ADC data is not valid")

```

6.Output

```

ADC channel 0 value:
| 0 | 1 | 2 | 3 |
-----
| 510 | 0 | 0 | 0 |

Acceleration (m/s^2):( 1.377, -2.524, 8.950)
XZ angle = 81.25deg YZ angle = 105.75deg

Magnetometer (gauss):( 98.636, -7.273, -417.959)
heading: 8.20 degrees

ADC channel 0 value:
| 0 | 1 | 2 | 3 |
-----
| 638 | 0 | 0 | 0 |

Acceleration (m/s^2):( 2.180, -2.639, 9.332)
XZ angle = 76.85deg YZ angle = 105.79deg

Magnetometer (gauss):( 97.364, 33.818, -417.959)
heading: 36.83 degrees

ADC channel 0 value:
| 0 | 1 | 2 | 3 |
-----
| 673 | 0 | 0 | 0 |

Acceleration (m/s^2):( 1.798, -2.639, 8.835)
XZ angle = 78.50deg YZ angle = 106.63deg

Magnetometer (gauss):( 97.727, 37.273, -417.959)
heading: 39.90 degrees

ADC Validation Efficiency:82.7%
ADC data is valid

```

6. Power Spectrum Graph

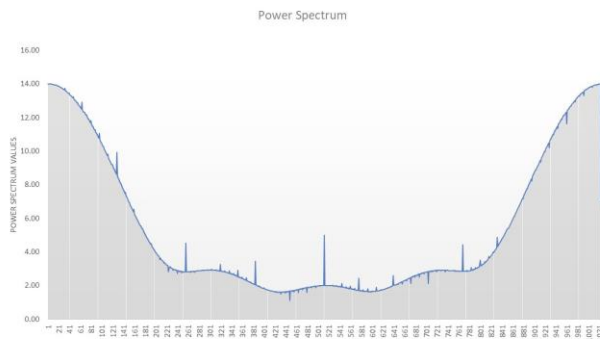


Fig 10. Power Spectrum Graph

7. Conclusion

The motor successfully rotates in clockwise and anticlockwise directions and the LSM303 measures the acceleration of it and we have achieved 82.7% efficiency which satisfies the Nyquist criteria.

8. Acknowledgement

I sincerely thank Dr. Hua Harry Li for teaching main concepts and guiding to reach the solutions.

9. Appendix

Project Demo Video YouTube Link:

<https://www.youtube.com/watch?v=WfaOvWt6TGM>

10. References

- [1] H. Li, "Author Guidelines for CMPE 146/242 Project Report", *Lecture Notes of CMPE 146/242*, Computer Engineering Department, College of Engineering, San Jose State University, March 6, 2006, pp. 1.
- [2] Stepper Motor interface with Raspberry Pi. <https://www.electronicwings.com/raspberry-pi/stepper-motor-interfacing-with-raspberry-pi>
- [3] LSM303 interface with Raspberry Pi <https://www.digikey.com/catalog/en/partgroup/lsm303-triple-axis-accelerometer-magnetometer-compass-board/54857>
- [4] ADC interface with Raspberry Pi <https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters>