Department of Mechanical and Mechatronics Engineering

MTE 201 – Term Project

Lego Block Measuring Device

Instructor:
Peter Teertstra

Armaan Sengupta	20991907
Armaan Rasheed	20993642
Aarush Jain	21024818
Andy Zhang	21006293

November 13th, 2023

Overall Design:

Summary:

Our measurement system measures linear distance by converting rotary motion measured via an encoder to linear distance. In this case, the sensor is the rotary encoder, the signal modification system is the microcontroller the encoder is connected to, and the indicator is an LCD screen controlled by the microcontroller. The device has several auxiliary features that don't directly assist with the measurement itself, but make the project more practical in the scenario it was developed for automated factory tolerance inspection. Our team designed the measurement system to be used in an industrial automation line to perform quality assurance validation on parts without human interference, this is described throughout the remainder of this document.

Theory of operation

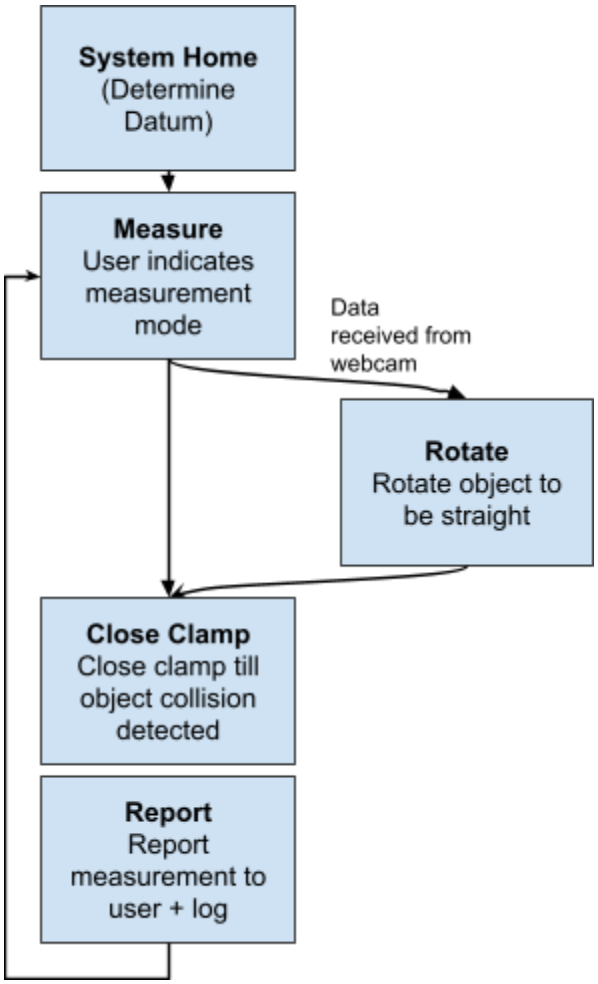

Figure 1: a flowchart describing the theory of operation of the general procedure

t

Above a flow chart for the system's operating procedure can be seen.

Assumptions:

1. The LEGO block is going to be hand-placed onto the rotational plate
2. The machine has to be manually turned on from the V5 GUI
3. For very long components that need length to be measured, the user would place the object length-wise as it’s common sense
4. For multiple objects, the platform is not used and the user would orient however they want

Mechanical:

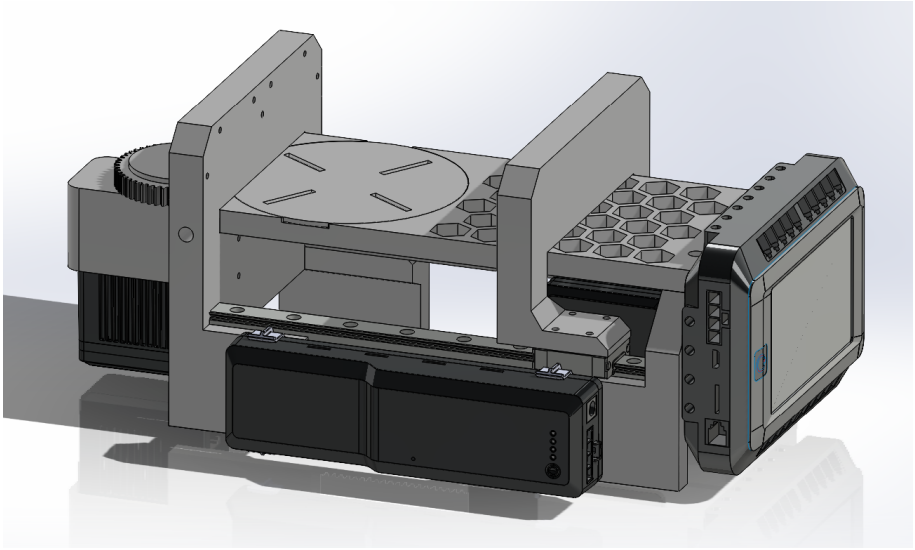

Figure 2: isometric view of the machine assembly

The overall design of the machine has 2 main systems, a linear motion system and a rotating platform system. This stemmed from brainstorming ideas to make a “vice” based measurement machine applicable from an engineering point of view. Adding the ability to rotate and measure a singular part allows for automation in the production line. Composed of mainly 3D printed PLA components, and using the VEX V5 Ecosystem, it allowed us to bypass the electronic scope of the project and focus on making the mechanical measuring systems and the software associated. Two days of quick CADing resulted in this final design, and leveraging the 3D printers owned by group members, all the necessary pieces were manufactured within a day. A lot of considerations went into tolerances, with the primary goal to keep it as tight as possible to improve accuracy when measuring. This resulted in a few redesigns and re-prints since all the pieces were very tight fitting. The whole machine is assembled using common M3 hardware and VEX-based imperial standard screws.

Linear Motion System:

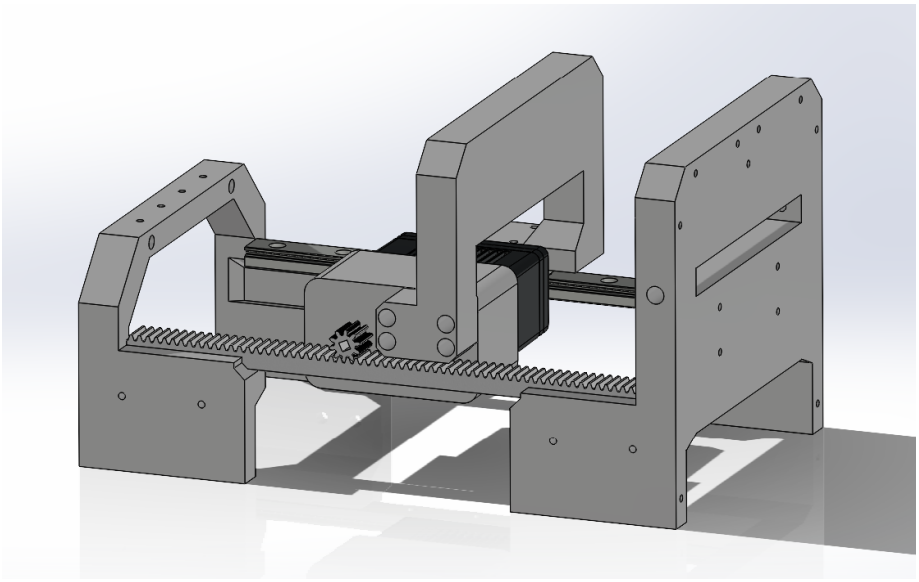

Figure 3: view of the linear rails and rotary encoder

Using a linear motion rail as the base of the design and with the constraint of the rotating platform, this is the final iteration of the motion system. Using only one rail on one side and a rack and pinion on the other, this unconventional design allowed us to minimize design complexity and allow the functionality of a rotating platform that won't interfere with the motor carriage.

Rotational System:

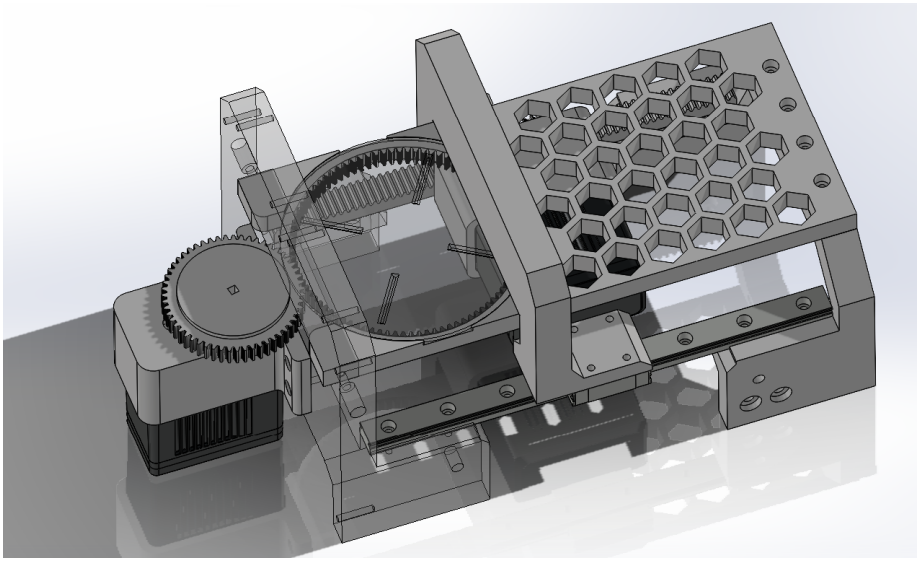

Figure 4: view of the rotational plate and servo attached to it

To make a clean, hidden rotating platform, a cutout was made into one of the ends of the vice-based machine and allowed a simple spur gear system to be integrated. Playing around

with the design allowed the gear disk to be self-supporting and the gear to be fully hidden within the end plate. Originally, the motor was to be placed right below the platform to allow ease of implementation but that caused clearance issues with the linear carriage. Thus the gear system was chosen in favor because of its external placement. This system utilized customs gears of 55 and 90 teeth and was designed using a spur generator.

Camera Vision Mount:

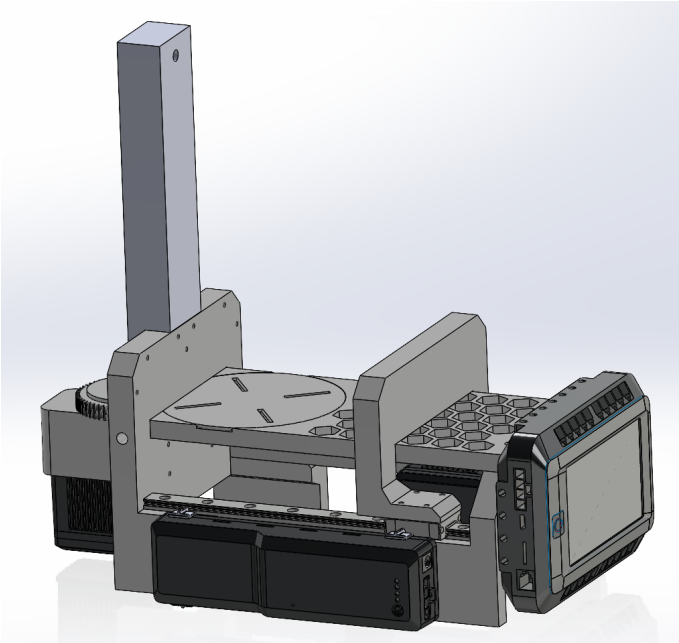

Figure 5: Long stem protruding from assembly shows camera mount for CV

The camera mount was designed considering the field of view of the camera and basic trigonometry was used to determine the height required for the entire rotating platform to be within view. A simple elevated rod with a counterbored hole to mount the camera via a single M6 screw was added to ensure the camera's position did not deviate and fine-tuning regarding (more details in software) could remain consistent.

Embedded Software:

Collision Detection:

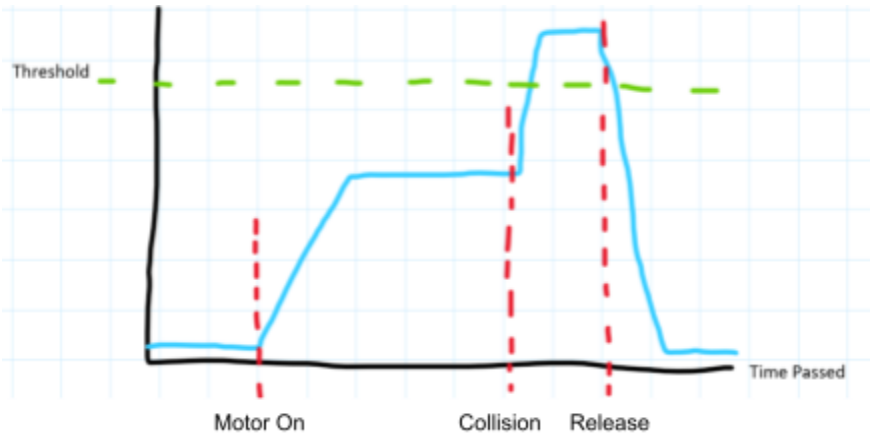

Figure 6: shows the communication protocol used for three-wire communication

Collision detection is quintessential to the operating of the system as it is required both for the homing of the system (determining a reference frame of zero distance) as well as measuring the length of the object. This is because we are not using any kind of limit switch which might have introduced further variability in the measurement of objects, and would have limited the size of objects we can measure. Thus we choose to utilize a method that would not require the external sensor to contact the object itself. By directly measuring the current that the motor draws we can detect a current spike, which indicates a sudden change in torque of the motor due to additional resistance; this can be attributed to colliding with an object. By running various trials we determined a threshold which separated normal operating conditions and object collision with 100% accuracy (over 45 trials). This technique was then used to implement “homing” the system, which involved it detecting a collision between the sliding plate and the fixed back plate to determine a datum of zero distance, as well as during each measurement to detect collision between the sliding back plate and the object.

Communication protocol:

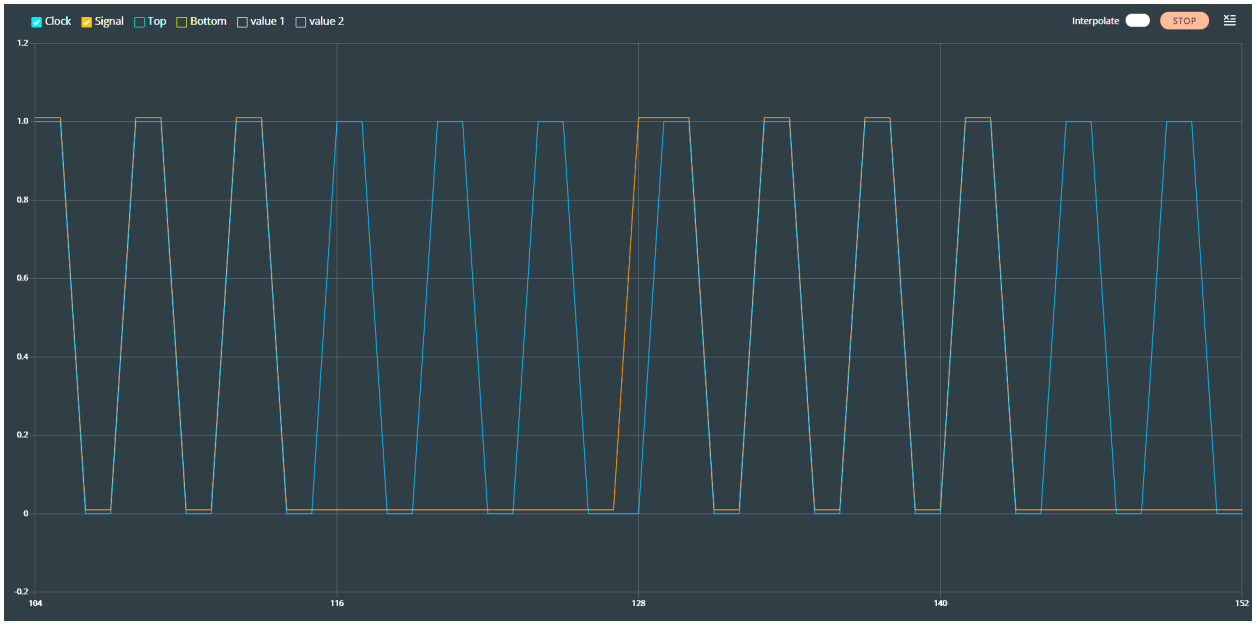

Figure 7: serial plotter describing the serial exchange between ESP32 and V5 Brain

The VEX V5 brain is required to receive data from a PC regarding what angle it should rotate the object to (the PC is using computer vision to determine this, more on this below). However, because the VEX V5 brain uses proprietary software and firmware, standard communication protocols like UART, I2C, SPI, etc… are not available on the brain. The only thing that was available was the ability to read a wire as being a digital high, or digital low (0 or 1). Based on this our team worked on creating a custom communication protocol such that an ESP32 could be used as a middleman to wirelessly receive data from the PC and then send that data over 2 wires to the brain using our custom uni-directional, peer-to-peer, communication protocol, which was loosely inspired by I2C, carrying over the concept of a clock and signal line. We began by converting our angle into binary, and then transmitting it using our communication protocol (90-degree angle shown in the image above) to the brain, and then decoding this signal back into motor encoder ticks for the turntable motor to rotate the object to the correct orientation. There were several challenges, and innovations made regarding this custom communication protocol, though the details are out of the scope of this project. The result however is the system is able to receive an angle input from a PC wirelessly thereby being able to rotate an object to its correct orientation.

User Interface:

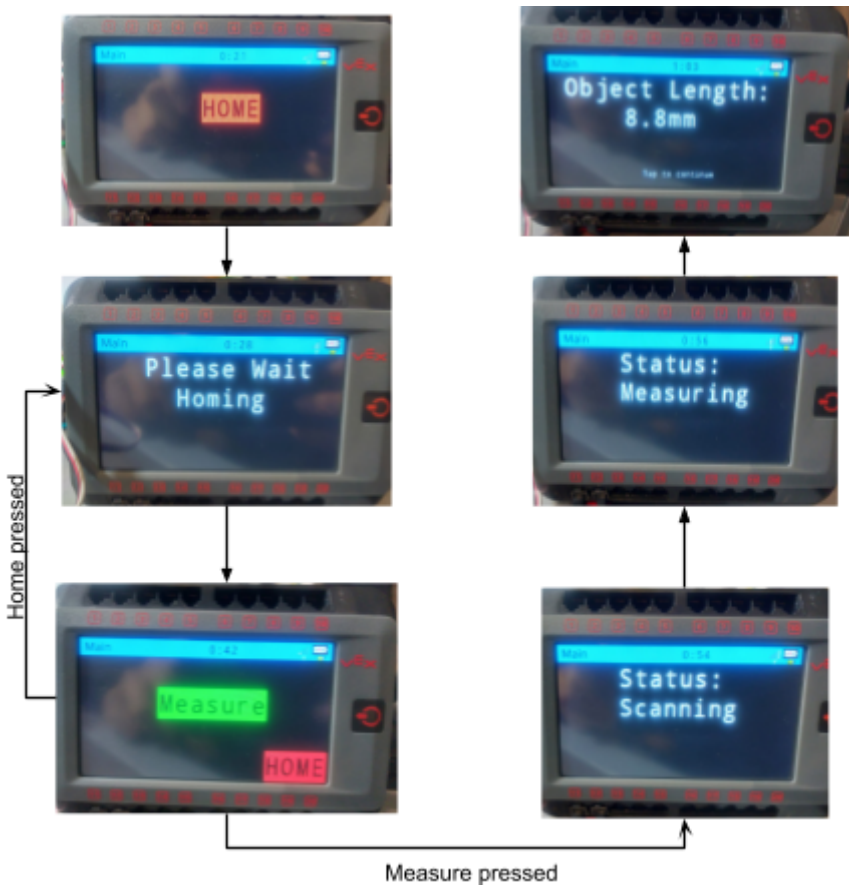

Figure 8: GUI on the V5 Brain to start and stop

A user interface was added to control the machine. A basic button and message object were created and reused that took advantage of the brain's capacitive touchscreen interface. While this interface was limited in what kinds of graphics could be displayed, it was sufficient for us to be able to effectively switch between all the operating modes of the system as shown in the flow chart above. Everything was written from scratch, including pixel rendering for objects and determining if objects were clicked using cartesian coordinates of touch inputs and corresponding saved object locations.

Computer Vision:

In order to place the Lego block in the correct orientation for the encoder to start measuring its distance, computer vision software was developed with the help of the OpenCV library to help identify the placement of a Lego block and then compute the angle the block is positioned with respect to the horizontal axis in the camera's frame of reference. The process is fairly simple and straightforward and begins with the camera positioned directly above the base plate looking down as part of a bird's eye view and operating on a live feed. The current algorithm running through the master computer is waiting to detect any non-moving rectangular objects that will come into focus. After a block is slid under the focus of the camera and comes to a complete

stop, the camera is able to detect this using object detection functions such as applying Gaussian and gray filters and then computing the contours to detect shapes. After the software is able to detect the block, it writes the frame at that exact moment to a JPG file and then stores it in its current working directory.

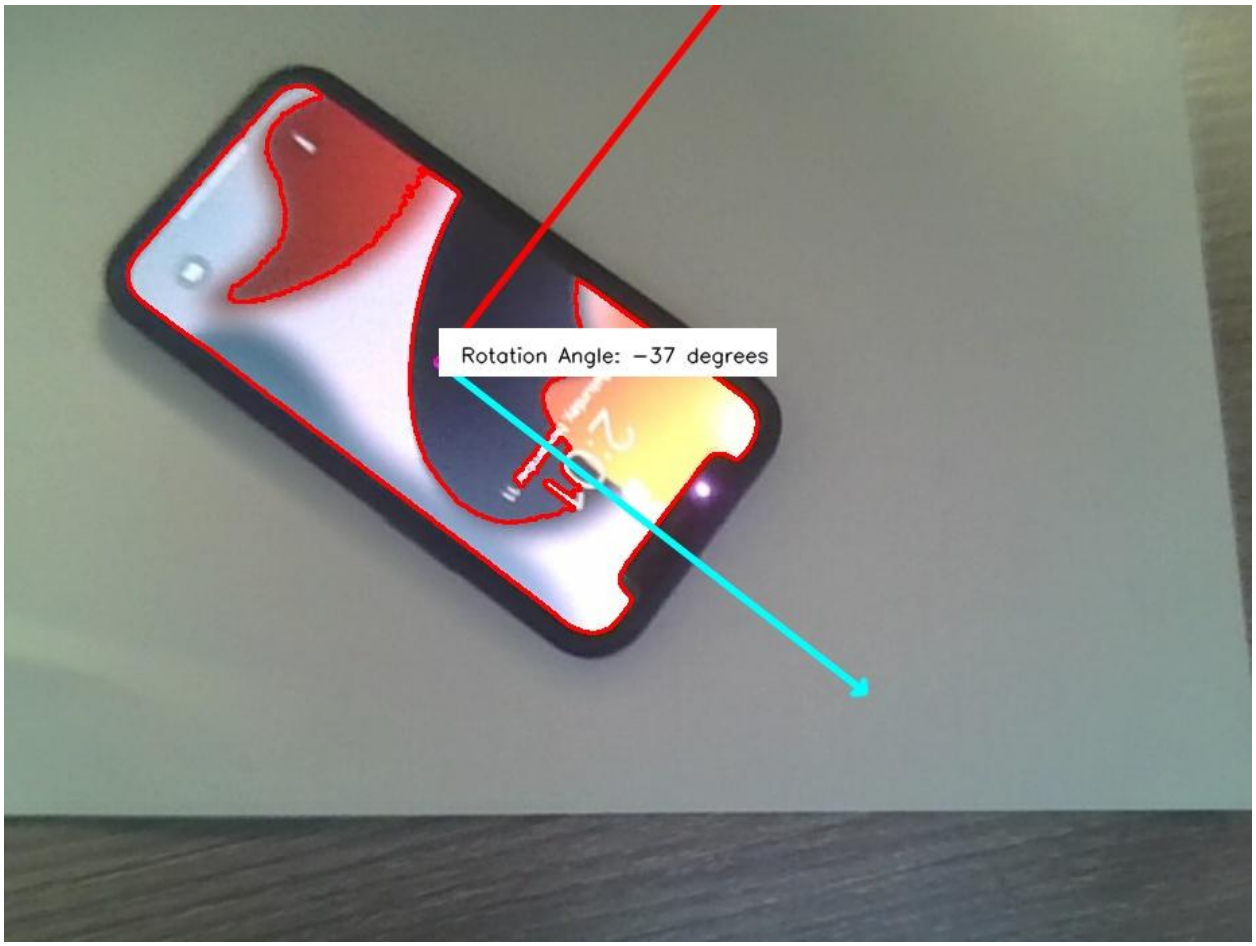

Figure 9: A sample demonstration of the angle of rotation computation of a phone acting as a block

After this process, the software proceeds to compute the angle the block makes with the horizontal axis with respect to the camera's frame of reference. The OpenCV script first converts the frame to a binary representation via a threshold function and then proceeds to find all the contours. It then iterates over all the contours calculated and calculates the area under each contour while simultaneously removing contours that are either too large or too small. After drawing each approved contour on the frame, the orientation is computed using a method called Principal Component Analysis where the data points of each contour are stored in a 2D array and computes the mean, eigenvectors, and eigenvalues of each set of points using the built-in PCACompute function. Afterwards, the program determines the two directions (eigenvectors) where the eigenvalues are highest, alluding to the highest variance of data. Afterwards, the program computes the angle between the two largest eigenvectors and stores it in a global

variable which is sent to the ESP32 microcontroller board via WebSocket for further processing and rotation.

Networking Architecture:

The design choice behind picking an ESP32 was the ability to use its WiFi capabilities to our advantage. It was configured to be a soft access point so that devices could connect to the ESP32 on a local area network. The backbone of our data acquisition system comprises an OpenCV script and server.py script which produced and forwarded an angle value to the ESP32. Once the ESP32 had recognized the laptop hosting the computer vision as a client within its network, it would receive an angle value from the scripts and store it onboard the ESP32. Next, the ESP32 would serially communicate with the V5 brain to convert this angle value into a servo value which would orient the LEGO block at the correct angle and prepare it for pushing. Furthermore, the ESP32 when acting as a soft access-point uses basic HTTP protocol to communicate with the laptop over the hosted server!

Calibration/Analysis:

Calibration Data & Calibration Curve

The calibration procedure consists of the mid-plate pushing all the way to the CV mount and zeroing at this position. Once it has reached this position, it will start going backwards till it reaches a calibration distance of 108mm at which point it will stop. The machine is now calibrated and ready for use. Data was recorded on how many times the encoder rotated versus the real distance measured using a digital caliper as can be seen in the table below

Table 1: Calibration Data

Degrees (Measurement System Output) (mm)	Caliper Reading (Real Value) (mm)
176	7.59
265	14.42
352	22.55
444	30.13
482	34.44
523	38.66
571	40.52
630	47.69
704	53.96
832	65.16

860	67.35
926	73.25
1003	79.87
1061	85.28
1033	82.3
465	32.14
567	39.65
789	58.94
980	76.65
1009	79.95

This data can then be plotted to yield Figure 10

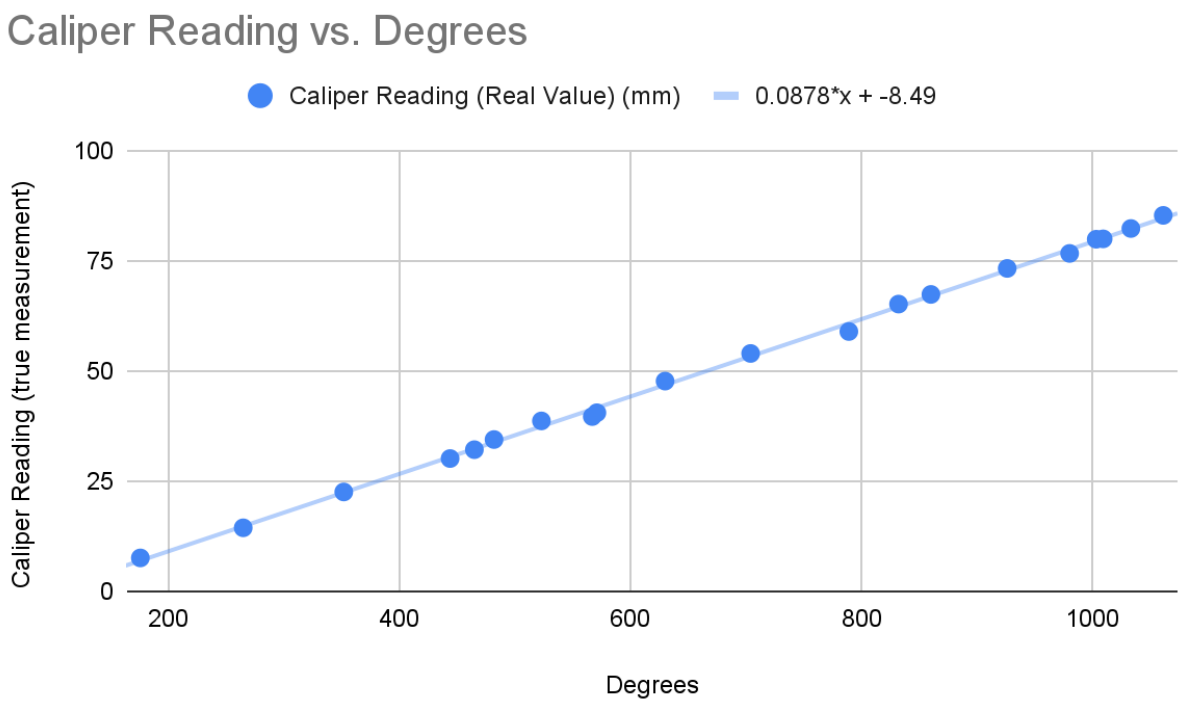

Figure 10: graph displaying caliper reading vs degrees for experimental data

Based on this, and our knowledge that the relationship between the two values is linear, a line of best fit can be used to model the data. We can see that there is an offset value (non-zero b term), which does not quite make sense because we know that an angle of zero degrees means zero inches. This is a result of inaccuracy within the data.

We can now plot the deviation for each distance to get Figure 11

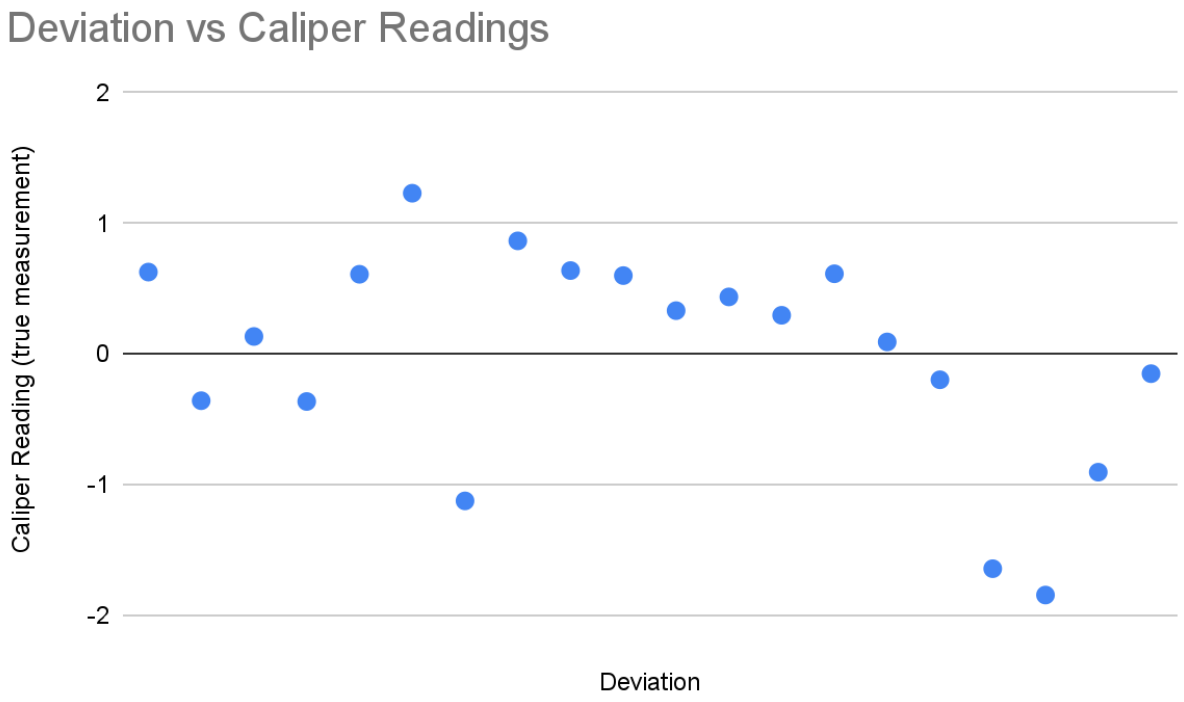

Figure 11: Deviation VS True Measurement

The estimate of the maximum uncertainty value is 1.8442mm based on the maximum deviation recorded between all the trials as can be seen on the graph above.

Evidence of systematic and random errors

The fact that we have an offset of 8.49mm, meaning that the calibration equation thinks zero degrees is -8.49mm is evidence of systematic error in either our instrument itself or, in the method we used to gather the "real" data. Random error is minimal as indicated by a R^2 value of 0.999 meaning the random variation within data samples was low enough to be approximately linearly, and since we expected a linear relationship, it stands to reason that the precision of the device is high even if the accuracy might not be.

Conclusion:

Reflecting on the overall design of the machine, there were many improvements that we could make to improve accuracy, user experience and functionality. The mechanical components were made of plastic, which is flexible and would create inaccuracy if there were any loading forces during operation. It would introduce deformations and skew the real measurement. Ideally, all the parts should be metal, and the mid plate that moves should have 2 linear rails; one at each end. In addition, having a sensor to detect collision would also greatly increase the accuracy, as

the system no longer relies on when the motor detects a current spike from hitting the object. This allows the system to stop more accurately.

For the firmware that controls the machine, it was simple and had no major issues that affected performance. If given more time, the user interface could use an overhaul and be designed to look more modern and similar to production line machines. Since our goal was to make a very applicable device, we wanted to have lots of opportunities to implement more functionality via software. This resulted in us running into a lot of issues, finding alternatives and scrapping features. The problem with using serial communication between a microcontroller and a computer for our use case was that we needed multiple files communication over the same “COM4” PORT. For example, the *server.py* script will be sending an angle value to the ESP32 over the COM4 PORT and then, the ESP32 would be using the same COM4 PORT to transmit information to the VEX-V5 brain. This is not possible due to the port already being busy because of the communication between the ESP32 and *server.py* script. Hence why, we opted to make the ESP32 a soft-access point for all devices except the V5 brain. In addition, we planned lots of features that could be very useful when implemented properly. If more time was given, we wanted to have a system that would be able to utilize the CV camera to recognize the sides of an object and project its measured dimensions onto a snapshot. This is a simple concept, but it could prove to be useful if this feature is implemented in a phone and caliper for example.

Overall, the project was a good learning opportunity for mechanical and software design, it challenged the group to design an applicable engineering device within the scope of the project. The timeline of the project also challenged everyone’s ability to work in parallel to design, manufacture, assemble, and program the device. It also allowed the group to explore and use the 3rd year mechatronics design studio and get familiar with their equipment, which is definitely helpful for future courses.