# Appendix A

This program reads analogue inputs from the ASEMG circuits as well as from the fedback circuit and produces the proportional Servo/buzzer responses. The analogue values are also transmitted via Serial port for debug purposes.

#include <Servo.h>

Servo MyServo;

//Variable Declaration PINs

int AInput7 = 7; //Analog Input pin A7 Band Position Control

int AInput5 = 5; //Analog Input pin A5 Sesorial position Feedback

int PWMOutput = 11; //PWM Output pin 11 Sesorial feedback

int ServOut = 10; //Servo signal connected to pin D10

int DirPIN = 3; //Digital Pin to change direction

int CalPIN = 4; //Digital Pin to start the calibration protocol

int ledPIN = 13; //Digital Pin with a led

//Global Variables

/\*\*Position Control\*\*/

int AValueCtrl = 0;

float RMS = 0;

int ServAng = 0;

int DeltaAng=0; //add/dim servo's angle

int counter=0; //use to calculate Signal's RMS value

/\*\*Sensorial Feedback\*\*/

int PWMDuty = 0; //Duty to control the vibrating motor

int AValueFbk = 0; //analogue value read from the analogue channel

int NSample=1; //counts to calculate the average

float AnaAverg=0; //to calculate the average of the last 10 samples

volatile int dir=-1; //to choose direction negative: close; positive: open;

/\*\*Code Timing\*\*/

int TimeIni = 0;

int TimeEnd = 0;

int DeltaT = 0;

bool Active = false;

void setup() {

// put your setup code here, to run once:

//Activate serial transmision

Serial.begin(115200);

//set analog referece for the ADC

analogReference(DEFAULT);

//Pin Setup

pinMode(A5, INPUT); //Used for the control signal

pinMode(A7, INPUT);//Used for the feedback signal

pinMode(ServOut, OUTPUT);//Claw psoition control

//Servo Setup

MyServo.attach(ServOut);

MyServo.write(40); //Initial position, Widely Open

//Interrupts Setup

attachInterrupt(digitalPinToInterrupt(DirPIN),ButtonRed,FALLING);

attachInterrupt(digitalPinToInterrupt(CalPIN),ButtonBlack,FALLING);

}//End Setup

void loop() {

/\*\*\*\*\*\*Position Control\*\*\*\*\*\*/

//Read a Value from the sensor:

AValueCtrl = analogRead(AInput7);

//Serial.println(AValueCtrl);

RMS = 0;

if (AValueCtrl>=550){

TimeIni = millis();

for (counter=0; counter<=100; counter++)

{ RMS = RMS + AValueCtrl;

// Serial.print(RMS);Serial.print(", ");

Serial.println(AValueCtrl);

// Serial.println(counter);

delay(5);

AValueCtrl = analogRead(AInput7);

if (AValueCtrl<550) {

break;

};

};

TimeEnd = millis();

DeltaT= TimeEnd-TimeIni;

// Serial.print("Delta T = ");Serial.println(DeltaT);

RMS = RMS\*(counter+1)\*(0.001);

Serial.print("RMS = ");Serial.print(RMS);Serial.print("/");Serial.println(counter);

DeltaAng = 0.01\*RMS;

Serial.print("DeltaAng= ");Serial.println(DeltaAng);

ServAng = ServAng + (DeltaAng\*dir);

if(ServAng>40){

ServAng = 40;

}; //Can't open wider!

if(ServAng<0){

ServAng = 0;

}; //Can't close narrower!

MyServo.write(ServAng);

Serial.print("ServAng= ");Serial.println(ServAng);

}

/\*\*\*\*\*\*End Position Control\*\*\*\*\*\*/

/\*\*\*\*\*\*Sensory Feedback Control\*\*\*\*\*\*/

//Digital Low-pass Filter:

//Reads 10 values from the AInput, and take the average:

//Sampling rate, 5ms

AnaAverg = 0;

for(int NSample=1; NSample<=20; NSample++){

//Read a Value from the sensor:

AValueFbk = analogRead(AInput5);

AnaAverg = AnaAverg + AValueFbk;

delay(5);

};

AnaAverg = AnaAverg/20;

/\*\*End Digital Low-pass Filter \*\*\*\*\*\*\*\*/

// Escale value:

PWMDuty = 0.25\*AnaAverg + 25;

if (PWMDuty<60) PWMDuty=70; //Min value that can be felt

if (PWMDuty>255) PWMDuty=255;//Max value in the motor

//Send the PMW value to the motor:

analogWrite(PWMOutput, PWMDuty);

// Print value via serial

Serial.print("Position= ");Serial.print(AnaAverg);Serial.print(", ");

Serial.print("Duty= ");Serial.println(PWMDuty);

/\*\*\*\*\*\*End Sensory Feedback Control\*\*\*\*\*\*/

}//End Loop

//Interruption Routine Service - Red Button

//When the button is pressed, the direction of the claw changes from opening to closing

//and viceverse

void ButtonRed(){

dir\*=-1;

if(dir<0){

Serial.println("closing");

}

else{

Serial.println("opening");

};

};//End Button Red

//Interruption Routine Service - Black Button

//When the button is pressed, the configuration mode for the feedback starts

void ButtonBlack(){

int Value = 0;

//Blink three times, to start calibration

digitalWrite(ledPIN, LOW);

delay(500);

digitalWrite(ledPIN, HIGH); //1

delay(300);

digitalWrite(ledPIN, LOW);

delay(300);

digitalWrite(ledPIN, HIGH); //2

delay(300);

digitalWrite(ledPIN, LOW);

delay(300);

digitalWrite(ledPIN, HIGH); //3

delay(300);

digitalWrite(ledPIN, LOW);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

do{

//Read the analogue port and change resistance in Zero Pot until it reaches 200 counts

Value = analogRead(AInput5);

delay(100);

}while(Value!=200);

//Blink once to continue calibration

digitalWrite(ledPIN, HIGH); //1

delay(300);

digitalWrite(ledPIN, LOW);

delay(300);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

do{

//Read the analogue port and change resistance in Span Pot until it reaches 950 counts

Value = analogRead(AInput5);

delay(100);

}while(Value!=950);

//Blink Twice to finish calibration

digitalWrite(ledPIN, HIGH); //1

delay(300);

digitalWrite(ledPIN, LOW);

delay(300);

digitalWrite(ledPIN, HIGH); //2

delay(300);

digitalWrite(ledPIN, LOW);

delay(300);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

loop();//Starts main program again

};//End Black Red

# Appendix B

#include <Servo.h>

Servo servo;

const int numvalue = 10;

int rI1 = 0;

int sum = 0;

int avg = 0;

int rI2 = 0;

int total = 0;

int average = 0;

const int threshold1 = 410;

const int DeltaAng=1;

const int threshold2 = 380;

int ang=0;

int inputpin = A0; // to measure pulses

int input[10]={0,0,0,0,0,0,0,0,0,0};

int fdbkpin = A5 ; //to measure current

int fdbk[10]={0,0,0,0,0,0,0,0,0,0};

int dir=-1;

boolean PulseFLG=false;//true if a pulse is detected

boolean ObjectFLG=false;//true if avercurrent detect an object when closing the claw

boolean OCurrentFLG=false;//true if overcurrent is detected

/\*\*\*\*\*\*\*\*\*Functions\*\*\*\*\*\*\*\*\*/

boolean ReadPulse(){

for (int rI1 = 0; rI1 < numvalue; rI1++) {

sum = sum - input[rI1];

input[rI1] = analogRead(inputpin);

Serial.print("band ");Serial.print(rI1);

Serial.print(" ");Serial.println(input[rI1]);

sum = sum + input[rI1];

delay(50);

if (rI1 >= numvalue) {rI1 = 0; };

}//end for

avg = sum / numvalue;

Serial.print(" band avg ");Serial.println(avg);

if((avg > threshold1)){ return true;}

else {return false;};//end if else

};//End Read Pulse

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

boolean OverCurrent(){

total=0;

average=0;

for(rI2=0; rI2<numvalue; rI2++){

fdbk[rI2]=analogRead(fdbkpin);

Serial.print(" current ");Serial.println(fdbk[rI2]);

total = total+fdbk[rI2];

}

average=total/10;

Serial.print(" average ");Serial.println(average);

if (average>threshold2){ return true;}

else {return false;};//end if else

};//End Overcurrent

void setup() {

pinMode(A0, INPUT);

pinMode(A5, INPUT);

Serial.begin(9600);

servo.attach(9);

ang=170;

servo.write(ang);delay(1000);

Serial.println("OK");

}

void loop() {

/\*\*\*\*\*Reset Flags\*\*\*\*/

PulseFLG=false;

ObjectFLG=false;

OCurrentFLG=false;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(!PulseFLG){//loop to detect pulse

PulseFLG=ReadPulse();

};//end While

do{//close claw until obect is detected

ang=ang+(DeltaAng\*dir);

if(ang==99){

ang=170;

PulseFLG=true;//to jump to reset

break;

};//limit condition, no object found

Serial.print(" new ang ");Serial.println(ang);

servo.write(ang);

OCurrentFLG=OverCurrent();//check for overcurrent

while (OCurrentFLG){//if Overcurrent is detected

ObjectFLG=true;

ang=ang-(DeltaAng\*dir);

servo.write(ang);

};//end while no overcurrent

}while(!ObjectFLG);//End DO-While

Serial.println("OBJECT");

delay(5000);

while(!PulseFLG){//loop to detect second pulse

PulseFLG=ReadPulse();

};//end While

/\*\*\*\*\*\*\*Reset Claw Position\*\*\*\*\*\*\*/

ang=170;

servo.write(ang);

delay(2000);// wait 2sec

Serial.println("reseted");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}