“InMoov” (3D printed) Humanoid-Robot Arm Control

ELG 4913: ELECTRICAL ENGINEER PROJECT 2

Winter 2020

Instructor: Professor Emil Petriu

Design team member:

Yanran Wang (7782241)

Baohui Xie (7685551)

Ren Guan (7867203)

Submission Date: 2020-04-06

Table of contents

Introduction:

In recent years, robotic arm has been widely used in industries system control, medical device control and entertainment or home service robot. Therefore, robotic arm control has become a significant research focus in electrical engineering area. As a scientific focal point, it is meaningful to the performance in its accuracy, stability and feasibility. This project is a research and experiment, which are focusing on designing a robotic arm control system and striving to develop a high stability and reliability real robotic arm, based on joint coordinate system and inMoov open sourced skills.

After we finished our project's theoretical design, we now move to specifically focus on the arm physically construction and control among the overall inMoov project. Coding, and 3D printing process; the assembly process will be included in this semester's workload.

At the end of the project, we are expecting to build up an inMoov robot left arm which is constituted by mostly plastic components. By using arduino board and the corresponding programming code, the user will be able to fully control the arm with the help of software on another platform through arduino applications.

Review the design of the arm

Proportional, Integral, Derivative control (PID control)

PID control is consist of proportional, integral and derivative. By adding these influences on the original signal, the system can perform an accurate and optimal control to the arm. First, the figure below illustrates the general idea of PID control of the arm:

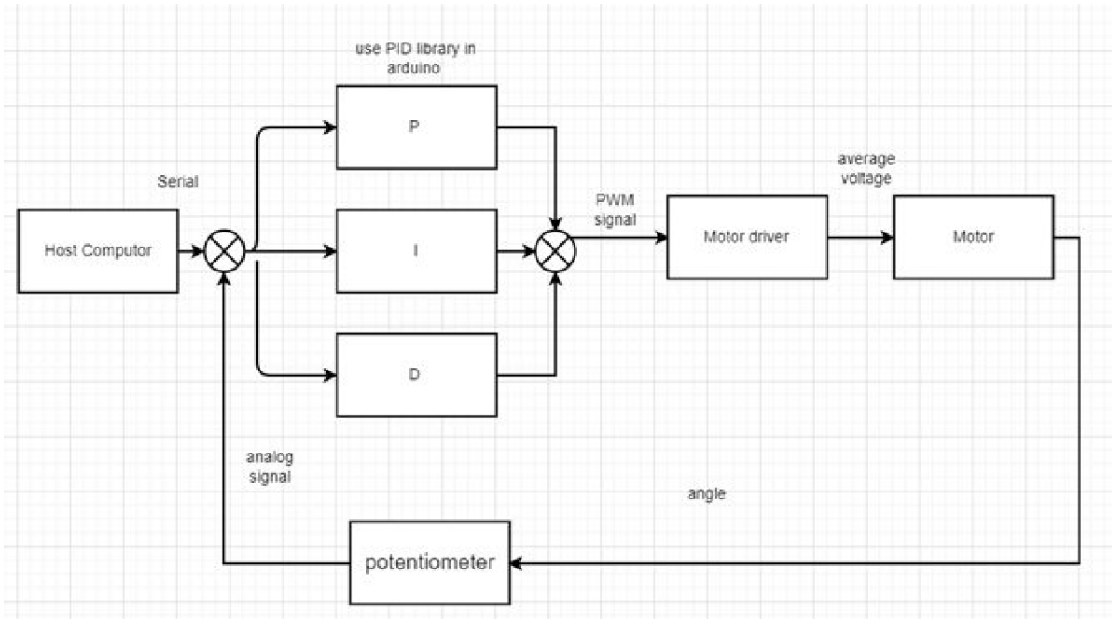


Figure 1: PID control of the arm

On the block diagram, we can see that the potentiometer is continuously sending the feedback signal to the Arduino UNO. And then, this feedback signal will merge with the value that the host sent to the Arduino before the real calculation of proportional, integral and derivative. Here we denote the signal after the merging as e(t). After the calculation of P, I and D with e(t), these three values will be summed together in order to get the control variable pulse-wide-modulation signal. Here we denote the sum of P, I and D as u(t), where

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(\tau)\, d\tau + K_{\mathrm{d}} \frac{de(t)}{dt}$$

The terms of Kp, Ki and Kd are all non-negative value, and they are the coefficients for the proportional, integral and derivative respectively.

The control variable pulse-width-modulation signal will apply to the motor drive as the input. And, the motor will rotate based on that input information, especially for the speeding. But motor part will be explant later on after this subsection. Before that we denote the output signal of whole system as y(t) and input signal of whole system as x(t). As we mentioned before, the signal before the P, I and D calculation is called e(t). Here now we can define the e(t) as: e(t)=x(t)-y(t) Now we have defined all the information that we may require in our code. But the problem is the PID algorithm cannot guarantee the stability of the arm. Therefore, in order to stabilize the arm, we need to apply the Routh-Hurwitz Criterion, we have to convert the u(t) in to transfer function first. That will give us:

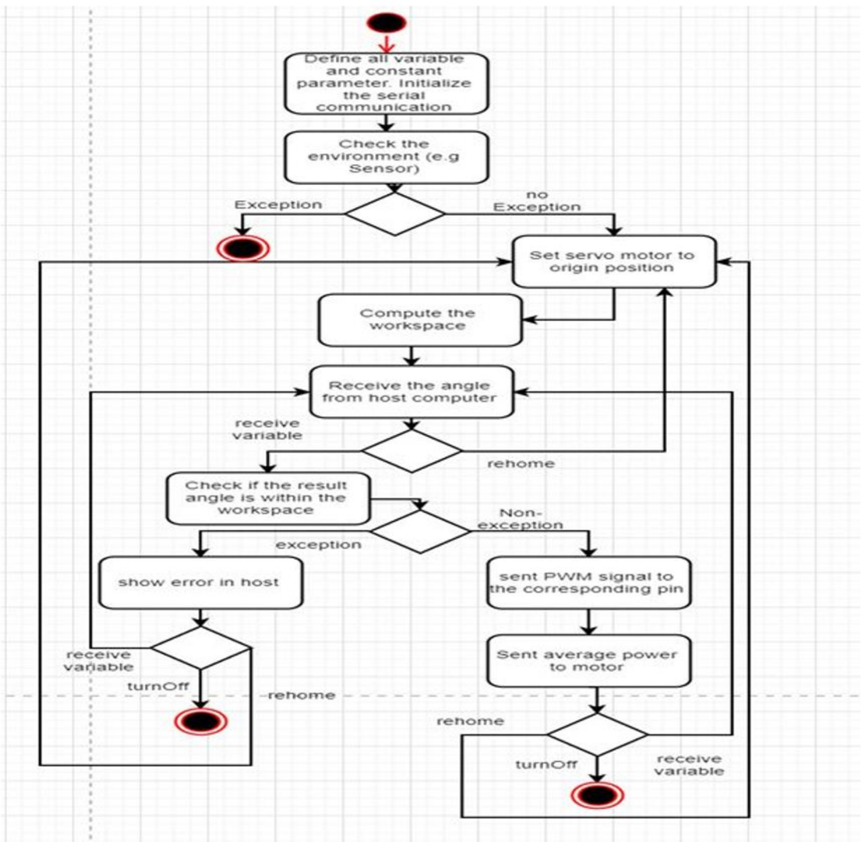
$$U(s)_{\mathrm{cmd}} = K_{\mathrm{P}}E(s) + K_{\mathrm{I}}\frac{E(s)}{s} + K_{\mathrm{D}}sE(s),$$

After the criterion, we will have a new transfer function. However, since we do not have the real data of the sensor yet, we cannot simulate by using this method.

Software diagram

Obviously, we can divide our InMoov Robotic Arm's behavior into two parts which are invisible and visible behavior respectively, and these two type behaviors can also be described by activity diagram and state diagram. The activity diagram represents the flow work that an object or component perform, and state diagram is used to present the externally visible arms operates of a system.

Activity Diagram:



Define all variable and constant parameter. Initialize the serial communication
Check the environment (e.g Sensor)
Exception
no Exception
Set servo motor to origin position
Compute the workspace
Receive the angle from host computer
receive variable
rehome
Check if the result angle is within the workspace
Non-exception
exception
show error in host
sent PWM signal to the corresponding pin
Sent average power to motor
receive variable
turnOff
rehome
rehome
turnOff
receive variable

Figure 2: Activity Diagram

As we mentioned before, we could know that the Arduino UNO board was controlled by serial input which was given by the host computer. Therefore, if we start to control our robotic arm, the first order we will give to the Arduino is defining all variables and constant parameters that we enter on the host and then it will initialize the serial communication between the host computer and the control board. In addition, control board also needs to check the environment variables from potentiometer to ensure there is no exception. If any exception is found, the system will automatically turn off, which means we should replace the current sensor by a new one. Secondly, if there is no exception found, the next step is going to set servo motor to origin position, which means the position of each arm's joint will be rehomed. Then build-in monitor will compute the workspace we enter is matched or not to the robotic arm's own workspace. For example, if the maximum rotation angle of joint 1 is 90 degrees but we enter 92 degrees, then the system will show error in host because the result angle is out of arm's original workspace. Therefore, if we want to make the robotic arm to reach the angle we enter, sometimes we will need to modify our robotic arm. After checking the result of angle with no exception, if the result is within the workspace, control board will send PWM signal through digital pin to the corresponding pin on H-bridge. After that, the H-bridge will control the motors to finish the position or angle command that we enter in the host. Finally, we can decide to turn off the system or operate other commands or just simply rehome our arm to the initial position, it is very flexible.

State Diagram:

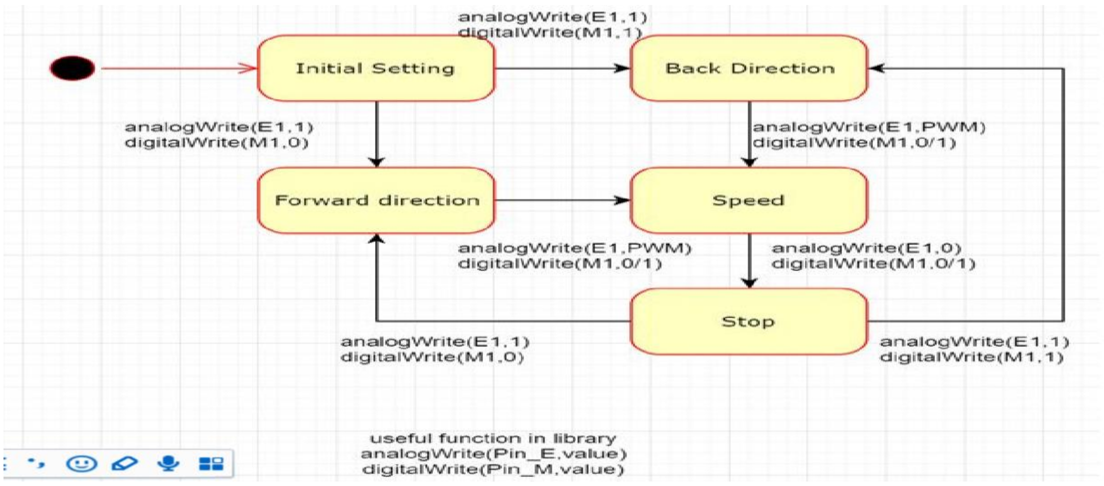


Figure 3: State Diagram

The above state diagram is an example for shoulder joint motor, which illustrates the movement steps of the robotic arm from initial state to final state. Firstly, if we want our robotic arm to do some actions in positive direction (forward direction), the Arduino control board will send a high analog signal and a low digital signal to H-bridge, then the H-bridge will control the motors to rotate in forward direction. After that, we can enter and run the special speed code in order to speed up our arm's action. For example, if we want to speed up our shoulder to reach its destination, we should give two commands to let control board give a PWM analog signal and a random (it means whatever motor signal pin receives high or low signal, it will not affect the reality action that the shoulder joint will do.) digital signal to H-bridge in order to acquire the result we want. Then our robotic shoulder will move in forward direction faster and faster until its joint reach the angle and position we entered on the host before. Finally, we can continue to give the new orders to our shoulder while it is stopped, such as forward or backward movement, but it will repeat the same process as we described above.

Work illustration

3D Printing:

As we know, InMoov robotic arm can be roughly divided into five parts shoulder, torso, bicep, forearm and hand. in accordance with the Assembly also carried out a block work. Firstly, we need to use a 3D printer to make one of the components needed for each part.

The working principle of 3D printer is heating nozzle diameter of about 2mm filamentous thermoplastic material (such as ABS, etc.) through the nozzle is heated to a certain temperature to Melt, heating nozzle nozzle diameter is generally between 0.2 mm to 0.6 mm, at the same time, the 3D printer slowly pushes the thermoplastic material forward at a rate that allows the molten material to be ejected from the nozzle. When the nozzle temperature reaches the molten thermoplastic material can be stabilized, i.e. in accordance with a stable rate of discharge of thermoplastic, the nozzle in accordance with the object to be printed in this layer of the slice model starts to start moving in the XY axis direction. When the layer after the printing is completed, the horizontal table in the Z-axis direction down a certain height, the nozzle continues to move in the horizontal direction, to complete the next layer of printing, and so on, and finally complete the printing of the entire object.

We will get the STL (Standard Template Library) model file for the part before printing it. The next step is to use the Cura software to slice the STL model, plan the route of the printer, and convert the STL model file into a file that the printer can recognize. The Cura software interface and slicing process, as shown in figure 1.4, requires an additional printing cradle when printing parts, and its purpose is to facilitate the printing of parts from the work platform up and down. According to the work of the original 3D printer route planning, progressive, and ultimately the entire print route planning STL model is completed to complete the slicing process. The sliced STL model file into the SD card, 3D printer will be able to work according to the SD card in the slice file, first print the base, and then according to the slice file in the line, layer by layer printing, and finally piled up the STL model corresponding parts.

Making the new wires for connections:

1.Due to taking out the potentiometer from DC motor, it mean we need to make some new connections between DC motor and Arduino. The reason that we took out the potentiometer from servo was we need to place potentiometer at proper joints as a sensor.

2.We also need to make some new wires for connecting the potentiometer and Arduino.
3.In making the wires, we used liquid tin as the joint between wire and potentiometer's ports.
4.As we mentioned above, we would use welding gun to melt down the tins and solder it with ports and wires.

Final prototype

Hardware Component

Following by the schedule made in the last semester, all the 3D printed components have been printed successfully and all of them have been fully assembled. The following diagram is the result of assembling:

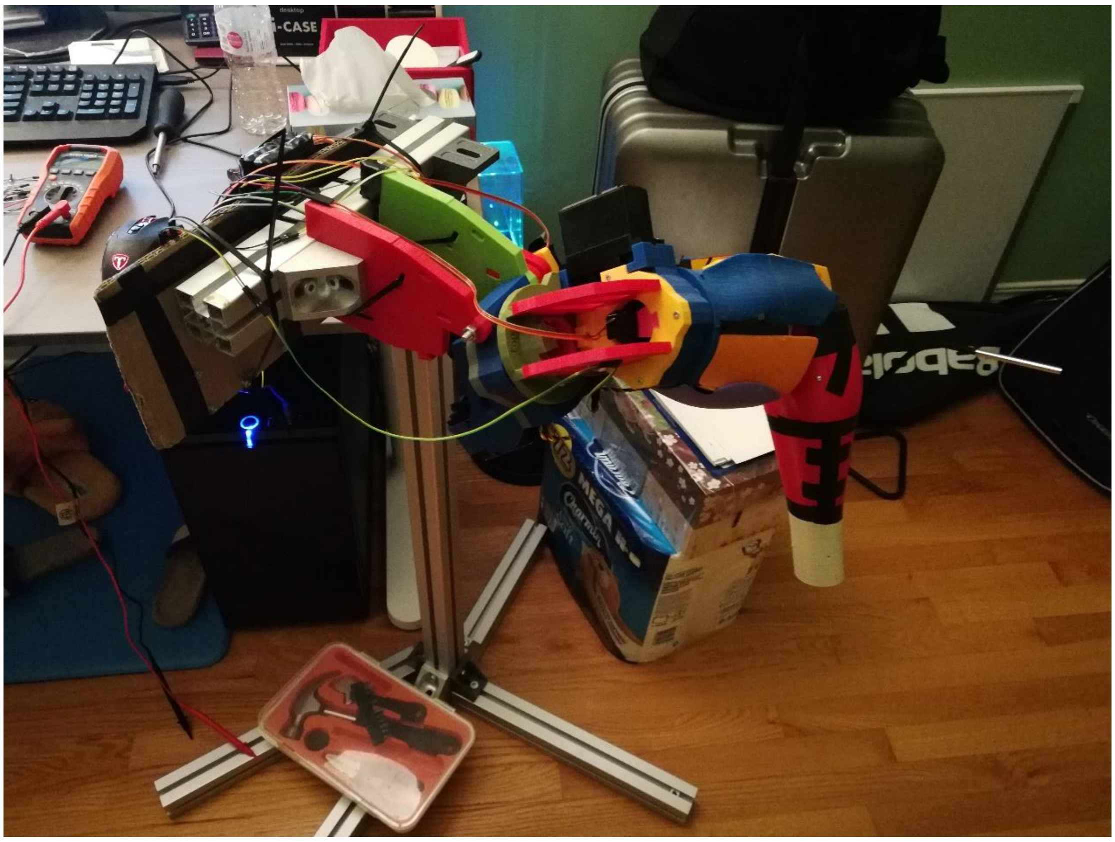


Figure 4: Final prototype

Control panel

The Arduino uno pad is chosen to be the core controller of the InMoov robot arm. We have already bought it and tested its usability. We can successfully send signal to the motor and control the switches. The digital pins on the Arduino pad will be used as the main connection media. One thing to be notice here is that we can not use the port “0” and “1” as to connect with the motor drive, since they are always be logic high if there is a communication between computer and Arduino. Also, it also should be notice that always connect the ground of the motor drive and the ground of Arduino.

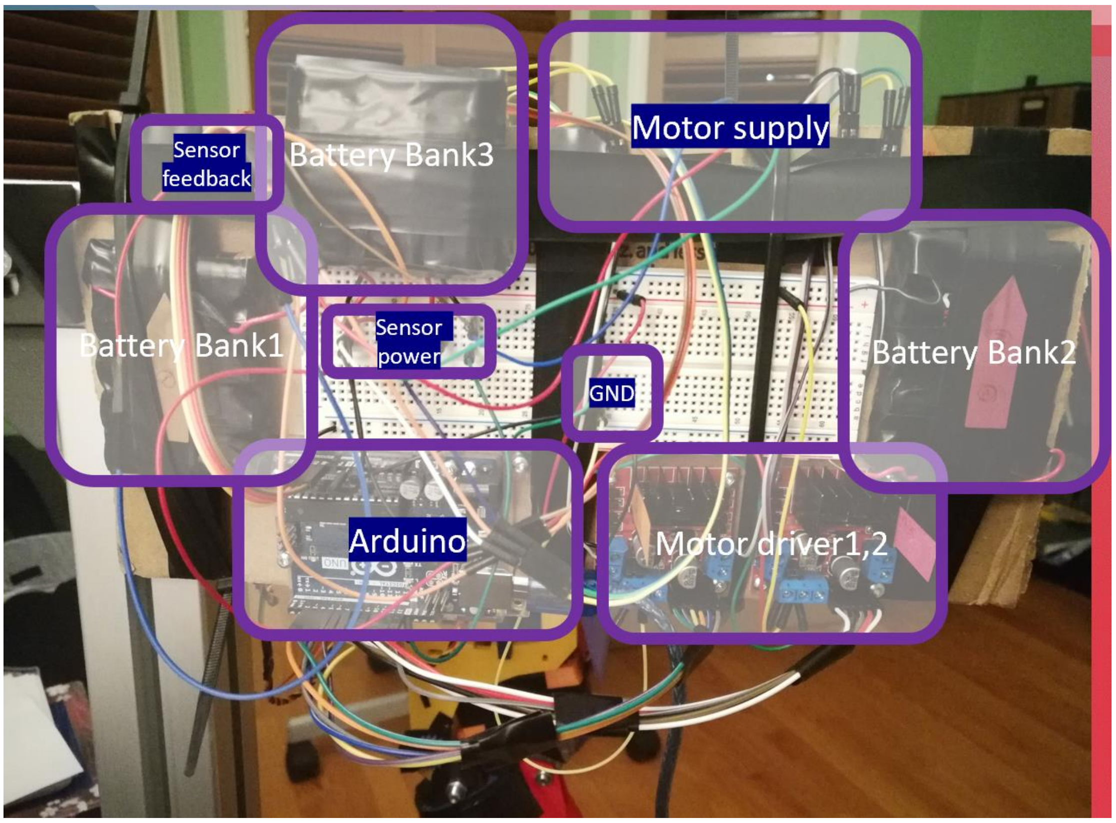
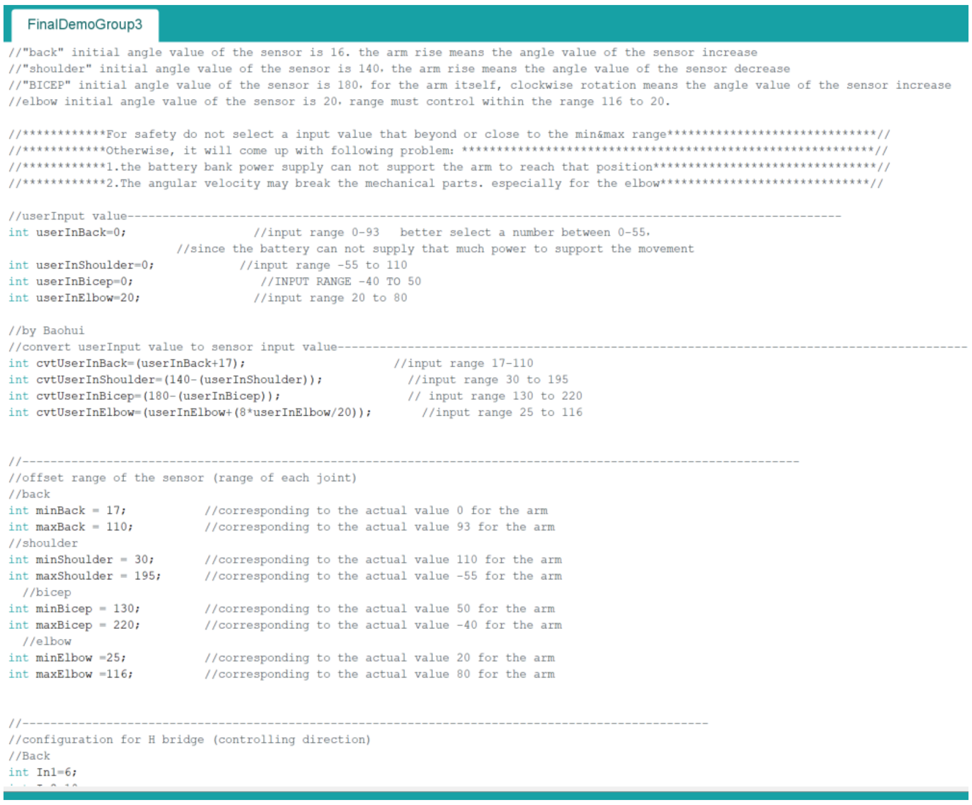
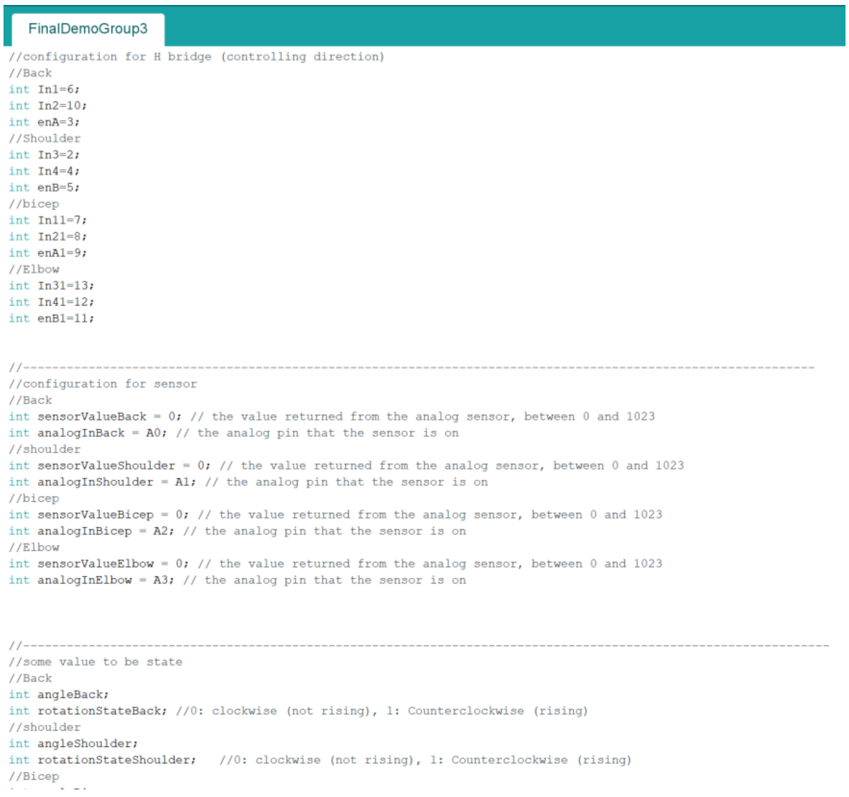


Figure 5: Control panel

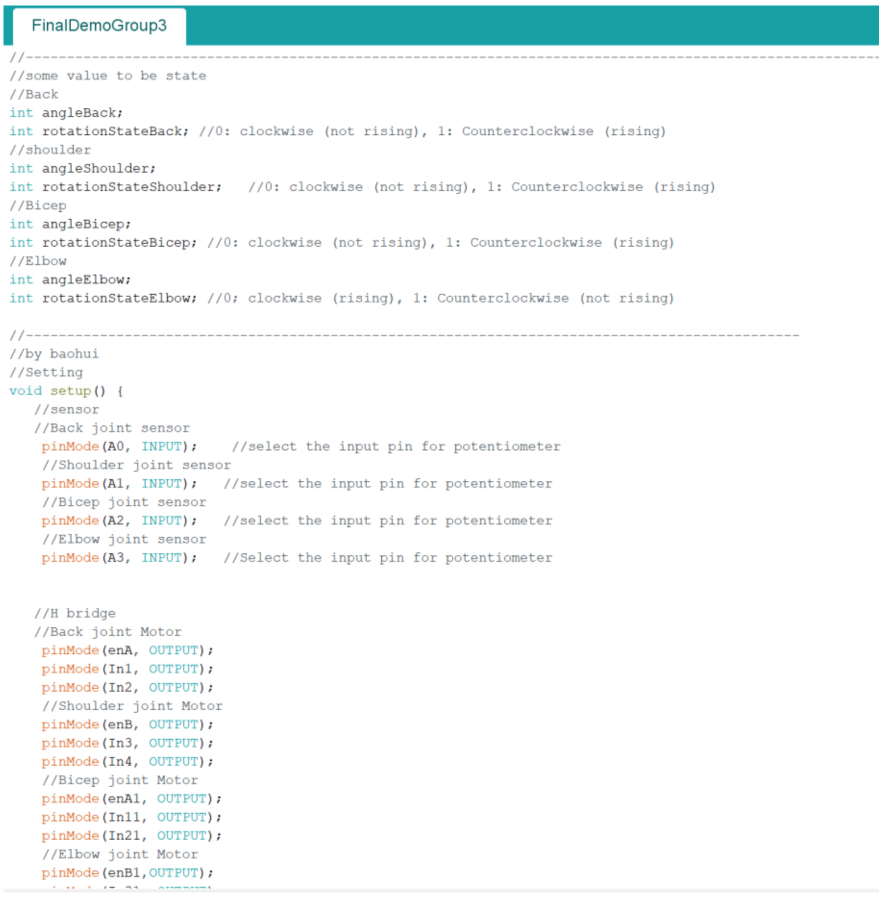
Arduino code



```
//"back" initial angle value of the sensor is 16. the arm rise means the angle value of the sensor increase
//"shoulder" initial angle value of the sensor is 140, the arm rise means the angle value of the sensor decrease
//"BICEP" initial angle value of the sensor is 180, for the arm itself, clockwise rotation means the angle value of the sensor increase
//elbow initial angle value of the sensor is 20, range must control within the range 116 to 20.

//************For safety do not select a input value that beyond or close to the min&max range******************************//
//************Otherwise, it will come up with following problem: ************************************************************//
//************1.the battery bank power supply can not support the arm to reach that position*********************************//
//************2.The angular velocity may break the mechanical parts. especially for the elbow*******************************//

//userInput value-----------------------------------------------------------------------------------------------------------
int userInBack=0;                  //input range 0-93   better select a number between 0-55,
                        //since the battery can not supply that much power to support the movement
int userInShoulder=0;             //input range -55 to 110
int userInBicep=0;                   //INPUT RANGE -40 TO 50
int userInElbow=20;                //input range 20 to 80

//by Baohui
//convert userInput value to sensor input value-----------------------------------------------------------------------------------------
int cvtUserInBack=(userInBack+17);                    //input range 17-110
int cvtUserInShoulder=(140-(userInShoulder));           //input range 30 to 195
int cvtUserInBicep=(180-(userInBicep));                 // input range 130 to 220
int cvtUserInElbow=(userInElbow+(8*userInElbow/20));       //input range 25 to 116


//------------------------------------------------------------------------------------------------------------------
//offset range of the sensor (range of each joint)
//back
int minBack = 17;           //corresponding to the actual value 0 for the arm
int maxBack = 110;          //corresponding to the actual value 93 for the arm
//shoulder
int minShoulder = 30;       //corresponding to the actual value 110 for the arm
int maxShoulder = 195;      //corresponding to the actual value -55 for the arm
  //bicep
int minBicep = 130;         //corresponding to the actual value 50 for the arm
int maxBicep = 220;         //corresponding to the actual value -40 for the arm
  //elbow
int minElbow =25;           //corresponding to the actual value 20 for the arm
int maxElbow =116;          //corresponding to the actual value 80 for the arm


//-----------------------------------------------------------------------------------------------------
//configuration for H bridge (controlling direction)
//Back
int In1=6;
```

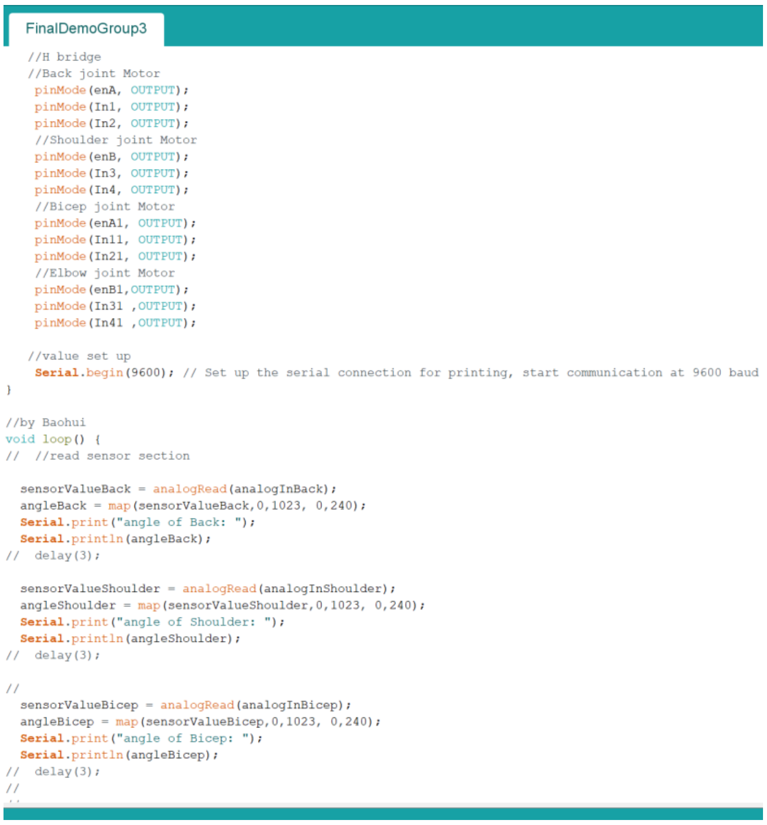


```
//configuration for H bridge (controlling direction)
//Back
int In1=6;
int In2=10;
int enA=3;
//Shoulder
int In3=2;
int In4=4;
int enB=5;
//bicep
int In11=7;
int In21=8;
int enA1=9;
//Elbow
int In31=13;
int In41=12;
int enB1=11;


//--------------------------------------------------------------------------------------------------------------------
//configuration for sensor
//Back
int sensorValueBack = 0; // the value returned from the analog sensor, between 0 and 1023
int analogInBack = A0; // the analog pin that the sensor is on
//shoulder
int sensorValueShoulder = 0; // the value returned from the analog sensor, between 0 and 1023
int analogInShoulder = A1; // the analog pin that the sensor is on
//bicep
int sensorValueBicep = 0; // the value returned from the analog sensor, between 0 and 1023
int analogInBicep = A2; // the analog pin that the sensor is on
//Elbow
int sensorValueElbow = 0; // the value returned from the analog sensor, between 0 and 1023
int analogInElbow = A3; // the analog pin that the sensor is on



//----------------------------------------------------------------------------------------------------------------------
//some value to be state
//Back
int angleBack;
int rotationStateBack; //0: clockwise (not rising), 1: Counterclockwise (rising)
//shoulder
int angleShoulder;
int rotationStateShoulder;   //0: clockwise (not rising), 1: Counterclockwise (rising)
//Bicep
```



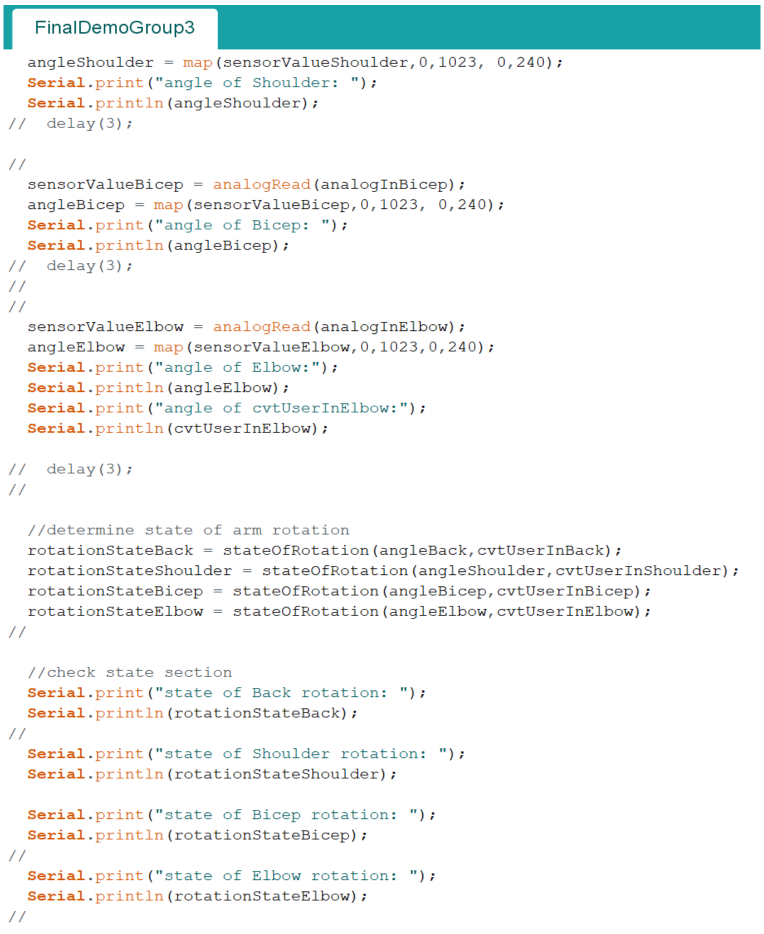
FinalDemoGroup3

```
//-----------------------------------------------------------------------------------------------------------
//some value to be state
//Back
int angleBack;
int rotationStateBack; //0: clockwise (not rising), 1: Counterclockwise (rising)
//shoulder
int angleShoulder;
int rotationStateShoulder;   //0: clockwise (not rising), 1: Counterclockwise (rising)
//Bicep
int angleBicep;
int rotationStateBicep; //0: clockwise (not rising), 1: Counterclockwise (rising)
//Elbow
int angleElbow;
int rotationStateElbow; //0; clockwise (rising), 1: Counterclockwise (not rising)

//-------------------------------------------------------------------------------------------------
//by baohui
//Setting
void setup() {
   //sensor
   //Back joint sensor
    pinMode(A0, INPUT);     //select the input pin for potentiometer
    //Shoulder joint sensor
    pinMode(A1, INPUT);    //select the input pin for potentiometer
    //Bicep joint sensor
    pinMode(A2, INPUT);    //select the input pin for potentiometer
    //Elbow joint sensor
    pinMode(A3, INPUT);    //Select the input pin for potentiometer


   //H bridge
   //Back joint Motor
    pinMode(enA, OUTPUT);
    pinMode(In1, OUTPUT);
    pinMode(In2, OUTPUT);
    //Shoulder joint Motor
    pinMode(enB, OUTPUT);
    pinMode(In3, OUTPUT);
    pinMode(In4, OUTPUT);
    //Bicep joint Motor
    pinMode(enA1, OUTPUT);
    pinMode(In11, OUTPUT);
    pinMode(In21, OUTPUT);
    //Elbow joint Motor
    pinMode(enB1,OUTPUT);
```

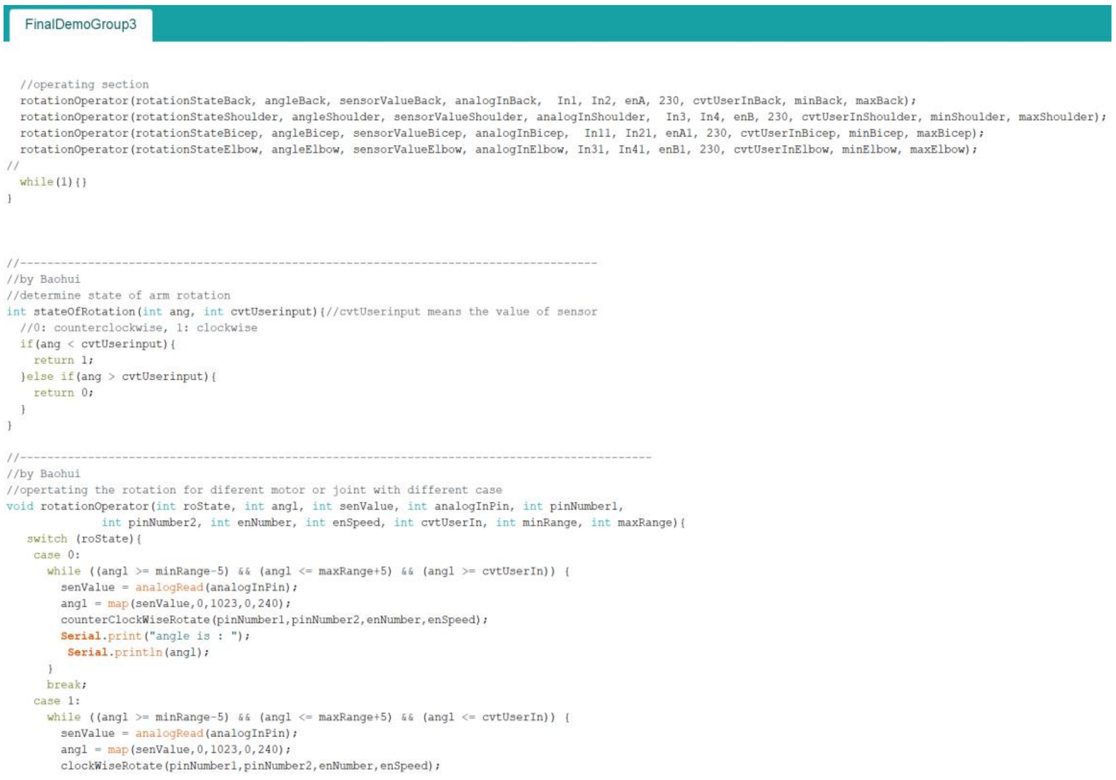


FinalDemoGroup3

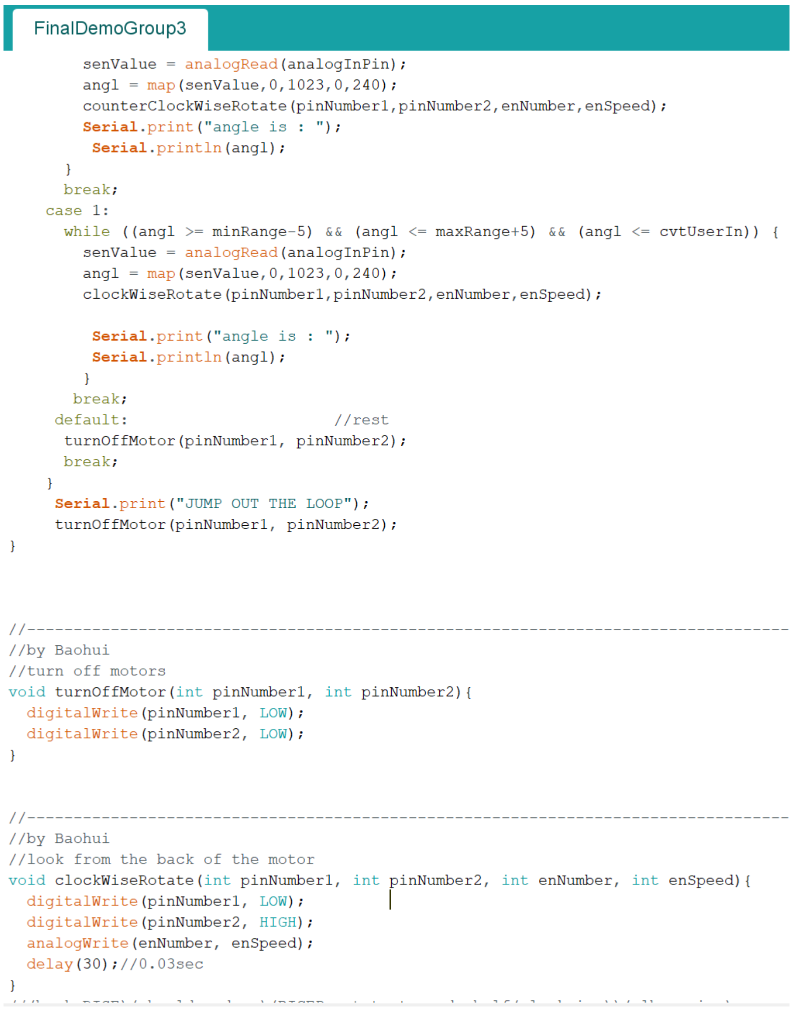
```
   //H bridge
   //Back joint Motor
    pinMode(enA, OUTPUT);
    pinMode(In1, OUTPUT);
    pinMode(In2, OUTPUT);
    //Shoulder joint Motor
    pinMode(enB, OUTPUT);
    pinMode(In3, OUTPUT);
    pinMode(In4, OUTPUT);
    //Bicep joint Motor
    pinMode(enA1, OUTPUT);
    pinMode(In11, OUTPUT);
    pinMode(In21, OUTPUT);
    //Elbow joint Motor
    pinMode(enB1,OUTPUT);
    pinMode(In31 ,OUTPUT);
    pinMode(In41 ,OUTPUT);

   //value set up
    Serial.begin(9600); // Set up the serial connection for printing, start communication at 9600 baud
}

//by Baohui
void loop() {
//  //read sensor section

  sensorValueBack = analogRead(analogInBack);
  angleBack = map(sensorValueBack,0,1023, 0,240);
  Serial.print("angle of Back: ");
  Serial.println(angleBack);
//  delay(3);

  sensorValueShoulder = analogRead(analogInShoulder);
  angleShoulder = map(sensorValueShoulder,0,1023, 0,240);
  Serial.print("angle of Shoulder: ");
  Serial.println(angleShoulder);
//  delay(3);

//
  sensorValueBicep = analogRead(analogInBicep);
  angleBicep = map(sensorValueBicep,0,1023, 0,240);
  Serial.print("angle of Bicep: ");
  Serial.println(angleBicep);
//  delay(3);
//
```



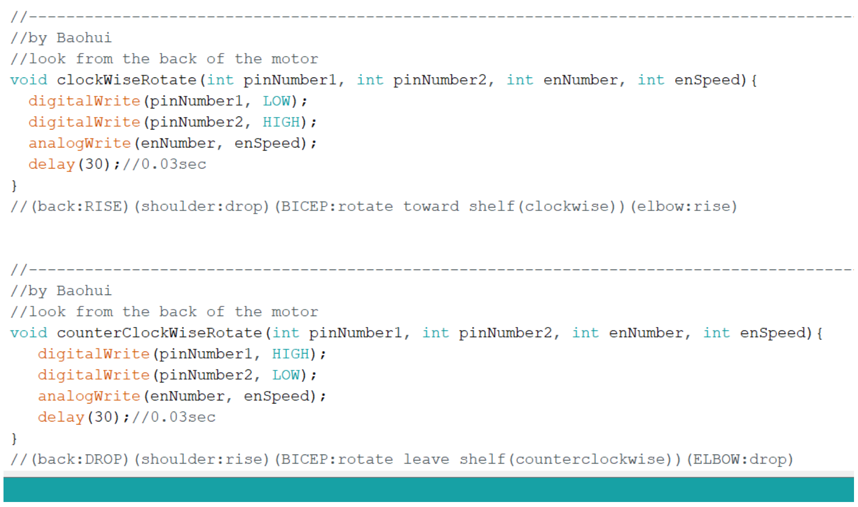
```
  angleShoulder = map(sensorValueShoulder,0,1023, 0,240);
  Serial.print("angle of Shoulder: ");
  Serial.println(angleShoulder);
//  delay(3);

//
  sensorValueBicep = analogRead(analogInBicep);
  angleBicep = map(sensorValueBicep,0,1023, 0,240);
  Serial.print("angle of Bicep: ");
  Serial.println(angleBicep);
//  delay(3);
//
//
  sensorValueElbow = analogRead(analogInElbow);
  angleElbow = map(sensorValueElbow,0,1023,0,240);
  Serial.print("angle of Elbow:");
  Serial.println(angleElbow);
  Serial.print("angle of cvtUserInElbow:");
  Serial.println(cvtUserInElbow);

//  delay(3);
//

  //determine state of arm rotation
  rotationStateBack = stateOfRotation(angleBack,cvtUserInBack);
  rotationStateShoulder = stateOfRotation(angleShoulder,cvtUserInShoulder);
  rotationStateBicep = stateOfRotation(angleBicep,cvtUserInBicep);
  rotationStateElbow = stateOfRotation(angleElbow,cvtUserInElbow);
//

  //check state section
  Serial.print("state of Back rotation: ");
  Serial.println(rotationStateBack);
//
  Serial.print("state of Shoulder rotation: ");
  Serial.println(rotationStateShoulder);

  Serial.print("state of Bicep rotation: ");
  Serial.println(rotationStateBicep);
//
  Serial.print("state of Elbow rotation: ");
  Serial.println(rotationStateElbow);
//
```



```
  //operating section
  rotationOperator(rotationStateBack, angleBack, sensorValueBack, analogInBack,  In1, In2, enA, 230, cvtUserInBack, minBack, maxBack);
  rotationOperator(rotationStateShoulder, angleShoulder, sensorValueShoulder, analogInShoulder,  In3, In4, enB, 230, cvtUserInShoulder, minShoulder, maxShoulder);
  rotationOperator(rotationStateBicep, angleBicep, sensorValueBicep, analogInBicep,  In11, In21, enA1, 230, cvtUserInBicep, minBicep, maxBicep);
  rotationOperator(rotationStateElbow, angleElbow, sensorValueElbow, analogInElbow, In31, In41, enB1, 230, cvtUserInElbow, minElbow, maxElbow);
//
  while(1){}
}



//-------------------------------------------------------------------------------------
//by Baohui
//determine state of arm rotation
int stateOfRotation(int ang, int cvtUserinput){//cvtUserinput means the value of sensor
  //0: counterclockwise, 1: clockwise
  if(ang < cvtUserinput){
    return 1;
  }else if(ang > cvtUserinput){
    return 0;
  }
}

//---------------------------------------------------------------------------------------------
//by Baohui
//opertating the rotation for diferent motor or joint with different case
void rotationOperator(int roState, int angl, int senValue, int analogInPin, int pinNumber1,
              int pinNumber2, int enNumber, int enSpeed, int cvtUserIn, int minRange, int maxRange){
   switch (roState){
    case 0:
      while ((angl >= minRange-5) && (angl <= maxRange+5) && (angl >= cvtUserIn)) {
        senValue = analogRead(analogInPin);
        angl = map(senValue,0,1023,0,240);
        counterClockWiseRotate(pinNumber1,pinNumber2,enNumber,enSpeed);
        Serial.print("angle is : ");
         Serial.println(angl);
      }
      break;
    case 1:
      while ((angl >= minRange-5) && (angl <= maxRange+5) && (angl <= cvtUserIn)) {
        senValue = analogRead(analogInPin);
        angl = map(senValue,0,1023,0,240);
        clockWiseRotate(pinNumber1,pinNumber2,enNumber,enSpeed);
```



```
        senValue = analogRead(analogInPin);
        angl = map(senValue,0,1023,0,240);
        counterClockWiseRotate(pinNumber1,pinNumber2,enNumber,enSpeed);
        Serial.print("angle is : ");
         Serial.println(angl);
      }
      break;
    case 1:
      while ((angl >= minRange-5) && (angl <= maxRange+5) && (angl <= cvtUserIn)) {
        senValue = analogRead(analogInPin);
        angl = map(senValue,0,1023,0,240);
        clockWiseRotate(pinNumber1,pinNumber2,enNumber,enSpeed);

         Serial.print("angle is : ");
         Serial.println(angl);
        }
       break;
     default:                      //rest
      turnOffMotor(pinNumber1, pinNumber2);
      break;
    }
     Serial.print("JUMP OUT THE LOOP");
     turnOffMotor(pinNumber1, pinNumber2);
}



//-------------------------------------------------------------------------------------
//by Baohui
//turn off motors
void turnOffMotor(int pinNumber1, int pinNumber2){
  digitalWrite(pinNumber1, LOW);
  digitalWrite(pinNumber2, LOW);
}


//-------------------------------------------------------------------------------------
//by Baohui
//look from the back of the motor
void clockWiseRotate(int pinNumber1, int pinNumber2, int enNumber, int enSpeed){
  digitalWrite(pinNumber1, LOW);
  digitalWrite(pinNumber2, HIGH);
  analogWrite(enNumber, enSpeed);
  delay(30);//0.03sec
}
```

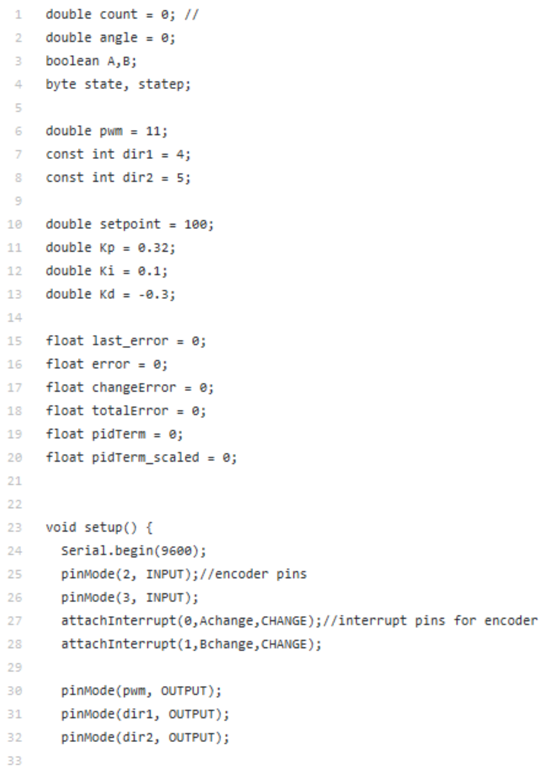


```
//-------------------------------------------------------------------------------------
//by Baohui
//look from the back of the motor
void clockWiseRotate(int pinNumber1, int pinNumber2, int enNumber, int enSpeed){
  digitalWrite(pinNumber1, LOW);
  digitalWrite(pinNumber2, HIGH);
  analogWrite(enNumber, enSpeed);
  delay(30);//0.03sec
}
//(back:RISE)(shoulder:drop)(BICEP:rotate toward shelf(clockwise))(elbow:rise)


//-------------------------------------------------------------------------------------
//by Baohui
//look from the back of the motor
void counterClockWiseRotate(int pinNumber1, int pinNumber2, int enNumber, int enSpeed){
   digitalWrite(pinNumber1, HIGH);
   digitalWrite(pinNumber2, LOW);
   analogWrite(enNumber, enSpeed);
   delay(30);//0.03sec
}
//(back:DROP)(shoulder:rise)(BICEP:rotate leave shelf(counterclockwise))(ELBOW:drop)
```

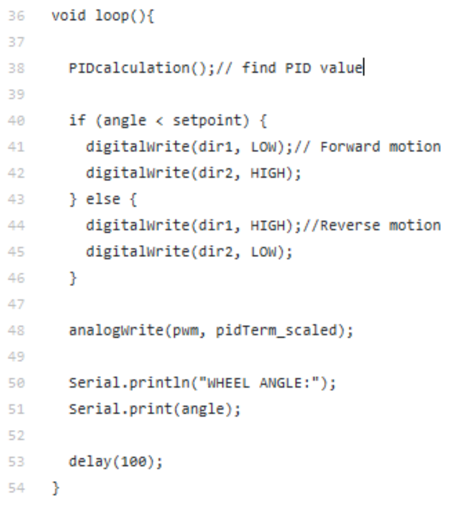
PID Controller Programming

The aim that I would be hoping to achieve was how to program a DC Motor in order to move through any specified angle that we wanted relative to our current position

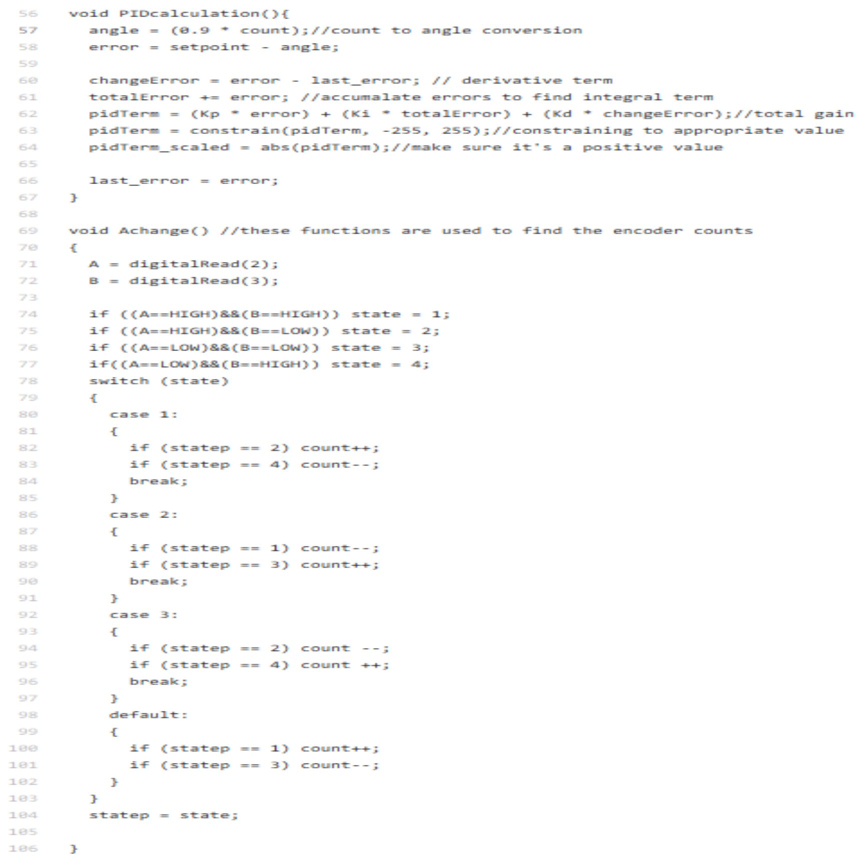
and for it to stay there. I wanted to achieve this transition in the most efficient way as possible, which all depended on how I set the fixed terms in this PID Controller.
The following diagrams are illustrating the codes which would be used in PID controller originally:



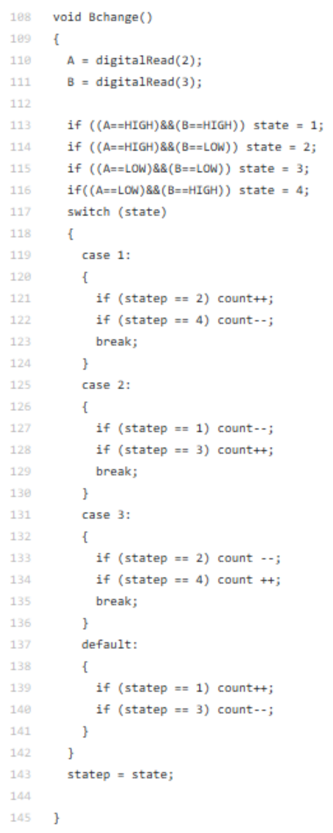
```
double count = 0; //
double angle = 0;
boolean A,B;
byte state, statep;

double pwm = 11;
const int dir1 = 4;
const int dir2 = 5;

double setpoint = 100;
double Kp = 0.32;
double Ki = 0.1;
double Kd = -0.3;

float last_error = 0;
float error = 0;
float changeError = 0;
float totalError = 0;
float pidTerm = 0;
float pidTerm_scaled = 0;


void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT);//encoder pins
  pinMode(3, INPUT);
  attachInterrupt(0,Achange,CHANGE);//interrupt pins for encoder
  attachInterrupt(1,Bchange,CHANGE);

  pinMode(pwm, OUTPUT);
  pinMode(dir1, OUTPUT);
  pinMode(dir2, OUTPUT);

```



```
void loop(){

  PIDcalculation();// find PID value

  if (angle < setpoint) {
    digitalWrite(dir1, LOW);// Forward motion
    digitalWrite(dir2, HIGH);
  } else {
    digitalWrite(dir1, HIGH);//Reverse motion
    digitalWrite(dir2, LOW);
  }

  analogWrite(pwm, pidTerm_scaled);

  Serial.println("WHEEL ANGLE:");
  Serial.print(angle);

  delay(100);
}
```



```
void PIDcalculation(){
  angle = (0.9 * count);//count to angle conversion
  error = setpoint - angle;

  changeError = error - last_error; // derivative term
  totalError += error; //accumalate errors to find integral term
  pidTerm = (Kp * error) + (Ki * totalError) + (Kd * changeError);//total gain
  pidTerm = constrain(pidTerm, -255, 255);//constraining to appropriate value
  pidTerm_scaled = abs(pidTerm);//make sure it's a positive value

  last_error = error;
}

void Achange() //these functions are used to find the encoder counts
{
  A = digitalRead(2);
  B = digitalRead(3);

  if ((A==HIGH)&&(B==HIGH)) state = 1;
  if ((A==HIGH)&&(B==LOW)) state = 2;
  if ((A==LOW)&&(B==LOW)) state = 3;
  if((A==LOW)&&(B==HIGH)) state = 4;
  switch (state)
  {
    case 1:
    {
      if (statep == 2) count++;
      if (statep == 4) count--;
      break;
    }
    case 2:
    {
      if (statep == 1) count--;
      if (statep == 3) count++;
      break;
    }
    case 3:
    {
      if (statep == 2) count --;
      if (statep == 4) count ++;
      break;
    }
    default:
    {
      if (statep == 1) count++;
      if (statep == 3) count--;
    }
  }
  statep = state;

}
```



```
void Bchange()
{
  A = digitalRead(2);
  B = digitalRead(3);

  if ((A==HIGH)&&(B==HIGH)) state = 1;
  if ((A==HIGH)&&(B==LOW)) state = 2;
  if ((A==LOW)&&(B==LOW)) state = 3;
  if((A==LOW)&&(B==HIGH)) state = 4;
  switch (state)
  {
    case 1:
    {
      if (statep == 2) count++;
      if (statep == 4) count--;
      break;
    }
    case 2:
    {
      if (statep == 1) count--;
      if (statep == 3) count++;
      break;
    }
    case 3:
    {
      if (statep == 2) count --;
      if (statep == 4) count ++;
      break;
    }
    default:
    {
      if (statep == 1) count++;
      if (statep == 3) count--;
    }
  }
  statep = state;

}
```

The complete ino file, which include the above code, will be sent separately with this report. Also, the result has been shared with professor and TA. The following link is the address of the video. Professor and TA who have not receive that email can view the result via this link.

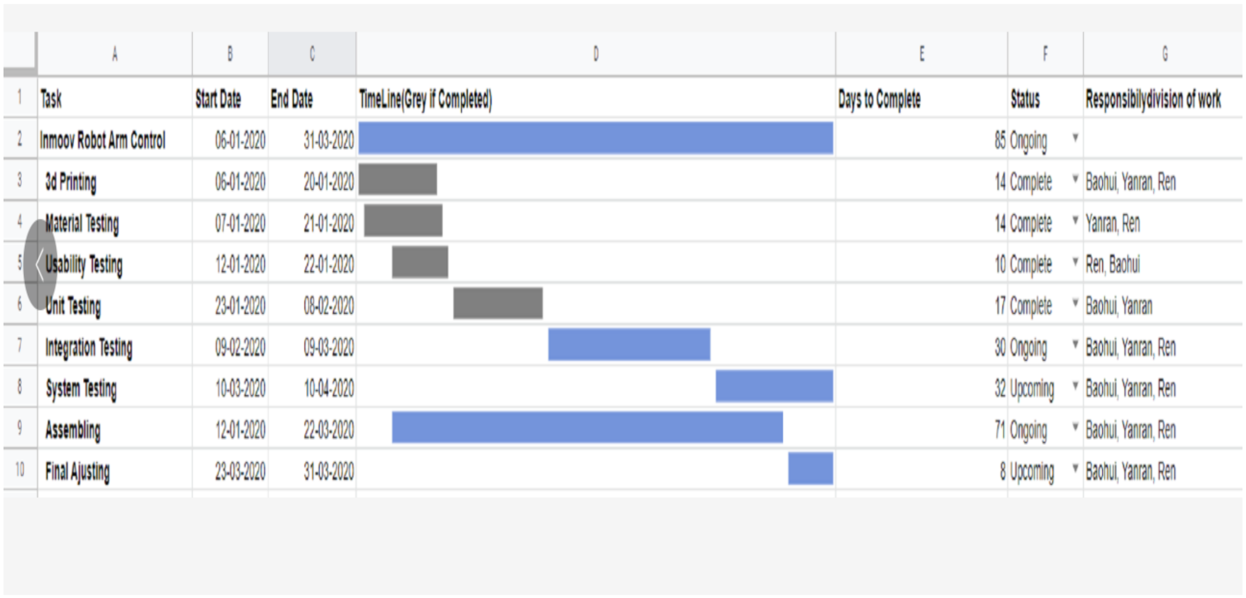
https://drive.google.com/open?id=1Xq5boR1C-lIS5bW4qk8VQ9HMb-vI-nzH

Financial analysis

Name	Price	Purpose	Name	Price	Purpose
AB Glue	25.98$	Object Bonding	HS-805BB	39.99$	Shoulder Joint 1 motor
Screw	25.98$	Fixing Connecting	HS-805BB	39.99$	Shoulder Joint 2 motor
Kite String	15.99$	Connecting	HS-755HB	47.99$	Elbow Motor
Arduino Board (Uno Rev3)	28.98$	Central Control	MG996	10.74$	Wrist Motor
Resistor	30.00$	Electrical Circuit Component	MD 1.3 Dual Motor Controller x5	12.99$	H-bridge
Wire (22AWG)	8.00$	Connection	Power Supply	11.99$	Power Supply
HS-805BB	39.99$	Bicep joint			
Total	**$174.92**		+	**$215.25**	**= $390.17**

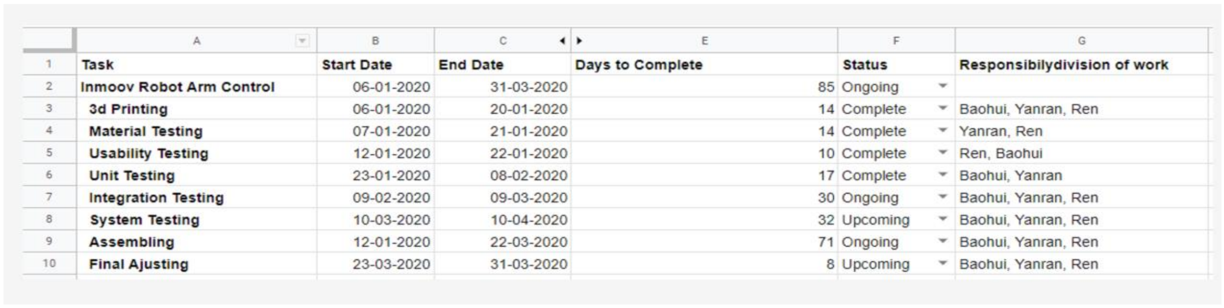
In this project, the actual consumption is 452.99. Obviously, it means our project is totally overestimate in budget. Due to serves failures, we had to purchase more material to fix our wrong steps, such as new glues and new nails. In addition, we also ordered an extra Arduino board for back up using. Fortunately, it has not been used since we are starting our project's assembly.

Gantt chart and job distribution



	A	B	C	D	E	F	G
1	Task	Start Date	End Date	TimeLine(Grey if Completed)	Days to Complete	Status	Responsibilydivision of work
2	Inmoov Robot Arm Control	06-01-2020	31-03-2020		85	Ongoing	
3	3d Printing	06-01-2020	20-01-2020		14	Complete	Baohui, Yanran, Ren
4	Material Testing	07-01-2020	21-01-2020		14	Complete	Yanran, Ren
5	Usability Testing	12-01-2020	22-01-2020		10	Complete	Ren, Baohui
6	Unit Testing	23-01-2020	08-02-2020		17	Complete	Baohui, Yanran
7	Integration Testing	09-02-2020	09-03-2020		30	Ongoing	Baohui, Yanran, Ren
8	System Testing	10-03-2020	10-04-2020		32	Upcoming	Baohui, Yanran, Ren
9	Assembling	12-01-2020	22-03-2020		71	Ongoing	Baohui, Yanran, Ren
10	Final Ajusting	23-03-2020	31-03-2020		8	Upcoming	Baohui, Yanran, Ren

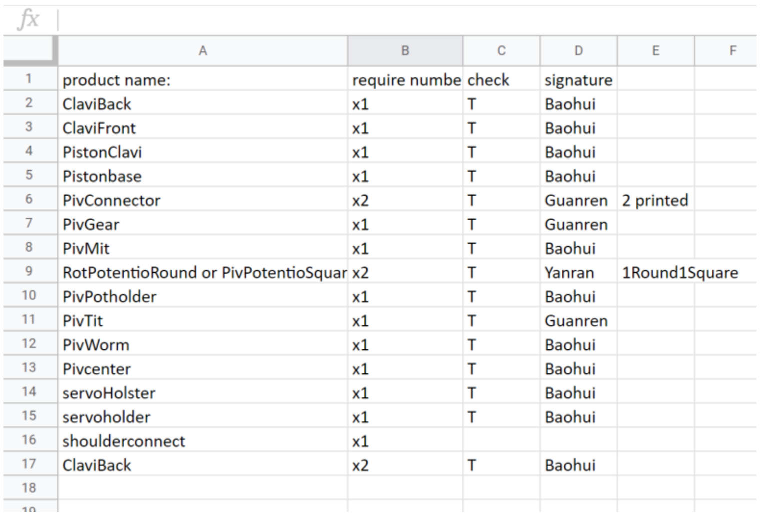
Figure 6: Gantt chart



	A	B	C	E	F	G
1	Task	Start Date	End Date	Days to Complete	Status	Responsibilydivision of work
2	Inmoov Robot Arm Control	06-01-2020	31-03-2020	85	Ongoing	
3	3d Printing	06-01-2020	20-01-2020	14	Complete	Baohui, Yanran, Ren
4	Material Testing	07-01-2020	21-01-2020	14	Complete	Yanran, Ren
5	Usability Testing	12-01-2020	22-01-2020	10	Complete	Ren, Baohui
6	Unit Testing	23-01-2020	08-02-2020	17	Complete	Baohui, Yanran
7	Integration Testing	09-02-2020	09-03-2020	30	Ongoing	Baohui, Yanran, Ren
8	System Testing	10-03-2020	10-04-2020	32	Upcoming	Baohui, Yanran, Ren
9	Assembling	12-01-2020	22-03-2020	71	Ongoing	Baohui, Yanran, Ren
10	Final Ajusting	23-03-2020	31-03-2020	8	Upcoming	Baohui, Yanran, Ren

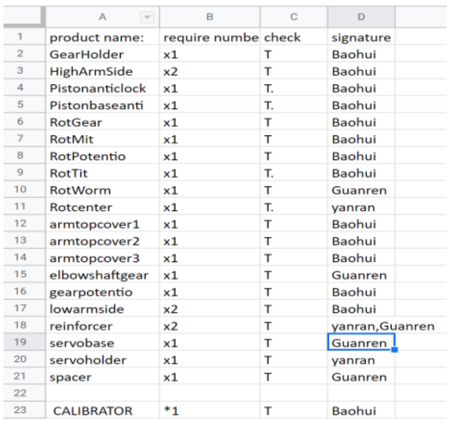
Figure 7: job distribution

Contribution on Printing



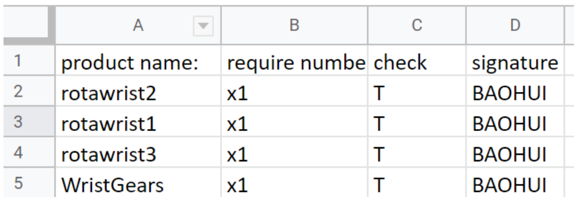
	A	B	C	D	E	F
1	product name:	require numbe	check	signature		
2	ClaviBack	x1	T	Baohui		
3	ClaviFront	x1	T	Baohui		
4	PistonClavi	x1	T	Baohui		
5	Pistonbase	x1	T	Baohui		
6	PivConnector	x2	T	Guanren	2 printed	
7	PivGear	x1	T	Guanren		
8	PivMit	x1	T	Baohui		
9	RotPotentioRound or PivPotentioSquar	x2	T	Yanran	1Round1Square	
10	PivPotholder	x1	T	Baohui		
11	PivTit	x1	T	Guanren		
12	PivWorm	x1	T	Baohui		
13	Pivcenter	x1	T	Baohui		
14	servoHolster	x1	T	Baohui		
15	servoholder	x1	T	Baohui		
16	shoulderconnect	x1				
17	ClaviBack	x2	T	Baohui		
18						

Figure 8. table for contribution on printing “Shoulder”



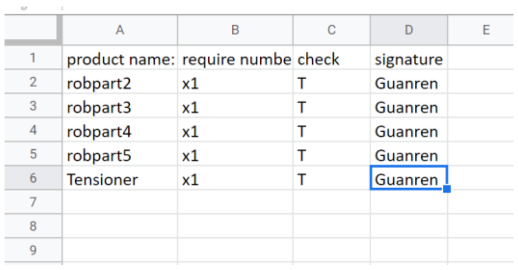
	A	B	C	D
1	product name:	require numbe	check	signature
2	GearHolder	x1	T	Baohui
3	HighArmSide	x2	T	Baohui
4	Pistonanticlock	x1	T.	Baohui
5	Pistonbaseanti	x1	T.	Baohui
6	RotGear	x1	T	Baohui
7	RotMit	x1	T	Baohui
8	RotPotentio	x1	T	Baohui
9	RotTit	x1	T.	Baohui
10	RotWorm	x1	T	Guanren
11	Rotcenter	x1	T.	yanran
12	armtopcover1	x1	T	Baohui
13	armtopcover2	x1	T	Baohui
14	armtopcover3	x1	T	Baohui
15	elbowshaftgear	x1	T	Guanren
16	gearpotentio	x1	T	Baohui
17	lowarmside	x2	T	Baohui
18	reinforcer	x2	T	yanran,Guanren
19	servobase	x1	T	Guanren
20	servoholder	x1	T	yanran
21	spacer	x1	T	Guanren
22				
23	CALIBRATOR	*1	T	Baohui

Figure 9. table for contribution on printing “Bicep”



	A	B	C	D
1	product name:	require numbe	check	signature
2	rotawrist2	x1	T	BAOHUI
3	rotawrist1	x1	T	BAOHUI
4	rotawrist3	x1	T	BAOHUI
5	WristGears	x1	T	BAOHUI

Figure 10. table for contribution on printing “forarm”



	A	B	C	D	E
1	product name:	require numbe	check	signature	
2	robpart2	x1	T	Guanren	
3	robpart3	x1	T	Guanren	
4	robpart4	x1	T	Guanren	
5	robpart5	x1	T	Guanren	
6	Tensioner	x1	T	Guanren	
7					
8					
9					

Figure 11. table for contribution on printing "Back"

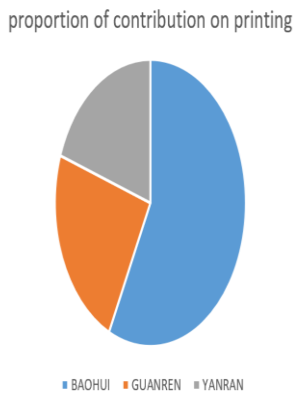


Figure 12. table for contribution on printing

Contribution for each member

A. GuanRan
   1. Printing 13 item out of 56
   2. Assembling mainly on back and forarm
   3. welding and making Dupont wire
   4. Writing PID control
   5. Final report (Work illustration, Financial analysis and PID control parts)
B. YanRanWang
   1. 3D Printing and disqualified components rework
   2. Assembling the arm, mainly on elbow and bicep parts
   3. welding wires and heads with solder wires and electric wire connectors
   4. Arduino coding mainly on user input conversion
   5. Making batteries with certain voltage
   6. Error detecting and debugging
C. BaohuiXie
   1. The entire control board (including wiring motor driver, sensor, battery bank and Arduino)
   2. Assembling the arm (especially the shoulder and bicep part)
   3. Handmade Dupont wire (welding and measuring)

4. Print 32 items out of 56 items
5. Coding: all the contribution is shown in the code common by noting as "Baohui" (setup(), loop(), stateOfRotation(), rotationOperator(), turnOffMotor(), clockWiseRotate(), counterClockWiseRotate())
6. Testing the code and measuring the error between input and actual value.
7. Come up with the minimum and maximum input range of each joint
8. Final report (Final prototype, Analysis of result and problems, Conclusion and recommendation of further development)

Analysis of result and problems

Again, here is the link for the result.

https://drive.google.com/open?id=1Xq5boR1C-lIS5bW4qk8VQ9HMb-vI-nzH

The result for our project is good. Nevertheless, two problems are still existing in certain input values.

First, for the first three joints, there is no difficulty with movement, and there is no phenomenon of rotating to the wrong direction and wrong angle value. However, for the elbow, there is an error between the input value and final measurement. But the error is always smaller than 5 degrees. The problem may be caused by the sensor holder beside the sensor, since we can not stick the holder to the bicep strongly with the current material that we have. Thus, the holder will move a little bit along with the rotation direction.

The second problem will be the power supply. Currently, we are using five 1.5V batteries to build up a 7.5V battery bank. By doing that, we reduce our budget and the weight of the control board. However, the disadvantage is that we can not set the input voltage to 8V or 8.5V. Since the maximum rated voltage for the motor is 8.5V, 7.5V is not enough for the motor to push the arm to the highest position. If we add one more battery to the battery bank, then the total output voltage will be 9V, which is higher than the 8.5V rated voltage of the motor. Thus, considering our budget, there is no way to fix this problem without a suitable DC power supply.

The overall result of the performance of the arm is great, and it has the ability for improvement. And the improvement will be concluded in the following section.

Conclusion and recommendation of further development

As we mentioned above, the performance of the arm can be improved. We believe that there are at least 3 improvements.

The first improvement is related to the power supply problem that we mentioned above. By replacing the hand made battery bank with a better DC power supply, the problem of the battery bank power supply cannot support the arm to reach that position will be solved. Or, we can build up a higher voltage battery bank and divide the voltage to around 8.5V.

Second, the PID control of the arm has only a delicate performance, so we have not put it on the demo code. To improve that, we may use the technique of autotuning before the signal goes to the PID section instead of manual tuning. The PID also affects the speed of the motor by changing the pulse width of the motor power signal. During the demo, we found that the arm will stop performing if the speed (or the pulse width of the power signal) becomes lower than a certain point. We think it may be caused by the problem of the voltage. Since our power supply is only 7.5V, a 50% pulse width power signal will reduce the voltage to 3.75V, which is lower than the lowest rated voltage of the motor. Thus, the motor stop and the performance of the arm will be interrupted.

Last, instead of rotating the joint one by one, we believe that we can write a delay function to synchronize the performance of each joint. By doing that, we can set a timer in the code or put all the motor enable code in the same loop. So that the second motor does not have to wait for the execution to jump out from the last loop.

All in all, everyone in the group works hard together for this section. Everyone in the group has try at least 3 different kinds of jobs. And we have made a very great achievement.