Department of Electronic & Telecommunication Engineering
University of Moratuwa

EN2160 - Electronic Design Realization

Report 02
Design Details

Torque Sensor

Group No. 17

AMARATHUNGA D.N.	210037G
PASIRA I.P.M.	210446J

Date - 2024.03.23

Contents

1 Comprehensive Design Details

1.1 Circuit Analysis

1.1.1 Power Supply

In our battery-powered setup, we utilize a regulator circuit to maintain stable voltage output, ensuring consistent power supply to both the microcontroller and strain gauges.

We selected BD450M5 due to its low dropout voltage, which is typically 0.2V. Its maximum current rating is 0.5A.

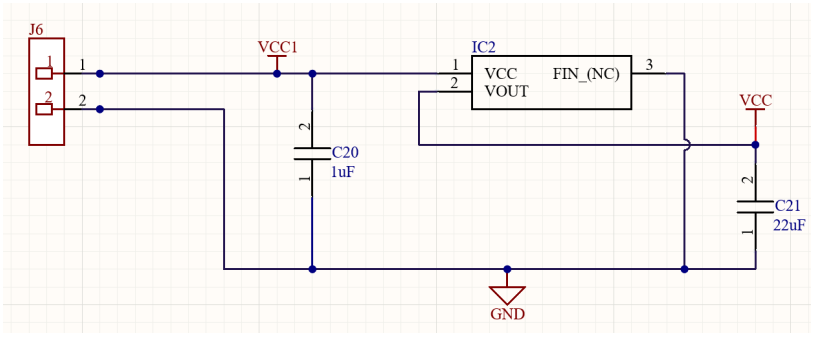


Figure 1: Power Circuit

1.1.2 Bridge Balancing Circuit

The initial unbalances in strain gauge circuits, caused by irregularities in gauge resistances, often exceed the actual strain-induced signal. The solution involves adjusting resistances within the Wheatstone Bridge configuration. However, manual methods pose challenges such as determining the arm needing adjustment and avoiding further imbalance.

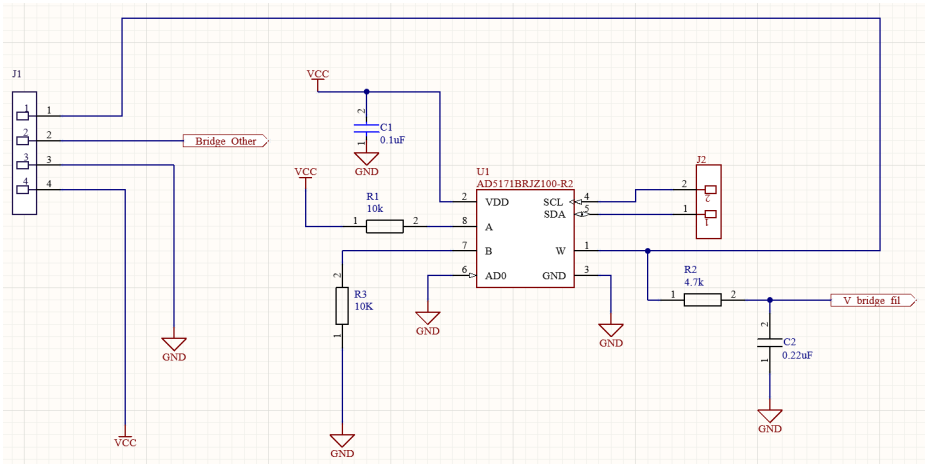


Figure 2: Bridge Balancing Circuit

We choosed AD5171 as the digital potentiometer to adjust resistance because,

- Stores the resistance value, maintaining the wiper position even when power is off and resuming it when power is restored.
- Permanently stores a chosen resistance value after OTP activation.

- Allows unlimited adjustments to the resistor value prior to OTP activation.
- 64 The AD5171 has 64 discrete wiper positions.
- This fine adjustment allows for precise control over the resistance settings.

1.1.3 Instrumentation Amplifier Circuit

To enhance the output signal from a Wheatstone bridge, we employ an instrumentation amplifier, known for its precision in measurement applications. Specifically, we've chosen the AD8293G160 for its cost-effectiveness and capability to amplify the signal without introducing significant noise. This amplifier maintains accuracy over time and operates efficiently with minimal power requirements. In addition, we have refined the signal further by incorporating filters to eliminate unwanted noise, ensuring a clean and reliable output for our measurement purposes. The circuit amplifies the voltage difference between the Wheatstone bridge strain gauge. Here why we used AD8293G160.

- Up to 160 MHz, suitable for high-speed signal processing.
- Typically 20 µV offset voltage and 0.3 µV/°C drift, minimizing signal processing errors.
- Typically 140 dB CMR, effective in rejecting common-mode noise.
- Includes overvoltage protection and thermal shutdown for reliable operation.
- The amplifier has a gain of 160, which is suitable for our application.

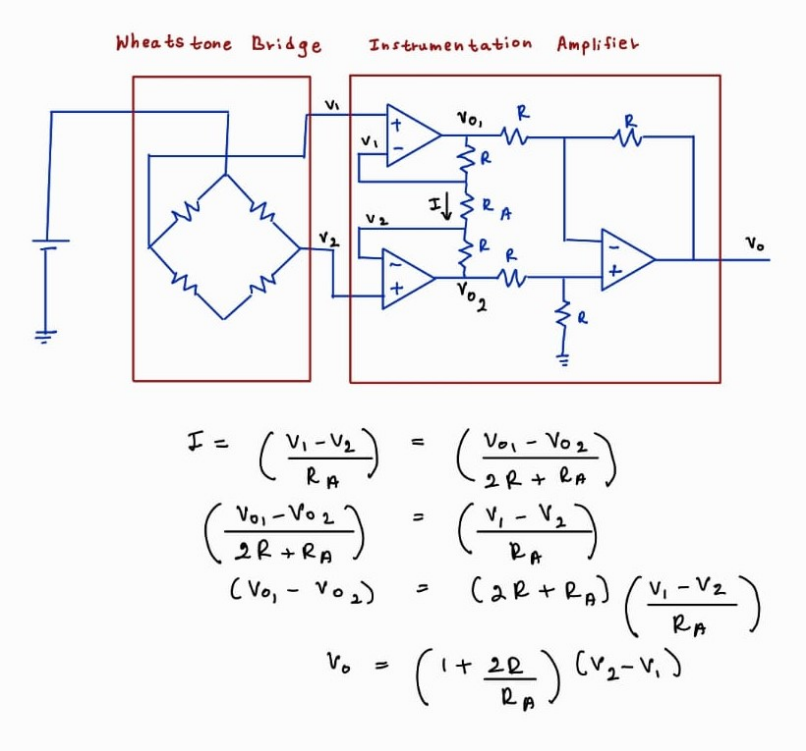


Figure 3: Instrumentation Amplifier Equation

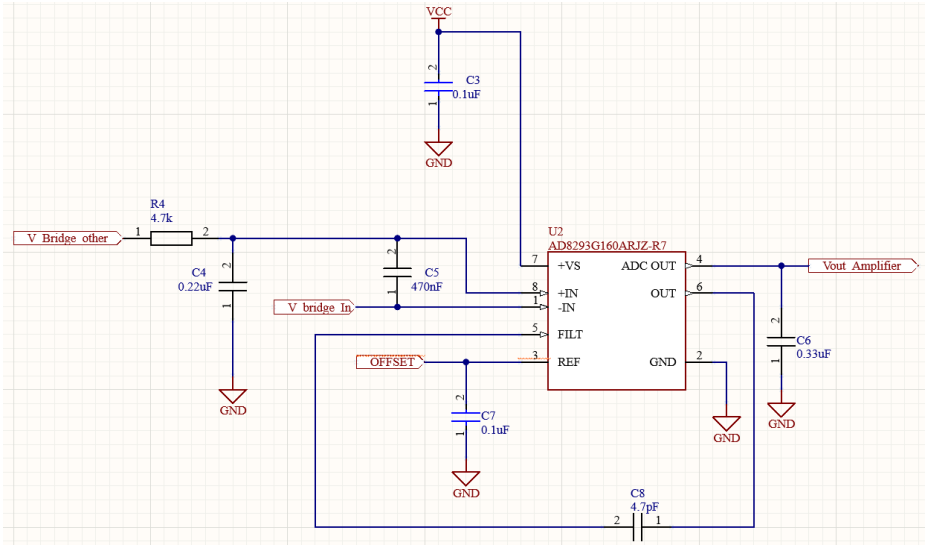


Figure 4: Instrumentation Amplifier

1.1.4 Offset Circuit for Instrumentation Amplifier

The combination of an instrumentation amplifier, such as the AD8293G160, with a digital-to-analog converter (DAC) like the MCP4728, offers precise offset adjustment capabilities in electronic circuits. The instrumentation amplifier serves to accurately amplify small differential signals while rejecting common-mode noise. However, in some applications, it's necessary to introduce an offset voltage to bias the amplifier's output to a specific level or compensate for certain system characteristics. By integrating a DAC into the circuit, engineers can dynamically adjust this offset voltage digitally. The MCP4728 provides multiple channels of analog output with high resolution, typically through an I2C interface. This allows for precise control over the offset voltage applied to the instrumentation amplifier.

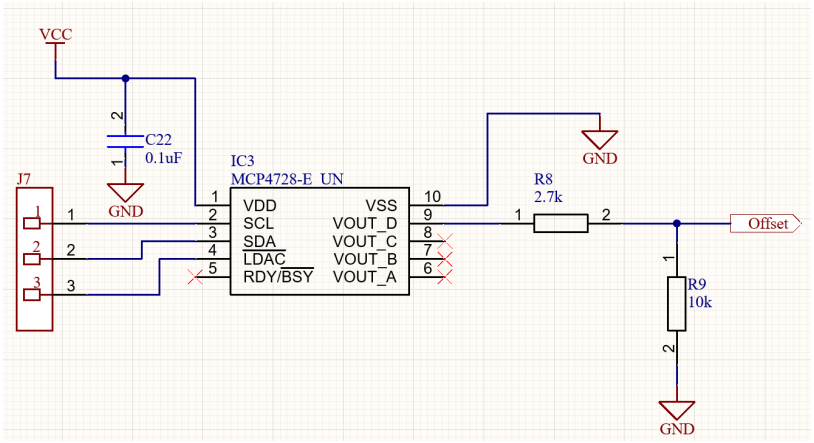


Figure 5: Offset Circuit

Here's why we used MCP4728 over other DACs.

- Provides up to 12-bit resolution for precise analog signal representation.
- Designed for efficient power usage, beneficial for battery-powered applications.
- Offers low noise, high linearity, and reliable operation in various conditions.

- Includes on-board nonvolatile memory (EEPROM) for storing DAC codes and configuration settings

1.1.5 Amplifier Circuit

To amplify the weak signal from the instrumentation amplifier, the circuit incorporates a secondary amplifier, the OPA336, which operates efficiently on low supply voltage and features a rail-to-rail output swing. Configured as a non-inverted amplifier with a single-pole low-pass filter, it ensures signal flexibility across different force ranges. Gain adjustments are made using the AD5171 digital potentiometer, which provides precise control with 64-step resolution for optimal signal amplification.

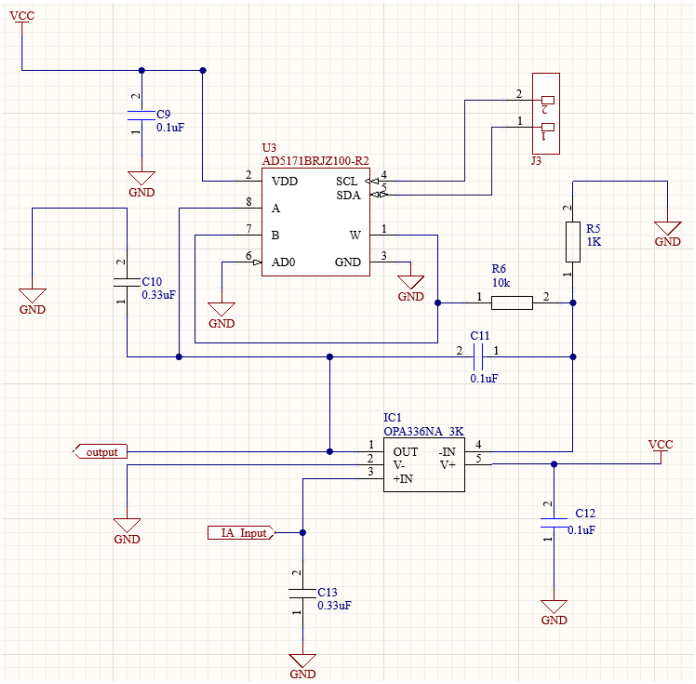


Figure 6: Amplifier Circuit

Here are the advantages of the OPA336 operational amplifier (op-amp) compared to other typical op-amps like the LM741 and TL072.

- Very low power consumption, ideal for battery-operated devices.
- High speed (up to 10 MHz bandwidth) and precision with low offset voltage.
- Low noise density and distortion, suitable for sensitive measurement.
- Designed for efficient single-supply operation.

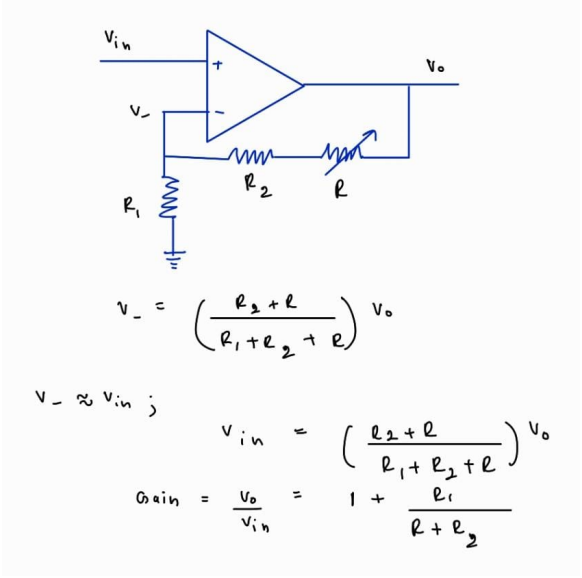


Figure 7: Non Inverting Amplifier Equation

1.1.6 Microcontroller Circuit

The microcontroller is used both to display the output torque value and to act as the ADC converter.

The unit converts the analog signal from the strain gauges into digital data, acting as an ADC, calculates the corresponding torque value, and controls the output signal, which may involve displaying the torque value or transmitting it to external devices.

The ADC converts the analog input signal into a 10-bit digital value, providing 1024 discrete levels of representation.

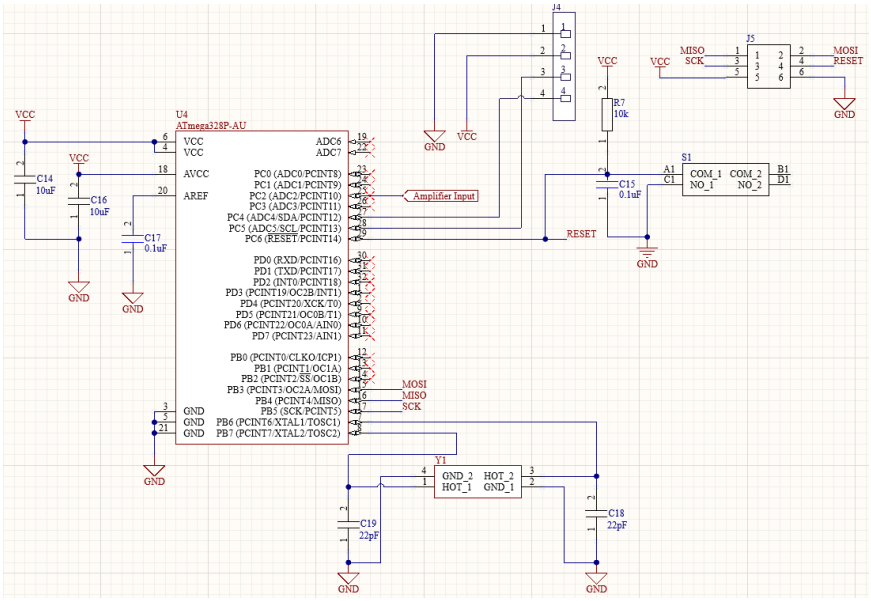


Figure 8: Microcontroller

1.1.7 Calculations

When no torque was applied, the resistor values of the strain gauges, measured using a multimeter, were 273, 278, 282, and 289 ohms. Based on those values, we performed the following calculation:

The calculated initial voltage difference was 7.961 mV, indicating an imbalance in the bridge circuit. To balance the Wheatstone bridge, we used a digital potentiometer with a maximum resistance of 100 kohms. The calculated resistance value needed to balance the Wheatstone bridge was 79,619 ohms, which should be set on the digital potentiometer.

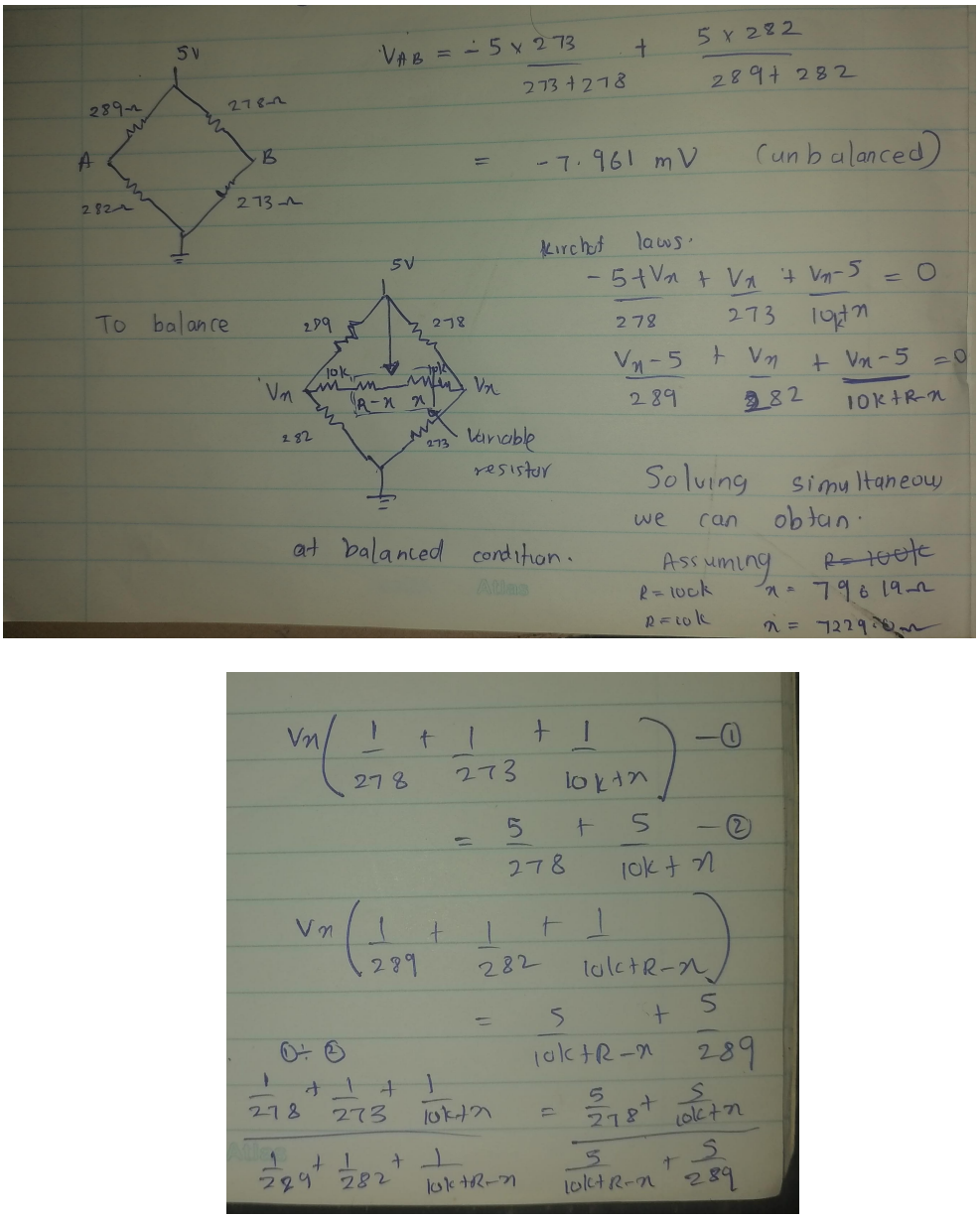


Figure 9: Calculation of resistor value to balance the Wheatstone bridge

The calculated polar moment of inertia of the shaft is $3.77 \times 10^{-9}\,\mathrm{m}^4$. When a 100 Nm torque is applied, the angle that the shaft rotated can be calculated as below. Based on that, changes in resistances were calculated. Assuming that the Wheatstone bridge is balanced by fixing the above-calculated resistor value, the theoretical voltage difference was calculated.

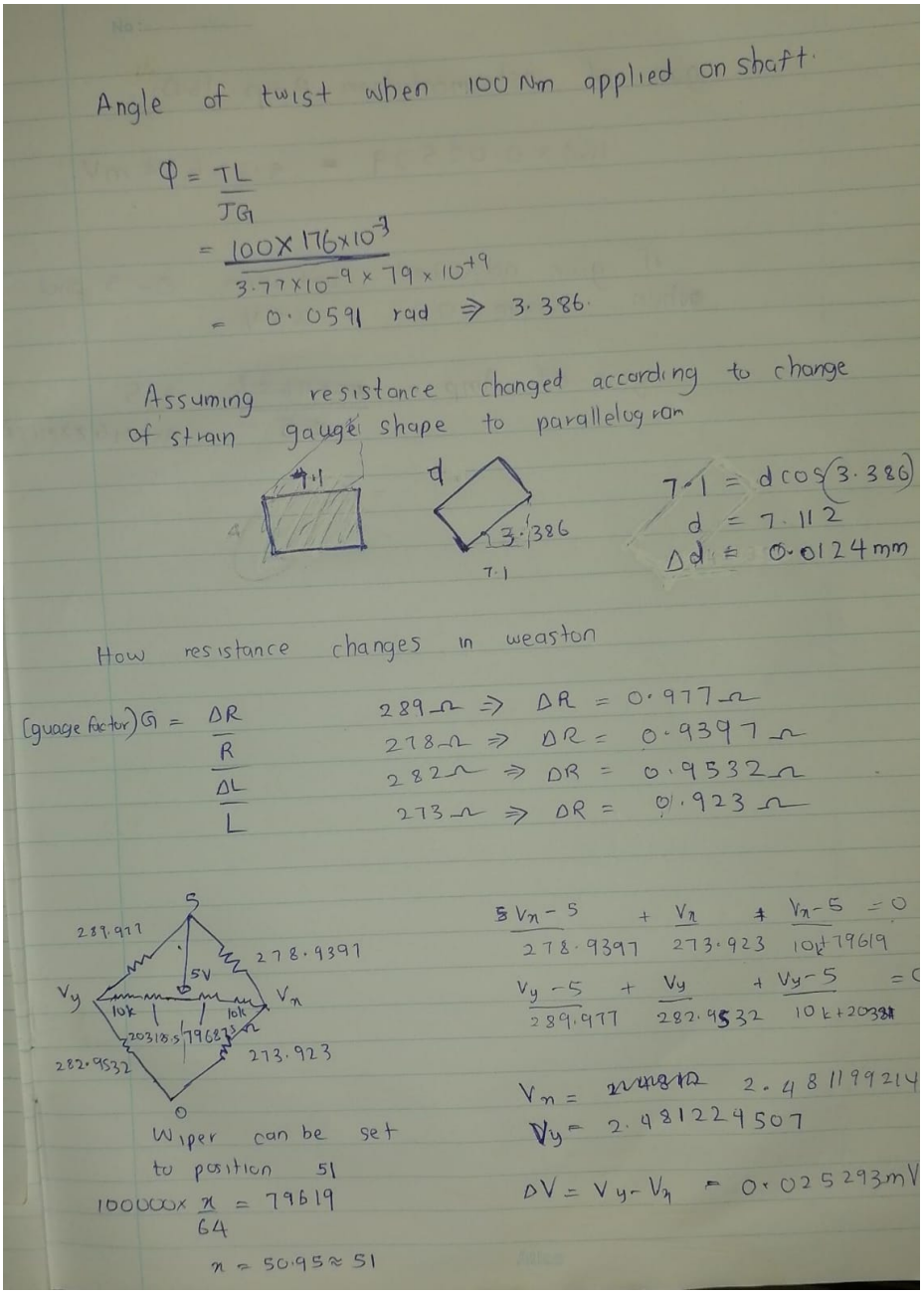


Figure 10: Calculation of voltage difference when a 100Nm torque is applied

After amplifying the voltage difference by instrumentation amplifier, the below value was obtained.

We have used separate circuits to adjust the offset and amplifier gain to desirable values. By adjusting them accordingly, we can map the 100Nm torque to 5V, which is the maximum input voltage for the ADC converter.

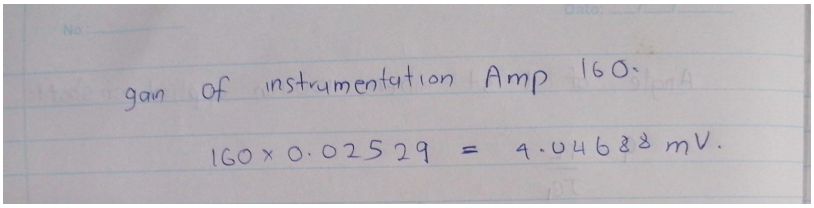


Figure 11: Value of the voltage difference after amplified by the instrumentation amplifier

1.2 Enclosure Design

Our torque sensor enclosure design, designed with SolidWorks, embodies precision engineering and innovation, ensuring functionality and ease of manufacturing. We've paid close attention to detail, incorporating draft angles and fillets strategically to facilitate smooth manufacturing processes, particularly molding.

Draft angles ensure smooth release of the molded part from both the core and cavity, preventing sticking and reducing the likelihood of production delays and defects. Fillets further aid in the ejection process by minimizing friction and promoting uniform stress distribution within the mold. Additionally, they improve material flow during injection, reducing the risk of air pockets or voids in the final part. By incorporating these design features into the core and cavity components, manufacturers can optimize molding efficiency, minimize costs, and achieve consistent, high-quality production outcomes.

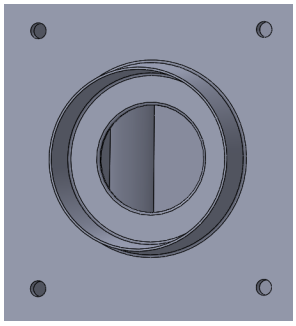
Our enclosure consists of four main parts the top, bottom (lid), and two side sections.

1.2.1 Top Part

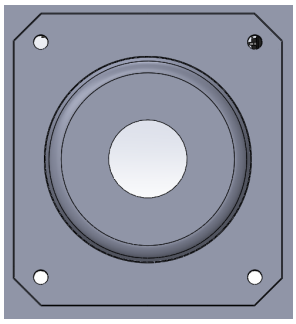
The top component acts as the main cover, offering access to internal parts while shielding them from external elements.

We strategically positioned the shaft holes near the floor to enhance stability. This design decision effectively lowers the center of gravity of the structure, minimizing the risk of tipping or imbalance. By implementing this approach, we ensure that our product remains secure and reliable, even when subjected to external forces or vibrations.

A circular protrusion is included on the side of the housing. This section allows for the insertion of the racer, ensuring a secure fit.



(a) Hollow space for to fix racers



(b) Side part fixed to the Main Housing

1.2.2 Bottom Part (Lid)

As the lid, the bottom part provides essential support to house the torque sensor securely.

1.2.3 Side Parts

Two side components are specifically designed to keep the racers in place. These components are securely attached to the main housing using nuts and bolts.

1.3 Programming

1.3.1 Atmega328P Microcontroller

We detail that the Atmega328P microcontroller serves as the central processing unit for showcasing torque variation. The output of the amplifier circuit is routed to one of the analog pins of the Atmega328P, where it is efficiently processed to generate torque data. This data is then utilized to construct a graphical representation of torque variations on the display. Using the analog-to-digital conversion capabilities of the Atmega328P, we ensure accurate translation of analog signals into digital data for visualization.

Figure 13: ATMEGA 328P-AU

```
#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 16000000UL
#define display_ADDRESS 0x3C
#define display_COMMAND 0x00
#define display_DATA 0x40

#define SCL_PIN PC5
#define SDA_PIN PC4

#define READ_PIN A0

uint8_t buffer[1024];
```

```
double ox, oy;
bool Redraw4 = true;
double x = 0, y = 0;

void setup() {
    DDRC |= (1 << SCL_PIN) | (1 << SDA_PIN);

    // Initialize I2C and display
    i2c_init();
    display_init();

    ADMUX = (1 << REFS0);
    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1);

    // Clear the display
    display_clear();
    display_show();
}

void loop() {
    ADCSRA |= (1 << ADSC);
    while (ADCSRA & (1 << ADSC)); // Wait for conversion to complete
    uint16_t adc_value = ADC;

    double bvolts = adc_value;
    x += 1;

    DrawCGraph(x, bvolts, 30, 50, 75, 30, 0, 100, 25, 0, 1024, 512, 0,
    "Bits vs Seconds", Redraw4);

    if (x > 100) {
        while (1) {}
    }

    _delay_ms(1000);
}

void i2c_init(void) {
    TWSR = 0x00;
    TWBR = ((F_CPU / 100000UL) - 16) / 2; // Set the TWI frequency to 100kHz
}

void i2c_start(void) {
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
    while (!(TWCR & (1 << TWINT)));
}

void i2c_stop(void) {
    TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);
    while (TWCR & (1 << TWSTO));
}

void i2c_write(uint8_t data) {
    TWDR = data;
```

```
    TWCR = (1 << TWINT) | (1 << TWEN);
    while (!(TWCR & (1 << TWINT)));
}

void display_command(uint8_t command) {
    i2c_start();
    i2c_write(display_ADDRESS << 1);
    i2c_write(display_COMMAND);
    i2c_write(command);
    i2c_stop();
}

void display_init(void) {
    _delay_ms(100);
    display_command(0xAE); // Display OFF
    display_command(0x20); // Set Memory Addressing Mode
    display_command(0x00); // Horizontal Addressing Mode
    display_command(0xB0); // Set Page Start Address for Page Addressing Mode
    display_command(0xC8);
    display_command(0x00); // Set low column address
    display_command(0x10); // Set high column address
    display_command(0x40); // Set start line address
    display_command(0x81); // Set contrast control
    display_command(0xFF); // Max contrast
    display_command(0xA1);
    display_command(0xA6); // Set Normal Display
    display_command(0xA8); // Set Multiplex Ratio
    display_command(0x3F);
    display_command(0xA4); // Entire Display ON, Output RAM to Display
    display_command(0xD3); // Set Display Offset
    display_command(0x00); // No offset
    display_command(0xD5);
    display_command(0xF0); // Highest frequency
    display_command(0xD9); // Set Pre-charge Period
    display_command(0x22); // Pre-charge period
    display_command(0xDA); // Set COM Pins Hardware Configuration
    display_command(0x12);
    display_command(0xDB);
    display_command(0x20);
    display_command(0x8D); // Set Charge Pump
    display_command(0x14);
    display_command(0xAF); // Display ON
}

void display_clear(void) {
    // Clear the display buffer
    for (int i = 0; i < 1024; i++) {
        buffer[i] = 0x00;
    }
}

void display_show(void) {
    // Send the display buffer content to the OLED display
    for (uint8_t page = 0; page < 8; page++) {
```

```
        display_command(0xB0 + page);
        display_command(0x00);
        display_command(0x10);
        i2c_start();
        i2c_write(display_ADDRESS << 1);
        i2c_write(display_DATA);
        for (uint8_t col = 0; col < 128; col++) {
            i2c_write(buffer[ col ]);   // Display buffer 128
        }
        i2c_stop();
    }
}

void display_draw_pixel(int x, int y, bool color) {
    if (x >= 0 && x < 128 && y >= 0 && y < 64) {
        if (color) {
            buffer[x + (y / 8) * 128] |= (1 << (y % 8)); // Set pixel on
        } else {
            buffer[x + (y / 8) * 128] &= ~(1 << (y % 8)); // Set pixel off
        }
    }
}

void DrawCGraph(double x, double y, double gx, double gy, double w, double h,
double xlo, double xhi, double xinc, double ylo, double yhi, double yinc,
double dig, const char* title, bool& Redraw) {

    double i;
    double temp;

    if (Redraw) {
        Redraw = false;

        // Draw the title
        for (int i = 0; title[i] != '\0'; i++) {
            display_draw_pixel(i * 6, 0, 1);
        }

        // Calculate initial display coordinates
        ox = (x - xlo) * (w) / (xhi - xlo) + gx;
        oy = (y - ylo) * (gy - h - gy) / (yhi - ylo) + gy;

        // Draw y scale
        for (i = ylo; i <= yhi; i += yinc) {
            temp = (i - ylo) * (gy - h - gy) / (yhi - ylo) + gy;
            if (i == 0) {
                for (int j = gx - 3; j < gx + w + 3; j++) {
                    display_draw_pixel(j, temp, 1);
                }
            } else {
                for (int j = gx - 3; j < gx; j++) {
                    display_draw_pixel(j, temp, 1);
                }
            }
```

```
        }

        // Draw x scale
        for (i = xlo; i <= xhi; i += xinc) {
            temp = (i - xlo) * (w) / (xhi - xlo) + gx;
            if (i == 0) {
                for (int j = gy - h; j < gy + 3; j++) {
                    display_draw_pixel(temp, j, 1);
                }
            } else {
                for (int j = gy; j < gy + 3; j++) {
                    display_draw_pixel(temp, j, 1);
                }
            }
        }
    }

    // Update current point and draw it on the display
    x = (x - xlo) * (w) / (xhi - xlo) + gx;
    y = (y - ylo) * (gy - h - gy) / (yhi - ylo) + gy;
    display_draw_pixel(ox, oy, 1);
    display_draw_pixel(x, y, 1);
    ox = x;
    oy = y;
    display_show();
}

// To execute loop function continuously
int main(void) {
    setup();
    while (1) {
        loop();
    }
    return 0;
}
```

1. **Port Configuration:**

   DDRC: Data Direction Register for Port C

   In the ATmega 328, analog pins 4 and 5 act as SDA (PC4) and SCL (PC5) respectively. In port manipulation, this analog range is defined as the C range. Therefore, we have set pins 4 and 5 in DDRC to one to configure them as outputs.

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 14: Data Direction Register for Port C

2. **Analog Reference:**

   ADMUX: ADC Multiplexer Selection Register

REFS1-0 and REFS0-1 was selected from the below four options using the ADC multiplexer to set the reference voltage to AVCC.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, internal V_{REF} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V voltage reference with external capacitor at AREF pin

(a) Voltage Reference Selections for ADC

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

(b) ADC Multiplexer Selection Register

Figure 15: Illustration of ADC Reference and Multiplexer Selections

3. **ADC Set up and Reading:**

   ADCSRA: ADC Control and Status Register

   In the ADCSRA, ADEN was set to one to enable the ADC and ADPS2=1, ADPS1=1, ADPS0=0 were set to select the prescaler factor of 64.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Figure 16: ADC Prescaler Selections

- ADC clock frequency of Atmega328 = 16 MHz
- Clock cycles per conversion = 13
- Prescaler factor = 64

$$ADC\ Sampling\ Rate = \frac{ADC\ clock\ frequency}{Prescaler \times Conversion\ clock\ cycles} = \frac{16 \times 10^6\ Hz}{64 \times 13} = 19.2\ kHz$$

To initiate the conversion, the ADSC bit is set to one, after which the microcontroller waits for the conversion to complete before proceeding to read the analog values.

4. **I2C Communication:**

   For I2C communication, the following four registers are used:

   - **TWSR (TWI Status Register)**

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 17: ADC Control and Status Register

- **TWBR (TWI Bit Rate Register)**
  The bit rate is determined using the following equation.
  - CPU frequrncy = 16 MHz
  - Desired TWI frequency = 100 kHz

$$\text{TWBR} = \left\lfloor \frac{\text{CPU\_FREQ}}{2 \times \text{TWI\_FREQ}} - 8 \right\rfloor = \frac{16 \times 10^6}{2 \times 100 \times 10^3} - 8 = 72$$

- **TWCR (TWI Control Register)**
  - TWEN (TWI Enable bit): To enable TWI (Two-Wire Interface) module.
  - TWINT (TWI Interrupt Flag): To indicates that the current TWI operation has completed.
  - TWSTA (TWI Start Condition Bit): To generate a start condition.
  - TWSTA (TWI Start Condition Bit): Tto generate a start condition.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

(a) START Condition

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

(b) WRITE Condition

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

(c) STOP Condition

Figure 18: TWI (I2C) START, WRITE, and STOP Conditions

- **TWDR (TWI Data Register)**

5. **Display Commands:**

   References:

   - SSD1306 DataSheet - Command Descriptions (pg 34-46)→Link
   - Reference code for writing display commands→Link

1.3.2 AD5171 Digital Potentiometer

Programming the AD5171 digital potentiometer with an Arduino offers a flexible and convenient way to control resistance values in electronic circuits digitally. The AD5171 is a single-channel, non-volatile digital potentiometer that can replace mechanical potentiometers in various applications, providing precise resistance adjustments under digital control.

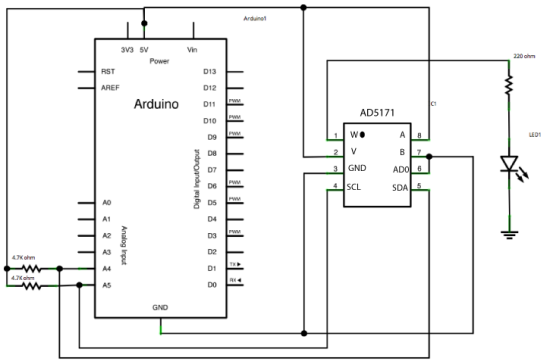


Figure 19: Schematic for Programming AD5171

Code for adjusting resister value:

```
#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 16000000UL

void i2c_init(void) {
  TWSR = 0x00;
  TWBR = ((F_CPU / 100000UL) - 16) / 2; // Set the TWI frequency to 100kHz
}

void i2c_start(void) {
  TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
  while (!(TWCR & (1 << TWINT)));
}

void i2c_stop(void) {
  TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);
  while (TWCR & (1 << TWSTO));
}

void i2c_write(uint8_t data) {
  TWDR = data;
  TWCR = (1 << TWINT) | (1 << TWEN);
  while (!(TWCR & (1 << TWINT)));
}

int main(void) {
  // Set up
  uint8_t val = 0;
  i2c_init();
```

```
  // Loop
  while (1) {
    i2c_start();
    i2c_write(0x2c << 1);
    i2c_write(0x00);
    i2c_write(val); // Send potentiometer value byte
    i2c_stop();

    val++; // Increment value
    if (val == 64) { // If reached maximum position
      val = 0; // Start over from lowest value
    }
    _delay_ms(500);
  }
}
```

OTP Activation:

After testing and finding the correct resistor value, the value should be fixed permanently. According to the AD5171 datasheet, to permanently fix the wiper position, the **T bit** in the SDA instruction byte should be set to logic 1.

S	0	1	0	1	1	0	AD0	0	A	T	X	X	X	X	X	X	A	X	X	D5	D4	D3	D2	D1	D0	A	P
	Slave Address Byte									Instruction Byte								Data Byte									

Figure 20: SDA Write Mode Bit Format

Hence, the int main(void) function should be changed as follows.

```
int main(void) {
  uint8_t val = 10; // Set wiper position
  i2c_init();
  i2c_start();
  i2c_write(0x2c << 1);
  i2c_write(0x80);
  i2c_write(val);
  i2c_stop();
}
```

1.3.3 MCP4728 Digital to Analog Converter

Using Arduino, We can easily control the MCP4728 DAC via the I2C communication protocol. By programming Arduino sketches, specific offset voltage levels can be defined and transmitted to the DAC, instructing it to output corresponding voltage levels. This integration offers a straightforward method to fine-tune the offset voltage given to instrumentation amplifier.

```
#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 16000000UL
#define MCP4728_ADDR 0x60 // MCP4728 I2C address

void i2c_init(void) {
  TWSR = 0x00;
  TWBR = ((F_CPU / 100000UL) - 16) / 2; // Set the TWI frequency to 100kHz
}
```

```
void i2c_start(void) {
  TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
  while (!(TWCR & (1 << TWINT)));
}

void i2c_stop(void) {
  TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);
  while (TWCR & (1 << TWSTO));
}

void i2c_write(uint8_t data) {
  TWDR = data;
  TWCR = (1 << TWINT) | (1 << TWEN);
  while (!(TWCR & (1 << TWINT)));
}

void mcp4728_set_output(uint16_t value) {
    uint8_t buffer[3];
    buffer[0] = 0b01011110;           // Command byte for fast write to channel D
    buffer[1] = (value >> 8) & 0xFF;  // High byte of DAC value
    buffer[2] = value & 0xFF;         // Low byte of DAC value

    i2c_start();
    i2c_write(MCP4728_ADDR << 1);     // Send MCP4728 address with write bit
    for (uint8_t i = 0; i < 3; i++) {
        i2c_write(buffer[i]);         // Send control byte and DAC value bytes
    }
    i2c_stop();
}

int main(void) {
    i2c_init();
    mcp4728_set_output(512); // Set MCP4728 DAC output to 512

    while (1) {
        _delay_ms(1000);
    }

    return 0;
}
```

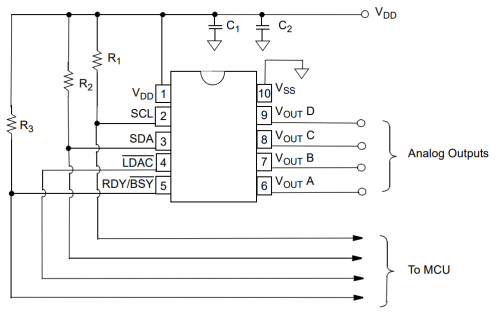


Figure 21: Schematic for MCP4728 programming

1.4 Mechanical Design

1.4.1 Shaft Design

During shaft design, it is necessary to ensure that the stresses acting on the shaft are within the permissible shear stress of the selected material (mild steel) while carrying torque.

To increase the sensitivity of the strain gauges to changes in torque, the section where the strain gauges are attached was made thinner. This thinner section allows for greater deformation in response to torque, resulting in a more pronounced change in electrical resistance measured by the strain gauges.

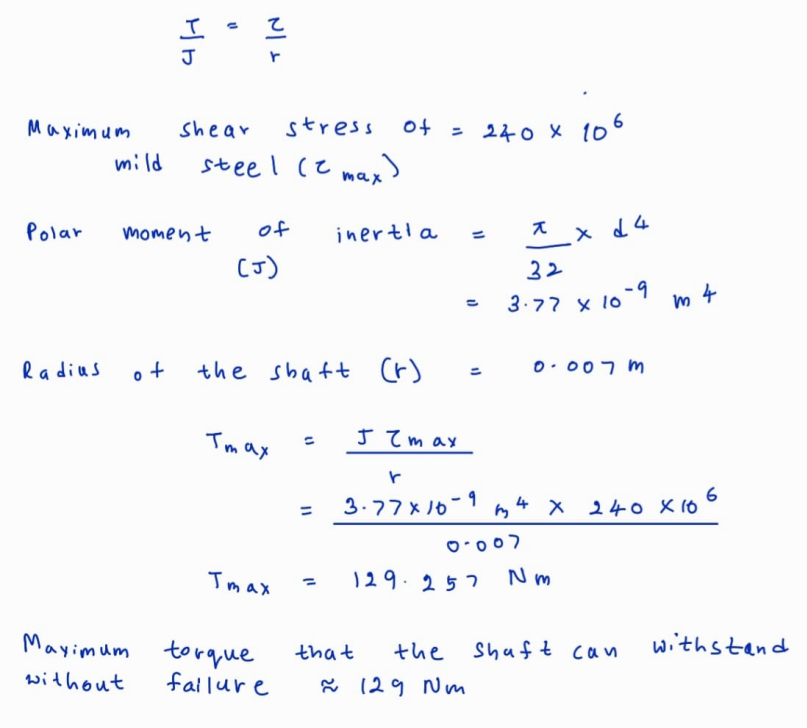


Figure 22: Maximum torque that the shaft can withstand

1.4.2 Slip Ring Design

We encountered difficulty in sourcing slip rings of the required size and faced challenges in exporting them within the permitted time frame. Therefore, we have decided to fabricate them ourselves. We utilized brass rings, nylon bushes, and carbon brushes as components in the fabrication of the slip rings using the following method.

1. **Inserting Nylon Bushing:** We start by sliding a nylon bushing onto the shaft. This bushing ensures insulation, preventing electrical contact between brass rings and the shaft while allowing them to rotate smoothly.

2. **Placing Brass Rings:** Next, we place four brass rings onto the nylon bushing, considering the number of circuits we need. Two rings are needed for power lines (Vcc and ground), and the other two brass rings are for taking signals from the Wheatstone bridge. We make sure they're evenly spaced to prevent electrical interference and ensure each circuit has its own ring.

3. **Considering Cable Entry:** We decide where our cables will enter the slip ring assembly. We soldered wires to each brass rings.

4. **Placing Carbon Brushes:** We insert the four carbon brushes into another component with four corresponding holes, which is mounted to the enclosure. These brushes ensure contact with the rotating brass rings. We ensure they're evenly spaced and aligned with each ring to create consistent electrical connections.

(a) Nylon Bushing

(b) Brass Tube

(c) Carbon Brushes

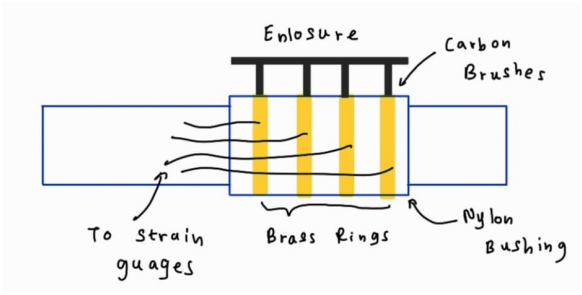


Figure 24: Slip Ring Placement

1.5 Calibration

Calibrating a torque sensor is essential to ensure accurate measurements. Building a bridge balancing circuit for zero calibration is a common method for calibrating sensors like torque sensors. Here's a simplified step-by-step guide to calibrate the torque sensor using a bridge balancing circuit.

1. **Zero Calibration:** Zero calibration is the process of setting the sensor's output to zero when no torque is applied. This step is crucial to eliminate any offset or bias in the sensor's reading. nitially, when there's no torque applied to the shaft, the Wheatstone bridge might not be perfectly balanced, leading to a non-zero output. To achieve a zero output when there's no torque, the bridge needs to be balanced using AD5171 digital potentiometer.

   The AD5171 digital potentiometer is a device that electronically simulates the behavior of a traditional mechanical potentiometer. Instead of physically rotating a knob to change resistance, the resistance of the AD5171 can be controlled digitally via communication protocols such as I2C or SPI. The Microcontroller sends commands to the AD5171 specifying the desired resistance value or steps by which the resistance should be adjusted. Code is shown above.

2. **Span Calibration:** Span calibration involves applying a known torque to the sensor and adjusting its output to match the expected value. This step ensures that the sensor provides accurate readings across its entire range. The known torque should cover the full range of expected measurements. The calibration procedure needs to be integrated into the code uploaded to the microcontroller, as the existing code maps the sensor's analog voltage output to a range of 0 to 1023. Essentially, adjustments are required within the code to ensure that the sensor's output is accurately represented within this numerical range, thereby facilitating precise torque measurements.

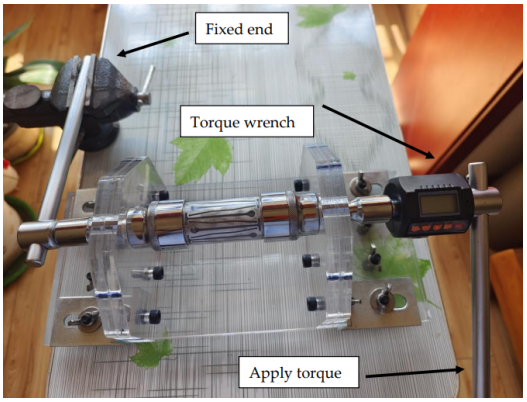


Figure 25: Set up for testing and calibration

1.6 System Integration

Insert the display wires through the holes in the enclosure. Connect the battery power to the PCB and place both the PCB and battery inside the enclosure. Secure the PCB to the enclosure using screws. Connect the display wires to the PCB via the relevant JST port.

Next, place the shaft so that it contacts the carbon brushes and slip rings. Connect the wires from the slip rings to the PCB. Two of the slip ring wires should be connected to the 5V and ground, which provide power to the Wheatstone bridge configuration of the strain gauge in the shaft. The other two wires carry the incoming signal to the PCB. Ensure all four wires are connected to the PCB using the appropriate JST port.

2 Daily Log Entries

2.1 Exploring Research Publications and Other Resources(22 February - 29 February)

In our exploration of torque sensor technologies, we thoroughly examined numerous master's theses and research publications. In our investigation into torque sensor technologies, our inquiry led us to uncover a variety of sensor types.

In our initial review of the first master's thesis, we encountered a range of methods, prominently including the strain gauge method. Drawing inspiration from these findings, we synthesized our own concepts and formulated preliminary designs. The strain gauge method is a widely used technique for measuring torque. It relies on the principle that when a torque is applied to a shaft or structure, it causes a deformation or strain in the material. A strain gauge is a small sensor that is bonded to the surface of the shaft. To measure torque using the strain gauge method, multiple strain gauges are often used in a configuration known as a Wheatstone bridge. This configuration allows for greater sensitivity and accuracy in detecting small changes in resistance.

By carefully examining the second master thesis, we gained valuable insights into signal processing techniques relevant to torque sensors. It became clear that multiple amplification stages are crucial, and reducing noise is essential for accurate measurements.

Through our analysis of these master's theses, we uncovered various methods for transmitting signals from the sensor system to the circuit. This exploration was particularly important because our torque sensor must be able to calculate rotary torque, necessitating a wireless connection between the sensor and circuitry, as direct wiring is not feasible.

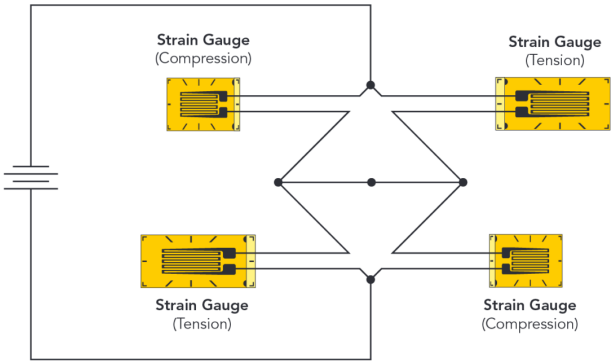
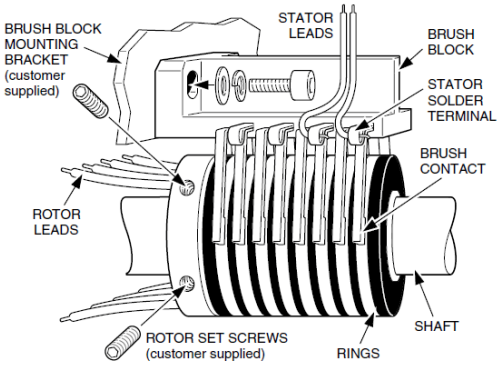


Figure 26: Strain Gauge method

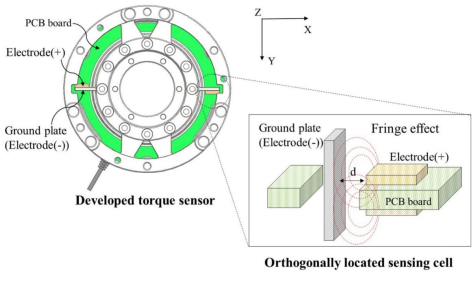
Among the methods we discovered were the slip ring method and the digital telemetry method. These approaches provide avenues for transmitting signals wirelessly from the torque sensor system to the circuit, facilitating the calculation of rotary torque without the constraints of direct wiring.

The slip ring method is a technique used to transmit signals or electrical power from a rotating component, such as a shaft, to a stationary component, without the need for physical connections like wires. It's commonly employed in systems where continuous rotation is required, such as in torque sensors measuring rotary torque. As the shaft rotates, the brushes or contacts maintain contact with the rings on the rotor, allowing electrical signals or power to be transmitted from the rotating component to the stationary component.

The digital telemetry method is also an advanced technique used for transmitting data wirelessly from a rotating component, such as a shaft, to a stationary receiver or data acquisition system. In this method, sensors mounted on the rotating component measure torque or other parameters and convert the measurements into digital signals. These digital signals are then encoded and modulated onto a carrier wave using techniques such as frequency-shift keying (FSK) or phase-shift keying (PSK). The modulated carrier wave is then transmitted wirelessly to a stationary receiver.



(a) Slip Ring Method



(b) Digital Telemetry Method

From the remaining two research publications, we encountered two additional torque sensor technologies: capacitive torque sensors and magnetostrictive torque sensor technology. Although challenging to implement, these methods proved immensely valuable in expanding our

understanding and fostering the development of conceptual designs.

Capacitive torque sensors utilize changes in capacitance to measure torque applied to a shaft or rotating component. This method relies on the principle that the capacitance between two conductive plates changes when the distance between them is altered. In a capacitive torque sensor, one of these plates is fixed while the other is attached to the rotating shaft.

The magnetostrictive method, also known as magnetostrictive torque sensing, is a technique used to measure torque by exploiting the magnetostrictive properties of certain materials. Magnetostriction is the phenomenon where a material changes shape or dimensions when subjected to a magnetic field. In magnetostrictive torque sensors, a torsional shaft is typically made from a magnetostrictive material such as nickel or iron. When a magnetic field is applied to this shaft, it undergoes a slight change in length proportional to the torque applied to it. This change in length is detected using a sensing element, such as a magnetostrictive waveguide or a magnetostrictive wire, which is placed along the shaft's axis.

- Torque measurement
- Six degree of freedom Force/ Torque sensor
- Magnetostrictive Torque sensor technology
- Capacitive Torque sensor technology

2.2 Stimulating Ideas and Developing Conceptual Designs (1 March- 9 March)

In this create phase, our focus was on generating innovative concepts for the torque sensor project through brainstorming and other creative techniques. We have detailed these concepts above, exploring various methods to accurately measure the relative displacement or twist of a shaft, which is fundamental for torque measurement. These efforts reflect our commitment to innovative design and advancing torque sensing technology.

2.3 Evaluating Conceptual Designs (10 March- 16 March)

We focused on evaluating the developed conceptual designs for the torque sensor projec. We considered a variety of factors across different aspects of the designs to ensure comprehensive assessment and informed decision-making. Finally, we decided to proceed with the strain gauge slip ring method.

2.4 Shaft and Slip Ring Designing (17 March - 24 March)

Utilizing our integration method, we designed a 14mm diameter shaft and a compatible slip ring. The design ensures seamless communication with the torque sensor and system components. This approach guarantees accurate data acquisition and reliable performance, enhancing overall system integration and functionality.

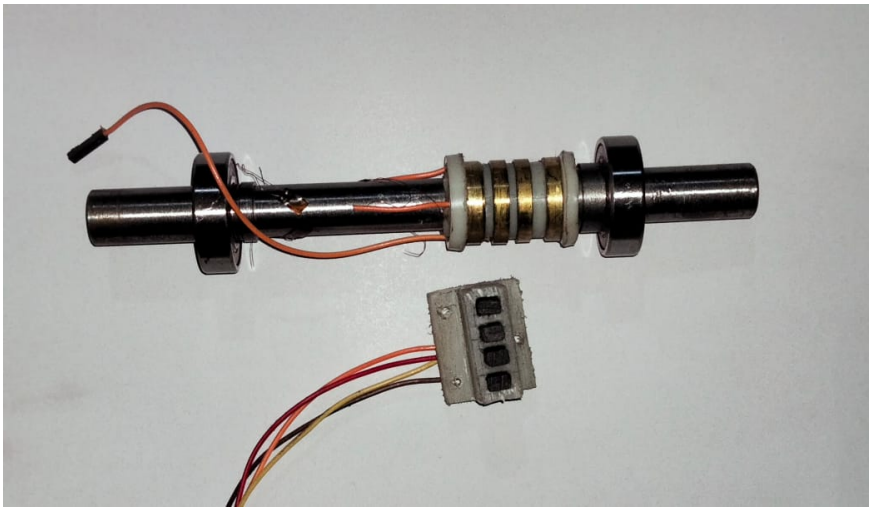


Figure 28: Designed Shaft and Slip Rings

2.5 Circut Designing and Component Selection (25 March - 31 March)

In circuit design, besides applying electronic principles, one of the major challenges we encountered was selecting components that best suited our requirements. This involved considering factors such as availability, cost, and other relevant aspects. We listed several alternatives for main components and evaluated them based on various criteria to determine the best fit for our project.

2.5.1 Bill of Materials

Component	Amount	Price($)
MCP4728-E/UN	1	2.13
AD8293G160ARJZ-R7	1	1.82
OPA336NA/3K	1	2.73
AD5171BRJZ100-R2	2	10.00
Crystals 16MHz 8pF	1	0.65
LDO Voltage Regulators 3-42V	1	1.23
Thin Film Resistors - SMD 1206 1.0Kohm	1	0.34
Thick Film Resistors - SMD ResHighPowerA 1206 4k7	2	1
Thin Film Resistors - SMD 1206 10Kohm	5	1.95
4.7pF C0G 1206	1	0.39
1206 16V 10uF	2	0.96
50V 0.47uF X7R	1	0.25
25V 22uF X5R	1	0.41
25V .22uF X7R	2	0.7
25V .33uF X7R X7R	3	1.2
25V 1uF X7R	1	0.27
25V 0.1uF X7R	5	1.5
25V 22pF C0G X7R	2	0.52
25V .22uF X7R	2	0.7
ATMEGA328P-AU	1	2.66

2.5.2 Data sheets and other details of selected components

- **MCP4728-E/UN (1)** - (DAC offset) Link
- **AD8293G160ARJZ-R7 (1)** - (instrumentation amplifier) Link
- **595-OPA336NA/3K (1)** - (OpAmp) Link
- **AD5171BRJZ100-R2 (2)** - (digital Pot) Link
- **ATmega328P-AU (1)** - (microcontroller) Link
- **CX2016DB16000D0GPSC1 (1)** - (Oscillator) Link
- **BD450M5FP-CE2 (1)** - (power regulator) Link
- **ERA-8AEB102V (1)** - 1K Link
- **ERA-8AEB272V (1)** - 2.7k Link
- **CHP1206AFX-4701ELF (2)** - 4.7K Link
- **ERA-8AEB103V (5)** - 10k Link
- **12063A220JAT2A (2)** - 22pF Link
- **12063A4R7DAT2A (1)** - 4.7pF Link
- **CGA5L1X7R1C106K160AC (2)** - 10uF Link
- **12065C474JAT2A (1)** - 470nF Link
- **12063D226MAT2A (1)** - 22uF Link
- **12063C224JAT2A (2)** - 0.22uF Link
- **12063C334MAT2A (3)** - 0.33uF Link
- **12063C105MAT2A (1)** - 1uF Link
- **C1206C104F3GACTU (5)** - 0.1uF Link
- **Fevicol Super Glue** (To attach strain gauges onto the shaft)
- **BF350-3AA Strain Gaugues** (4)

PRODUCT BF350-3AA Strain Gauge

FEATURE

The use of anti-static packaging products, to avoid the potential harm of static electricity

All products are tested for stability, consistency and reliability. Ensure product excellence.

Welcome Guide, our worry-free 6-month refund and friendly customer service.

This product is widely used in product development, student experiments, maintenance , production, etc.

If you have any questions, please contact us

PRODUCT DESCRIPTION

Type: BF350-3AA

Resistance: 349.8 ±0.1Ω

Sensitivity coefficient(gauge factor): 2.0 -2.20

Accuracy class: 0.02

Strain limit: 2.0% (2000 microstrain)

Monolithic size: 7.1mm*4.1mm

MATERIALS

Conductive material: Constantan, copper-nickel alloy 55% copper /45% (constant resistance over temperature)

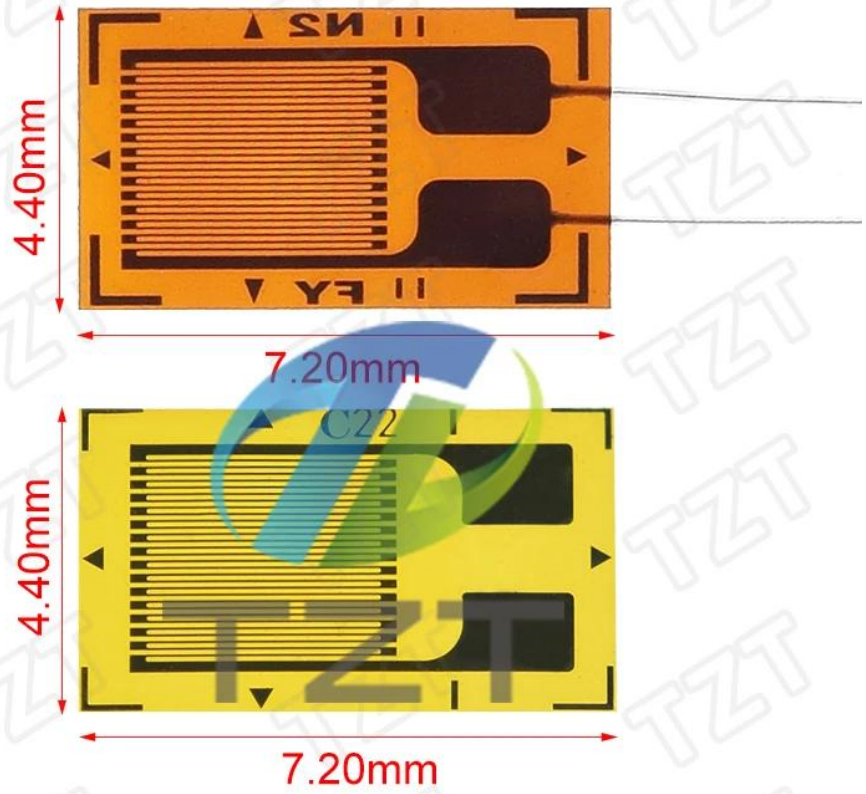
Substrate backing: modified phenolic resin foil

ADHESIVES

Surface must be polished, remove any oxides, residues before applying adhesive.

Long term adhesives: epoxy glue hot curing adhesive (H-610 or equivalent)

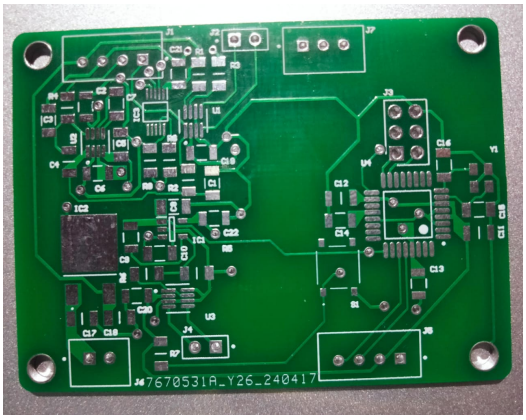
Short term adhesives: Cyanoacrylic glues.TZT.



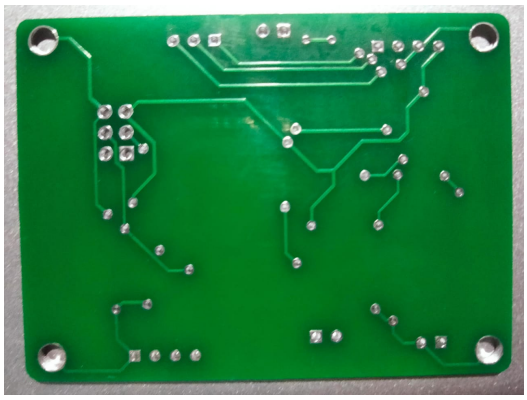
2.6 Schemetic, PCB and Solidwork Designing (1 April - 14 April)

After reviewing the schematics of a previously implemented torque sensor from a master thesis, we thoroughly analyzed them and began designing our own schematics in Altium. We incorporated additional protective measures for the circuits and included a feature for graphically displaying the torque on a screen using atmega chip. We adopted a hierarchical design technique to design our schematic, breaking it into smaller sections or modules. Each part represented a specific function of the torque sensor. This made our schematic easier to understand and work with. We made sure each module connected smoothly with the others, so everything worked together seamlessly. This approach helped us collaborate effectively and fix any issues quickly.

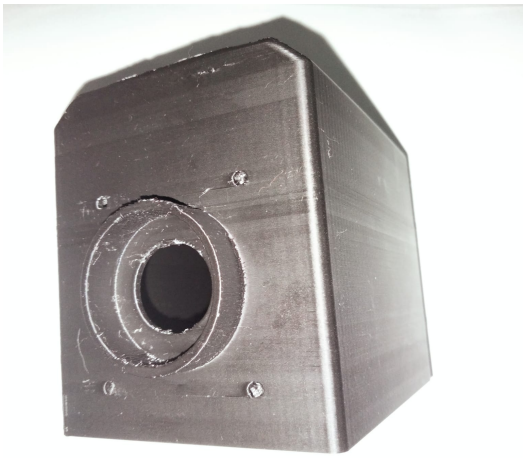
After completing the schematics following standard procedures, we proceeded to the PCB design phase. In our PCB designing process, we employed various routing techniques and component placement strategies to create a compact and efficient layout. We carefully placed components to minimize signal path lengths, ensuring optimal performance. By using advanced routing methods, we managed to maintain a high level of functionality while keeping the board size as small as possible. This approach not only enhanced the overall design but also improved the ease of handling and integration of the torque sensor system.



(a) Front view of bare PCB

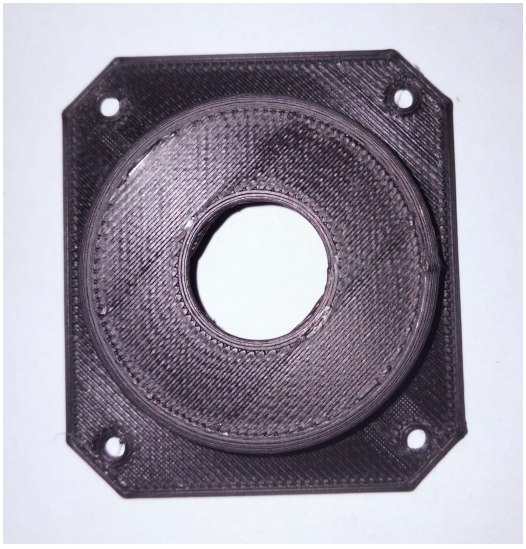


(b) Back view of bare PCB

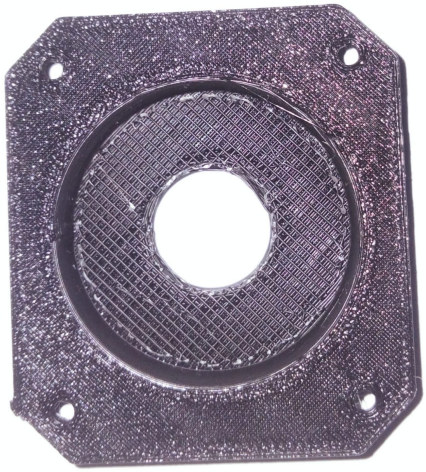


(a) Top Part - Side View

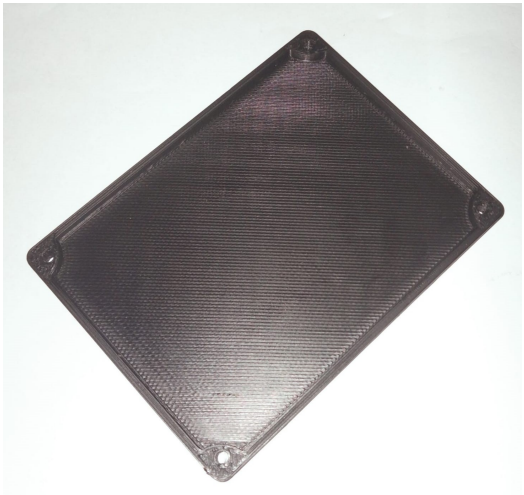
(b) Top Part - Bottom View



(c) Side Part - Top View



(d) Side Part - Bottom View



(e) Bottom Part

Figure 30: Printed Enclosure

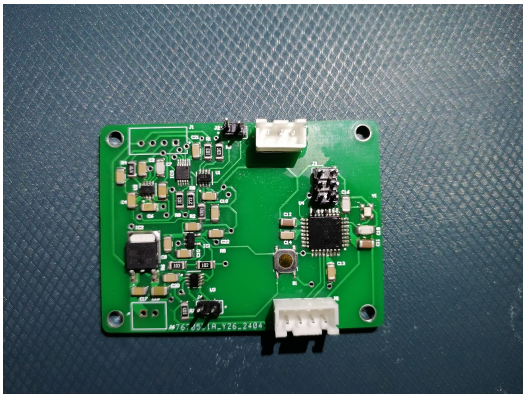
2.7 Programming of Components (14 April - 28 April)

Incorporating programmable digital potentiometers was essential to ensure precise resistance values in our design. We developed the necessary code to program them by thoroughly examining the available resources.

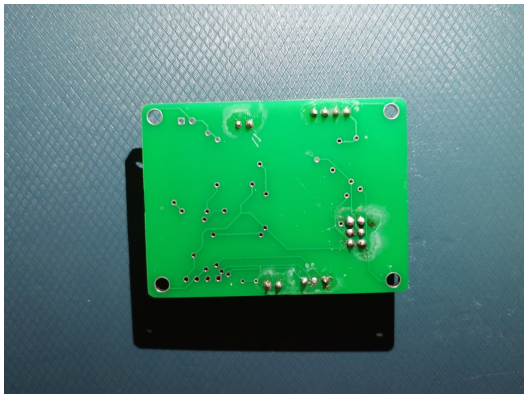
Furthermore, we programmed the Atmega chip to facilitate the conversion of torque readings into voltage values, which were then displayed on an screen. This involved writing code to translate the torque data acquired by the sensor into voltage measurements compatible with the display. By implementing this feature, users could conveniently visualize torque values in real-time, enhancing the usability and accessibility of our torque sensor system.

2.8 PCB Soldering (29 April - 5 May)

We advanced our project's assembly process by soldering components. To solder surface-mount device (SMD) components, we applied solder paste to the pads of the circuit board, precisely positioned the SMD components, and then used hot air to melt the solder, creating strong and reliable connections.



(a) Top View of PCB



(b) Back View of PCB

2.9 Designing the Torque Supply System (5 May - 12 May)

We have designed two metal components to be fixed to the shaft to supply torque. One component will be kept fixed, while torque is applied to the other end. The torque-applying part features a sequence of holes positioned at varying distances, allowing forces to be applied at different points to measure the resulting torque accurately.

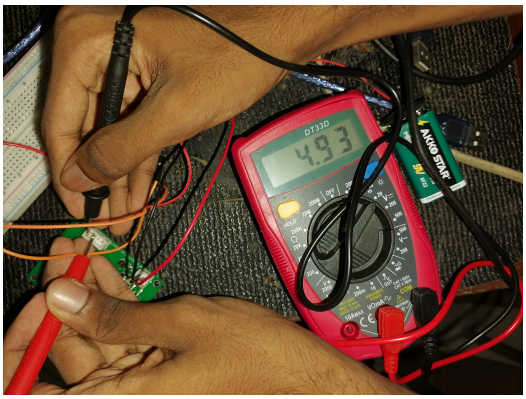
Figure 32: Designed parts to apply torque

2.10 Testing (12 May - Onwards)

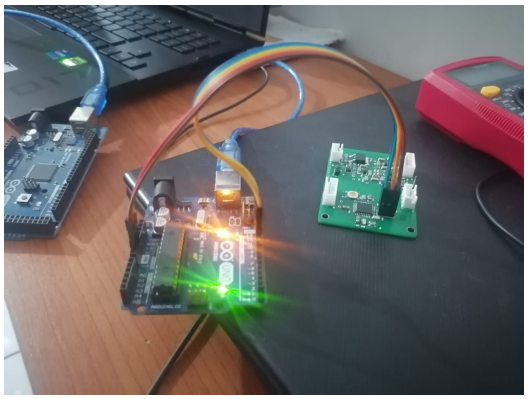
In our torque-providing setup, we achieve precise torque application by manipulating known weighted objects while keeping the opposite side fixed. This method allows us to calibrate the torque sensor accurately, ensuring reliable and consistent measurements.

In the initial testing process, we provide a 9V battery supply to our main power port and measure the voltage between Vcc and ground to test the voltage regulation of our circuitry. This process works effectively, as we consistently obtain a regulated output of 4.93V.

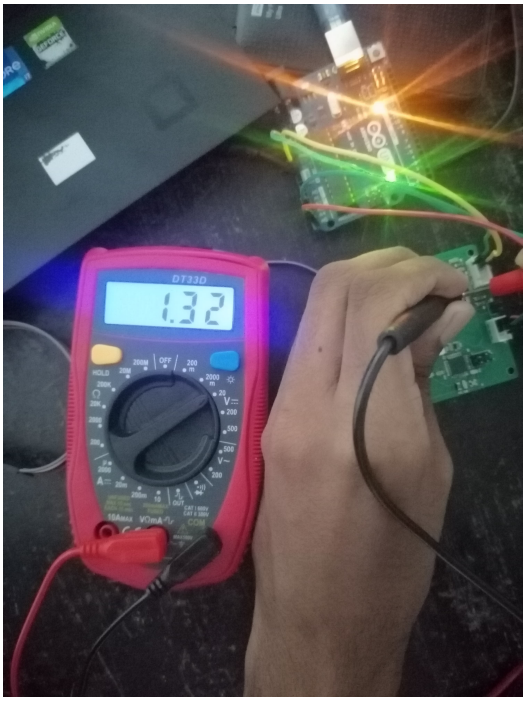
We utilized Arduino programming to configure these potentiometers according to our specifications. By programming the digital potentiometers, we could finely adjust their resistance settings, ensuring optimal performance and accuracy in our weaston bridge configuration of strain gauge.



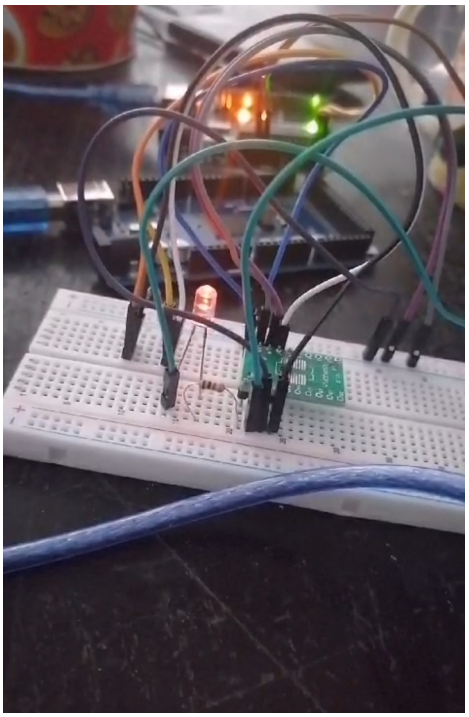
(a) Voltage Testing



(b) Programming Atmega328



(a) MCP4728 DAC programming



(b) Digital Potentiometer testing

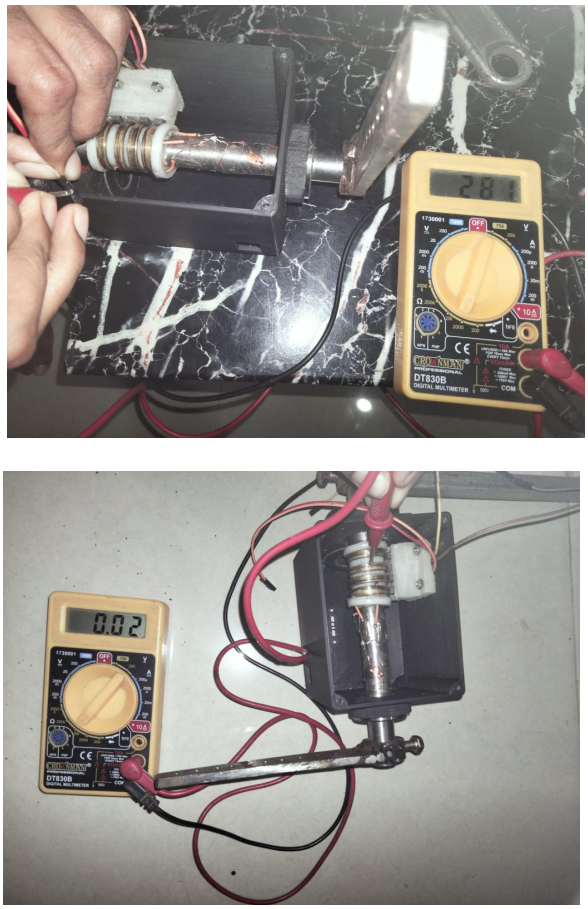


Figure 35: Mechanical Testing

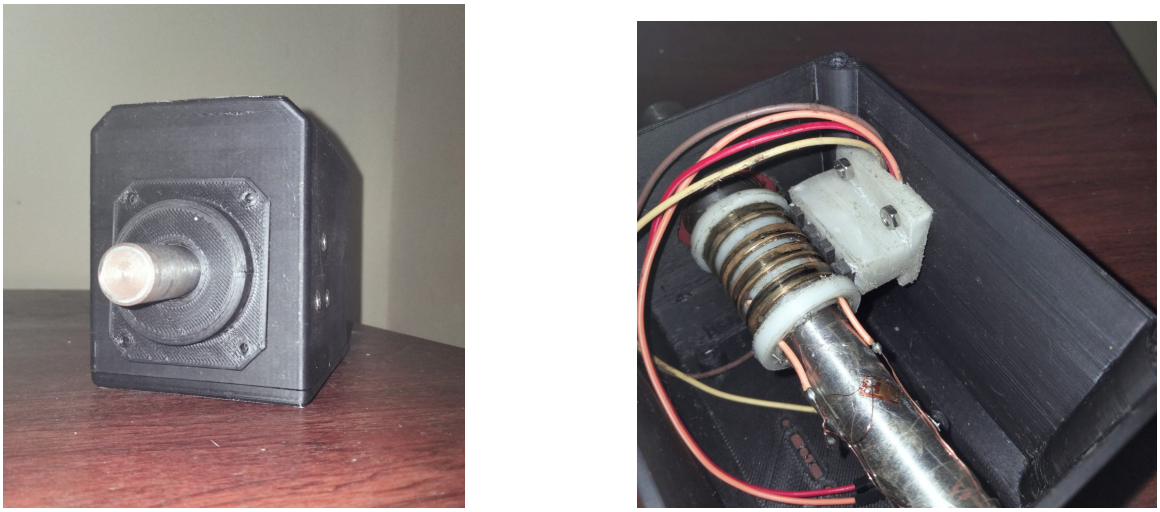


Figure 36: Integrated Torque Sensor

3 Appendix

3.1 Codes with libraries

(Please refer to pages 10-19 for the register level code)

3.1.1 Atmega328P Microcontroller

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define READ_PIN A0
#define OLED_RESET A4
#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH 16

double volts;
double bvolts;
double x = 0, y = 0;
bool Redraw1 = true;
bool Redraw2 = true;
bool Redraw3 = true;
bool Redraw4 = true;
double ox, oy;

Adafruit_SSD1306 display(OLED_RESET);

void setup() {
    Serial.begin(9600);
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
    display.display();
    display.clearDisplay();
    pinMode(A0, INPUT);
}

unsigned long OldTime;
unsigned long counter;

void loop(void) {
    bvolts = analogRead(A0);
    volts = (bvolts / 204.6);
    DrawCGraph(display, x++, bvolts, 30, 50, 75, 30, 0, 100, 25, 0, 1024, 512, 0,
"Bits vs Seconds", Redraw4);
    if (x > 100) {
        while (1) {}
    }
    delay(1000);
}

void DrawCGraph(Adafruit_SSD1306 &d, double x, double y, double gx, double gy,
double w, double h, double xlo, double xhi, double xinc, double ylo, double yhi,
double yinc, double dig, String title, boolean &Redraw) {

    double i;
```

```
    double temp;
    int rot, newrot;

    if (Redraw == true) {
        Redraw = false;
        d.fillRect(0, 0, 127, 16, SSD1306_WHITE);
        d.setTextColor(SSD1306_BLACK, SSD1306_WHITE);
        d.setTextSize(1);
        d.setCursor(2, 4);
        d.println(title);
        ox = (x - xlo) * (w) / (xhi - xlo) + gx;
        oy = (y - ylo) * (gy - h - gy) / (yhi - ylo) + gy;
        d.setTextSize(1);
        d.setTextColor(SSD1306_WHITE, SSD1306_BLACK);
        for (i = ylo; i <= yhi; i += yinc) {
            temp = (i - ylo) * (gy - h - gy) / (yhi - ylo) + gy;
            if (i == 0) {
                d.drawFastHLine(gx - 3, temp, w + 3, SSD1306_WHITE);
            } else {
                d.drawFastHLine(gx - 3, temp, 3, SSD1306_WHITE);
            }
            d.setCursor(gx - 27, temp - 3);
            d.println(i, dig);
        }
        for (i = xlo; i <= xhi; i += xinc) {
            d.setTextSize(1);
            d.setTextColor(SSD1306_WHITE, SSD1306_BLACK);
            temp = (i - xlo) * (w) / (xhi - xlo) + gx;
            if (i == 0) {
                d.drawFastVLine(temp, gy - h, h + 3, SSD1306_WHITE);
            } else {
                d.drawFastVLine(temp, gy, 3, SSD1306_WHITE);
            }
            d.setCursor(temp, gy + 6);
            d.println(i, dig);
        }
    }

    x = (x - xlo) * (w) / (xhi - xlo) + gx;
    y = (y - ylo) * (gy - h - gy) / (yhi - ylo) + gy;
    d.drawLine(ox, oy, x, y, SSD1306_WHITE);
    d.drawLine(ox, oy - 1, x, y - 1, SSD1306_WHITE);
    ox = x;
    oy = y;
    d.display();
}
```

3.1.2 AD5171 Digital Potentiometer

```
#include <Wire.h>

void setup() {
    Wire.begin(); // join I2C bus (address optional for master)
}
```

```
byte val = 0;

void loop() {
    Wire.beginTransmission(44); // transmit to device #44 (0x2C)
    // device address is specified in datasheet
    Wire.write(byte(0x00)); // sends instruction byte
    Wire.write(val); // sends potentiometer value byte
    Wire.endTransmission(); // stop transmitting

    val++; // increment value
    if (val == 64) { // if reached 64th position (max)
        val = 0; // start over from lowest value
    }

    delay(500);
}
```

3.1.3 MCP4728 Digital to Analog Converter

```
#include <Adafruit_MCP4728.h>
#include <Wire.h>

Adafruit_MCP4728 mcp;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10);

  Serial.println("Adafruit MCP4728 test!");

  if (!mcp.begin()) {
    Serial.println("Failed to find MCP4728 chip");
    while (1) {
      delay(10);
    }
  }
  mcp.setChannelValue(MCP4728_CHANNEL_D, 512);
}

void loop() {
  delay(1000);
}
```

The report was reviewed by: