Jay A. Huber
U.C. Berkeley, Fall 2020

Joystick-Controlled Single-Motor Robotic Hand

OBJECTIVE

The objective of this project was to build a 3D printed robotic hand that would open and close in response to a joystick input. The fingers were not designed to move independently of one another, and I did not attempt to design robust joints and linkages that were capable of holding anything. The joint flexing behavior is cable driven, with a common spool pulling five individual cables, one for each finger. Thus, the device is designed to be capable of one 'grabbing' motion, with the finger position corresponding to that of the joystick.

MECHANICAL DESIGN

(a)

(b)

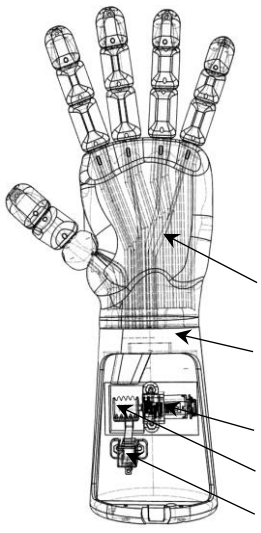


Figure 1: Fusion360 model of the assembly, (a) back side of rendering, and (b) front side, wireframe view.

This robotic hand assembly is composed of nineteen 3D printed parts; these are the palm, wrist, electronics compartment cover, three segments to form each finger, and a spool mounted on the motor shaft (see Appendix I for a comprehensive parts list, including electrical parts and hardware). The dimensions roughly correspond to those of my own left hand. The parts were designed in Fusion360, then built with a Prusa 3D printer (see Figures 1 through 6). The mechanical system is cable-driven. Stiff cables (monofilament fishing line) are used to flex the joints in each finger, while bungee cord (has an elastic core in a protective nylon sheath) is used for assembly. One channel runs along the inner side of each finger piece for the monofilament cable, and two channels run along the outer side of each finger piece for the elastic cable. Each finger is held together by a single piece of elastic cable, which is threaded down one side, around the tip of the finger, and down the other side (see Figure 3). The palm has a network of interior channels and a compartment to tie the cables (see Figure 1b).

Each finger is composed of three segments, each of which have edges that slope down at approximately forty-five degrees at the proximal and distal ends (see Figure 4). This allows the finger linkages (held in an open position by the elastic) to contract together into a flexed shape when the monofilament line is pulled. See the video (linked at the end of the paper) or the Engineering Drawings in Appendix II for more detail.

Because the motor position determined from the encoder is not necessarily accurate, particularly after startup, I added a switch to provide the absolute position. This roller switch is mounted next to the spool such that they roll across each other and the switch is triggered by a ridge on the spool (see Figure 2 and Figure 5e).

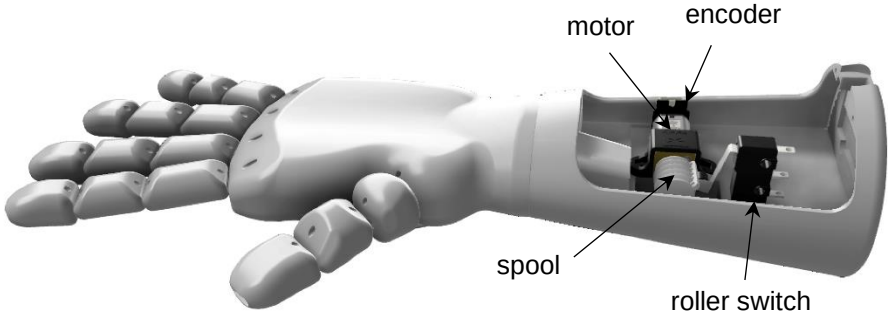


Figure 2: Fusion360 rendering of the assembly (front side). This image does not show the electronics compartment cover.

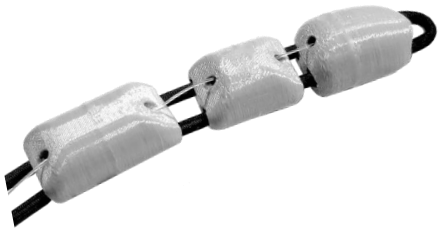


Figure 3: How cables are threaded through the fingers.

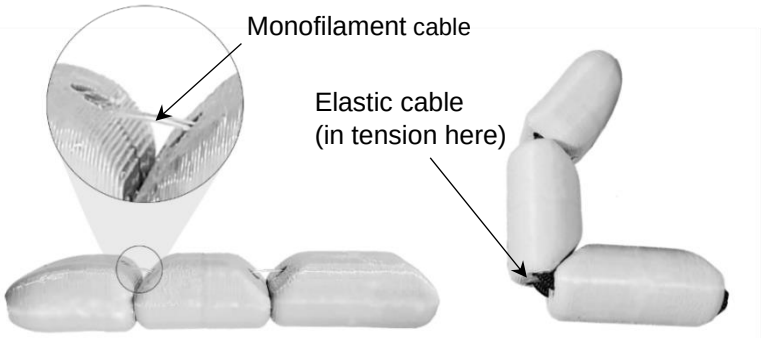


Figure 4: Pointer finger prototype, relaxed (left) and flexed (right).

CONSTRUCTION AND ASSEMBLY

The parts were printed in two segments each, because each print needed a flat side to print on due to limitations of the printer. This required printing many pieces, but I accomplished it in only a few prints because the pieces are small. The first step to assembly involved gluing component pieces together, then running the cables up and down each finger. After the elastic cables were threaded through the palm, they were tied with double-knots. Next, the wrist and palm were screwed together, and each monofilament cable was tied to the spool. See Figure 5 (below) and the demonstration video (linked on page 5) for documentation of the process. I used screws instead of glue so the device can be easily assembled/disassembled in the case that cables loosen or break and need to be replaced. The screws I used are self-tapping because the 3D printer could not produce the necessary sized threads. See Figure 6 (next page) for a photograph of the final assembly.

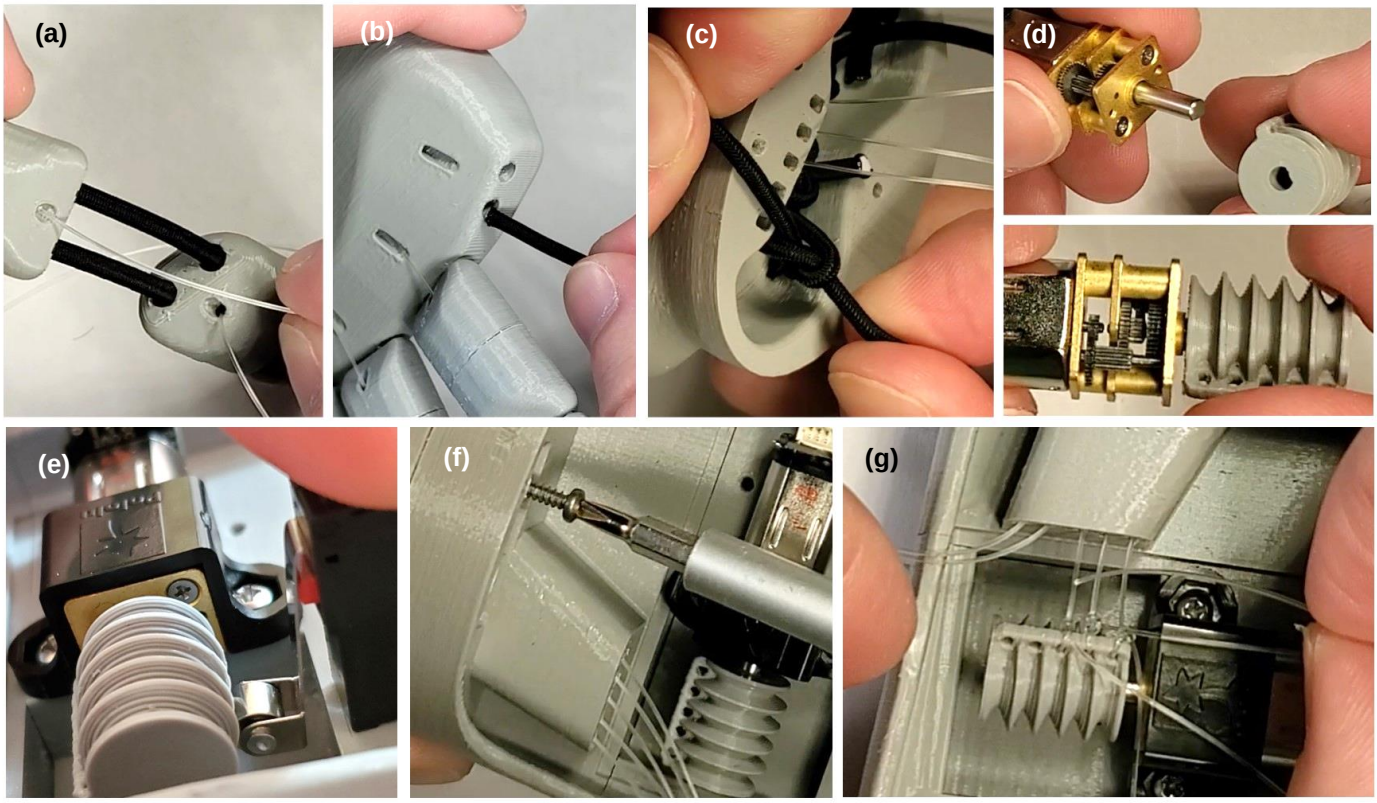


Figure 5: Assembly process. (a) Threading cables through the fingers, (b) Threading cables through the palm, (c) Tying elastic at the base of the palm, (d) Mounting the spool on the motor, (e) Placing the roller switch next to the spool, (f) Screwing the motor, palm, and wrist together into one assembly, (g) Tying the monofilament cables to the spool.

ELECTRICAL DESIGN

Figure 6: Completed hand assembly.

The electrical system is composed of a microcontroller, micro metal gearmotor, magnetic encoder, motor driver carrier, roller switch, lithium polymer battery, 5V power supply, 2-axis analog joystick, a capacitor (10μF), as well as some miscellaneous JST cables and jumper wires. Figure 7 (below) is a diagram of the wiring of the circuit. See the next section for information about the state machine and Appendix III for the code (done in C++).

The microcontroller (a PJRC Teensy development board) runs code generated and uploaded from the Arduino IDE, and is powered by a 3.7V 500mAh Lithium Polymer battery. The motor is run through a separate motor driver carrier and uses a 5V power supply (separate from the otherwise 3.3V system) to power it. I chose this microcontroller specifically because it runs using the Arduino IDE and is small enough to fit in the wrist portion of the device.

A ~25 mm pull of the monofilament cables will result in full hand flexion. Because the circumference of the spool is ~27 mm, this means full flexion is approximately one rotation of the motor. This rotation is mapped to half the vertical range of the joystick used for control. A dual axis joystick was used, but useful input is restricted to input from the vertical direction (+Y for extension and -Y for flexion).

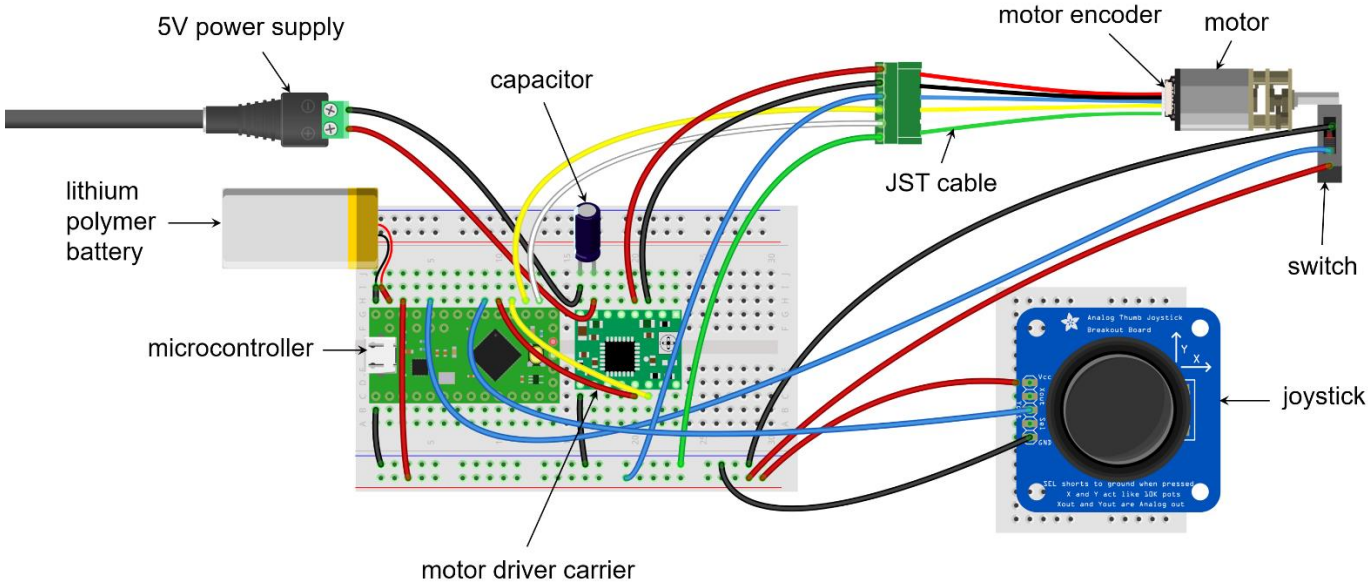


Figure 7: Electrical assembly.

STATE MACHINE

Movement of the joystick in +Y is an event that triggers extension, whereas -Y triggers flexion. At the device's physical limits, flexion and extension are stopped by software defined events. The amount of hand flexion (thus position of the motor) is controlled by the degree of the joystick. Therefore, the further the joystick is from the nominal position, the further it will flex or extend. The finite state machine is detailed in Figure 8 (see next page). The state machine implements a proportional-integral-derivative (PID) control methodology using feedback from the magnetic encoder. Additionally, the roller switch mounted next to the spool triggers each time the motor makes a full rotation. Because only one rotation is

desired, the behavior consists of the motor moving forwards and backwards in one cycle. The roller switch mounted to the motor is triggered just before each time the motor is at the ‘closed’ (or hand fully flexed) state.

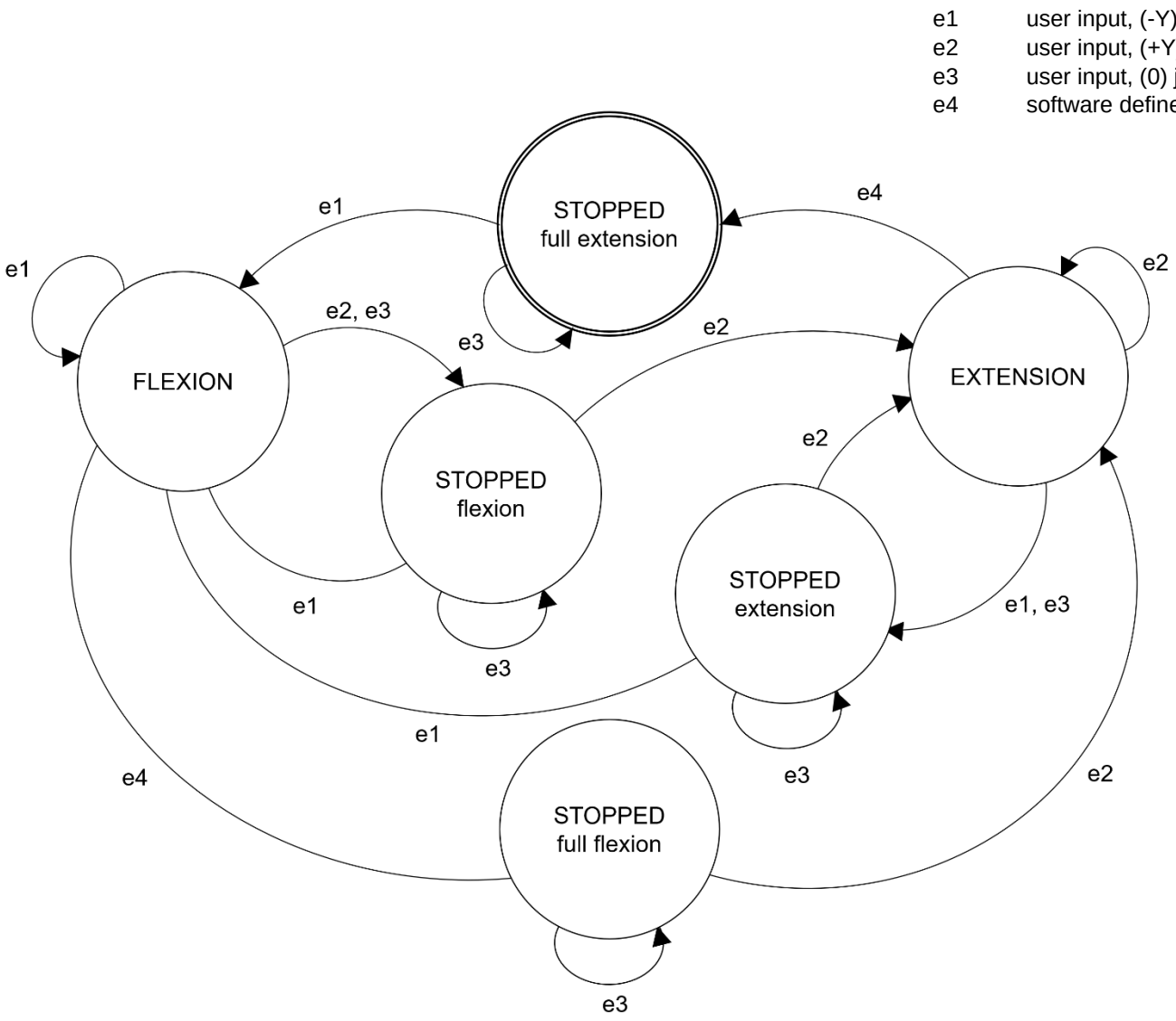


Figure 8: Finite State Machine for this system.

PERFORMANCE AND FUTURE WORK

The device largely performs as I expected. Unfortunately, the micro metal gearmotor used does not have enough torque to fully close all fingers at once. I would like to further develop the device in the future by replacing the motor with either a different model or using multiple motors of the same type (to control different fingers separately), so that all the fingers can fully flex. Ultimately, I untied the thumb, pointer, and pinkie finger cables from the spool because I did not want to cause damage to the motor by repeatedly stalling it. The final flexed hand is shown in Figure 9 and in the demonstration video.

Figure 9: Flexed hand.

AKNOWLEDGEMENTS

A huge thank you to my friend Mike B., who let me use his 3D printer for this project! Also, thank you to the ME 102B course staff for answering questions and providing resources.

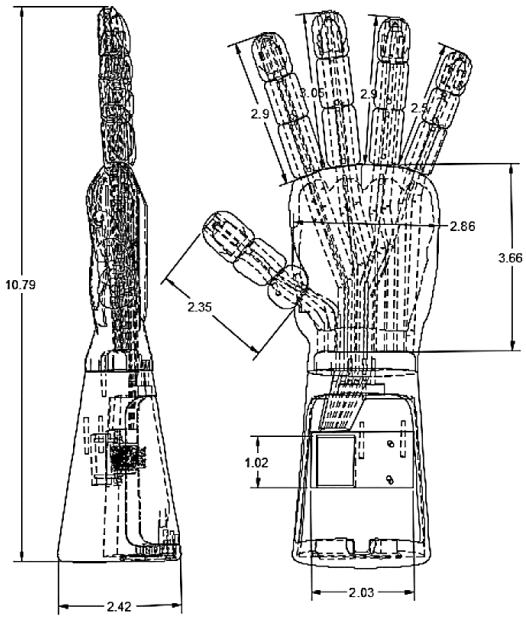
APPENDIX I

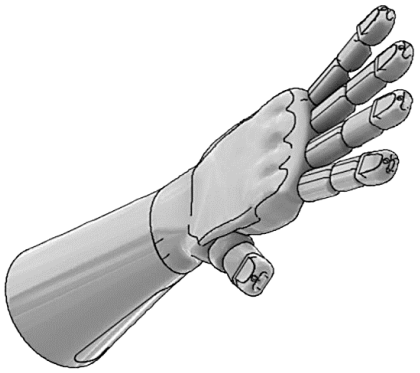
Parts List

	PART	DESCRIPTION	APPLICATION
1	Microcontroller	PJRC Teensy LC development board (runs Arduino) https://www.pjrc.com/store/teensylc.html	Controlling system
2	Lithium polymer battery	3.7V, 500 mAh Lithium Polymer battery https://www.adafruit.com/product/1578	Powering microcontroller
3	Micro metal gearmotor	Pololu #2215, 75:1 Micro Metal Gearmotor HP 6V with extended motor shaft https://www.pololu.com/product/2215	Pull cables onto spool
4	Encoder for micro metal gearmotor	Magnetic Encoder Pair Kit with Side-Entry Connector for Micro Metal Gearmotors, 12 CPR, 2.7-18V https://www.pololu.com/product/4761	Mounted to motor, power/data connection
5	5V power supply	5V 2A switching power supply https://www.adafruit.com/product/276	Powering motor
6	Motor driver carrier	DRV8833 Dual Motor Driver Carrier, item #2130 https://www.pololu.com/product/2130	Control motors, report data to controller
7	Roller switch	10T85 SPDT micro roller switch, 5A 125V http://www.farnell.com/datasheets/1685049.pdf	Provide absolute position
8	Capacitor, 10μF	Jackcon, 10μF 35V https://www.jackcon.com.tw/aluminum-electrolytic-capacitor.htm	Mitigate horrible motor noise
9	Female JST SH-style cable	6-Pin Female JST SH-Style Cable 30cm https://www.pololu.com/product/4763	Motor information to board
10	Bracket for micro metal gearmotor	Pololu #989 Micro Metal Gearmotor Bracket https://www.pololu.com/product/989	Mounts motor in place
11	Analog joystick	2-axis analog joystick with select button https://www.adafruit.com/product/512	Motor control input
12	Fishing line	South Bend Essentials, monofilament fishing line, 20lb test tension, 140 yd http://www.south-bend.com/	Joint flexing cable
13	Elastic cord	Elastic bungee cord, 2.5 mm x 25 ft https://www.paracordplanet.com/	Constructing/tensioning the joints
14	3D printed parts (19)	19 components printed on a Prusa in gray PETG. Three sections of each finger (15 pc), palm (1), wrist (1), electronics compartment cover (1), spool (1) https://prusament.com/	Mechanical Parts
15	Screws (6)	Self-tapping screws for plastic https://www.mcmaster.com/plastic-screws/thread-cutting-screws-for-plastic/	Holding plastic parts together, securing motor bracket

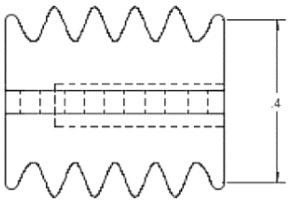
APPENDIX II

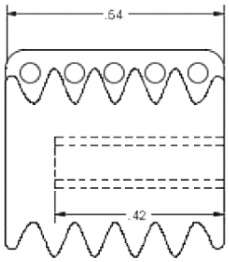
Engineering Drawings

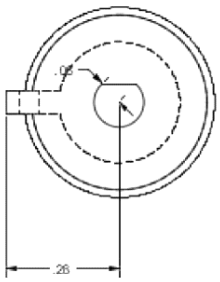


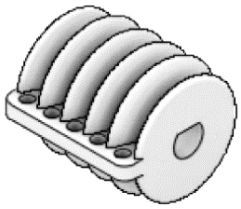


	PROJECT: Mechatronics, Arm				
	TITLE: Assembly				
APPROVED	SIZE: B	CODE	DWG NO		REV
CHECKED					
DRAWN Jay Huber 11/13/2020	SCALE 2:3	WEIGHT	SHEET 1/1		









	PROJECT: Mechatronics, Arm				
	TITLE: spool				
APPROVED	SIZE: B	CODE	DWG NO		REV
CHECKED					
DRAWN Jay Huber 11/13/2020	SCALE 1:1	WEIGHT	SHEET 1/1		

APPENDIX III

Code

```
//----------------------------------------------------------------------
// DEFINING GLOBAL VARIABLES
//----------------------------------------------------------------------
int motor_a1 = 16;
int motor_a2 = 17;
int enc_a = 14;
int enc_b = 15;
int joystick_pin = A4;
int roller_switch_pin = A8;

//----------------------------------------------------------------------
// ENCODER STATE MACHINE
//----------------------------------------------------------------------
int pos = 0;
int errors = 0;
void update_encoder(void){
  static int current_state = 0;
  int next_state = digitalRead(enc_b)<<1 | digitalRead(enc_a);
  switch(current_state){
    case 0:
      switch(next_state){
        case 0: break;             // no change
        case 1: pos++; break;      // forward
        case 2: pos--; break;      // reverse
        default: errors++; break; // error
      }
    break;
    case 1:
      switch(next_state){
        case 0: pos--; break;      // reverse
        case 1: break;             // no change
        case 2: errors++; break;   // error
        default: pos++; break;     // forward
      }
    break;
    case 2:
      switch(next_state){
        case 0: pos++; break;      // forward
        case 1: errors++; break;   // error
        case 2: break;             // no change
        default: pos--; break;     // reverse
      }
    break;
    default: // 3
      switch(next_state){
        case 0: errors++; break;   // error
        case 1: pos--; break;      // reverse
        case 2: pos++; break;      // forward
        default:break;             // no change
      }
    break;
  }
  current_state = next_state;
}

//----------------------------------------------------------------------
// INPUTS
//----------------------------------------------------------------------
int joystick = 0;
volatile byte roller_switch = LOW;
void update_inputs(void){
  joystick = analogRead(joystick_pin);
  roller_switch = analogRead(roller_switch_pin);
}

//----------------------------------------------------------------------
// MOTOR CONTROL
//----------------------------------------------------------------------
void motor_stop(void){
  digitalWrite(motor_a1, LOW);
  digitalWrite(motor_a2, LOW);
}
void motor_open(void){
  digitalWrite(motor_a1, HIGH);
  digitalWrite(motor_a2, LOW);
}
void motor_close(void){
  digitalWrite(motor_a1, LOW);
  digitalWrite(motor_a2, HIGH);
}
#define MOTOR_MAX 100
#define MOTOR_MIN 0

void update_motor_control(void){
  int current_pos = pos;
  if (current_pos > joystick+2){
    motor_close();
  }
  else if (current_pos < joystick-5){
    motor_open();
  }
  else if (roller_switch == HIGH ){
    motor_stop();
  }
  else {
    motor_stop();
  }
}

//----------------------------------------------------------------------
// STATS
//----------------------------------------------------------------------
void update_stats(void){
  static int loops = 0;
  if (loops++ > 10000){
    loops = 0;
  }
}

//----------------------------------------------------------------------
// MAIN
//----------------------------------------------------------------------
void setup(){
  Serial.begin(9600);
  pinMode(motor_a1, OUTPUT);
  pinMode(motor_a2, OUTPUT);
  pinMode(enc_a, INPUT);
  pinMode(enc_b, INPUT);
  pinMode(joystick_pin, INPUT);
  pinMode(roller_switch_pin, INPUT);
}
void loop() {
  update_encoder();
  update_motor_control();
  update_inputs();
  update_stats();
}
```