Product Development File

1. Specification and Planning Phase

(a) Statement of Need / Problem Statement

Design a device which can differentiate good from bad tesseracts and translate them to the appropriate location in either mode of operation.

(b) Product Design Requirements (Understanding the problem)

- Must fit in an MSE locker
- Must be able to move through a space with dimensions (18 cm by 25 cm)
- Must be able to locate good tesseracts
- Needs to keep track of its position on the field
    - Either in absolute terms or relative to some marker on the field
- Must be able to locate and move tesseracts in under 40s.
- Must move
- Muset locate tesseracts
- Must verify good tesseracts
- Must keep track of position in the “arena”
- Must fit under the gate
- Must be repaired/modified quickly
- Must work in 2 modes
- Must reach the underside of the gate
- Must fit between the bars of the gate
- Must fit in a locker
- Must work in varying arena sizes
- Must work in room 3109

(c) Program Plan (Planning and scheduling)

- Code must use procedural programming
    - Functions/procedures will keep the code streamlined, making logic writing/coding easier on the designers
- Subroutines should exist to avoid crashing into other robots
- Avoid delays in the code (time is money)

2. Conceptual Design Phase

(d) Conceptual Design Development (Concept generation)

See the Quality Function Deployment (QFD) included in the Product Development File folder.

How to manipulate tesseracts:

a) Gripper arm
b) Magnet

How to lift tesseracts

a) Arm
b) Scissor lift

Drive base:
Round drive base

Motor driving:

a) Servo motors
b) Stepper motors

(e) Concept Evaluation (Concept selection)

Methods of evaluation included:

1. Voting/Consensus after considering the pros and cons
    a. Agreed by consensus that, if using a gripper arm to pick up the tesseracts, we would not use the VEX claw arm.

3. Product (Detail) Design Phase

(f) Product Generation (Refine design concepts)

Using an arm with a claw attached to it would enable straightforward manipulation of tesseracts, but since it doesn't have any natural guards, it could pick up non-magnetic tesseracts. Using a magnet to pick up tesseracts would only pick up tesseracts with sufficient magnetic flux but not the bad tesseracts, which would eliminate the possibility of manipulating bad tesseracts. That makes this idea better than the gripper arm.

Using an arm would allow us to pick up tesseracts with reasonable dexterity. Using a scissor lift would make it easy to lift the tesseracts up to the bar in mode 2 without taking up a lot of space.

The End Effector

Many different concepts were considered when designing the end effector for the device. Using a claw as an actuated gripping mechanism was the first idea examined, since this was the actuation method employed during the previous design labs. The group quickly agreed that any gripping mechanism employed would have to improve upon the unreliable performance of the Vex Gripping arm (check name and add trademark), which often allowed the flag to slip during previous lab exercises. The flaw with this device was that the curved pincers were spaced far enough apart that the flag, which was just larger than the opening in the pincers, could be jostled and dislodged from the pincers. It was decided that if we were going to move tesseracts in this way, it would be most appropriate to redesign the claw. Using a magnet to pick up tesseracts was the second idea that we examined, because it could not pick up bad tesseracts. The difficulty we faced was in the selection of a magnet, since we did not know what field strength characteristics we would need to successfully pick the magnet off the sheet metal arena.

Designing the end effector involved more than choosing between a magnet and a claw. Once the group had chosen to use a magnet, it was imperative that we find a way to house the magnet in such that the good tesseracts would only stick to the intended face of the effector, the line tracking infrared (IR) sensor could be moved to scan the arena's edge, and would allow us to easily use the Hall Effect sensor to verify the location of a tesseract during the scanning process. Intelligently incorporating these components would simplify the programming and movement of the mechatronic device.

During the conceptual and prototype design phase, Tyler and Chris diverged in their implementation ideas for the end effector. Tyler's idea focused on using servo in combination with lipstick-tube mechanism to translate the rotational motion into linear motion. Tyler's concept used a cap to allow the magnet to rotate freely so that it could orient with either pole, allowing it to pick up magnets sitting on the arena's edge in any orientation. Chris's idea focused on using a servo's rotational motion with a wire to create a piston. Chris's connection to the magnet was planned to be something similar to a cap, but for prototyping, used one of the magnet's poles to attach to the connecting rod. To decide between the two mechanisms, the group compared the good and bad elements of each design.

	Chris's Method	Tyler's Method
Pros	● Easily adjustable on the servo arm ● Classical fixturing mechanism	● Magnet stored in the cap will allow for free rotation ● Innovative fixturing

		mechanism

Cons	• Magnetic rod between the servo and magnet fixes the pole of the magnet by connecting to the rod
	  ○ Makes picking up tesseracts off a magnetic surface harder
		• Not easily adjustable
		  ○ Length is fixed and dependent on lipstick tube length
		• Magnet height change is restricted by the servo rotation
		  ○ Low change in height vs. rotational range of the servo

Given the advantages of both designs, a hybrid design taking the best elements of the two was selected for the prototype and for the final design. The hybrid design will use a servo-piston mechanism - labelled servo_magnet in the code - to drive the magnet holder. The magnet holder will allow the magnet to re-orient to match the tesseracts. The magnet holder will sit inside an outer shell to keep it aligned with the bar connecting the mechanism to the rest of the arm. The bottom of the end effector will also have a small circular plastic piece to prevent tesseracts from contacting and sticking to the side of the outer shell. On this circular plastic piece, the Hall Effect sensor will be connected

(g) Product Evaluation (Product design vs. requirements)

Mode 2

When writing mode 2 the most challenging aspect was determining the location of the robot with relation to other objects. To determine the position the robot would drive forward until the ultrasonic sensor detected the uprights of the goal post. It would then drive back straight for a set distance using encoders to get behind all of the potential tesseract holding places. Because of the unreliability of encoders the robot could not rely solely on encoder counts to drive it straight forward and straight back. Due to this unreliability the robot was made to use a wall follower code half the time to correct for any deviation caused by encoder counts.

After driving straight back, the robot would recalibrate the hall effect sensor by taking several readings and averaging them. It would then set the bounds of the hall effect sensor that would

indicate a tesseract was found. Then the robot would move forward using a wall follow code with the ping sensors. The robot would drive forward over all tesseract holding areas with a Hall Effect sensor continuously scanning over the tesseract area. If the Hall Effect sensor went out of the specified range the robot would stop, move forward a set distance until the pickup device was directly over the tesseract. The magnet would be extended and the arm would move to pick up the tesseract. Next the arm would be positioned to fit under the gate with the tesseract in tow. It would then drive forward following the wall until it reached the end of the wall, the drive straight function would then be called to move ahead a set number of encoder counts to get in position to place the tesseract on the top piece of the gate
If a tesseract was not detected the robot would continuously move back and forth over the tesseract loading area - as was shown during the showcase - without losing its place.
During the showcase the hall effect sensor had too high of a sensitivity which prevented it from detecting tesseracts, unfortunately during testing the sensitivity was always fluctuating making it hard to find a set value to use for sensitivity.
For the second run of mode two the sensitivity was adjusted and the magnet was detected. Unfortunately the magnets along the side of the wall in the course were much stronger and closer to the tesseracts than predicted by the group. In the future the group should ask for clarification. This prevented the actual pickup of the tesseract due to the stronger magnetic force pulling the tesseract to the loading area. During testing for mode 2 only one magnet was used to "hold down" the tesseract. With only one magnet the robot was easily able to pick up the tesseract with our much larger and stronger magnet.

The robot would be potentially moving three tesseracts; in order to ensure that the tesseracts were not overlapped there were three separate placement areas programmed into the robot.
A code was in development to make the robot parallel to the wall using the ultrasonic sensors on the side of the robot. This would have been implemented into the main mode 2 code at some areas, and would potentially simplify the code.

Lipstick idea

After applying lipstick it became apparent that the lipstick worked very similar to a linear actuator that could be used to move the magnet up and down for pickup and drop off of the tesseract. All that would be required given the materials readily available would be a servo to turn the base and a mount for the main shaft of the lipstick. The mounted main shaft would be held in place while the servo twists the base of the lipstick to extend and retract the magnet.

Through some experimentation it was found that the 180 degrees that the servo offered would move the magnet far enough in both directions that it would both pickup and dropoff the tesseract. In this way the lipstick model satisfied the requirements; however it was difficult to mount the servo in such a way that it would carry out the task. Another

limitation of this design was the amount of vertical space required for the entire unit. As the magnet was too large to fit in the actual lipstick shaft, an extension piece had to be added increasing the height of the unit. The height of this entire unit was too much given the limitations of our end effector shaft length for placement of the tesseracts.

Linkage vs. String

When considering how to move the magnet up and down to manipulate the tesseract, the group considered two methods: a string and a linkage system. While the string would be very simple to use with the servo and may provide a larger variation in height of the magnet, the location of the magnet in the tube would be unknown. With a linkage system the magnet could be forced both up and down instead of just up as was the case with a string. This offered a significant advantage by allowing the group to move the magnet to an exact location. With the string it was always a range of 0 to the length of string let out, which was not ideal for manipulation of the tesseracts.

Scissor Lift vs. Arm

The main decision when deciding between an arm and the scissor lift was whether the group wanted to manipulate two systems or one. With a scissor lift the height could be easily attained for mode 2 and only one motor would be needed to lift it; however there may be some difficulty in placing the tesseract directly on the scissor lift after picking it up from the wall. In order to make a scissor lift that could be moved by the motors in the lab, the scissor lift would have to be made of light parts without a lot of friction. To reduce friction parts would probably have to be loose which may introduce deviation from the straight path up that was planned. Also keeping the tesseract directly on top of the scissor lift could introduce a problem given the tesseract may slide with the slightest angle of the scissor lift platform.

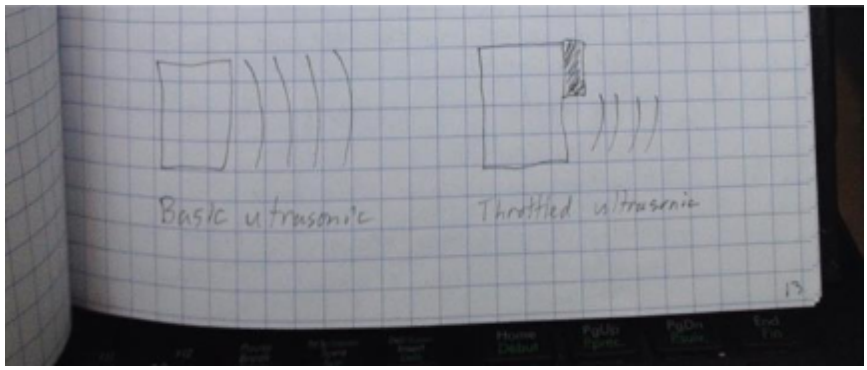
With the arm more motors would be involved however there wouldn't be a worry of the tesseract slipping. Also putting the tesseracts in 3 different places would be greatly simplified with an arm. When using the arm the greatest drawback was the play in the vex gears which was greatly amplified the further it is from the horizontal. With the arm more play would be involved however compensating for the play would be much simpler than compensating for play in a scissor lift.

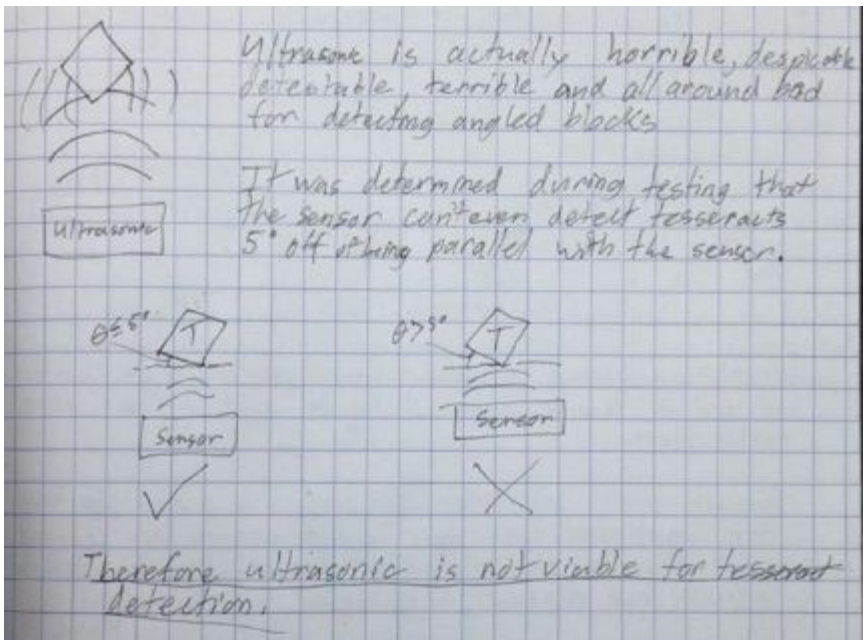
The consensus was that the coding would be greatly simplified if only one system was used to perform all tasks. This led the group to decide on using a jointed arm over a small arm in conjunction with scissor lift.

Ultrasonic Sensors

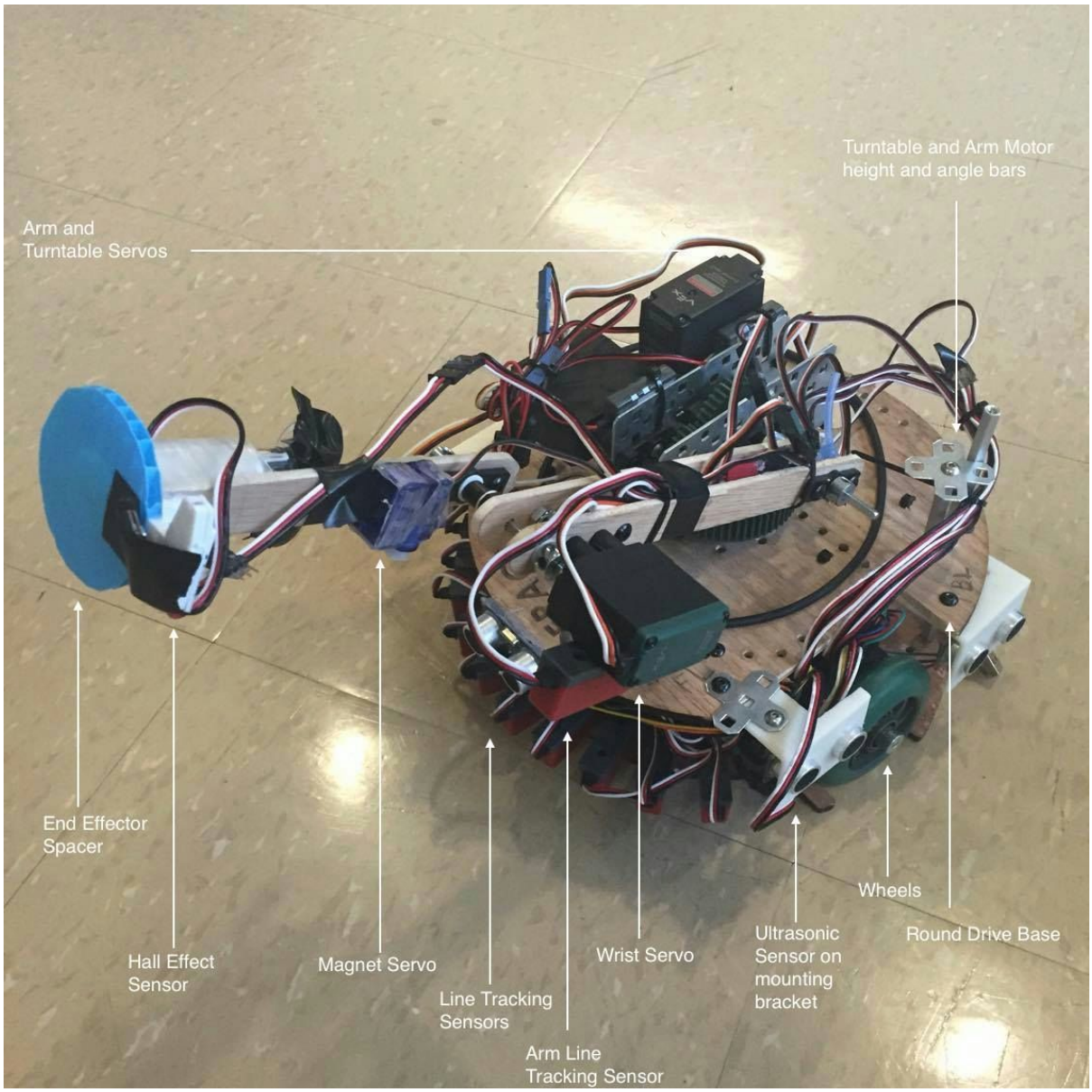
Originally the group thought to locate the tesseracts using an array of ultrasonic sensors, however the group quickly discovered that they were unreliable. There are multiple reason

for their lack of reliability. The first reason is that the ultrasonic waves fire in a cone shape making them spread out in a 40 degree cone. This would allow the ultrasonic sensors to fire directly over the tesseracts. The issue with this is that the tesseracts would be more or less ignored. The group did come up with a solution to this problem which was to throttle the ultrasonic sensors output. This would work by covering the top half of the ultrasonic sensor that way the 40 degree cone would never go higher than the tesseracts. It was discovered however, that this solution would not work at all due to the now reduced size of the ultrasonic waves. In addition there were also issues with detecting angled tesseract. Due to the nature of sound waves, the ultrasonic waves would simply bounce off angled surfaces and never return to the sensor to give it a reading. It was determined during testing that the sensors waves had to be less than 5 degrees to being perpendicular to the tesseracts for the ultrasonic sensor to actually read the tesseract. This led the group to avoid using ultrasonic sensors as a means of tesseract detection.





Final Prototype



Code

Arm Code

void sweepServo(Servo servo, int desiredPosition) {

The sweepServo function takes a servo class object and a desired servo angle. When writing a new angle to a servo, the servo moves at max speed. This is undesirable for the wrist servo, as fast movements could result in accidentally dropping a tesseract off the magnet. The sweepServo function simple writes new servo angles 1 degree at a time, with a small delay in between. The end result is a servo that moves smoothly and slightly slower.

void moveTurntable(int desiredPosition) {

The arm turntable is powered by a geared DC motor, and position feedback is given by an encoder. To move the arm to a new position, the moveTurnTable function was created. int desiredPosition is the encoder position to which the motor/turntable should move. The function uses a simple Proportional speed adjustment, relative to the current position and desired position. The final product should incorporate a more advanced PID position controller, but the simple P controller worked well during testing.

void moveArm(int desiredPosition) {

The arm is powered by a DC motor, and position feedback is given by an encoder. The arrangement is very similar to the arm turntable, and this function is very similar also. The moveArm position did not work well; when the arm is swinging it develops a non-negligible amount of momentum. This often caused the arm to overshoot the desired position and oscillate about the desired position. To remedy this issue, a rubber band was added to the arm to counteract momentum, and different P gains were used based on arm position. The results of these modifications were inconsistent.

void moveArmSweep(int desiredPosition) {

This function is called in the same manner as moveArm. Inside the moveArmSweep function, the regular moveArm function is called 10 encoder ticks at a time. This results in the arm moving slower, and not developing nearly as much momentum, eliminating the overshooting problems. The final product robot code should use a PID position controller, but the nested moveArm function inside moveArmSweep function did work reliably.

bool tesseractArmScan() {

This functions is designed to be called when a tesseract was detected by the infrared sensor array. The function will cause the robot to back straight up, sweep the arm from left to right above the ground, searching for magnetic flux using the arm hall effect sensor. If strong magnetic flux was detected, the arm will return to that position, actuate the magnet to pickup the tesseract, move the arm to a carrying position, and return true. If strong magnetic flux was not detected, the arm will lower, sweep the bad tesseract to the side, move the arm to a carry position, and return false.

Drive Code

bool driveStraightAheadEncoders(int ci_drive_speed, int desiredPosition) {

bool driveStraightReverseEncoders(int ci_drive_speed, int desiredPosition) {
These functions drive the robot straight ahead or reveres a desired amount of encoder ticks at a desired speed. While driving, the robot is polling the infrared sensor array and will immediately return true if a tesseract is detected. If a tesseract is not detected, the function will return false after reaching the desired encoder position. The motor control inside these functions is as follows. To start; the right motor is set to the passed speed and the left motor is set to the passed speed + a speed offset (derived experimentally). From there, the encoder positions between wheels are compared every 20mS. If the left motor did not move far enough the speed is increased, and if the left motor moved too far the speed is decreased. These speed adjustments based on encoder values are made every 20mS until the motors reach the desired position.

void driveStraight(const int ci_drive_speed) {
void driveStraightReverse(const int ci_drive_speed) {
These functions are similar to the driveStraightAheadEncoders and driveStraightReverseEncoders functions, but do not poll the ultrasonic distance sensors and do not return a bool. These functions must be called continuously for the robot to continue moving.

void skidsteerNinetyLeft(int ci_drive_speed) {
void skidsteerNinetyRight(int ci_drive_speed) {
These functions cause the robot to turn 90 degrees in place at a desired speed. The motor control code is very similar to the driveStraight functions, but in reverse, causing both motors to traverse equal encoder ticks but with the controlled motor in reverse. The amount of encoder ticks to traverse was derived experimentally to be 439.

void skidsteerTinyLeft(int ci_drive_speed) {
void skidsteerTinyRight(int ci_drive_speed) {
void parallel(char wall) {
The skidsteerTinyLeft and skidsteerTinyRight functions are essentially the same as the skidsteerNinetyLeft and skidsteerNinetyRight functions, but instead of turning the robot a full 90 degrees when called, they only turn the robot about 2 degrees each call. These functions are used inside the parallel function. The parallel function turns the robot in place to line it up parallel to a side wall using the ultrasonic distance sensors. Whether the reference wall is to the left or right of the robot is selected by passing char “l,” “L,” “r,” or “R.” The robot will skidsteer 2 degrees at a time until the front and rear ultrasonic distance sensor readings are equal.

void followWall(int ci_drive_speed, char wallSide, int desiredDistance) {
This is one of the main functions inside the robot code. This function can be called to drive the robot parallel to a wall using ultrasonic distance sensors. The parameters passed are

desired drive speed, which side of the robot the reference wall is, and what distance in centimeters to drive from the reference wall.

Troubleshooting Code

void printPingSensorReadings() {
void printEncoderValues() {
void printSensorReadings() {
During testing it was often helpful to print various readings to the serial monitor. Rather than write small code snippets whenever needed, some functions were written. This gave the option to print nicely formatted data of all sensors and encoders whenever needed by making a simple function all.

Infrared Sensor Array

An Infrared Reflective Optical Sensor Array (IRROSA) was chosen as the means to detect tesseracts. The IRROSA was chosen because it is capable of detecting both magnetic and non-magnetic tesseracts without disturbing or moving the tesseracts in a repeatable manner. The IRROSA principle is as follows; IR reflective optical sensors are positioned in a semi circle around the front of the robot. The IR sensors are mounted 2.5cm from the ground, so that a 2cm tesseract is free to pass below the sensors, and the IR sensors are positioned less than 2cm apart from each other along the arc. When a tesseracts moves underneath an IR sensor, more IR light is reflected to the sensor, giving a change in sensor reading value. Preliminary sketches may be viewed in the figure below. The prototype used 10 IR sensors, but the final product should use more sensors, perhaps 16, to minimize the possibility of a tesseracts passing between sensors.

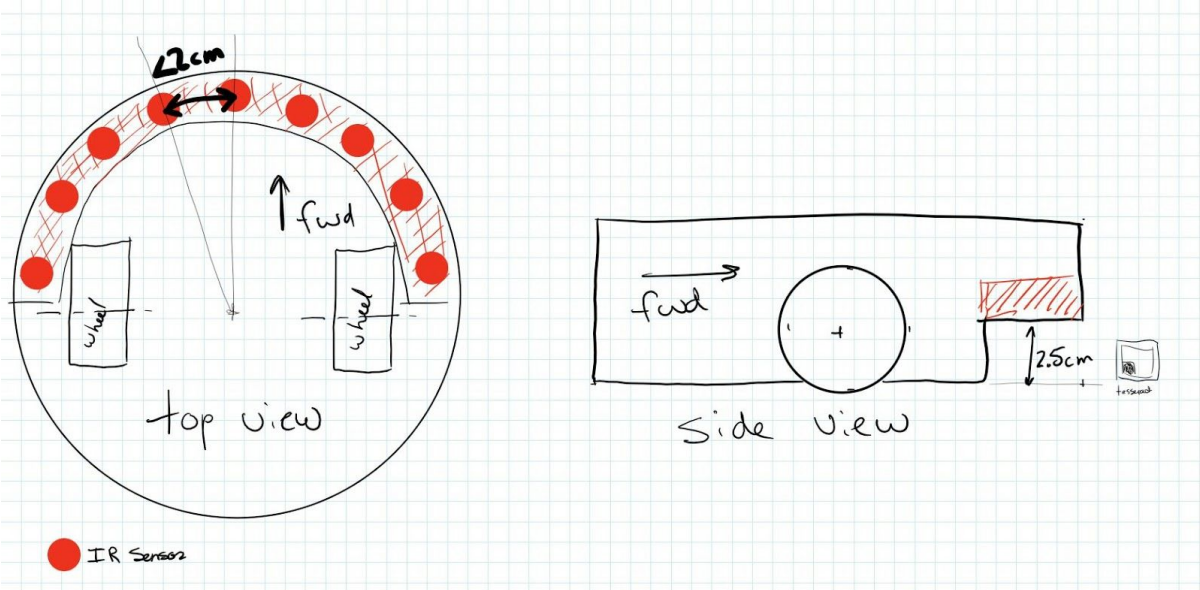


Figure 1: Infrared Reflective Optical Sensor Array concept sketches

The sensors chosen for the robot are Vishay TRTC5000L Reflective Optical Sensor with Transistor Output, available from www.digikey.ca for $0.6975 / pc when buying in quantities of 1000 or more. The sensors can be powered from 5v when used with a simple current limiting resistor, and provide an analog reading when used with a simple voltage divider using another resistor. An image of a TRTC5000L, as well as a circuit diagram image is

pictured below.

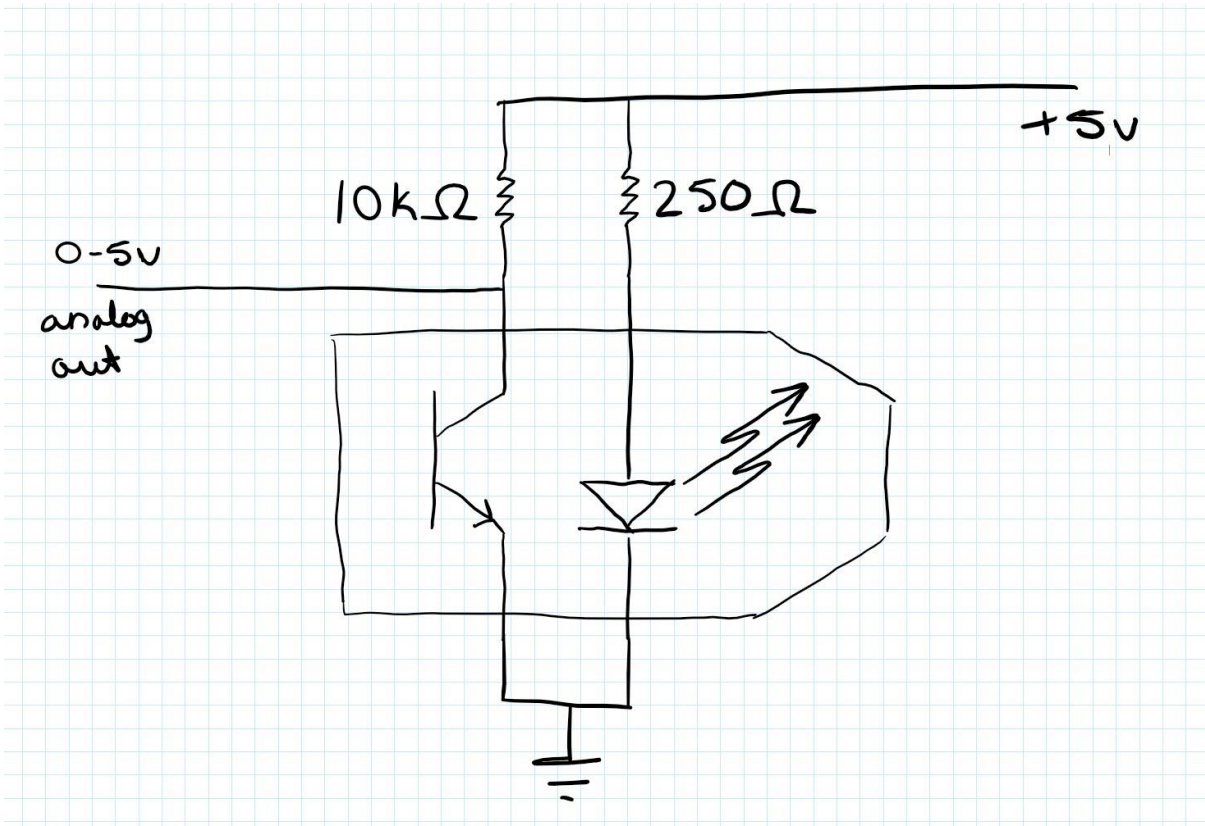


Figure 2: TCRT5000L sensor and circuit diagram for analog output

Several TCRT5000L sensors were purchased and assembled onto protoboard with the appropriate resistors for the prototype robot as show in the next figure. The sensors performed well under testing, producing a change in sensor value change of ~260 (10 bit) between detecting and not detecting a tesseract. This is a very strong (25%) change in sensor reading which is one of the reasons this type of sensor system was selected compared to other sensor types (Hall Effect, ultrasonic, etc). After creating 3 TCRT5000L sensors with associated resistors and wiring onto protoboard (shown below), it was decided to use Vex line trackers for prototype testing to save assembly time. The Vex line tracker sensors operate in a similar manner and provide similar readings to the TCRT5000L sensors.

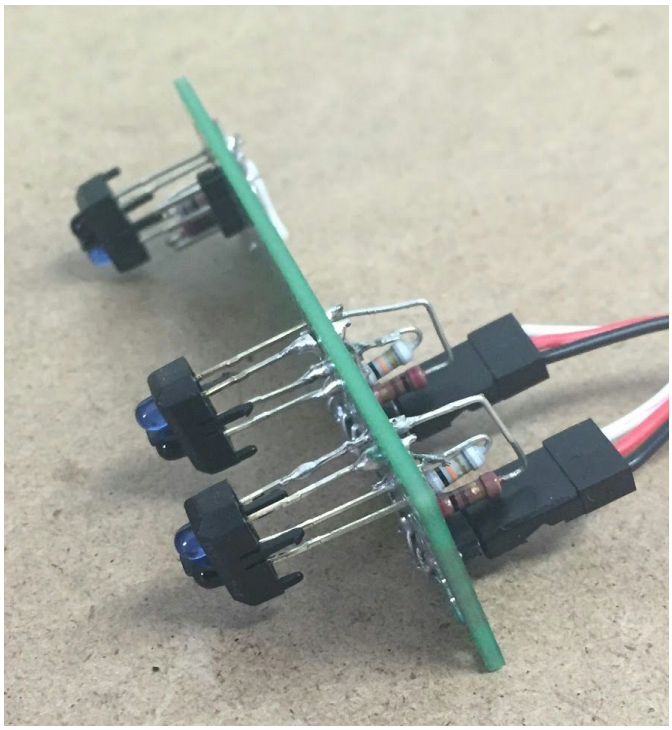
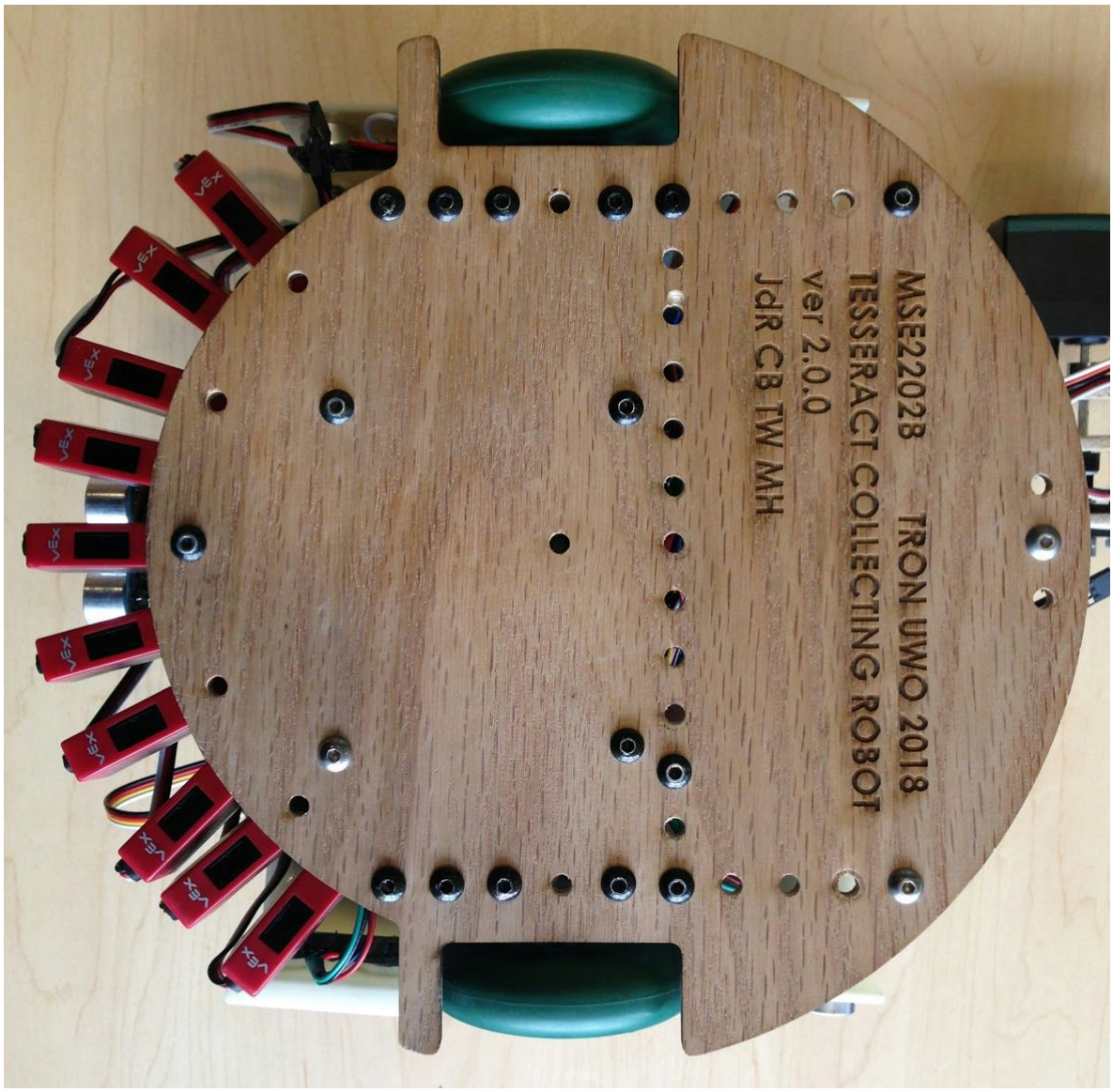


Figure 3: TCRT5000L prototype circuit

To read multiple analog sensors, a microcontroller with enough analog pins is required. Most microcontrollers do not have enough pins to read 10+ sensors. The final product could make use of an analog multiplexer/de-multiplexer such as the Texas Instruments CD74HC4067 16-Channel multiplexer. This small IC, available from www.digikey.ca, is available for $0.44505 / pc and is able to connect 16 analog sensors to a single microcontroller analog pin. Selecting which sensor is currently connected to the analog pin is done through 4 digital pins and a truth table. For the prototype; connecting the IR sensors to the surface mount CD74HC4067 would have required the fabrication of a custom circuit board, as well as waiting for the component to arrive. To save time; the sensors were connected to an Arduino Mega 2560 which is equipped with 16 analog pins. An image of the prototype using vex sensors is shown below, the Vex sensors being the red rectangular objects.



In programming; the desired output of the sensor is a true/false, signifying whether a tesseract is under a sensor or not. Since the sensors are analog and each sensor is slightly different and reads a different normal value for the same conditions, some calibration code is needed. Through experimentation it was discovered that when a tesseract is under a sensor, the analog value is 260 (10 bit value) lower than when there no tesseract value under the sensor. The actual sensor readings varied drastically between sensors, but the delta of 260 was constant between every sensor. To create a sensor threshold, you could average the tesseract and no-tesseract sensor values. If the instantaneous reading is above that threshold, there is no tesseract, and if the instantaneous reading is below that value, there is a tesseract. To simplify the calibration function, the constant 260 value was halved and subtracted from the no-tesseract reading to create a threshold. The threshold is then stored in the EEPROM so that the sensors do not need to be calibrated every power cycle. Since the sensor readings during the calibration cycle are of utmost importance, an initial sensor reading should be taken and discarded to give the microcontroller ADC time to settle. Multiple sensor

measurements should then be taken and average to have the cleanest data possible. A code snipped shown below shows this functionality, the full sensor code may be seen in the file TesseractSensorCode.ino.

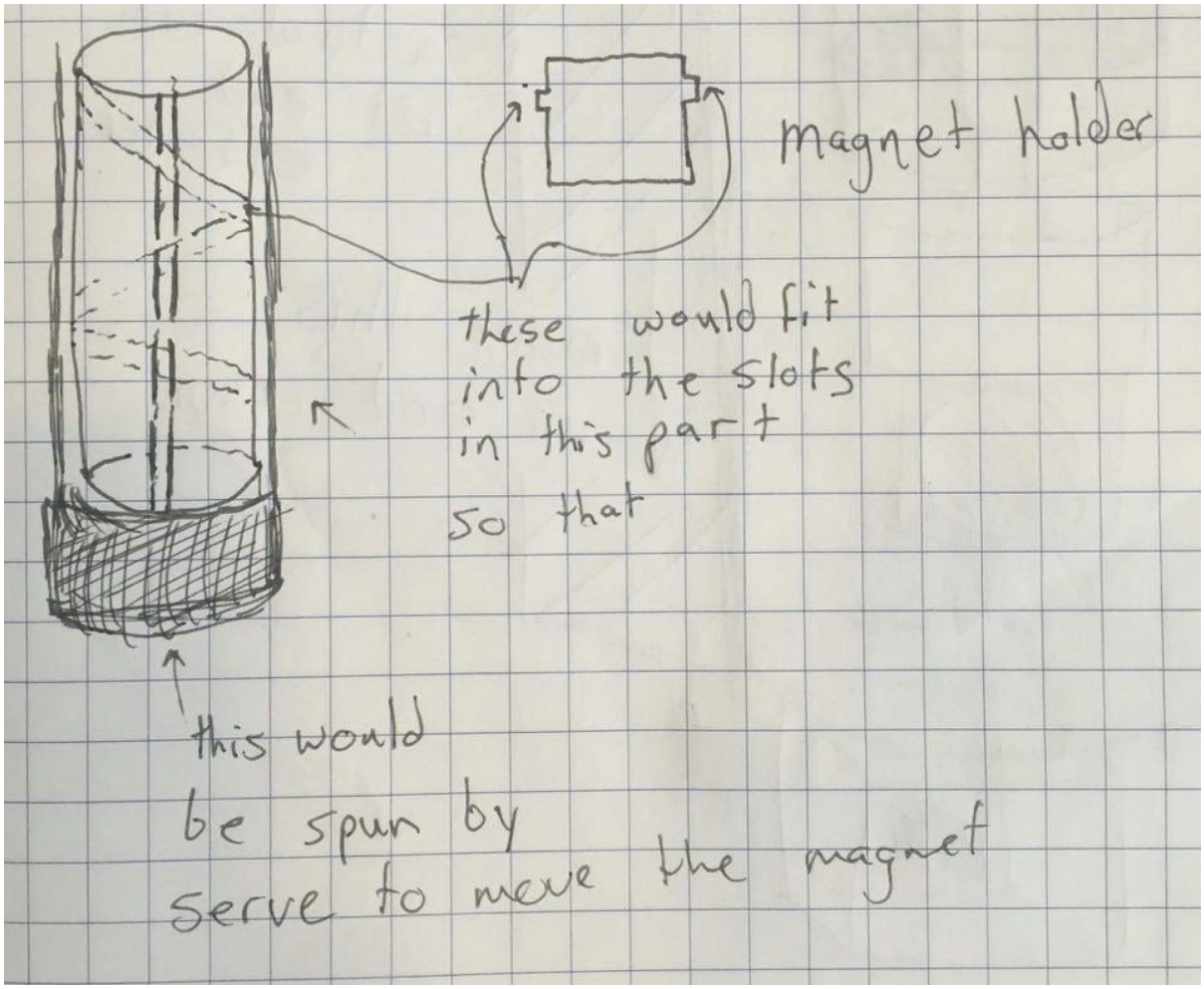
```
for (int i = 0; i < numberOfSensors; i++) {
  analogRead(analogPins[i]);                                          // just a garbage measurement to give ADC time to settle, this measurement is discarded
  for (int j = 0; j < 10; j++) {
    newThresholdValues[i] = newThresholdValues[i] + analogRead(analogPins[i]);    // read sensor 10 times, sum the values
  }
  newThresholdValues[i] = newThresholdValues[i] / 10;                 // average the sensors readings
  newThresholdValues[i] = (newThresholdValues[i] - sensorHalfDelta) / 4;          // create new threshold, convert 10 bit value to 8 bit value (EEPROM is8 bit)
  EEPROM.write(eepromAddress[i], newThresholdValues[i]);              // burn the array values to EEPROM
}
```

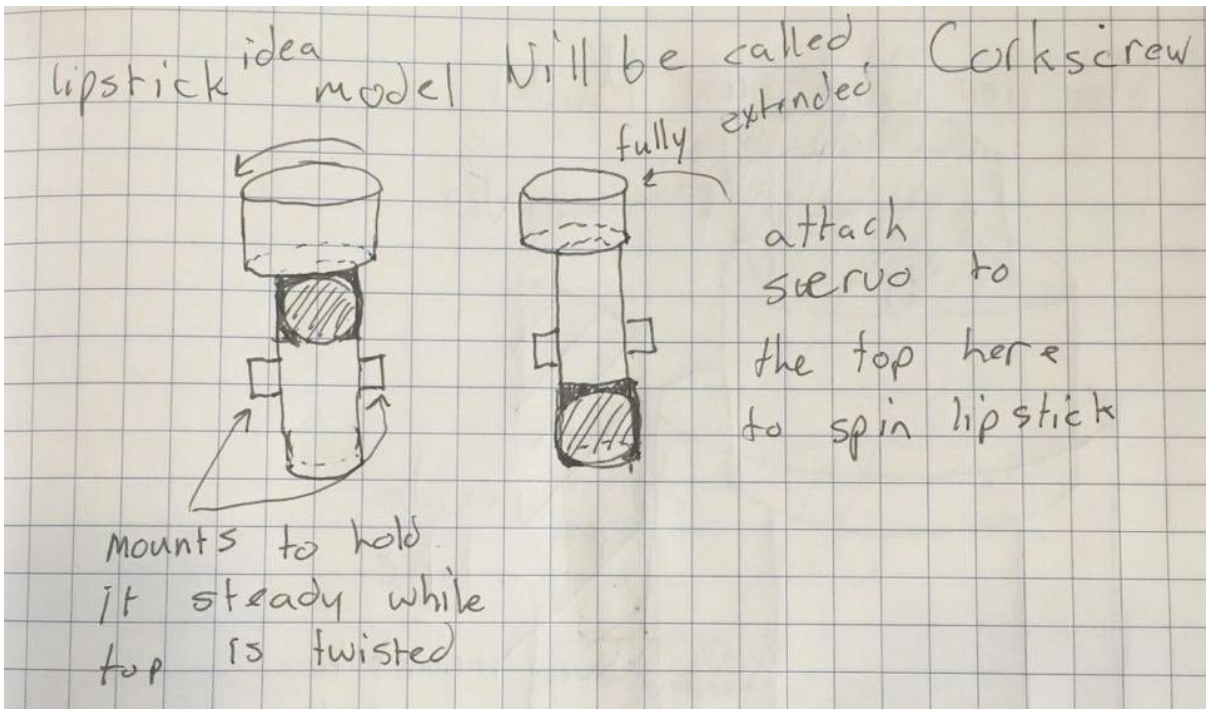
Figure 4: Calibration Code Snippet

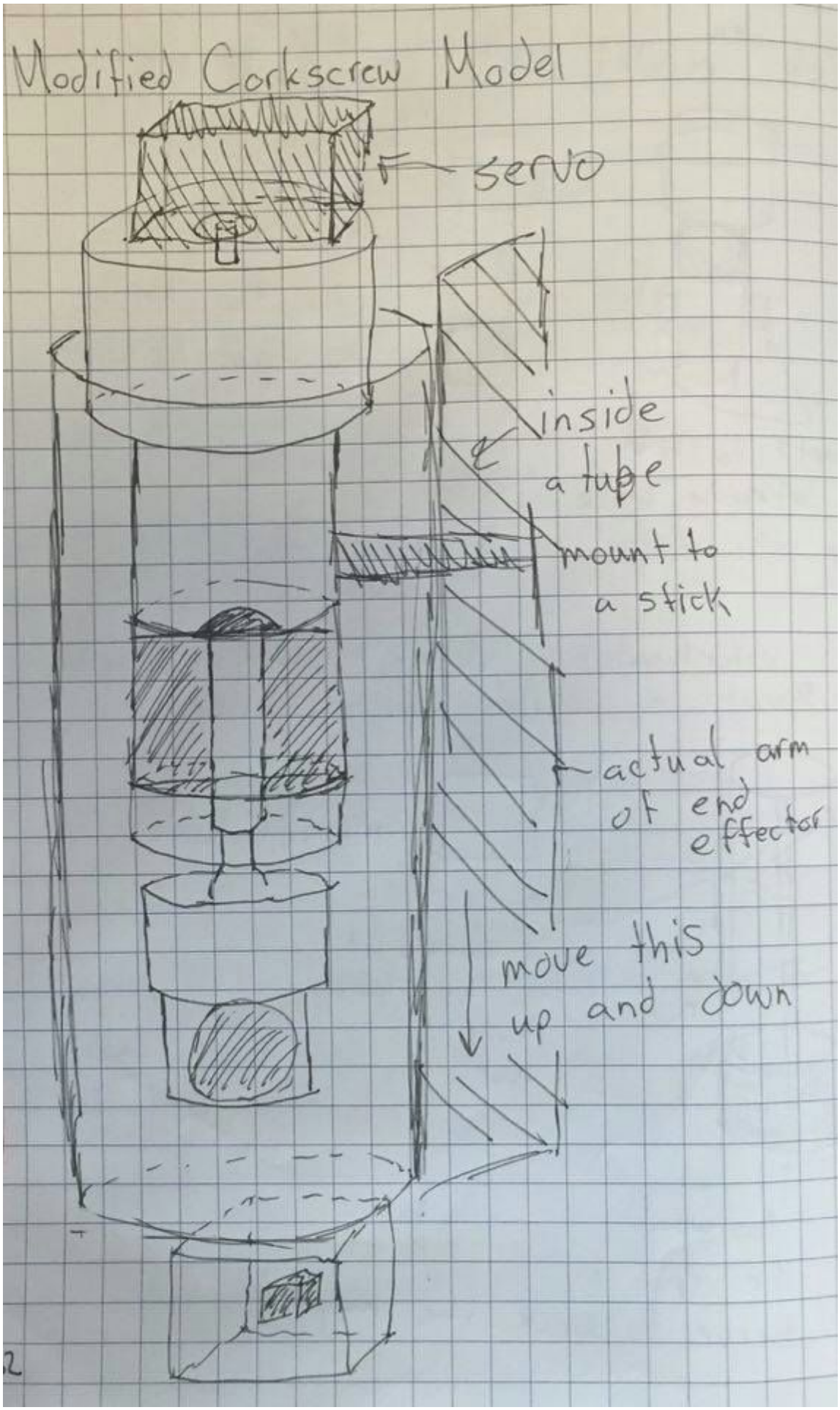
For the prototype code, a specific pin shorted to ground during power up activated the calibration function, and the internal Arduino pin flashed when the calibration function was complete. The final product robot should have a button to calibrate the sensor, and should maintain the led to signify that calibration is underway/complete. The prototype robot used an LED to visually show whether a tesseract had been detected and this was helpful during troubleshooting, the final robot should keep this functionality as well.

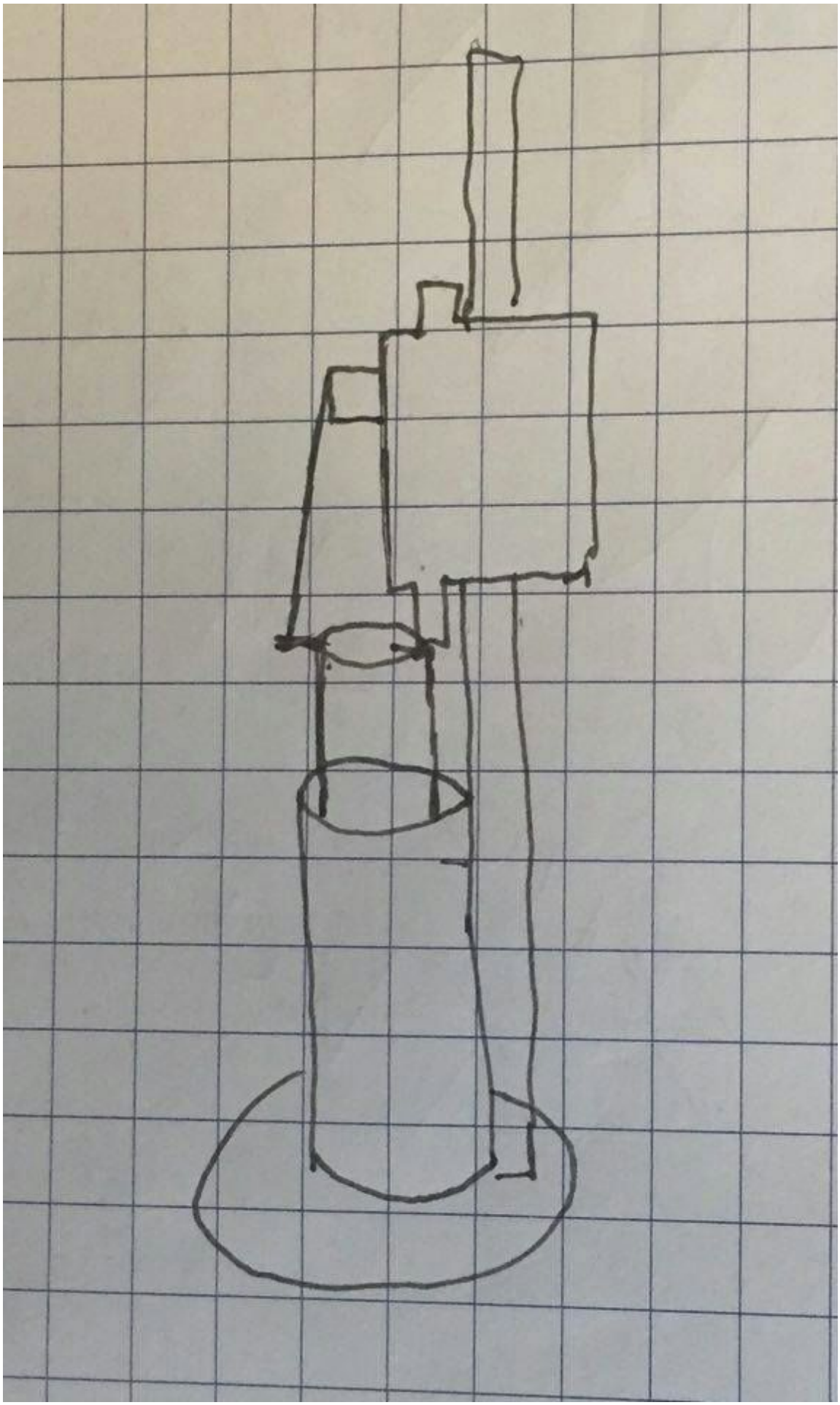
In conclusion; the Infrared Reflective Optical Sensor Array worked extremely well and did not present any problems when testing with the prototype. The sensors had a near 100% detection rate and did not give any false positives. After the IRROSA prototype assembly was completed and attached to the robot base prototype; calibration was completed only once and the IRROSA did not need to be re-calibrated for all of prototype testing.

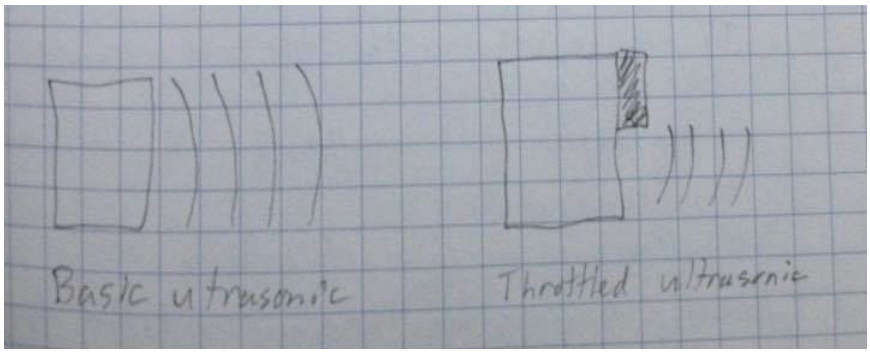
Miscellaneous Design Sketches

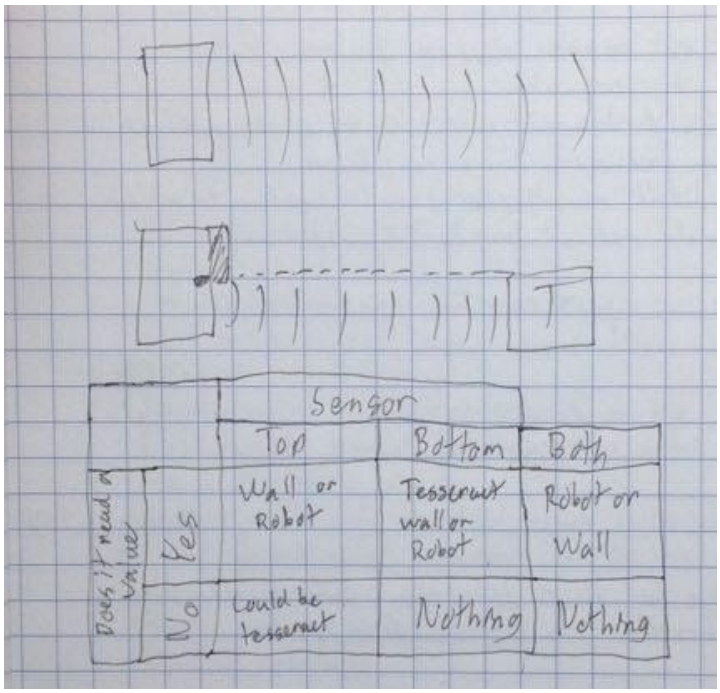


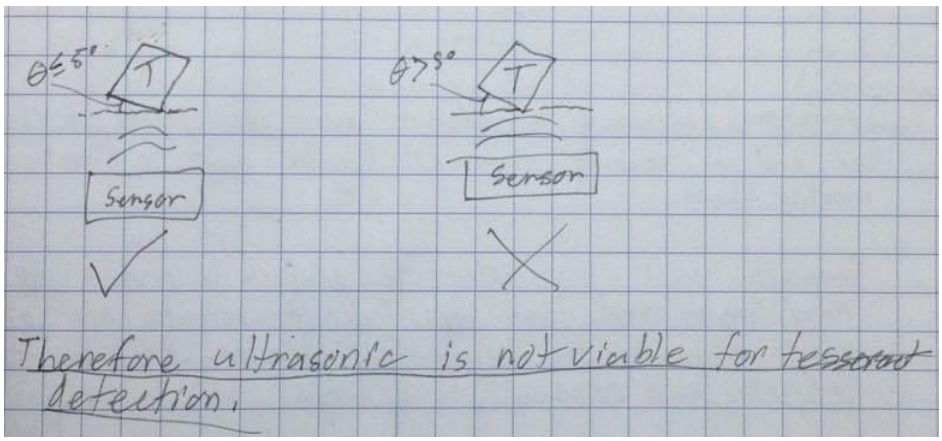



Modified Corkscrew Model
servo
inside a tube
mount to a stick
actual arm of end effector
move this up and down

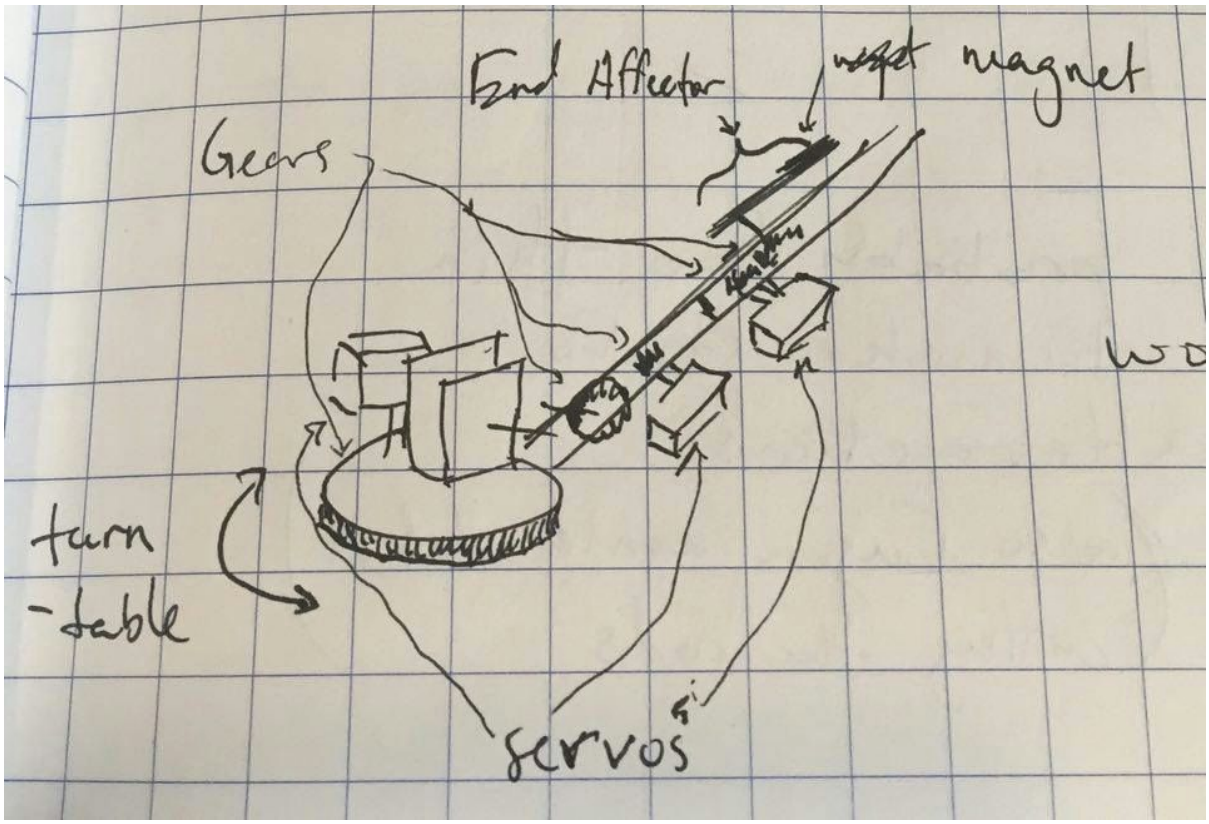


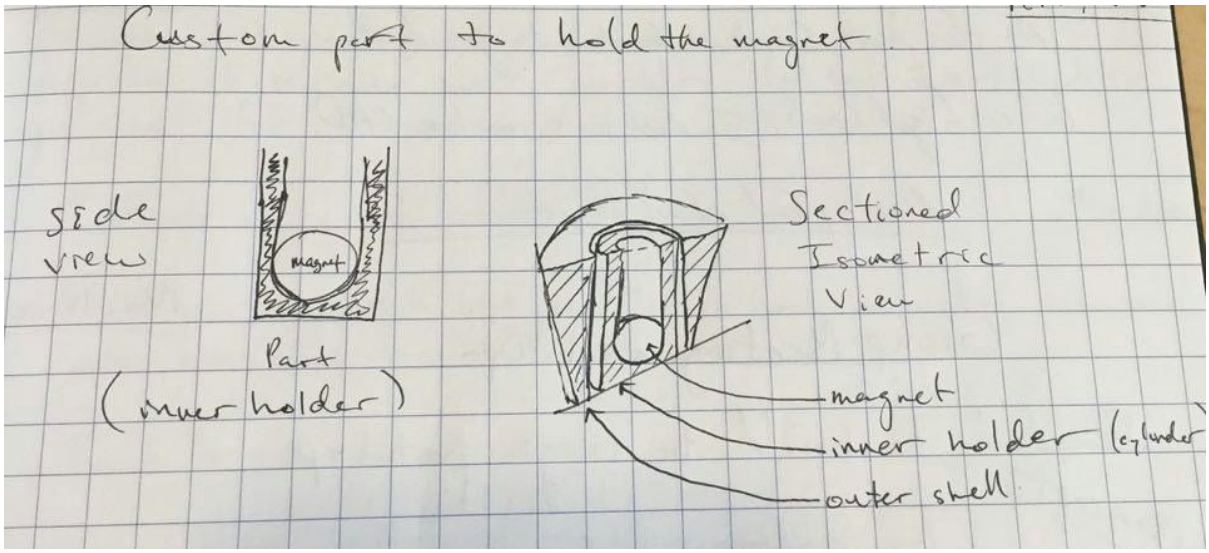


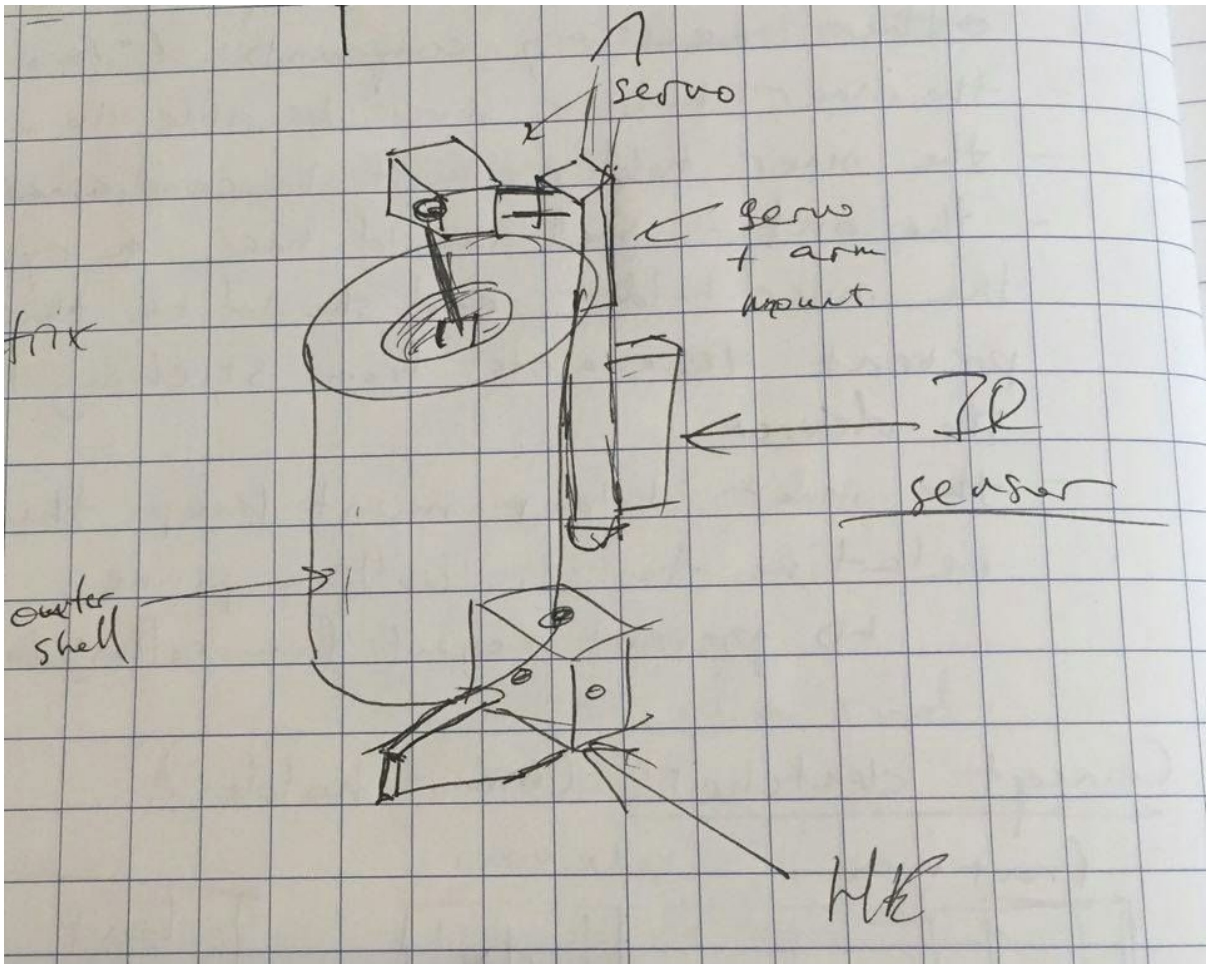




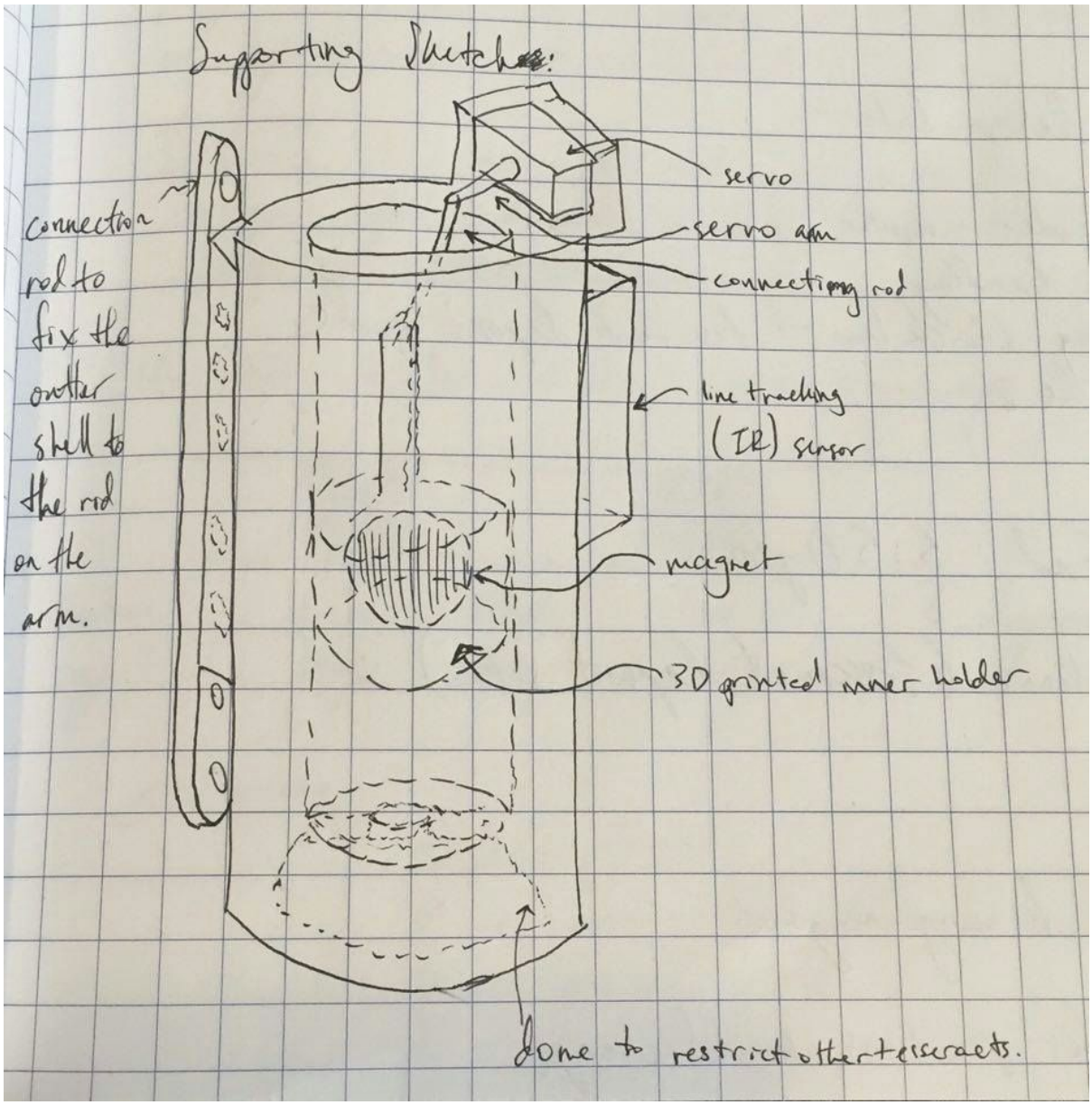
Therefore ultrasonic is not viable for tesseract detection.

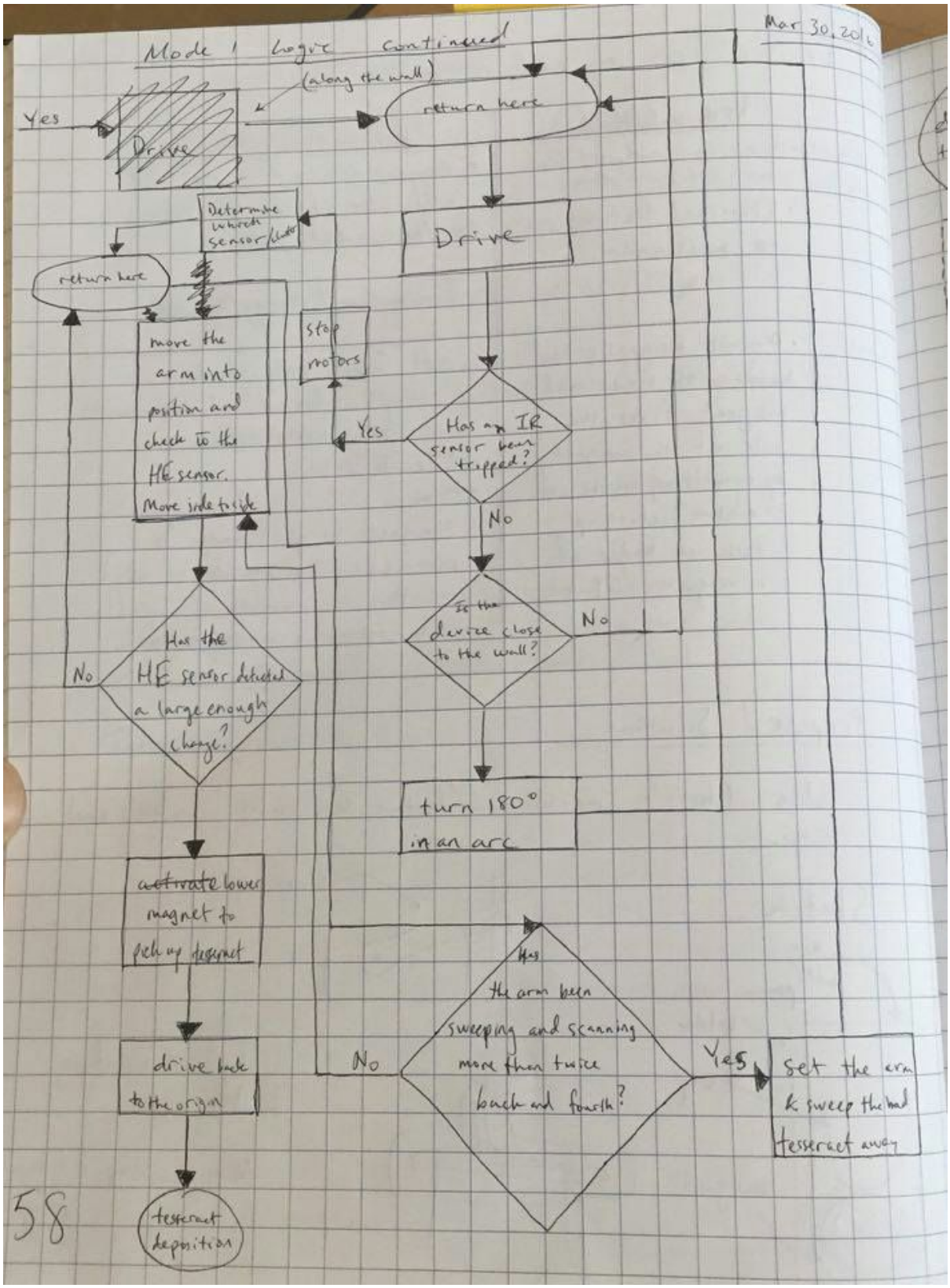

End Affector
weight magnet
Gears
turn
-table
servos

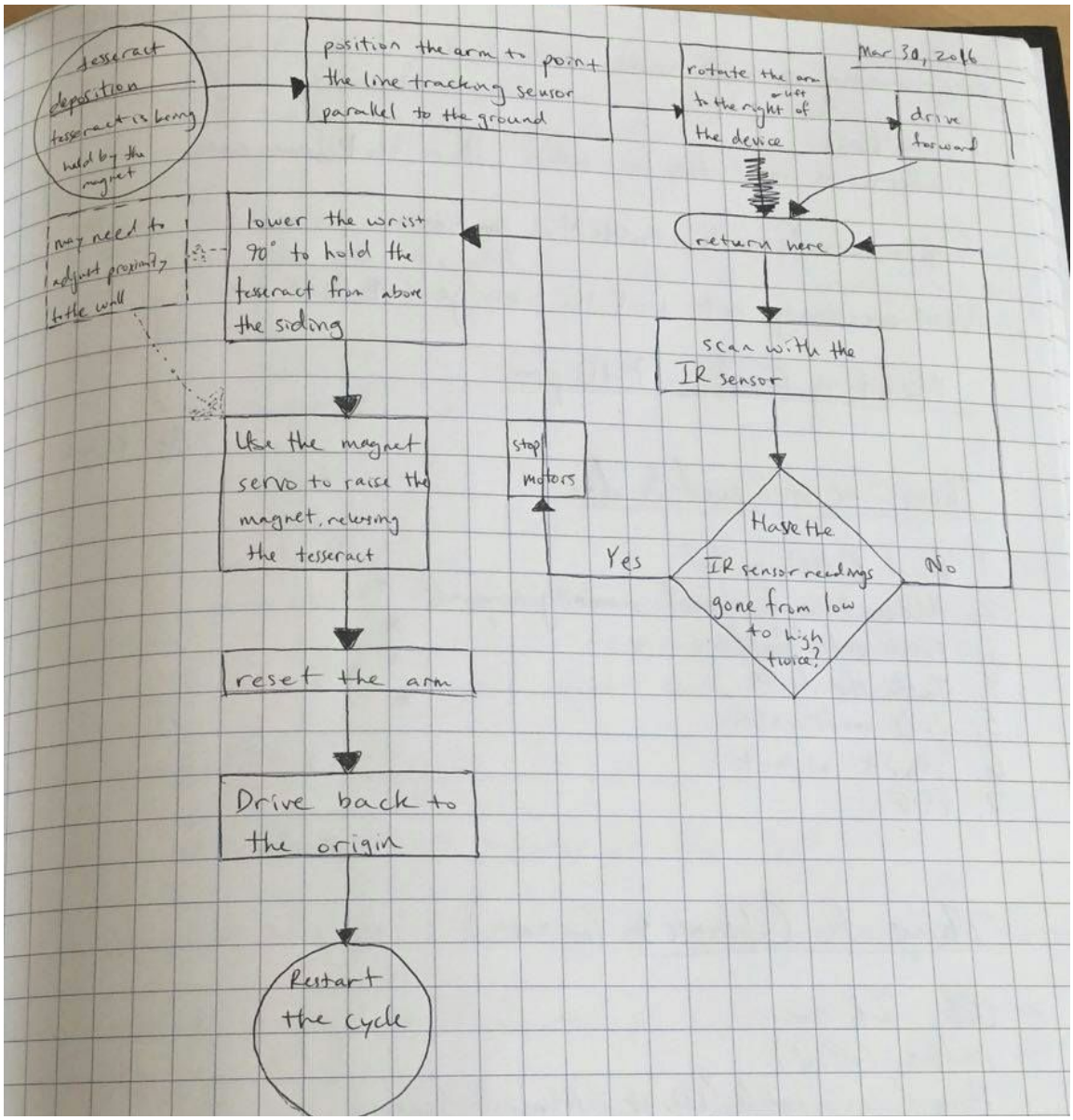

Custom part to hold the magnet.
side
view
magnet
Part
(inner holder)
Sectioned
Isometric
View
magnet
inner holder (cylinder
outer shell

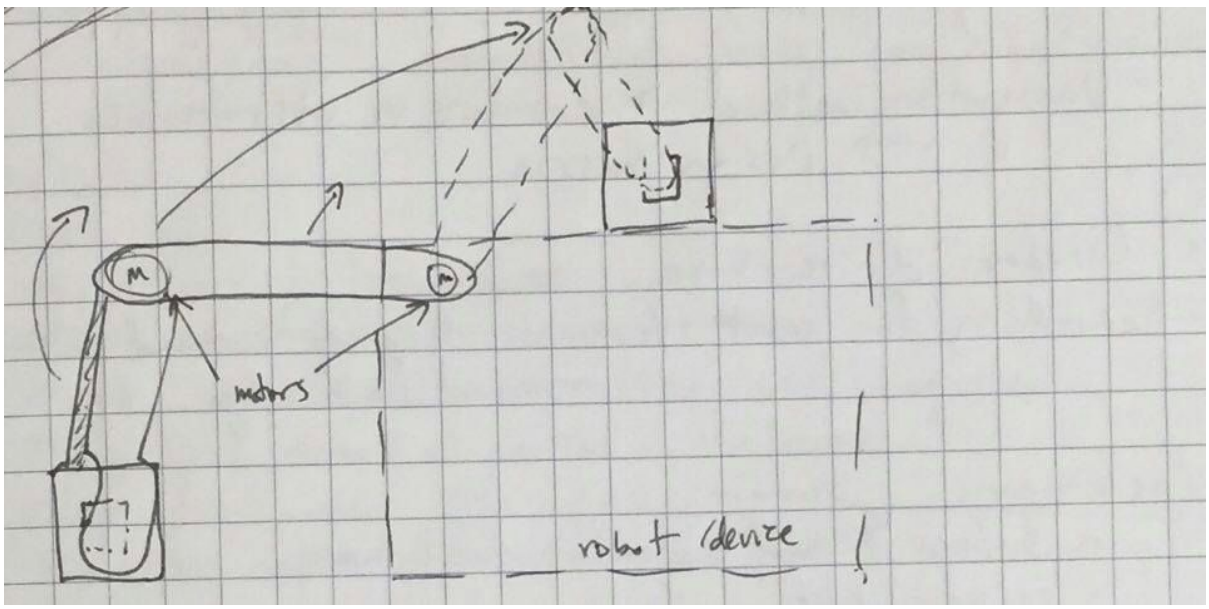

servo
servo + arm mount
fix
TOF sensor
outer shell

Supporting Sketches:

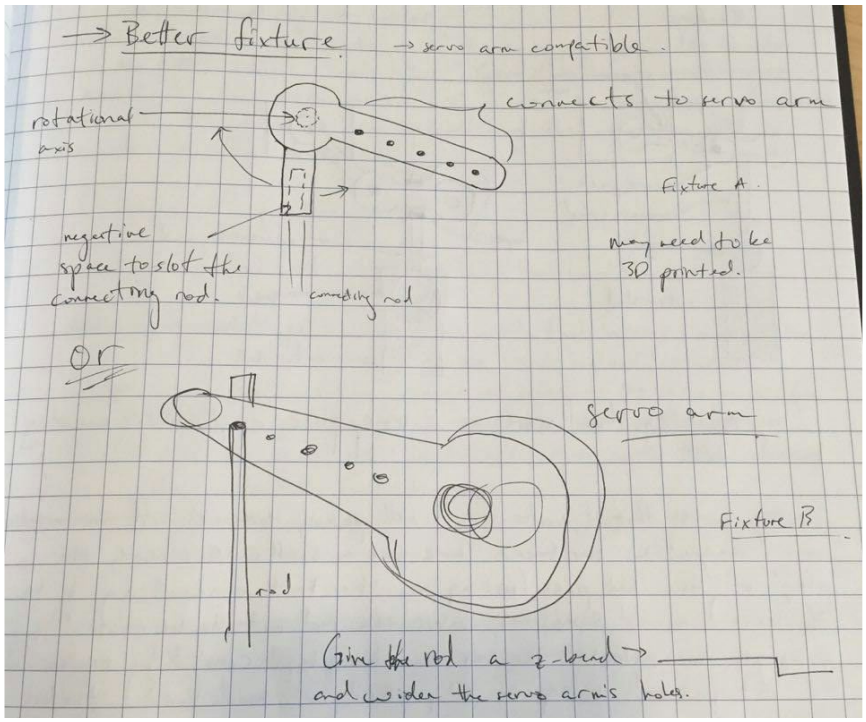



Mar 30, 2016
Mode 1 Logic continued
(along the wall)
Yes
Drive
return here
Drive
Determine which sensor/limit
return here
move the arm into position and check to the HE sensor. Move side to side
stop motors
Yes
Has an IR sensor been tripped?
No
Is the device close to the wall?
No
Has the HE sensor detected a large enough change?
No
turn 180° in an arc
activate lower magnet to pick up tesseract
drive back to the origin
tesseract deposition
Has the arm been sweeping and scanning more than twice back and forth?
No
Yes
set the arm & sweep the mad tesseract away
58


tesseract deposition
tesseract is being held by the magnet
position the arm to point the line tracking sensor parallel to the ground
rotate the arm to the right of the device
Mar 30, 2016
drive forward
may need to adjust proximity to the wall
lower the wrist 90° to hold the tesseract from above the siding
return here
scan with the IR sensor
Use the magnet servo to raise the magnet, releasing the tesseract
stop motors
Yes
Have the IR sensor readings gone from low to high twice?
No
reset the arm
Drive back to the origin
Restart the cycle


M
M
motors
robot device

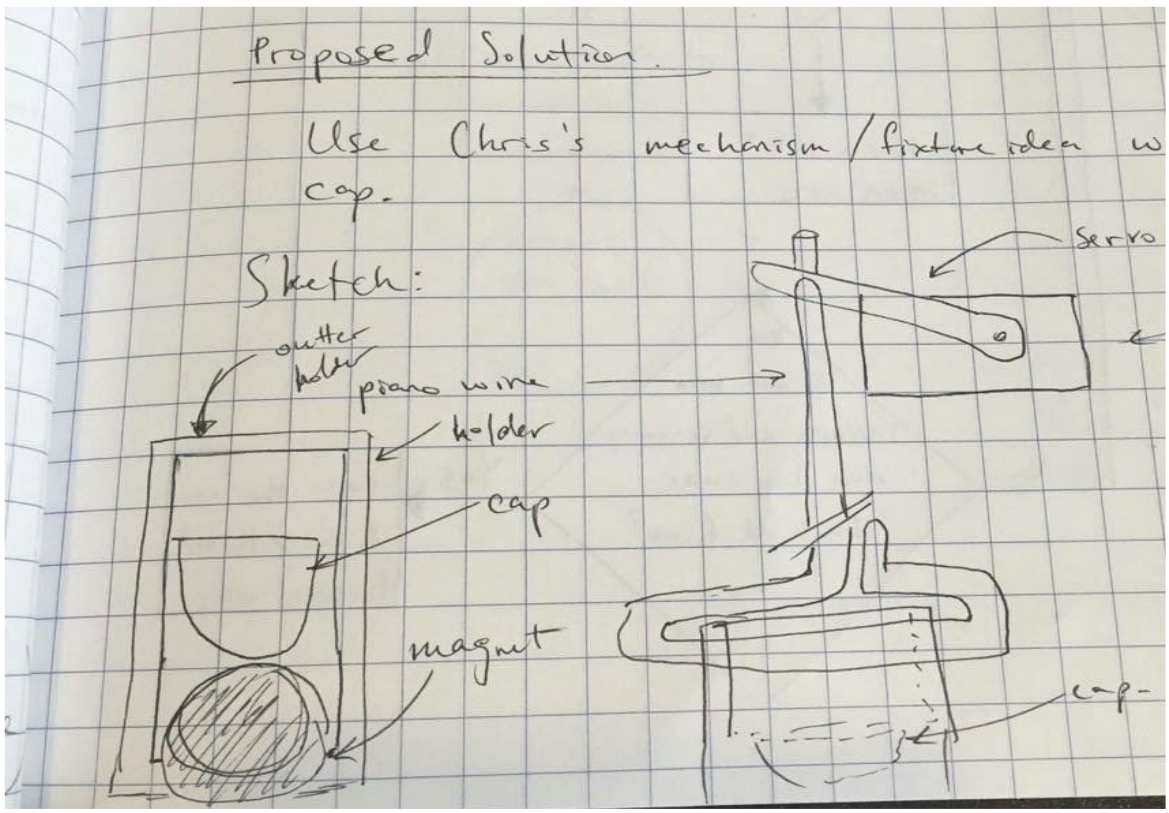
→ Better fixture → servo arm compatible.



Proposed Solution.

Use Chris's mechanism / fixture idea w cap.

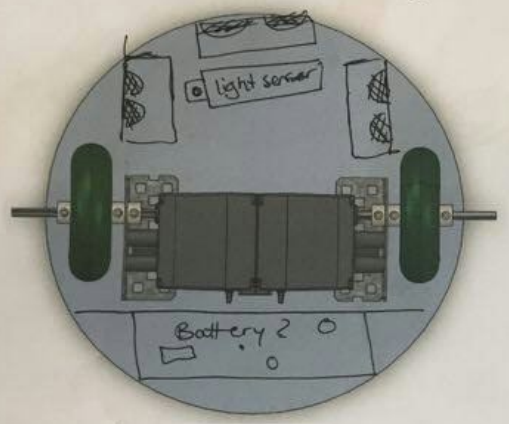
Sketch:



Jal

Layered Construction

ultrasonic range finders 3x



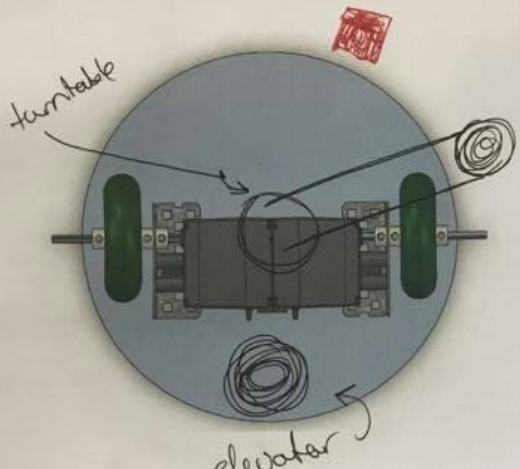
- sandwich design – hold some sensors + battery or uC inside?
- nice flat top to work with, and some sensors/motors out of the way
- separate tops w stand offs
- can go taller to hold more "stuff" ie uC, turntable, elevator parts
- can go multilayer
- easy 2d pieces to manufactur

8"

is as small as possible really, can widen slightly

General Ideas

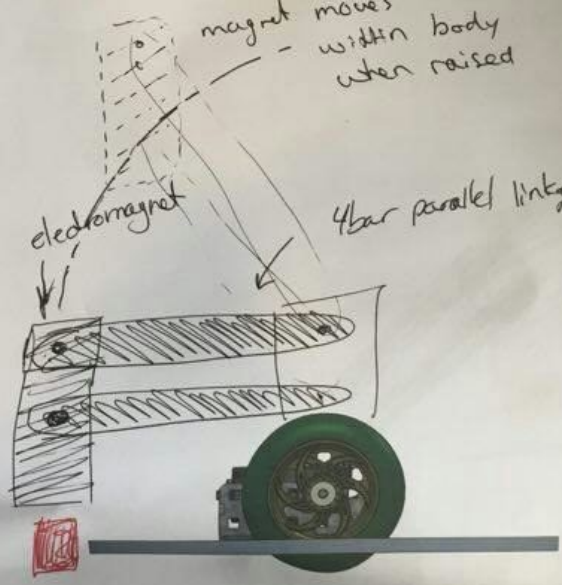
- tesseract



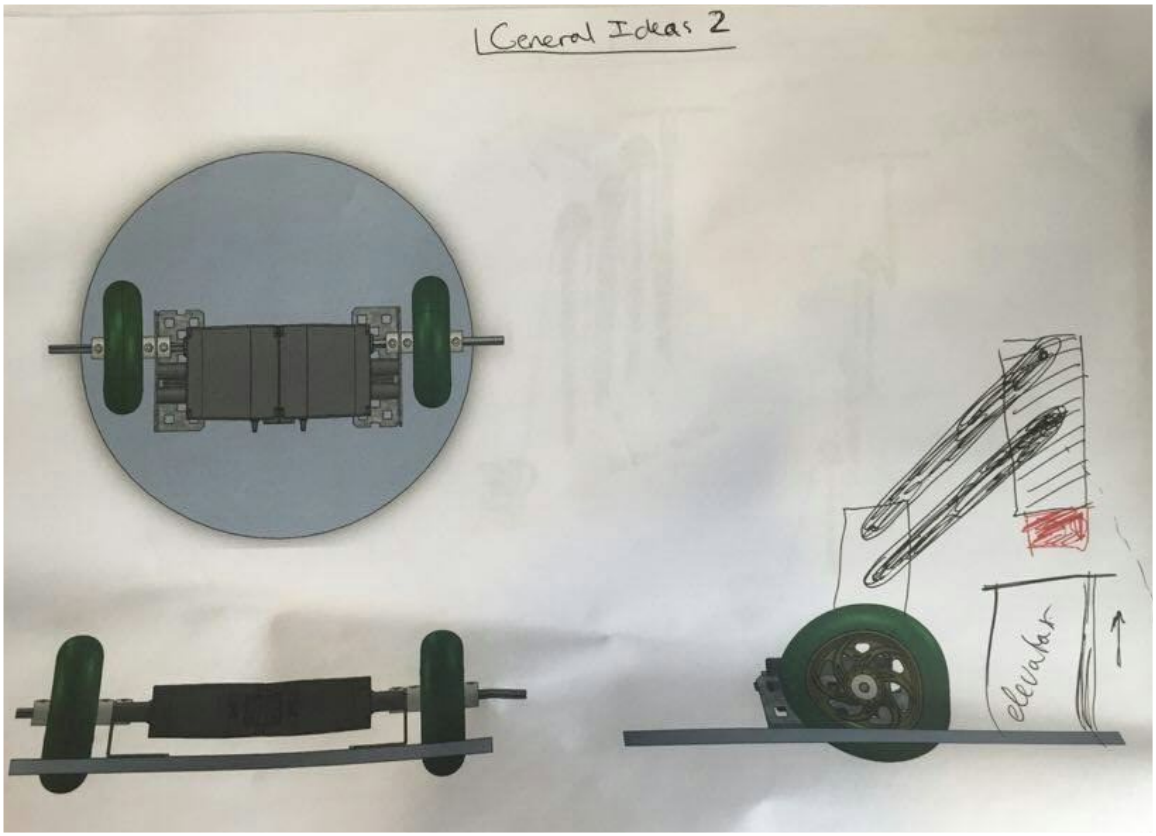
magnet moves within body when raised

electromagnet

4bar parallel linkge

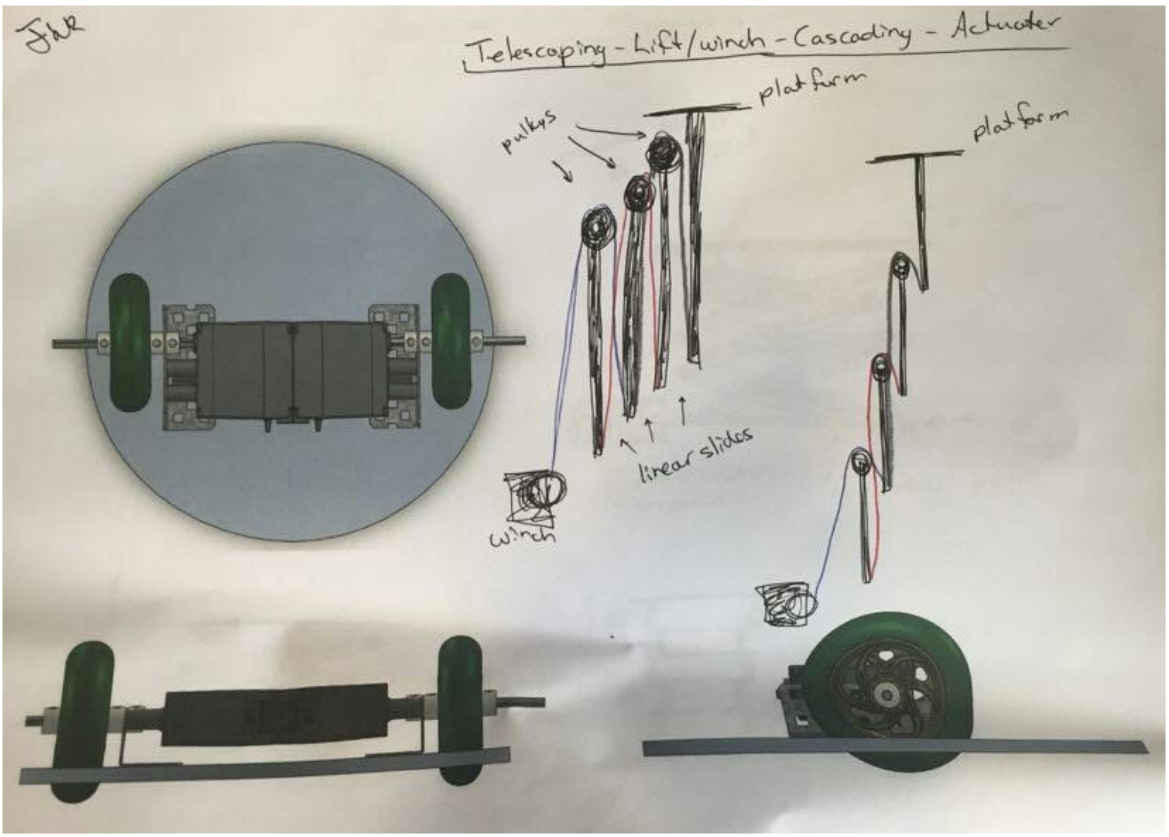


General Ideas 2



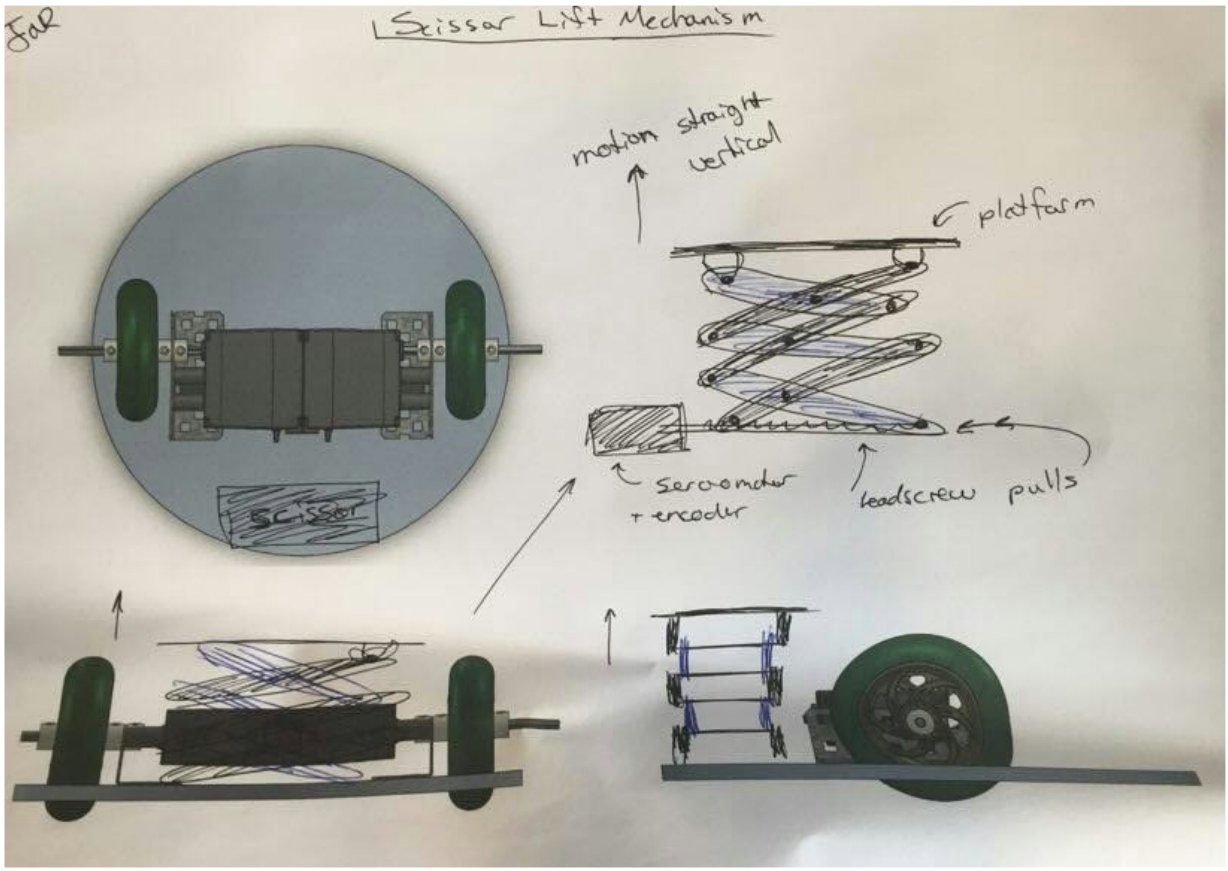
JdR

Telescoping - Lift/winch - Cascading - Actuator



JaR

Scissor Lift Mechanism



Half Scissor Lift