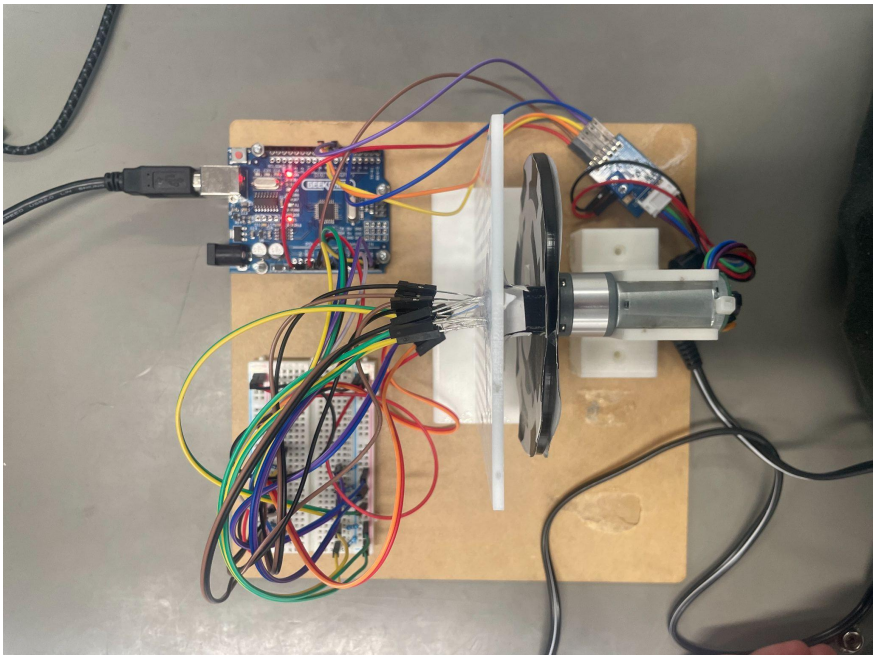
Project B - Encoder Design

Group 2



Name	UPI	ID
Eloise Beattie	ebea412	187448578
Luke Hynds	lhyn722	778830429
Oliver Reedy	oree985	748723004
Otto Walker	owal322	183423349

Table of Contents

Chapter 1 - Absolute Encoder Design

The absolute encoder needed to be able to accurately determine the current position of the shaft. This required a number of design considerations, all of which are detailed in this section, including the component selection, sensing circuit design and the absolute encoder pattern used.

1.1 Component Selection & Sensing Circuit Design

For the sensing circuit, shown in Figure 1.1.1, consideration was given to the components used in the design to meet project requirements. In the design, an infrared LED and infrared phototransistor are placed next to each other in a black-out housing. These components were aligned to measure reflectivity of the encoder plate placed next to them. This allowed for the detection of black and white encoder patterns.

For the infrared LEDs, the TSAL6200 was the designated choice. This IR LED provides high-power infrared emissions. A 68 ohm resistor was selected to limit the current through the IR LED to under 100 mA, to meet the specifications in the datasheet. This choice balanced the need for enough current to ensure light could be detected by the phototransistor, while not putting too much current through and reducing the longevity of the LED.

$$r = v/i = 5\,V\,/\,0.075\,A = 66.67\,\Omega \approx 68\,\Omega$$

For detecting reflected light, the LTR-3208E phototransistor was specified, and its characteristics were considered when designing the circuit. The phototransistor was configured in a voltage divider configuration. This configuration enabled the measurement of the voltage drop across the phototransistor. A 100k ohm resistor was chosen for this voltage divider circuit.

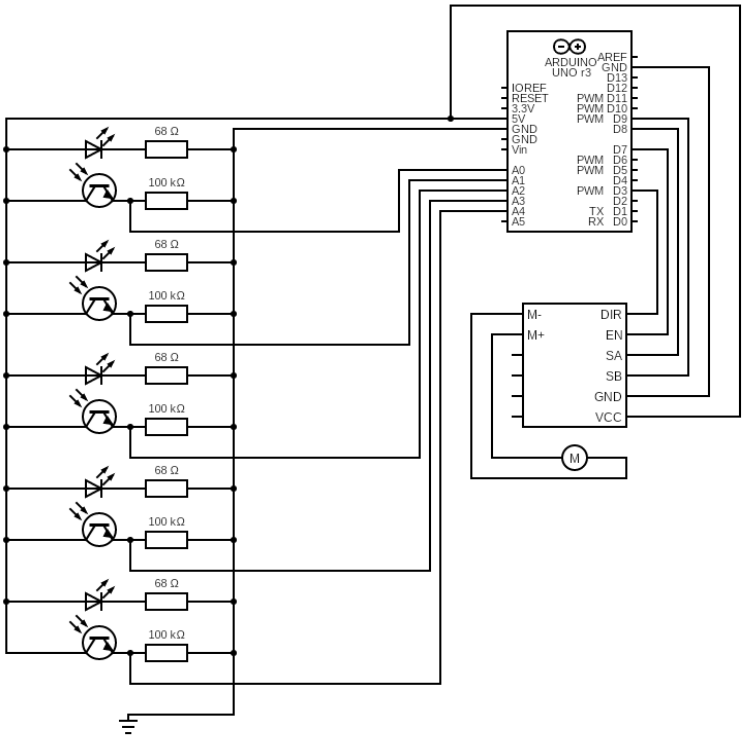


Figure 1.1.1 - Schematic of Sensor Configuration

The voltage drop across the phototransistor was measured by recording the voltage between the phototransistor and resistor with respect to ground using the Arduino's Analog-to-Digital Converter (ADC). The recorded range was approximately 0.25V to 0.5V, which corresponded to digital values in the range of 50 to 100 when comparing the reflectivity of black and white paper. As light levels changed, the voltage measured by the ADC changed proportionally, allowing for accurate capture of changes in position.

1.2 Sensor Positioning

Positioning of the sensors is important to acquire accurate readings. Given the design requirements for accuracy, it was decided that 5-bits of position data were required, and therefore five rings on our Absolute Encoder. This requires five sensors positioned within the system. Instead of positioning all five sensors next to each other on the encoder disk, we opted for a strategic placement strategy. We placed three sensors at the top of the encoder disk and positioned two sensors at the bottom. This arrangement allowed for several advantages in our design:

By distributing the sensors across the top and bottom of the encoder disk, it was possible to design the rings on the encoder disk to be slightly smaller. This optimised the use of the available space means sensor readings are slightly more accurate, as rings placed closer to the centre of the disc have a smaller circumference, and therefore more likely for a sensor of the same size to make false readings.

The positioning of three sensors at the top and two at the bottom created a scenario where the sensor housings could theoretically overlap. This design feature enabled us to better control individual sensor alignment, ensuring that the sensors passed consistently over the rings on the encoder disk.

1.3 Absolute Encoder Disk Design

The design of the encoder disk pattern was very important to achieve accurate sensor readings. To achieve the most accurate readings possible, it was decided that grey code would be used. Grey code was chosen for its property that only one bit changes at a time. This simplifies detection as bits change less often, therefore there are larger sections of the encoder that remain the same colour.

As mentioned in the previous section, it was decided that five rings were required on our encoder to achieve the accuracy required. Therefore we used 5-bit grey code. As seen in Figure 1.3.1, on the encoder disk, the outermost ring held the least significant bit, while the innermost ring contained the most significant bit. This design allows the distinct bit sequences while ensuring that the innermost ring, with its smaller circumference, features two large segments; one black and one white. This means that the least significant bit, which changes the most often, has the widest segments due to the larger circumference.

A significant consideration was the placement of two sensors at the bottom of the encoder. To obtain true grey code readings, we introduced a 180 degree offset for the corresponding rings. This offset allowed for consistent and accurate grey code value readings, aligning with the objectives of the project.

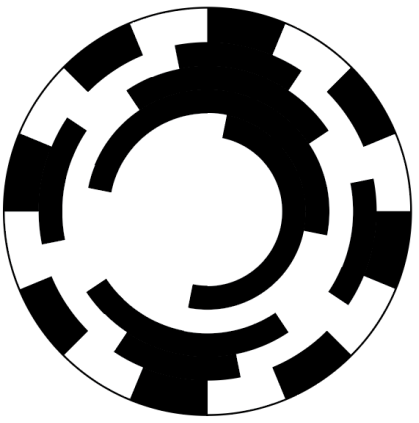


Figure 1.3.1 - Absolute Encoder Disk

Chapter 2 - Absolute Encoder Performance

2.1 Signal Processing and Code Design

To process the signals from the phototransistor, we created an arduino program with the input pins and their corresponding sensor numbers defined in the pre-setup. Additionally, for use during functions, each sensor was given a boolean that would correspond to the colour present on the wheel.

With robust sensor inputs, we realised that consistently checking for changes such as rising edges was not needed. The first step for changing the readings into a number was creating a function to convert the aforementioned booleans into the correct number. returnColour was created, which required a sensor pin number, and a pointer to a boolean as inputs. The value of the inputted pin was stored onto an integer using analogRead. Next, mentioned in the circuit design, a high number would result when white was present and reflecting, and vice versa for black. To prevent false readings, a universal upper and lower boundary were created, where if the signal passed a certain boundary, the boolean pointer would be assigned to a new value, with high readings being assigned true, i.e white, and low readings false, i.e black. The purpose of the dead zone was vital for preventing slight outlier readings from incorrectly switching the boolean leading to a wrong number.

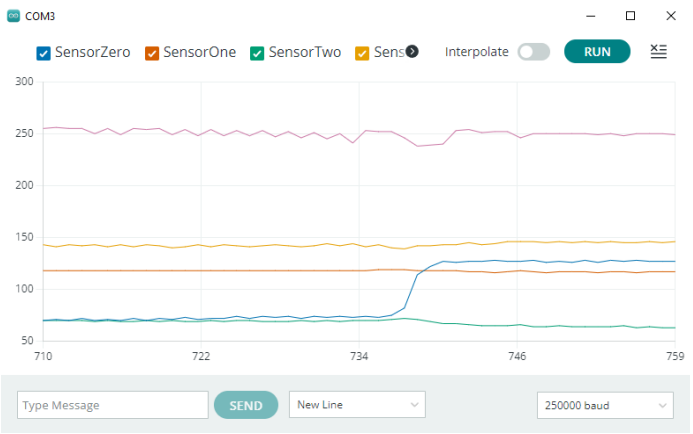


Figure 2.1.1 - Sensor Reading from a number change

However, it was found a universal boundary was too broad to encompass every photoresistor's range of values. Variations in resistors, photoresistors, strength of infrared LEDs, and small variations in the physical distances would lead to significant differences when compounded. The solution was to create unique boundaries for each sensor. Each time we wished to test the absolute encoder we would calibrate each sensor by seeing what values they would read as white and black, and assigned boundaries approximately 15 above or below to encompass an average reading. When stored on an array with indexes corresponding to their sensor number, the universal boundary could be negated, without the need to add additional inputs to the function.

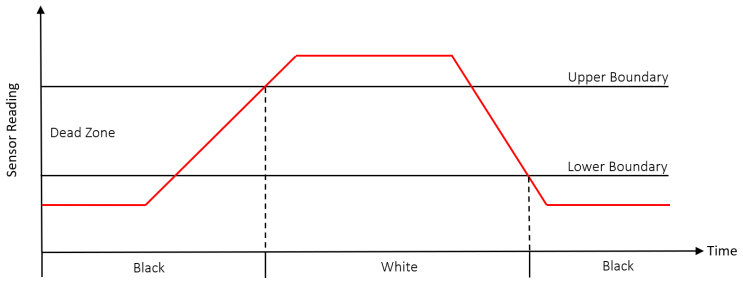


Figure 2.1.2 - Example of converting sensor readings to black or white

To return a position at any point in time, the function calculatePosition was created. It took no inputs, instead running returnColour for all five sensors, then returning the value from the

detectPosition function within itself. detectPosition, again with no inputs, converted the booleans from grey code to binary, and from binary to decimal ranging from 0 to 31. This number was then converted to angle in degrees.

Calculating the change in angle was straightforward once the encoder was returning accurate numbers. Before each spin of the motor, the initial angle was saved. Once the motor was finished turning, the final angle was saved. These two positions were passed into the calcChange function. As per the requirements of the test, no angles larger than 90 degrees were to be tested. This allowed us to figure out the four conditional situations for each change in angle which accounted for crossing the overflow point, and their corresponding directions. 'Change' refers to the final angle minus the initial angle. For example, if the change was negative, two situations are possible. Firstly, the change is less than 90, and thus represents what truly occurred. The true change is saved as the absolute of such, and the direction is set as anticlockwise. Alternatively, the change is above 90, implying that the overflow point was crossed. Consequently, the direction is set to clockwise, and true angle change is saved as the complementary angle. The same logic applies vice versa.

2.2 Sensor Testing and Performance

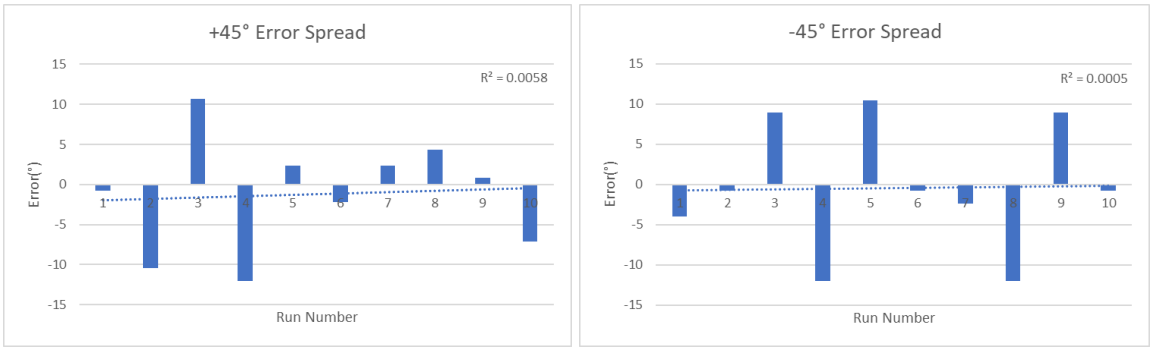


Figure 2.2.1 - Errors across the ±45° runs.

For our final demonstration, we were asked to perform angles of ±45. During our display, we had 1 outlying datum, out of 20. Shown above is a repeat of the experiment with identical conditions. All values landed within the ±14° range required to achieve maximum marks. The accuracy can be further proved by taking an average of the data points, as the shown data for +45° averaged to -1.2, and for -45° to -0.45. This trend continues no matter the inputted desired angle. The data points show a trend of being accurate but not precise, as seen by the variation in ranges and order. An R^2 of approximately 0.005 can be fitted with a trendline, which shows there is insignificant evidence of drift, or any pattern, to the errors for each run.

To ensure meaningful analysis, we computed the absolute errors when calculating the average values across various angles. Subsequently, we visualised the data within a range of ±75° at 7.5° intervals, but no discernible trend emerged. For spins with non-zero angles, errors consistently fell within the 4 to 8° range, while angles within the ±7.5° exhibited significantly higher accuracy.

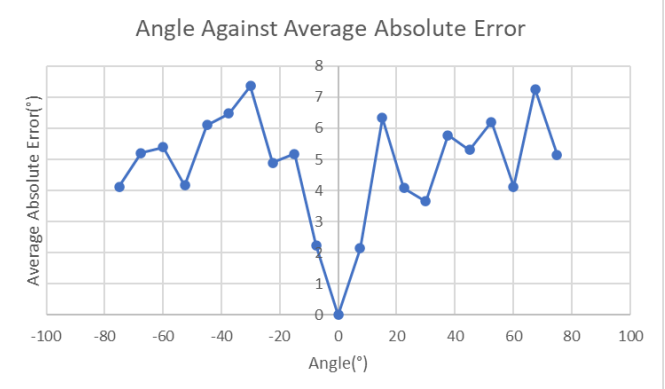


Figure 2.2.2 - Average Absolute Errors across ±75°

Chapter 3 - Quadrature Encoder Design

3.1 Sensing Circuit Design

The quadrature encoder circuit uses two infrared LED/phototransistor pairs as specified in chapter 1, each pair placed side by side in an opaque case to avoid outside light leakage. They work by the LED shining infrared light onto the disk, and the phototransistor picking up the different amounts of reflected light returned by bouncing off either white or black paper. The phototransistors are wired with a voltage divider to be sensitive to white with a 100kOhm resistor in series between output and ground as shown in figure 1. This resistor value was chosen after tests of multiple values showed 100kOhms resulted in the largest difference between white and black readings. It was decided to have this value resistor for both phototransistors as more fine tuning would have a disproportionate time cost for any effectiveness gained. The LEDs are wired in series with a 68 Ohm resistor between each LED and ground to achieve maximum brightness while avoiding excess current damaging the LED, as calculated in chapter 1.

3.2 Disk Pattern Design and analysis

The disk pattern, shown on right, is 100mm in diameter and consists of 32 segments, alternating between black and white. The segments of the inner ring are a half segment width out of line with the outer ring as a way to tell direction by which change is detected first, as well as fit more data for accuracy. The width of the segments is half that of the absolute encoder, as it was found that this allowed the inner ring to have the smallest width still detectable by the sensor, where the absolute design was held back by having sensors much closer to the middle of the disk where there is less space.
The width of the segments in the smallest ring of the absolute encoder was ~10mm, so the quadrature encoder design was based on the knowledge that the absolute encoder could pick up signals at that width.

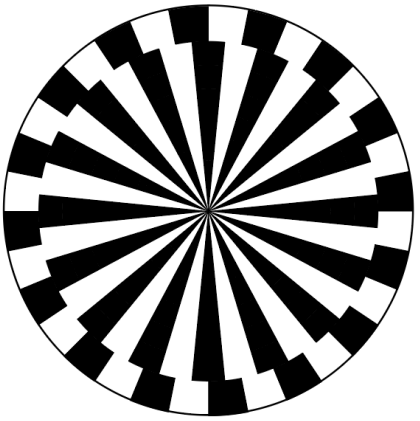


Figure 3.2.1: Quadrature Encoder Disk

$$\pi \times D = C,\ C \div no.\ segments = segment\ width \rightarrow \pi \times (100 - 12) \div 32 \approx 10mm$$

3.3 Hardware Positioning

The quadrature encoder uses the same sensing circuit placement as the absolute encoder, with only the outside two pairs used for measurement. These measure on opposite sides of the disk, as the spacing of the matrix plate holes with the cases made it impossible to fit two pairs one above the other. This means that each sensor is reading on opposite sides of the disk, however this does not matter as unlike the quadrature encoder, the absolute encoder pattern is the same whether upside down or not.

Chapter 4 - Quadrature Encoder Design

4.1 Sensor Testing and Performance

The quadrature encoder utilised two sensors, each comprising an infrared LED and a phototransistor enclosed within a sensor case. Sensor readings were obtained using the Arduino package's analog read command, which provided integer values corresponding to the phototransistor's light intensity readings. Initially, colour ranges were defined by a ±20 range around the values recorded for white and black, leading to suboptimal performance.

Issues stemmed from inconsistent sensor readings and significantly lower values attributed to the design of the 32-component disk pattern, which did not fully cover the sensor case, resulting in blended readings of black and white. To address this, a dynamic testing method was implemented to identify the precise transition points from black to white. Although this improved the accuracy for one sensor on one day, discrepancies persisted among the sensors and different times of day, likely due to ambient light conditions, paper colour, shadows, and slight manufacturing variations. Consequently, a daily calibration procedure was established, involving the serial printing of raw analog values to the monitor and subsequent adjustment of sensor reading boundaries. See figures 4.1.1 and 4.1.2 for details.

Figure 4.1.1 and 4.1.2: Serial Plot and Serial Monitor values of sensors zero and one as the disk rotated.

The calibration process involved analysing the peaks and troughs in the figures, representing black and white colours, respectively. Noting the similarity between the figures, a single boundary condition was chosen to apply to both sensors.

4.2 Signal Processing and Code Design

To streamline operations and minimise complications, a simplified code design technique was adopted, leveraging a method commonly used in motor encoders. By relying on the offset of the encoder disk, direction calculation was executed, ensuring that any errors were primarily attributed to hardware and connections rather than the code. This technique involved polling the sensors, with sensor one peaking prior to sensor zero during clockwise rotation, as depicted in Figure 4.1.1. The code was programmed to monitor sensor zero's transition from black to white, subsequently checking the status of sensor one. If sensor one indicated white, a clockwise rotation was confirmed; conversely, a black reading signified an anticlockwise rotation.

To determine the motor speed, a counting method was implemented. This entailed incrementing an integer count each time sensor zero changed its state, also achieved

through polling. With 32 counts per revolution and the count incremented over a 5-second interval, the RPM (Revolutions Per Minute) was calculated using a specific function at the end of the 5-second period. The count was then reset to zero to prepare for the subsequent interval.
Count to RPM equation:

$$speed = count \times \frac{12}{32}$$

4.3 Encoder Performance

The evaluation of the quadrature encoder's performance involved a comparison with the motor encoder readings displayed in the Serial Monitor. The error was determined by the formula:

$$error = optical\ encoder\ measured\ value - motor\ encoder\ measured\ value$$

To comprehensively assess the encoder's performance, accuracy and precision were calculated across a range of selected RPM values. Two accuracy metrics were employed: Accuracy % (1) was calculated as the Mean Error divided by the True Value, while Accuracy % (2) was determined as the Mean Error divided by the Full Span, both expressed as percentages.

Figure 4.3.1 illustrates a decrease in accuracy at higher speeds, attributed to the increased likelihood of the optical encoder missing counts, resulting in higher errors. Despite a favourable resolution of 0.375 RPM per count over a 5-second interval, the escalating speed induced a proportional increase in error, leading to significant deviations.

Furthermore, repeatability analysis was conducted for the 200 RPM case, which represented the majority of the tests performed. The calculated repeatability was determined to be 0.14% RPM, indicative of highly consistent results, with minimal deviations observed except for occasional outliers. A comparison of mean error for clockwise and anticlockwise tests showed minimal change, therefore it was concluded that the direction was not a factor in determining accuracy.

	100 RPM	125 RPM	150 RPM	175 RPM	200 RPM
Accuracy % (1)	1.00	0.73	0.64	0.65	0.88
Accuracy % (2)	0.28	0.26	0.27	0.32	0.49

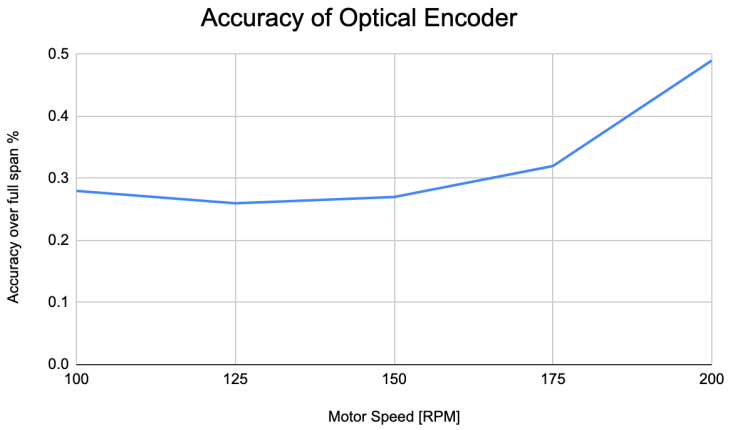


Figure 4.3.1: Full span accuracy over 100 RPM range

Chapter 5 - Health & Safety by Design

The health and safety considerations of our optical encoder project are critical, necessitating the minimization of risks to individuals' well-being and the implementation of robust operating procedures. This section meticulously delineates safety aspects, particularly focusing on electrical safety, infrared radiation, and mechanical hazards. Furthermore, we scrutinise the applications of our encoder, emphasising potential risks associated with inaccuracies in specific applications, such as a robotic arm.

Our project significantly prioritises electrical safety by utilising low-voltage DC for all electronic components. This approach effectively mitigates electrical risks, and the incorporation of circuit protection measures, including overcurrent and short circuit protection in the power supply, further ensures secure operations, safeguarding both equipment and personnel.

Given the inclusion of infrared LEDs, we have carefully considered the safety implications of infrared radiation. While the risk is minimal, we have communicated the potential risks to team members and instructed them to avoid direct eye exposure to infrared LEDs. To address potential overheating issues, we have regulated the current supplied to the LEDs below the specified forward current outlined in the data sheet.

Mitigating risks associated with moving components, such as the motor and rotating disk, is crucial. To address this, we propose the implementation of an enclosure to house all moving parts, preventing inadvertent contact that could lead to injuries. This enclosure also serves to avert contact with electronic components, minimising the risks of electrical shock or burns.

Considering the intended application in a robotic arm, particular attention is given to addressing potential errors in the absolute encoder. We recommend incorporating touch sensors for redundancy to prevent over-rotation, regular calibration procedures, and a secure holder for the sensors to prevent misalignment. These measures aim to prevent errors that could result in hazardous arm positioning or damage, ensuring the safe and accurate operation of the robotic arm.

The Engineering New Zealand Health and Safety practise note states that health and safety by design is an intentional and deliberate process undertaken to consider what can practicably be done in the design to eliminate hazards, reduce risks or otherwise minimise the potential for harm throughout the product or asset lifecycle. This overarching philosophy has been meticulously observed, ensuring an understanding of operations, thorough hazard assessment, strategic design modifications to mitigate risks, clear communication of remaining risks, and comprehensive documentation of decisions for assurance.

I, Eloise Beattie, wrote the Quadrature Encoder Performance section of the report, the quadrature code and managed final code edits and wrote part of the Health & Safety by Design section of the report.

I, Luke Hynds, wrote the Absolute Encoder Design section of the report, calculated required resistors, circuit design, absolute encoder disk design, quadrature encoder disk design, general debugging, and wrote part of the Health & Safety by Design section of the report.

I, Oliver Reedy, wrote the Absolute Encoder Performance section of the report, wrote the signal processing code, the initial testing/base code, and laser cut a disk with better tolerances.

I, Otto Walker, wrote the Quadrature Encoder Design section of the report, assisted in the design of the encoder disk patterns, assembled the physical components, and consulted on code design.