MECH 442
Mechatronics

LOAD SENSITIVE DRILL PROJECT REPORT

Koç University
Mechanical Engineering

Instructor:

İsmail Lazoğlu

Team Members:

Ezgi Çevik
Cem Tekin

1. Introduction:

In this project a load sensitive drill was designed. During orthopedic surgeries, damaging the muscles under the bones is a critical problem. The surgeon needs to be precise to avoid the penetration of the soft tissue. This results a poor stability in the surgery. The problem can be solved using a drill which will stop working when drilling of the bone is finished. The drill will be terminated immediately when it moves into another material which will prevent the drilling of the other material.

Ma et al. designed an intelligent bone drill and control method, which could provide automatic stop and reverse. They used torque sensor to understand the interaction of the cutting tool and the bone [1]. The design of the drill is shown in Figure 1.

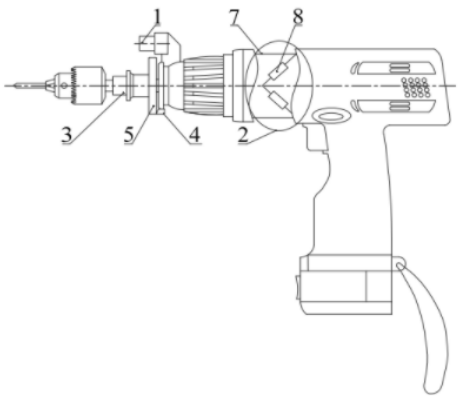


Figure 1: Intelligent bone drill [3].

1. Ultrasonic transducer
2. Strain torque sensor
3. Drill bit
4. Hall sensor
5. Metal reading dial
7. Annular metal part
8. Strain gages.

The automatic cranial drill invented by Li et al. could sense drilling-through and stop feeding immediately, protecting brain safety of patients by using strain gauge [2].

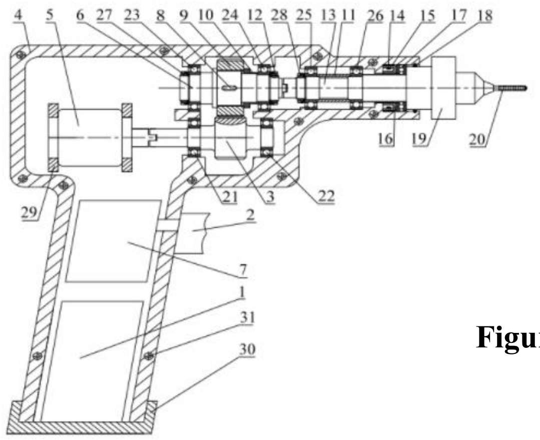


Figure 2: Axial force controllable bone drill [3].
14. Resistance strain gage

Marcos Louredoa , Inaki Diaz, Jorge Juan Gila used an optical encoder to track drill bit’s position error as shown in Figure 3,4,5 [4].

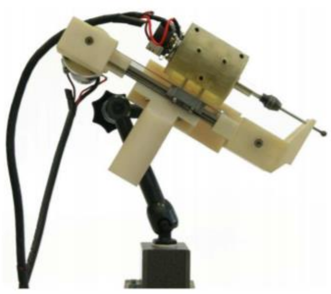


Figure 3: DRIBON system [3]

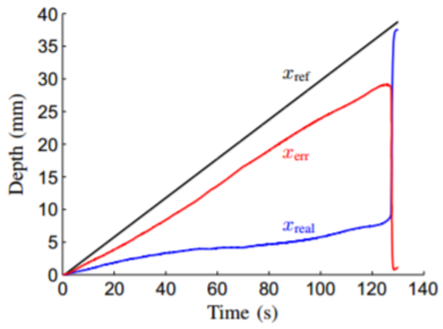


Figure 4: Position signals measured during the drilling process (t = 0 s bone drilling starts, t = 125 s bone protrusion starts) [3].

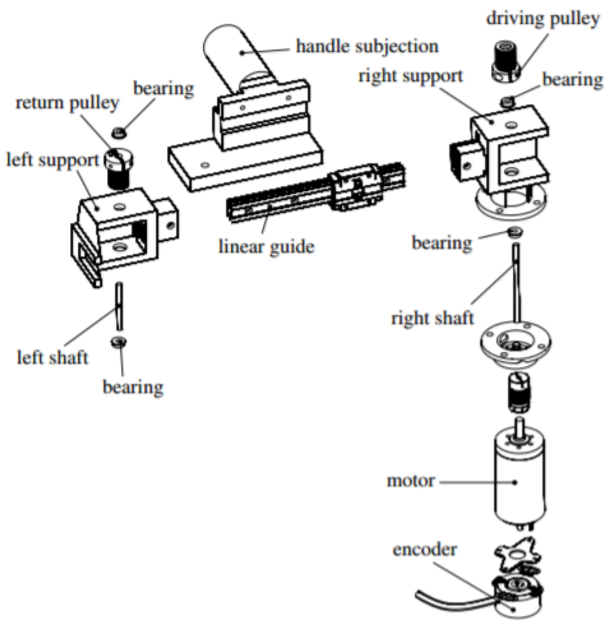


Figure 5: Components of the drilling guide and body of the system [3].

After the literature survey is done, we construct a design by using the relationship between the current and the torque in the DC motor. The design and the method will be explained in section 2 - 3, and the results will be discussed in section 4.

2. Experimental Setup and Budget

1. Spindle Motor: 300 TL

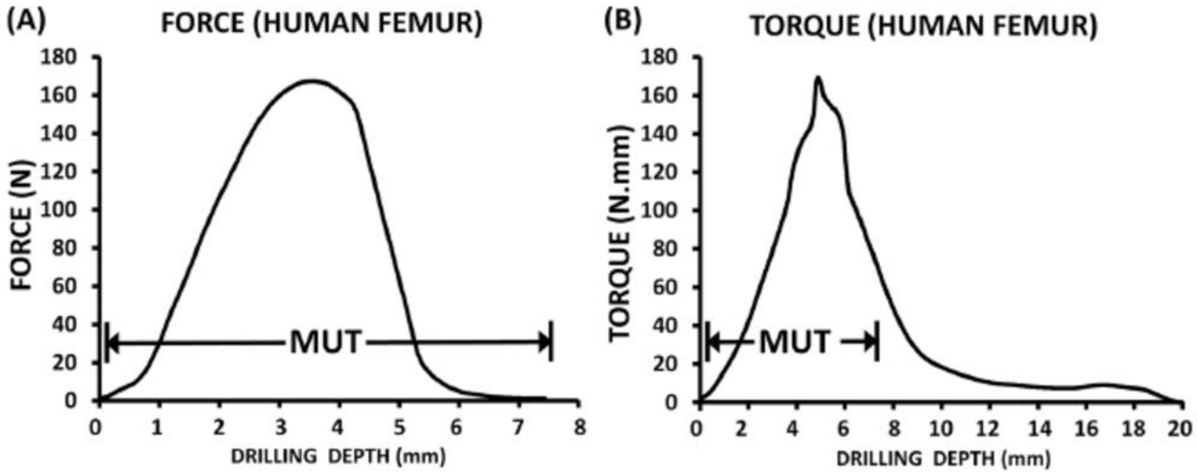


Figure 6: Force and Torque data of drilling a bone [5]

The figure above shows the force and torque data collected while drilling a femur bone. From this figure, it can be said that at least a spindle motor with torque of 180 Nmm. In addition, Pandey and Panda stated that the rotational speed of the drill should be around 3000 rpm [6]. There was not any dc spindle motor in Turkey which satisfied these requirements. Therefore, we bought the motor on eBay. It is a 400W 48V DC spindle motor whose maximum torque and speed are 500 Nmm and 12000 rpm respectively.

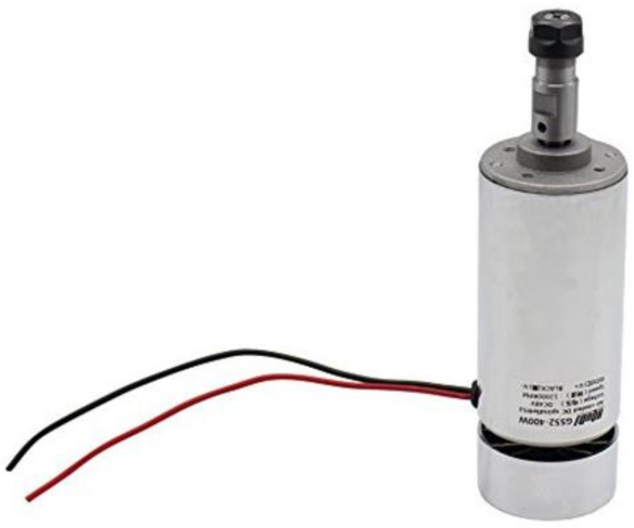


Figure 7: DC Spindle Motor

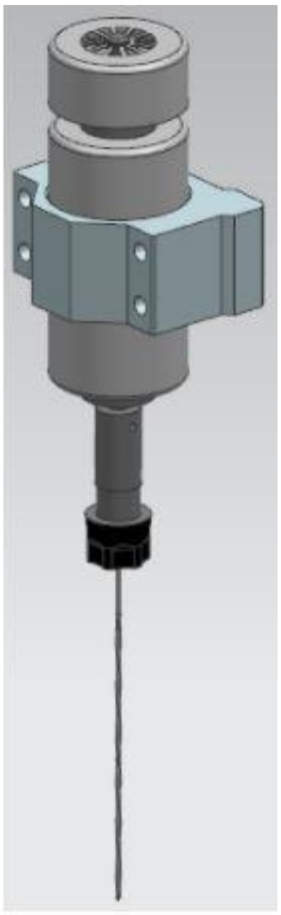


Figure 8: CAD drawing of the spindle motor with its mounting and the drill bit

2. IRFZ48N Power MOSFET

The MOSFET and an Arduino MEGA are used to control the DC spindle motor by PWM signal. The control circuit is given below.

Figure 9: MOSFET

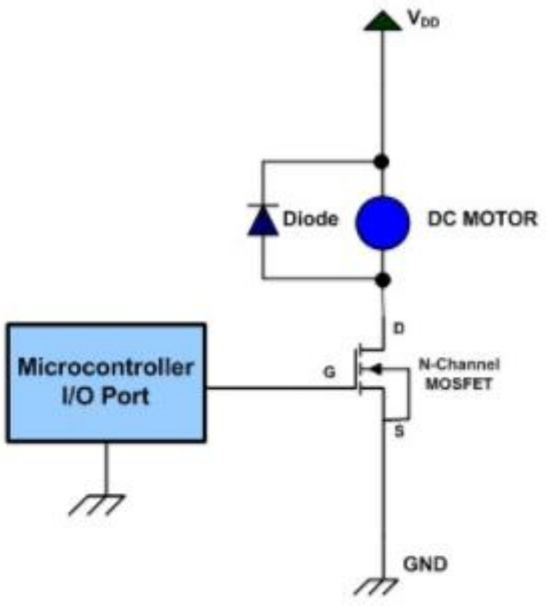


Figure 10: DC Motor control

3. Drill Bit

Drill bits for bone drilling are different from the regular drill bits. They have to have small diameters; however, they have to be longer than usual drill bits. They Pandey and Panda stated that optimal drill bit size for bone drilling is around 3 mm [6]. Therefore, we used a 3 mm drill bit.

Figure 11: Drill bit

4. Stepper Motor. 200 TL

Macavelia et al. found that at least a drill force of 180 N required [5]. We took their findings into account while selecting the stepper motor.

The motor that we chose is a hybrid NEMA size 34 1.8^{o} 2-phase stepper motor which has a torque of 2.2 Nm.

Figure 12: Stepper motor

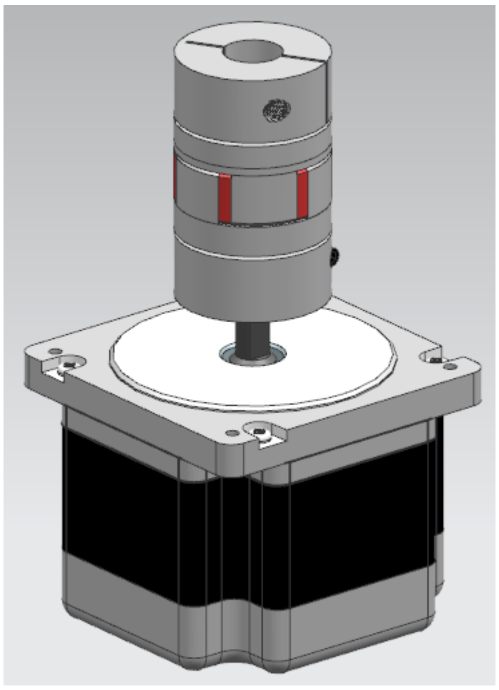


Figure 13: The CAD drawing of the stepper motor and coupling

5. Stepper Motor Driver: 380 TL

We bought an H-Bridge, 2 Phase Bi-polar Micro-stepping Drive. It is Power Step 2M982. The current passing through it is adjustable. The driver and its specifications are shown in the figures below.

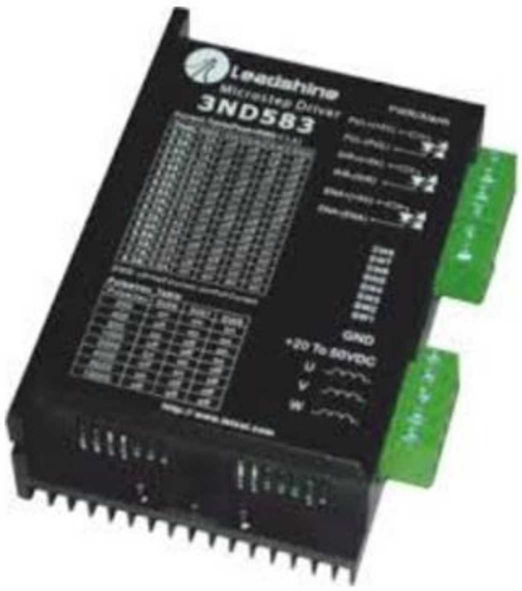
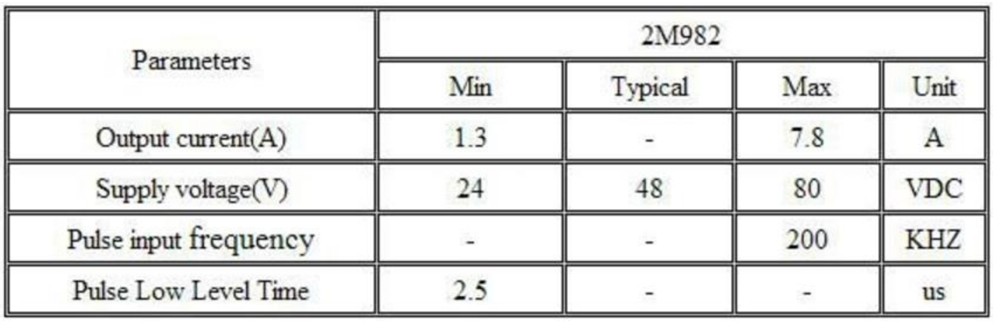


Figure 14: The stepper motor driver



Parameters	2M982			
	Min	Typical	Max	Unit
Output current(A)	1.3	-	7.8	A
Supply voltage(V)	24	48	80	VDC
Pulse input frequency	-	-	200	KHZ
Pulse Low Level Time	2.5	-	-	us

Table 1: The stepper motor specifications

6. Coupling: 30 TL

It is just for coupling the ball screw shaft and the stepper motor.

7. Linear Guides: 2x120=240 TL

They are coupled with the ball screw to create linear motion from the rotational motion of the stepper motor. 2 of them are used to eliminate the bending and drill stays vertical.

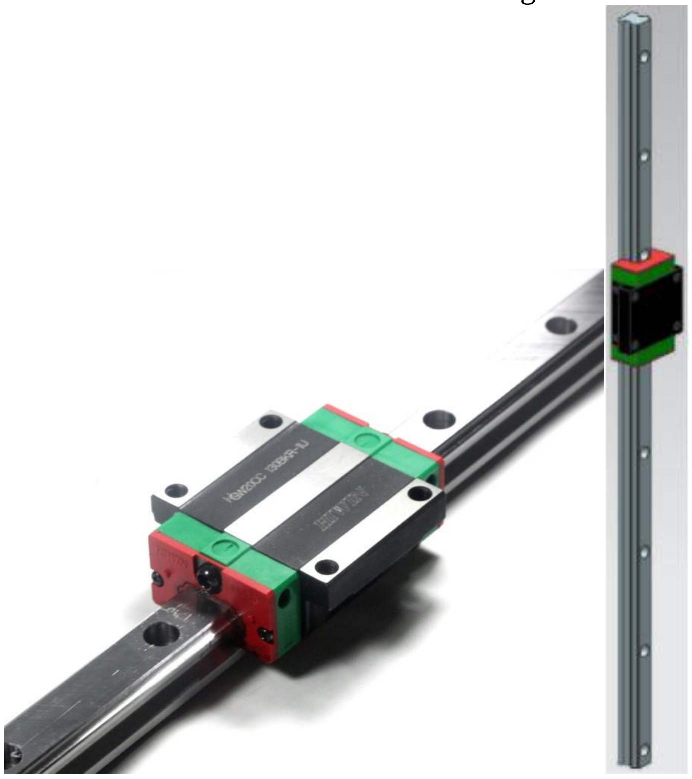


Figure 15: The linear guide

8. Ball Screw: 300 TL

The ball screw is used to convert rotational motion of the stepper motor to linear motion.

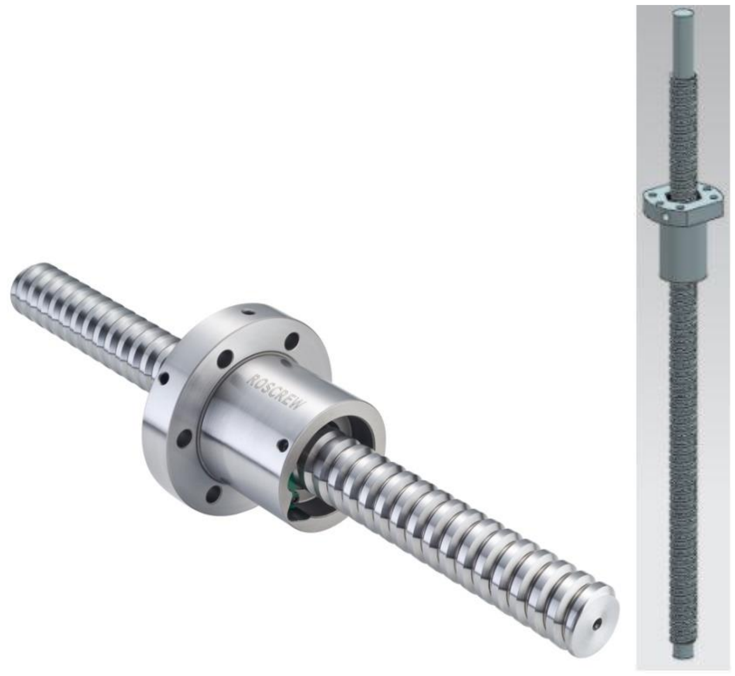


Figure 16: Ball screw

9. Ball bearing: 10 TL

The shaft of the ball bearing was machined for this 12 mm ball bearing. It is mounted on an aluminum plate.

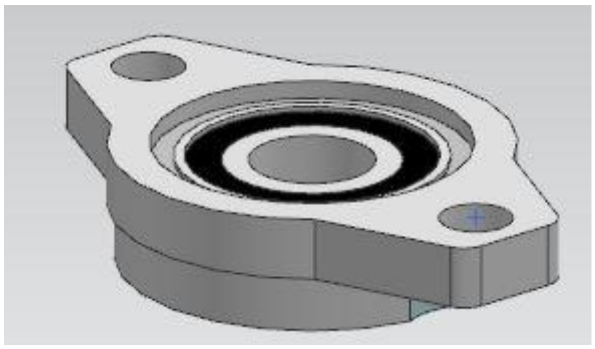


Figure 17: Ball Bearing

10. Connector

The connector is manufactured from steel to couple the spindle motor, linear guides and ball bearing.

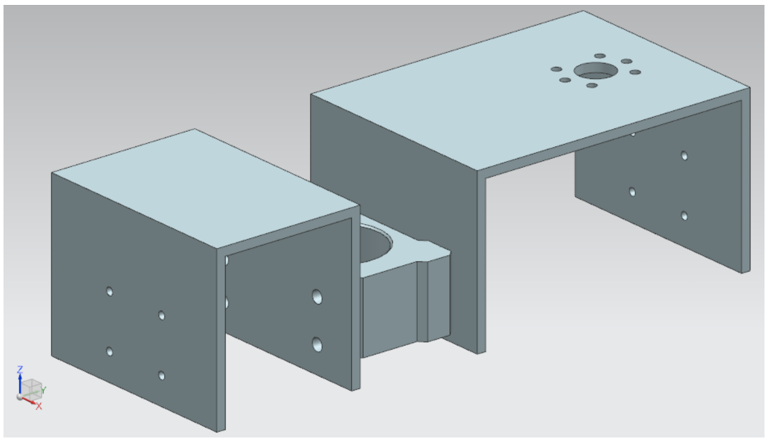


Figure 18: The CAD drawing of the connector

11. Sigma Profiles: 45 TL

Two 600mm sigma profiles with 45mmx45mm cross section are used to eliminate the bending of the system.

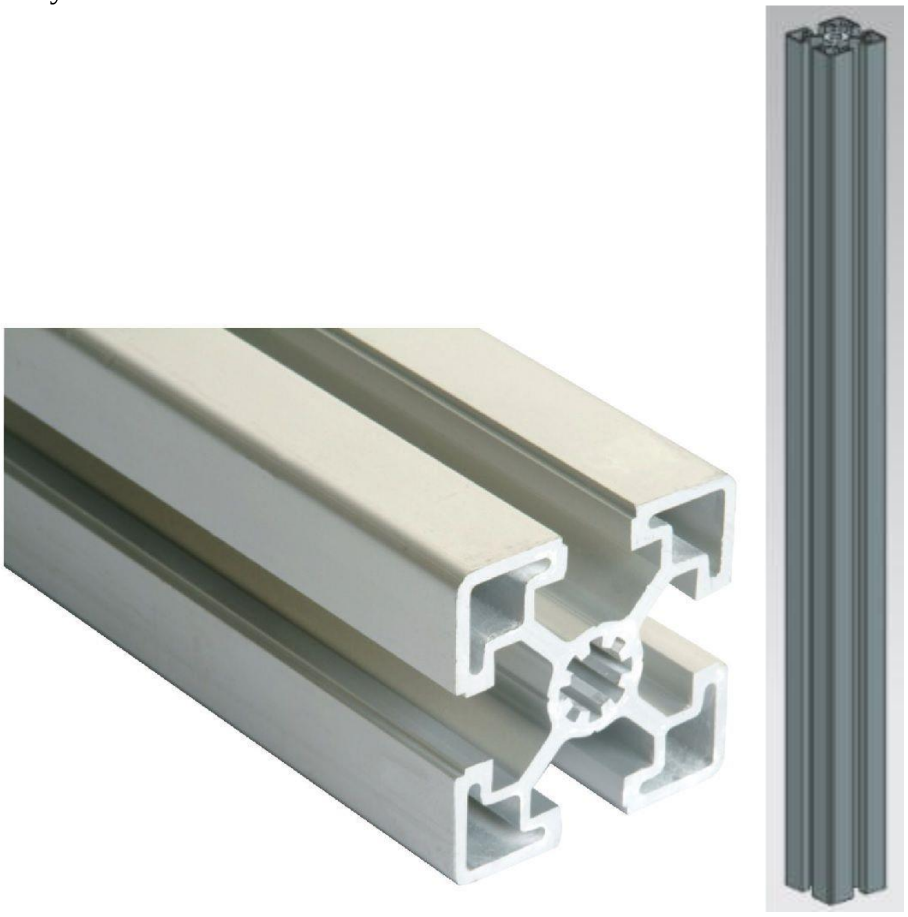


Figure 19: Sigma Profiles

12. Arduino MEGA: 50 TL

Arduino MEGA was used as a microcontroller to provide PWM signal to the DC motor and get the current data passing through the motor.

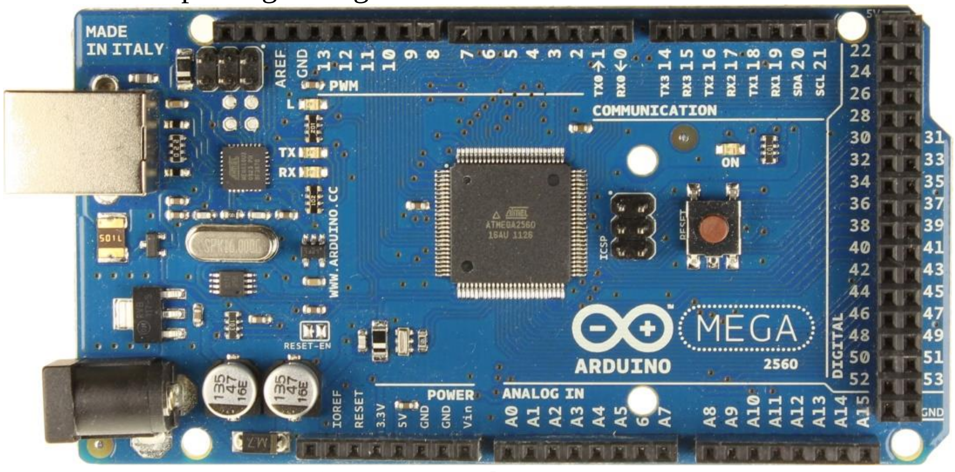


Figure 20: Arduino MEGA

13. Current sensor: 10 TL

AC712 Current Sensor was used to measure the current passing through the spindle motor. The sensitivity of the sensor is 66 mV/A and the resolution is 58.6 mA which are enough because during the drilling, the current increases at least 500 mA.

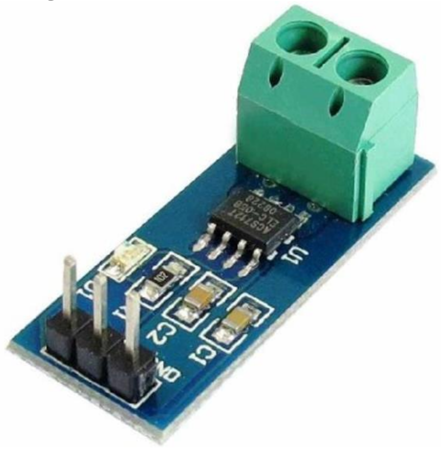


Figure 21: AC712 Current Sensor

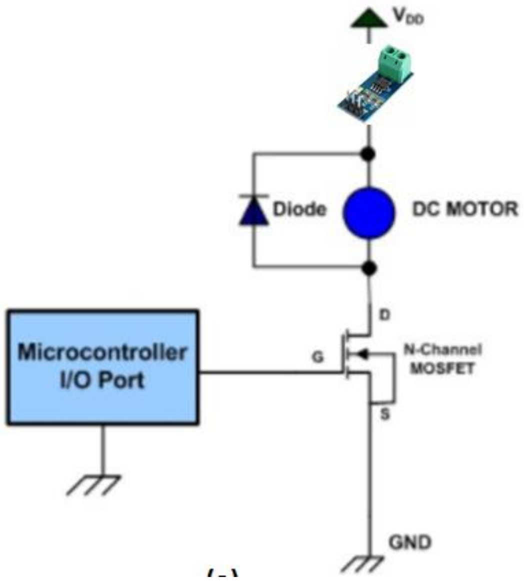


Figure 22: Current Sensor placement

There was a problem with the measurement of sensor. Since the signal given to the DC spindle motor is PWM voltage, the current measured was also in PWM signal. Thus, a low pass filter is used to stabilize the measurement.

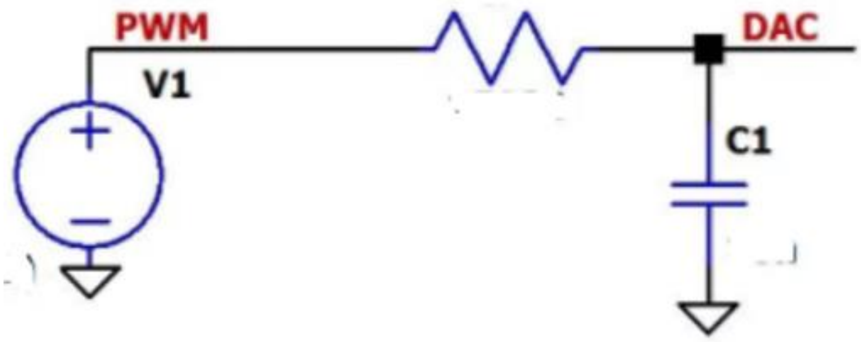


Figure 23: Single pole low pass filter

14. The assembly

The assembly is shown in the figure below.

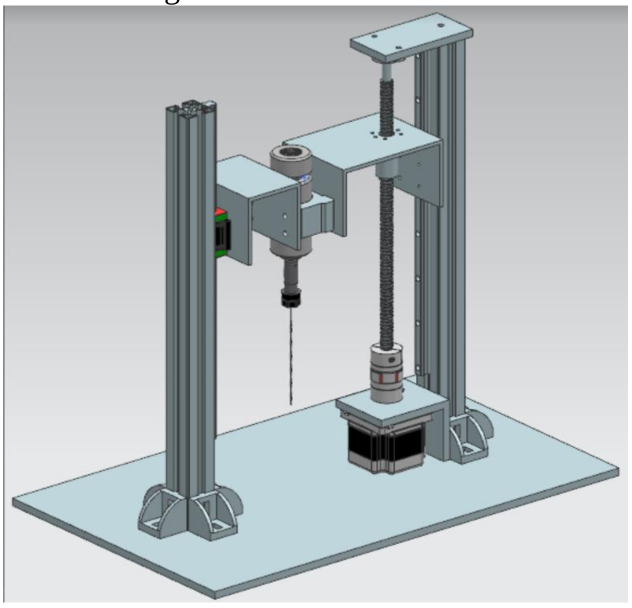


Figure 24: The assembly

3. **Method:**

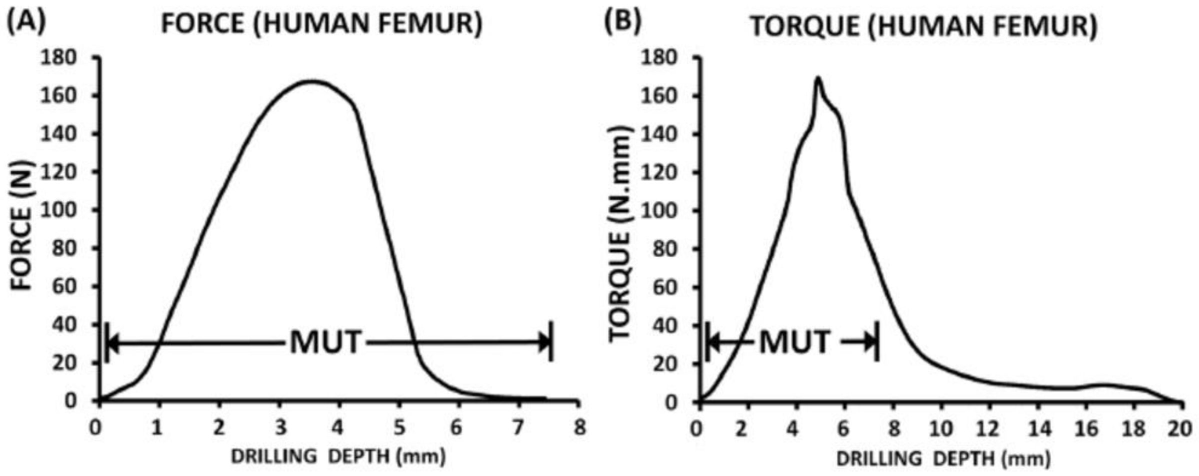


Figure 25: Force and torque data of femur bone drilling [5] (MUT=drill bit tip inside the bone)

In the figure above, the force and torque data collected while bone drilling is shown. As it can be seen from the figure, both drilling force and torque of the drill depends on the drilling depth. In addition, when the drill bit tip gets out of the bone, there are significant changes on both drilling force and torque. Thus, there are two main approaches to accomplish our objective which is stopping the drill when it is out of the bone. First, monitoring the drilling force and second monitoring the torque of the drill. We chose the second approach, because it is a much cheaper solution. For monitoring the drilling force, a load sensor is needed which can be very expensive. On the other hand, if we measure the current passing through the drill, we can monitor the torque. To measure the current passing through the drill, we just bought a 10 TL current sensor which can be powered by Arduino.

In the figure below, current data collected with our system is shown. As it can be seen from the figure, the current passing through the spindle motor is around 1A. As it enters the bone, the current starts to increases. At last, the current decreases significantly when it is out of the bone.

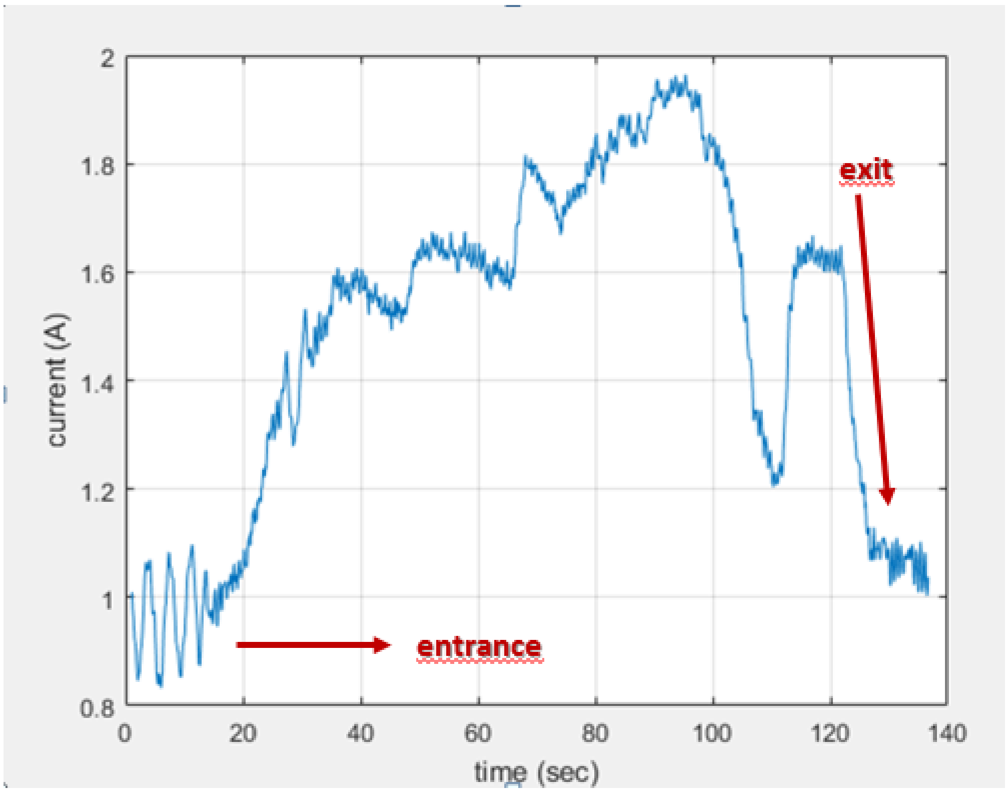


Figure 26: Current data collected by our system

In the code that we wrote, first the stepper motor starts to rotate which makes the drill move downwards and the spindle motor starts to rate to get the reference current value. Based on this reference current value and instantaneous current value, the code defines the stage of the drilling. There are three stages of drilling which consist of the time periods that the drill bit (1) before enters the bone, (2) is in the bone, and (3) is out of the bone. In every loop, the code compares the instantaneous current value with the reference current value and defines the stage. The code is shared in the Appendix.

In the first stage, the code checks if the instantaneous current is significantly higher than the reference current. If it is higher than the reference, the second stage starts. In the second stage, the code checks if the instantaneous current is almost as low as the reference current and if it is, stage three starts. In stage three, the stepper motor starts to rotate in the reverse direction and after the drill bit gets out of the bone completely, the stepper motor and the spindle motor stop.

The feed rate is 0.5mm/s and the speed of the spindle motor is around 6000 rpm.

4. Results and Discussion

All of the parts explained in the Experimental Setup were assembled. The assembly can be seen in the figure below. As the stepper motor rotates in the clockwise direction, the spindle motor goes up and as it rotates in the counter clockwise direction, the spindle motor goes down. Stepper motor is controlled by a micro stepping drive and the spindle motor is controlled by Arduino’s PWM signal.

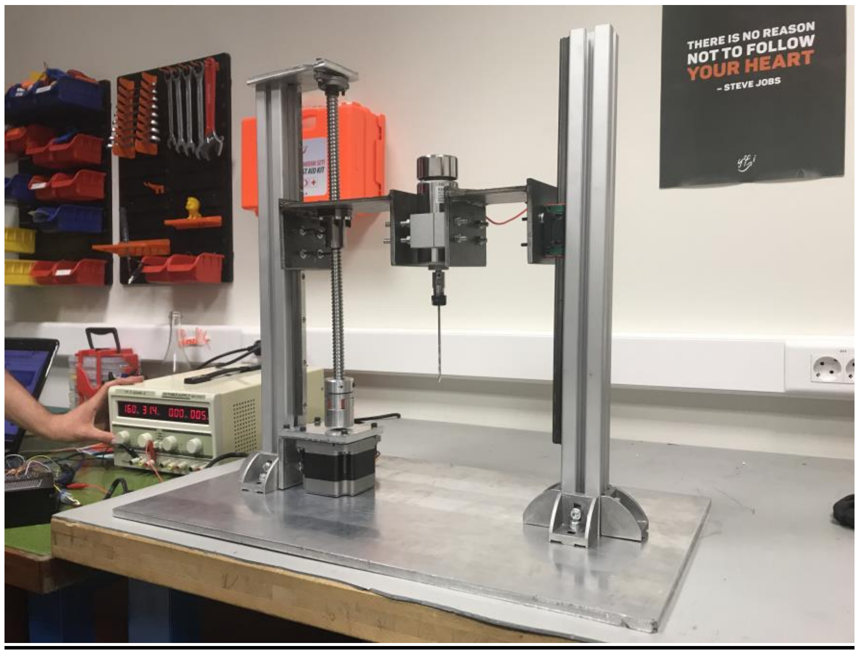


Figure 27: The Assembly

We did a lot of experiments on a calf femur bone. We collected current data and wrote the Arduino code based on the experimental data collected. The current data collected during a bone drilling experiment is shown in the figure below.

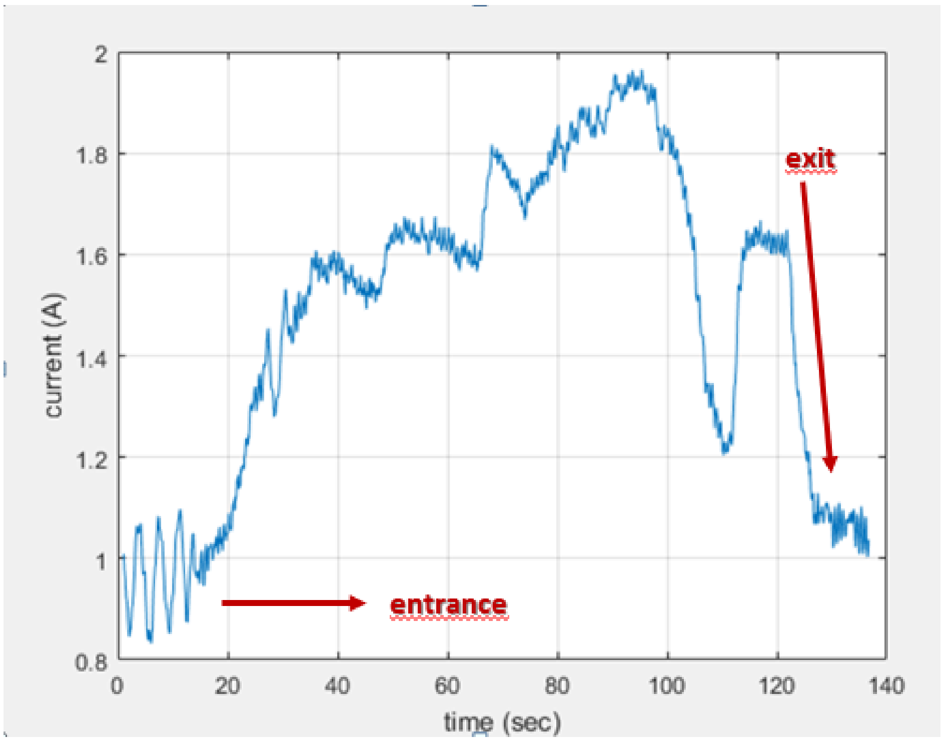


Figure 28: Current data of a bone drilling experiment

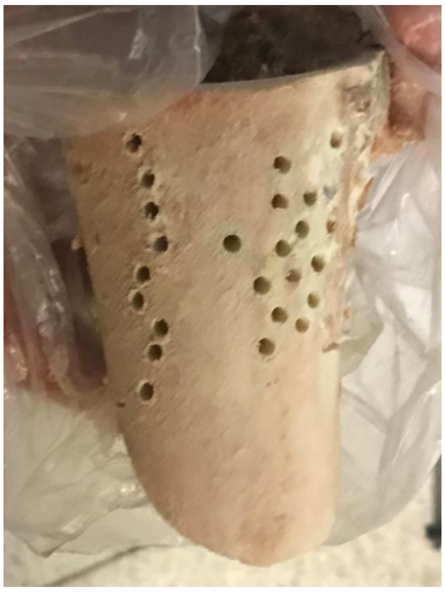


Figure 29: Bone specimen

We successfully wrote the code where the stepper motor rotates to counter clockwise to raise the spindle motor as soon as the drill bit gets out of the bone. Either a layer of the bone or both layers of the bone can be drilled. The videos of the drilling were shared with the report.

The performance of the drill is good; however, it can be improved. The current data can be collected at different feed rates. Then the feed rate that allows the fastest current change at the end of the bone can be selected. In addition, the same experiments can be done with different drilling speeds. The drilling speed can be selected based on the speed of the current response when the drill bit gets out of the bone.

Another improvement can be adding a speed feedback to the spindle motor. To make sure that the drilling speed is in the allowance range, a speed feedback can be very useful. In addition, with the current feedback, torque of the motor can be controlled while drilling the bone.

5. <u>Conclusion:</u>

Damaging the soft tissue, is a crucial problem during drilling the bone in orthopedic surgeries. An automated system, a load sensitive drill is designed and built in this project. Since the current and the motor torque is proportional, one can sense if the drill is in the bone or not via the change in the current of the DC motor. To measure the current a current sensor is used and the motor is terminated when the current is significantly changed. The experiments were done to prove that the drill is working properly. To conclude, there might be some improvements that can be implemented on the mechanism such as: collecting data at different feed rates, adding a speed feedback to the system.

References

[1] Yong, D.X., Jun, Q., Zhenyu, L., Yu, Y.D., Zhang, Y., Zhu, J.S., Wang, X.H., Chong, X.L., Xuesong, H. Intelligent bone drill and control method thereof. CN101530341 (2009).

[2] Fu, H.R., Liu, H.L., Zhang, Y.R. Automated skull drilling mechanism. CN102309354 (2012).

[3] Min Yang, Changhe Li, Benkai Li, Yaogang Wang, Yali Hou, *«Advances and Patents about Medical Surgical Operation Bone Drilling Equipment»,* Recent Patents on Mechanical Engineering, 2015, 8, 99-111.

[4] Louredo M, Díaz I, Gil JJ. DRIBON: A mechatronic bone drilling tool. Mechatronics 2012; 22(8): 1060-66.

[5] Macavelia, T., Salahi, M., Olsen, M., Crookshank, M., Schemitsch, E.H., Ghasempoor, A., Janabi-Sharifi, F., Zdero, R., 2012. Biomechanical measurements of surgical drilling force and torque in human versus artificial femurs. J. Biomech. Eng. 134 (12) (124503-124503)

[6] R.K. Pandey, S.S. Panda, Drilling of bone: a comprehensive review, J. Clin. Orthop. Trauma 4 (2013) 15–30.

Appendix

The code for drilling the one layer of the bone

```
int motorPin = 12; // DC motor
const int sensPin = A4;  // current sensor
int mVperAmp = 66; // sensitivity
int RawValue= 0;
int ACSoffset = 2500;
double Voltage = 0;
double Amps = 0;
int direct = 52; // direction pin of the stepper motor
int pulse = 53; // pulse pin of the stepper motor
unsigned long time;
unsigned long timeprev;
int boo = 0; // for taking the current value in 0.1 seconds
int boo2 = 0; // 1 when the dc motor is drillng the bone
int start = 1;
double ref = 0;  //reference current
int counter = 0;
int counter2 = 0;
void setup() {
  // put your setup code here, to run once:
  pinMode(motorPin,OUTPUT);
  pinMode(sensPin,INPUT);
  pinMode(direct,OUTPUT);
  pinMode(pulse,OUTPUT);
  digitalWrite(direct,HIGH); // HIGH down, LOW up
  digitalWrite(pulse,LOW);
  Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
  if (counter > 5000) {
    analogWrite(motorPin,0);
  }
  else {
    if (start == 1) {
      analogWrite(motorPin,255);
      delay(5000);

      digitalWrite(pulse,HIGH);
      delayMicroseconds(500000);
      digitalWrite(pulse,LOW);
      delayMicroseconds(500000);
      time = millis();
      if (time>10000) {
        start =0;
        int i =0;
        while (i < 10){
          digitalWrite(pulse,HIGH);
          delayMicroseconds(500000);
          digitalWrite(pulse,LOW);
          delayMicroseconds(500000);
          timeprev=time;
          time = millis();
          time = time % 10;
          if (time < timeprev) {
            RawValue = RawValue+analogRead(sensPin)/10;
            i++;
          }
        }
        Voltage = (RawValue / 1024.0) * 5000; // Gets you mV
        ref = ((Voltage - ACSoffset) / mVperAmp);
        RawValue=0;
        Serial.print("Reference current is ");
        Serial.print(ref);
        Serial.println("A");
      }
    }
    else {
      if (Amps > ref-0.25 & boo2 == 1)  {
        analogWrite(motorPin,124);
        digitalWrite(direct,LOW);
        digitalWrite(pulse,HIGH);
        delayMicroseconds(200000);
        digitalWrite(pulse,LOW);
        delayMicroseconds(200000);
        counter++;
      }
      else {
        digitalWrite(pulse,HIGH);
        delayMicroseconds(500000);
        digitalWrite(pulse,LOW);
        delayMicroseconds(500000);
        int i = 0;
        while (i < 10){
          digitalWrite(pulse,HIGH);
          delayMicroseconds(500000);
          digitalWrite(pulse,LOW);
          delayMicroseconds(500000);
          timeprev=time;
          time = millis();
          time = time % 10;
          if (time < timeprev) {
            RawValue = RawValue+analogRead(sensPin)/10;
            i++;
          }
        }
        Voltage = (RawValue / 1024.0) * 5000; // Gets you mV
        Amps = ((Voltage - ACSoffset) / mVperAmp);
        RawValue=0;
        if (Amps<ref-0.3){
          counter2++;
        }
        if (counter2>4){
          boo2=1;
        }
        Serial.println(Amps);

      }
    }
    timeprev = time;
  }
}
```

The code for drilling the both layers

```
int motorPin = 12; // DC motor
const int sensPin = A4;  // current sensor
int mVperAmp = 66; // sensitivity
int RawValue= 0;
int ACSoffset = 2500;
double Voltage = 0;
double Amps = 0;
int direct = 52; // direction pin of the stepper motor
int pulse = 53; // pulse pin of the stepper motor
unsigned long time;
unsigned long timeprev;
int boo = 0; // for taking the current value in 0.1 seconds
int boo2 = 0; // 1 when the dc motor is drillng the bone
int start = 1;
double ref = 0;  //reference current
int counter = 0;
int counter2 = 0;
int counter3 = 0;
void setup() {
  // put your setup code here, to run once:
  pinMode(motorPin,OUTPUT);
  pinMode(sensPin,INPUT);
  pinMode(direct,OUTPUT);
  pinMode(pulse,OUTPUT);
  digitalWrite(direct,HIGH); // HIGH down, LOW up
  digitalWrite(pulse,LOW);
  Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
  if (counter > 5000) {
    analogWrite(motorPin,0);
  }
  else {
    if (start == 1) {
      analogWrite(motorPin,255);
      digitalWrite(pulse,HIGH);
      delayMicroseconds(500000);
      digitalWrite(pulse,LOW);
      delayMicroseconds(500000);
      time = millis();
      if (time>10000) {
        start =0;
        int i =0;
        while (i < 5){
          digitalWrite(pulse,HIGH);
          delayMicroseconds(500000);
          digitalWrite(pulse,LOW);
          delayMicroseconds(500000);
          timeprev=time;
          time = millis();
          time = time % 20;
          if (time < timeprev) {
```

```
        RawValue = RawValue+analogRead(sensPin)/5;
        i++;
      }
    }
    Voltage = (RawValue / 1024.0) * 5000; // Gets you mV
    ref = ((Voltage - ACSoffset) / mVperAmp);
    RawValue=0;
    Serial.print("Reference current is ");
    Serial.print(ref);
    Serial.println("A");
  }
}
else {
  if (Amps > ref-0.15 & boo2 == 3)  {
    analogWrite(motorPin,124);
    digitalWrite(direct,LOW);
    digitalWrite(pulse,HIGH);
    delayMicroseconds(200000);
    digitalWrite(pulse,LOW);
    delayMicroseconds(200000);
    counter++;
  }
  else if (boo2 == 2)  {
    digitalWrite(pulse,HIGH);
    delayMicroseconds(500000);
    digitalWrite(pulse,LOW);
    delayMicroseconds(500000);
    int i = 0;

    while (i < 5){
      digitalWrite(pulse,HIGH);
      delayMicroseconds(500000);
      digitalWrite(pulse,LOW);
      delayMicroseconds(500000);
      timeprev=time;
      time = millis();
      time = time % 20;
      if (time < timeprev) {
        RawValue = RawValue+analogRead(sensPin)/5;
        i++;
      }
    }
    Voltage = (RawValue / 1024.0) * 5000; // Gets you mV
    Amps = ((Voltage - ACSoffset) / mVperAmp);
    RawValue=0;
    if (Amps<ref-0.4){
      counter3++;
    }
    if (counter3>10){
      boo2=3;
```

```
        }
        Serial.println(Amps);

      }
      else if (Amps > ref-0.2 & boo2 == 1)  {
        boo2 = 2;
      }
      else {

        digitalWrite(pulse,HIGH);
        delayMicroseconds(500000);
        digitalWrite(pulse,LOW);
        delayMicroseconds(500000);

        int i = 0;
        while (i < 5){
          digitalWrite(pulse,HIGH);
          delayMicroseconds(500000);
          digitalWrite(pulse,LOW);
          delayMicroseconds(500000);
          timeprev=time;
          time = millis();
          time = time % 20;
          if (time < timeprev) {
            RawValue = RawValue+analogRead(sensPin)/5;
            i++;
          }
        }
        Voltage = (RawValue / 1024.0) * 5000; // Gets you mV
        Amps = ((Voltage - ACSoffset) / mVperAmp);
        RawValue=0;
        if (Amps<ref-0.3){
          counter2++;
        }
        if (counter2>4){
          boo2=1;
        }
        Serial.println(Amps);

      }
    }
    timeprev = time;
  }
}
```