Main Fuel and Oxidizer Valve Actuator Design

## Revision History

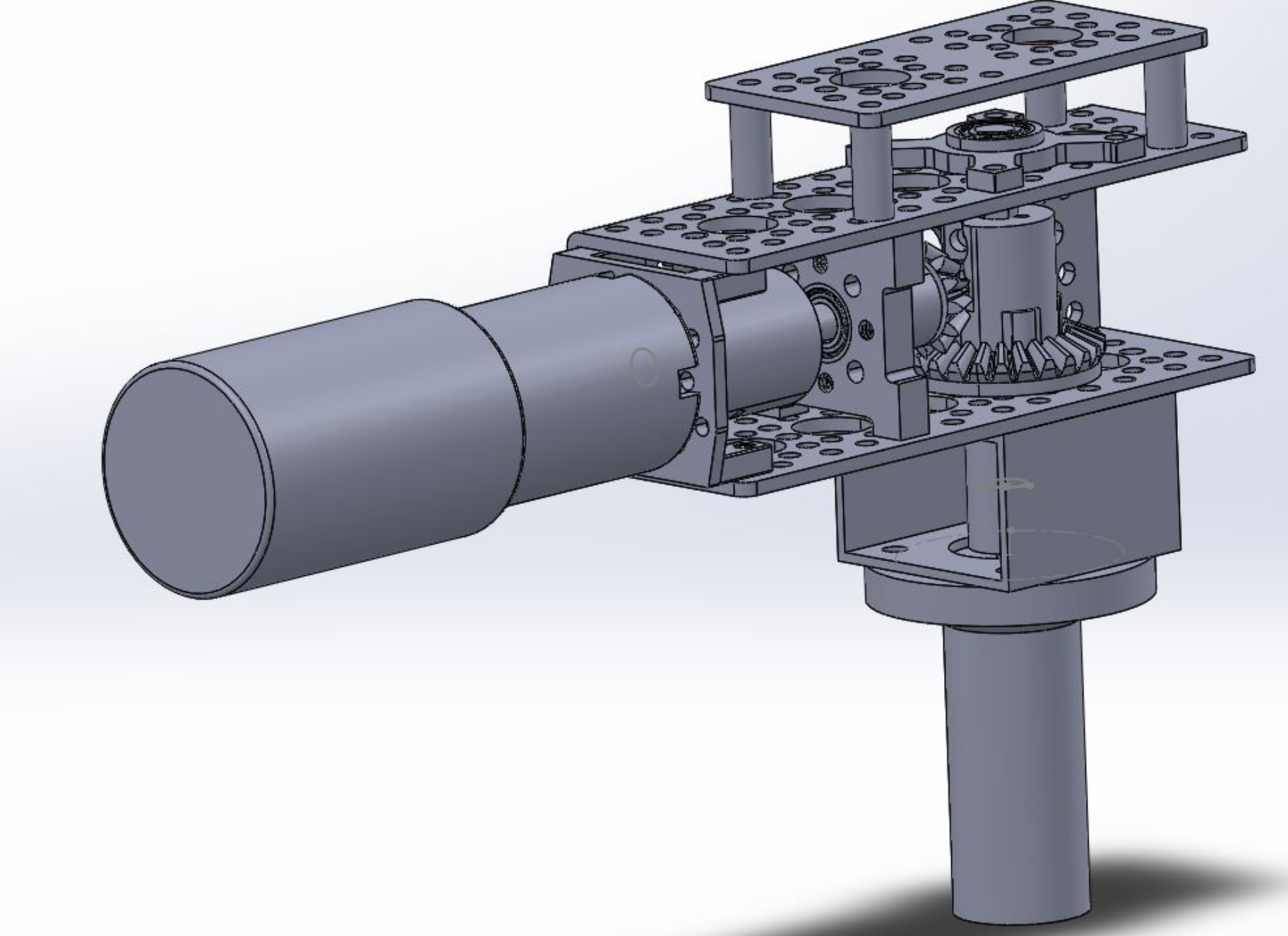
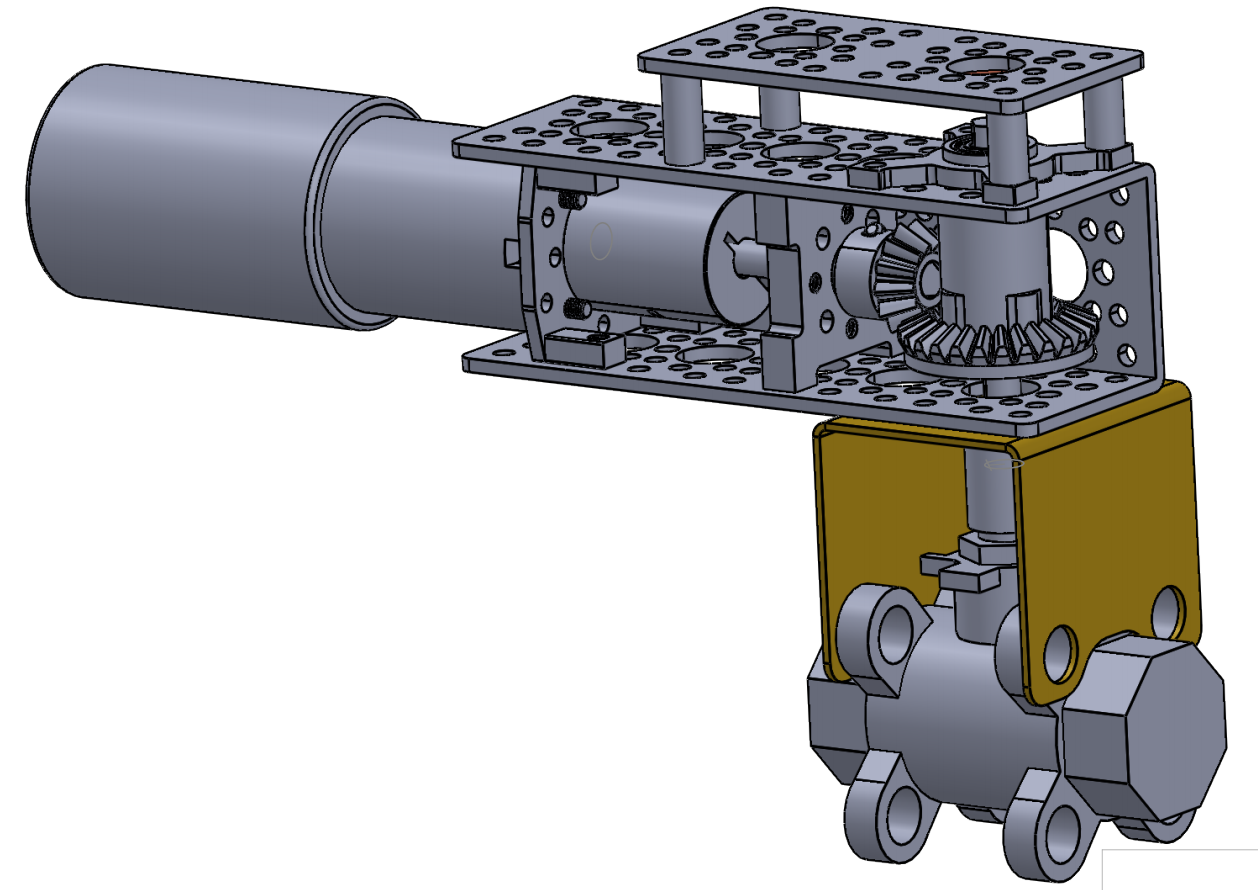
|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Name | Update Notes |
| 6/18/19 | 1.0 | Petr Mazhinikov | Initial writing |
| 11/19/19 | 2.0 | Marc Wasserman | Major overhaul, added content from 2018 capstone reports, removed double spaces after periods (argh!) |
|  |  |  |  |
|  |  |  |  |

## Background

The last valves before propellant reaches the engine are the main fuel valve and the main oxidizer valve. These valves are responsible for the control of the pressurized propellants to the engine. The main fuel valve will be located on the fuel side of the test stand and carry isopropyl alcohol. The main oxidizer valve will be located on the oxidizer side of the test stand which carries liquid oxygen. This document contains information on the valve actuators, which are required to attach to a manual ball valve and actuate the valve at pressures of >500psi. Additionally, the liquid oxygen valve will need a system to prevent frost both inside and outside of the valve. Finally, the actuator must be controlled remotely and provide absolute position sensing.

## Actuator Design

Current design files can be found in the [Liquid Engine Test Stand](https://github.com/psas/liquid-engine-test-stand/tree/master/main%20propellant%20valves/MFV%20%26%20LOX%20Actuator) GitHub repo.



*Figure 1: Main Fuel Valve Figure 2: Main Oxidizer Valve*

The configuration of the actuator went through many iterations and ended with a 90-degree bevel gear design shown in Figure 1 and 2. Both designs for the LOX and fuel valve actuators are very similar in nature. The only difference between them are the valves themselves and the types of brackets used to connect the actuators to the valves. To crack open and close the valves, a 12V 23 rpm DC motor is used, which will open the valve to open in about 1.25 seconds. The frame, motors, and supports are all parts specified from from servocity.com and are primarily standard Actobotics brand components. The brackets were made in-house using sheet metal and hand tools.

The purpose of placing the motor at a right angle is to provide access to the gear shaft to allow mounting of a the position sensor. The position sensor specified for the project is a magnetic hall-effect sensor that utilizes a special magnet which must be rotated near the sensor IC. The sensor is mounted via a second non-magnetic coupler placed on top of the gear shaft and the sensor board is mounted to the underside of the top plate. Two secondary advantages of the bevel gear is that is allows for the mounting of different motors without changing the gears and also provides a 2:1 reduction which doubles the torque.

## Position Sensing

The position sensing for the actuator will be accomplished using an AMS AS5047 magnetic position sensor. This sensor utilizes the Hall Effect to detect the rotation of a magnet placed near to the sensor IC as shown in the diagram below.

2.1 Mounting the AS5047P adapter board 
Figure 3: Mounting and dimensions 
o. 5-3mm 
22mm 
00000000 
000010000 
w/ 
000 
4x2.6mm 

*Figure X: AS5047 evaluation board with clearances and mounting shown,* [*https://media.digikey.com/pdf/Data%20Sheets/Austriamicrosystems%20PDFs/AS5047P-TS\_EK\_AB.pdf*](https://media.digikey.com/pdf/Data%20Sheets/Austriamicrosystems%20PDFs/AS5047P-TS_EK_AB.pdf)

The sensor can be read over a standard serial interface and provides absolute position sensing with a 14-bit resolution. This type of sensor is common in industrial and automotive applications and is more robust than a potentiometer-based sensor as there is no contact between the magnet and sensor. Care must be taken, however, to ensure clearances are maintained and ferrous materials are placed far enough away from the sensor to avoid interference.

## Liquid Nitrogen Testing



Figure: bucket full of liquid nitrogen with a valve shoved in it

A test was conducted in spring 2019 to gather information on any torque changes that happen when exposed to cryogenic liquids, and if a greater torque motor was required. The test consisted of using liquid nitrogen, which is much safer to handle than liquid oxygen, to simulate cryogenic flow. The LOX valve was tested at room temperature and after being submerged for an extended time (no record of amount of time) in liquid nitrogen. A moment arm and force meter were used to determine actuation torque (no record of length of arm) and a thermocouple was used to measure valve temperature. Through varying time intervals of the valve being submerged the torque was found to vary as seen in Fig. 5 and Fig. 6.

|  |  |
| --- | --- |
| *Figure X: Valve at room temperature. First peak (2.35 ft lb) is the valve starting to open, followed by a second peak when the valve is fully open* | *Figure X: Valve at first exposure to cryogenic temperature. First peak (3.03 ft lb)* |

The valve requires an increase of 0.68 ft lbs in torque when it was first exposed to the cryogenic temperature shown in figure 6. The valve itself underwent a temperature change of -74.5 C and maintained an average temperature of - 93.2 C. After the initial exposure, the valve went back to the same torque range as the room temperature one shown in figure 5. From the data gathered it was concluded that using a motor with about 3.5 ft-lb (4.75 Nm) of torque is sufficient to open the valve under cryogenic temperatures. It is not known how torque requirements will change with pressure, however due to a miscommunication between teams, the motor and gear train was specified to provide 35 ft-lb of torque, so it may not matter.

## Valve Freeze Mitigation

In consultation with industry advisors, the team became aware that a challenge with the cryogenic valves in a rocket system is the tendency for moisture to condense out of the air and freeze either inside or outside of the valve body, causing the valve to freeze in place. Mitigation strategies for this will involve an inert gas purge of the lines and tanks before filling and an inert gas purge of the LOX valve continuously beginning when the liquid oxygen tank is filled.

At this time, the team plans to design a case for the liquid oxygen valve and actuator with ports to allow for a slow N2 purge to prevent moisture buildup, however this has not yet been designed or manufactured.

## Valve Actuator CAD

<need to insert drawings of the valve actuators>

Link to current CAD: <https://github.com/psas/liquid-engine-test-stand/tree/master/main%20propellant%20valves>

## Valve Actuator BOM

Link to current BOM: <https://docs.google.com/spreadsheets/d/1h27afDgpPqDNQtMSQFRr19zuC3EtCLTjsB5sXuKFS4k/edit?usp=sharing>

|  |  |  |
| --- | --- | --- |
| Main Fuel Valve Actuator | |  |
| Part # | Manufacturer | Description |
| 39036 | Sharpe | Main Fuel Valve |
| - | Custom | Custom Base, sheet metal |
| 535110 | Servo City | Pillow block & bearing, 1/4" bore |
| 555188 | Servo City | Motor mount |
| 625112 | Servo City | Coupler, set screw, 1/4" to 5/16" |
| 634070 | Servo City | D-shaft, 0.25x2.25 in |
| 625100 | Servo City | Coupler, clamping 1/4" |
| 535130 | Servo City | Bearing block, 1/4" bore |
| 637222 | Servo City | Bevel Gear Set |
| 585444 | Servo City | Standard channel, 4.50" |
| 634068 | Servo City | D-shaft, 0.25x2.0in |
| AS5047Eval | Digikey | AS5047 Evaluation Board (magnetic sensor) |
| 585422 | Servo City | Flat plate, 1.5x3in |
| ? | Servo City | Standoff, 1/2in, 4 pack |
| ? | Servo City | Standoff, 5/8in, 4 pack |
| 632110 | TBD | 6-32 screws, length TBD, 25-pack (3/8"?) |
|  | Pololu | G2 high current motor driver |
| 638266 | Servo City | Heavy duty motor, 23rpm |
|  |  |  |
|  |  |  |
| **LOX Valve Actuator** | | (these parts are in addition to the parts above except as noted) |
| Part # | Manufacturer | Description |
|  | Custom | Custom bracket |
| 634074 | Servo City | d-shaft 0.25x2.5 |
| 634076 | Servo City | d-shaft 0.25x2.75 |
| 625114 | Servo City | shaft coupler 1/4" to 3/8" |
| 625106 | Servo City | shaft coupler 1/4" to 6mm |

## Initial Test Results

The main fuel valve actuator was initially assembled and some basic arduino code was written to test the functionality of the actuator. The motor specified had more than enough torque to operate the valve, however it was discovered upon testing that the commercial couplers attaching the gear shaft to the top of the valve stem was inadequate. The coupler allowed ~5 degrees of play in both directions, causing the valve to wander over multiple open/close operations. A custom coupler with a machined slot may need to be manufactured to reduce play. Additionally, to facilitate testing, the coupler that holds the position sensing magnet to the top of the gear shaft was 3d printed. A metal version will need to be manufactured for use in the actual hardware.

## Next Steps and Recommendations

1. Manufacture a custom coupler with tighter tolerances and retest actuator assembly
2. Manufacture a metal coupler to hold the position sensing magnet
3. Review and revise arduino test code to ensure:
   1. Bug-free operation
   2. Error handling
   3. I/O compatibility with the TSAR central controller
4. Design an external N2 purge system for the LOX valve assembly
5. Integrate valves and actuators into test stand and test operation under pressure

## Actuator Arduino Test Code

Please check [github for code updates](https://github.com/psas/liquid-engine-test-stand/tree/master/main%20propellant%20valves/Arduino%20code%20for%20Testing/New%20Test%20Code/New_Test_Code), however sample code for initial valve operation testing is provided below for reference:

/\*\*\*\*\*\*\*Hydrotest code for Main Propellant Valves\*\*\*\*\*\*\*\*\*

7/30/19

Functions:

- Reads data from AS5047 rotary encoder over SPI interface

- Drives valve open 90 deg or closed 90 deg on button press via Pololu G2 motor driver

- Allows adjustments of position in 3 degree increments (or arbitrary)

- Note - program re-zeroes on startup, so reset arduino after making open/closed position adjustment

- Note - motor driver accepts PWM for speed control, however full speed is desired, so motor pin is driven HIGH or LOW.

SPI Fuctions

UNO: MOSI pin 11

MISO pin 12

CLK pin 13

CSN pin 10

\*/

#include <SPI.h>

//Variables

int dir;

int actuate;

int set\_angle;

int error\_angle;

int zero\_angle;

int delta;

int home\_delta;

int buttonstate1;

int buttonstate2;

int buttonstate3;

int CSN = 10;

int SO = 12;

int SI = 11;

int CLK = 13 ;

unsigned int angle;

int pos;

//Definitions for motor output & button input pins

int pin\_motor = 5;

int pin\_motordir = 4;

int pin\_button1 = 7;

int pin\_button2 = 8;

int pin\_button3 = 9;

void setup(){

//Set Pin Modes for magnetic encoder SPI interface

pinMode(CSN, OUTPUT);

pinMode(SI, OUTPUT);

pinMode(SO, INPUT);

pinMode(CLK, OUTPUT);

//Set Pin Modes for motor & buttons

pinMode(pin\_motor, OUTPUT);

pinMode(pin\_motordir, OUTPUT);

pinMode(pin\_button1, INPUT\_PULLUP);

pinMode(pin\_button2, INPUT\_PULLUP);

pinMode(pin\_button3, INPUT\_PULLUP);

//Set Slave Select High to Start i.e disable SPI chip

digitalWrite(CSN, HIGH);

//Initialize SPI

SPI.begin();

Serial.begin(9600);

SensorRead();

zero\_angle = pos; //Read current position & save as zero\_angle on startup.

Serial.println("Ready. Zero angle =");

Serial.println(zero\_angle);

delay(500);

}

void loop() {

buttonstate1 = digitalRead(pin\_button1);

buttonstate2 = digitalRead(pin\_button2);

buttonstate3 = digitalRead(pin\_button3);

if(buttonstate1 == LOW){

SensorRead();

home\_delta = abs(zero\_angle - pos); //Determine if currently in the home position

if(home\_delta > 180) {

home\_delta = abs(home\_delta - 360);

}

Serial.println("degrees from home pos:");

Serial.println(home\_delta);

if(home\_delta < 10) { //Less than 10 degrees from home, drive to +90

set\_angle = (zero\_angle + 90);

set\_angle %= 360;

dir = 0;

}

if(home\_delta >= 10) { //More than 10 deg from home, drive to home position

set\_angle = zero\_angle;

dir = 1;

}

// set\_angle = (pos + 90); //Old logic, removed due to possible accumulating errors

// set\_angle %= 360;

Serial.println("set angle:");

Serial.println(set\_angle);

DriveLogic();

}

// Experiment to see if you can budge the valve with a button press

if(buttonstate2 == LOW){

SensorRead();

set\_angle = pos + 3;

set\_angle %= 360;

Serial.println("Adjust +3 degrees");

dir = 0;

DriveLogic();

}

if(buttonstate3 == LOW){

SensorRead();

set\_angle = pos - 3;

if(set\_angle < 0){

set\_angle += 360;

}

Serial.println("Adjust -3 degrees");

dir = 1;

DriveLogic();

}

/\*\*\*\*\*\*\*\*\*\*Test of functions:\*\*\*\*\*\*\*\*\*\*\*\*\*

MotorCommand(1);

delay(1000);

SensorRead();

Serial.println(pos);

MotorCommand(0);

delay(1000);

SensorRead();

Serial.println(pos);

\*/

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Auxiliary functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SensorRead() {

SPI.beginTransaction(SPISettings(10000000, MSBFIRST, SPI\_MODE1));

//Send the Command Frame

digitalWrite(CSN, LOW);

delayMicroseconds(1);

SPI.transfer16(0x3FFF);

digitalWrite(CSN,HIGH);

//Read data frame

digitalWrite(CSN, LOW);

delayMicroseconds(1);

angle = SPI.transfer16(0xC000);

digitalWrite(CSN, HIGH);

SPI.endTransaction;

angle = (angle & (0x3FFF)); //This is a bitwise AND - not sure why it is here?

pos = ( (unsigned long) angle)\*360UL/16384UL; //convert 14 bit to angle

}

// \*\*\*\*\*MOTORCOMMAND,(1)=ON,(0)=OFF\*\*\*\*\*

void MotorCommand(int actuate) {

if(actuate == 1) {

digitalWrite(pin\_motordir,dir);

digitalWrite(pin\_motor,HIGH);

}

else {

digitalWrite(pin\_motor,LOW);

}

}

void DriveLogic() {

SensorRead();

delta = abs(set\_angle - pos); //intermediate variable to determine min angle between

if(delta > 180) {

error\_angle = abs(delta - 360);

}

else { error\_angle = delta;

}

while(error\_angle >= 3) { //Adding several degrees to allow for stopping time

SensorRead();

delta = abs(set\_angle - pos); //intermediate variable for 360 degree rollover handling - needs optimization

if(delta > 180) {

error\_angle = abs(delta - 360);

}

else { error\_angle = delta;

}

// if(error\_angle > 95){ //Dheck if error overrun exists & stop motor

// break;

// }

MotorCommand(1);

Serial.println("Motor on, error\_angle =");

Serial.println(error\_angle);

}

MotorCommand(0);

}