Y2 project

Robotic Arm Ball Game

By: Ziming Xu 201565645, Muyuan Yan 201677316, Tianyou Li 201676614,

Jiankun Li 201676580, Zaynab Abdulkadir 201432505

Supervised by: Waleed Al-Nuaimy

Department of Electrical Engineering and Electronics

Abstract

Our Year 2 project consisted of building a robotic arm capable of picking up small objects. We designed this robotic arm to be accompanied by a simple ball game, intended to be used as a demonstration on the university's open days to attract potential new students to the department. Ultimately, we have successfully produced a functioning product. The purpose of this report is to present the design and implementation of our product that allowed us to reach our goals, and the results we found as a result.

Declaration

Table of Contents

List of Figures

Introduction, Motivation and Objectives

The idea of creating a robotic arm was collectively suggested by some of the group members, as none of us had prior experience working on this project. We believed that this project would provide the perfect learning experience for each member of the team.

Though many similar robotic arm projects had been done in the past by different groups, we made it a point to ensure that our project is unique. Through analysing the previous Year 2 projects, we decided to use unique control methods as well as provide an accompanying game to allow our product to stand out amongst other similar projects. In addition to this, the entire product design, from the structure of the arm to the software designed and implemented from scratch, was created by the team members. This solidifies the uniqueness of the project, as well as creates more opportunity for our skills to flourish.

The goal of this project was to produce a functional robotic arm that can pick up small objects. The arm should have then been able to move the object over a hoop and drop it through it. We planned for the hoop to detect when an object passed through it, increasing a score on an LED display until a built-in timer reached zero, resetting the score. We planned for the structure of the arm to be strong yet lightweight for it to not only be durable but easily manoeuvred.

For this project to be used in the university's open day as intended, we believed that the robotic arm would need to be easy to use, portable and aesthetically pleasing to attract and impress the users.

Materials and Methods

As mentioned previously, the robotic arm was designed from scratch by our team. Singlehandedly designing the mechanical structure of the robotic arm had great advantages. As opposed to purchasing premade parts, designing the structure ourselves meant that we could independently set the parameters of each part to meet the needs of the robot arm and adjust them at any time if needed. After having a team discussion, factors such as the size of each motor, the wiring arrangement, and the length of the branch arm were determined to prevent us from encountering issues later whilst creating the arm.

The main component of this project was the Arduino Mega2560 board, where all the software parts were uploaded to. The Arduino was used in our project as a bridge between the different parts of the arm, the LED display and the IR sensor in the hoop. We used two Arduino boards, one to control the robotic arm and one to change the score on the display once an object had been detected in the hoop. We had no choice but to do this as parallel programming is nearly impossible on an Arduino board [1].

The hinges on the arms were controlled using two HiTec 112081 HS-81 Micro servo motors, at the base and shoulder of the arm, as well as three Rapid Pro SG90 mini servo motors, used on the elbow wrist and claw of the arm. The HiTec servo motors are larger and more powerful thus we decided to use them at the areas of the arm which would be enduring the most weight. This was done to ensure the arm had enough power to move around smoothly. The user input was done using three Adafruit 512 Analog 2-axis Thumb Joysticks.

3D Modelling and Structure Design

The reason why we decided to use an acrylic plate as the basic structure of the arm is that the material has the advantages of portability and low cost. The use of this material helped us quickly produce the first prototype of the arm and made finding problems in the design easier.

We decided to use resin materials for 3D printing, because resin is high-quality, cheap and products made from resin can be created quickly. Resin has significant advantages over other 3D-printed materials in terms of producing high-quality components, making it one of the preferred choices of material to use in our project.

In the first version of the design (Figure 1), acrylic sheets were used to determine the basic structure of the entire arm, and the connections between the components were fixed using traditional screws.

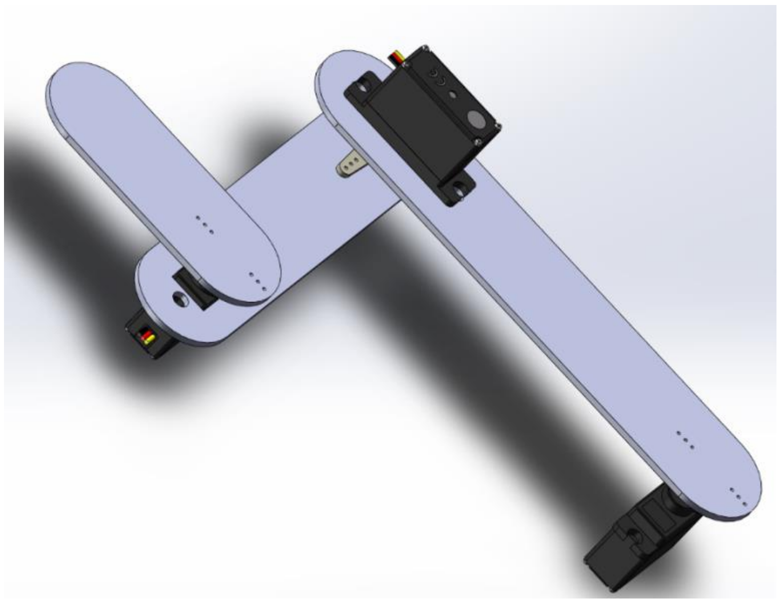


Figure 1 First Prototype design

In the first version of the design, there were several issues that needed to be addressed to achieve better performance and functionality of the robotic arm. Firstly, when connection the acrylic plates with the motors, due to failure of finding appropriate screws, the connection points needed to be redesigned to ensure that the arm would not come apart.

Secondly, the length of the arm needed to be experimented with to achieve higher control ability. In addition, to add the self-cycle 180-degree rotation function, the control system needed to be upgraded, including the addition of motors and circuits, and the reprogramming of the controller to achieve greater accuracy and reliability.

To reduce the load of the rotating base motor and reduce the resistance caused by the friction between the components, we installed bearings at the base section. To maintain the stability of the arm during rotation, a tapered design was given to the base to improve its stability, as seen figure 2.

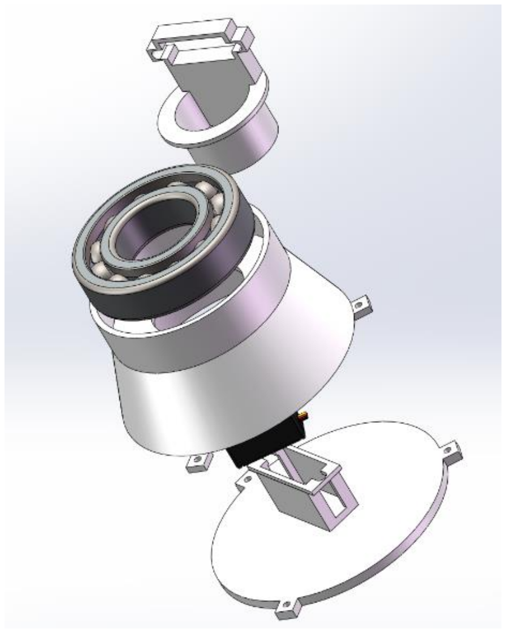


Figure 2 Arm Base Design

As for the problem of not being able to use screws, we adopted a new connection method after a discussion within the group. We engraved shapes on the printed components in the same shape as the motor connections and fixed them with M3 screws (as seen in figure 3).

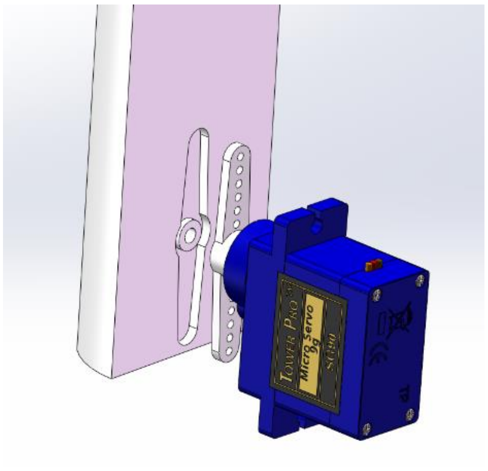


Figure 3 Connection between motor and arm.

For the claw part, considering the load capacity of the motor and the size of the mechanical arm, we decided to use three claws, in which only one claw was moved by the motor and the other two were fixed, shown below in figure 4.

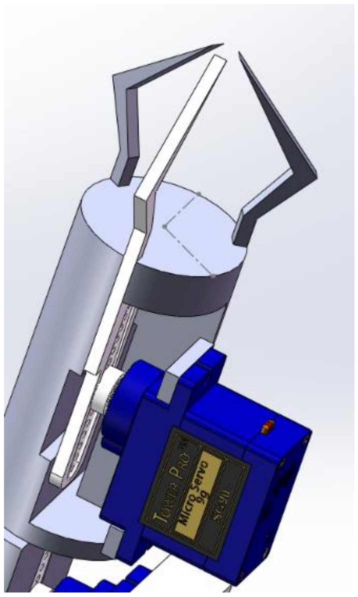


Figure 4 Claw design

Software

Source codes are available in Appendix C

Arm Control Method 1: Angle Control

The syntax of the Arduino code for controlling the motor was as shown below:

$$ServoPin.write(Angle)$$

The first control method was straightforward, using the joystick to determine the size of angle the servo should move by. Before demonstrating the coding of this version, it is important to know the syntax of joysticks. Each joystick outputs a two-dimension array, which shows the current position of the cord.

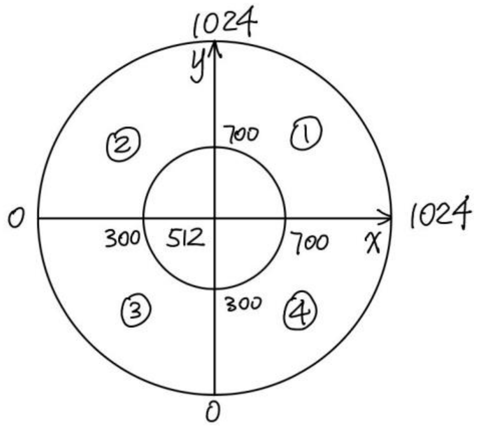
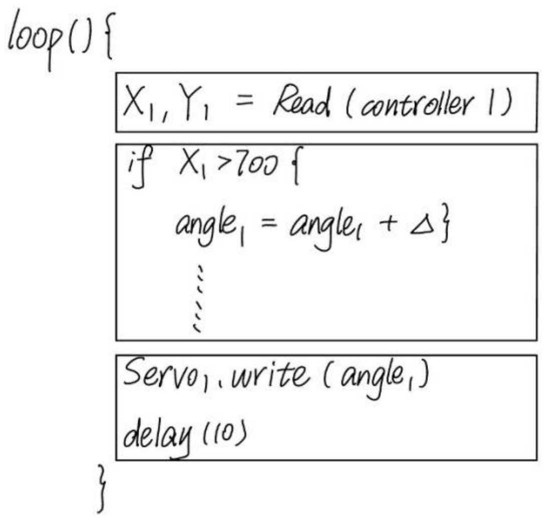


Figure 5 Controller output Sketch

As shown in figure 5 above, the one dimension of the output ranges from 0 to 1024, which is the most basic way to control the servo motors. We chose x to control Servo1 and y to control Servo2. If $X > 700$, the angle of Servo1 increased by Δ :

$$angle = angle + \Delta$$

It is changed at the end of the loop() function by $Servo1.write(angle)$. If $x < 300$, the angle of Servo1 decreases. And the same for Servo2. Therefore, the entire structure of the code is as follows, shown in Figure 6:



```
loop() {
    X1, Y1 = Read (controller 1)
    if X1 >700 {
        angle1 = angle1 + Δ}
        ⋮
    Servo1.write (angle1)
    delay (10)
}
```

Figure 6 Basic Servo Controlling System

This version was quite straightforward, but not intuitive. Additionally, it was hard to remember which joystick controlled each motor and the movement of the arm was not precise enough. Thus, another control method was developed.

Arm Control Method 2: Inverse Kinematics

Based on the goal of our robotic arm's claw moving in a straight line, a new control method was chosen: inverse kinematics. Inverse Kinematics is the use of kinematic equations to 'determine the motion of a robot to reach a desired position' [2]. Using this control method allowed for us to create a robotic arm that moved smoothly and precisely, creating a seamless, positive user experience.

The plane that contains the robotic arm, shown in Figure 7 below, was chosen to be the reference plane. The aim of this control method was for us to control the position of any of the servo motors in any direction.

Since only two servos are involved in this process, instead of using Jacobian matrix to convert the $dxdy$ in the controlling space to $d\theta_1 d\theta_2$ in the joint angle space, we used a more direct way to calculate the angles.

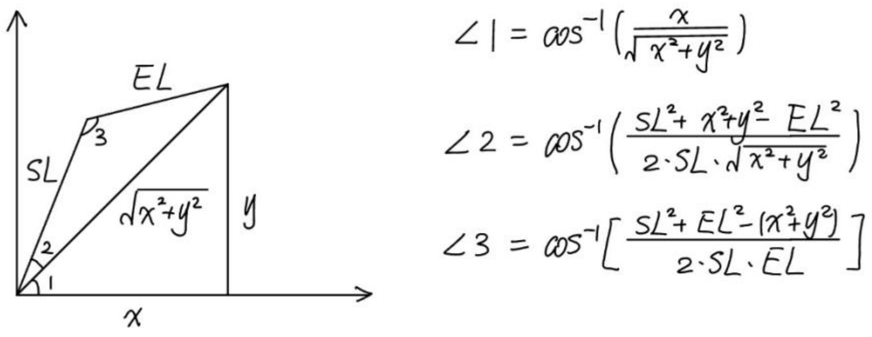

EL
SL
3
2
1
$\sqrt{x^2+y^2}$
y
x
$\angle 1 = \cos^{-1}\left(\frac{x}{\sqrt{x^2+y^2}}\right)$
$\angle 2 = \cos^{-1}\left(\frac{SL^2+x^2+y^2-EL^2}{2\cdot SL\cdot\sqrt{x^2+y^2}}\right)$
$\angle 3 = \cos^{-1}\left[\frac{SL^2+EL^2-(x^2+y^2)}{2\cdot SL\cdot EL}\right]$

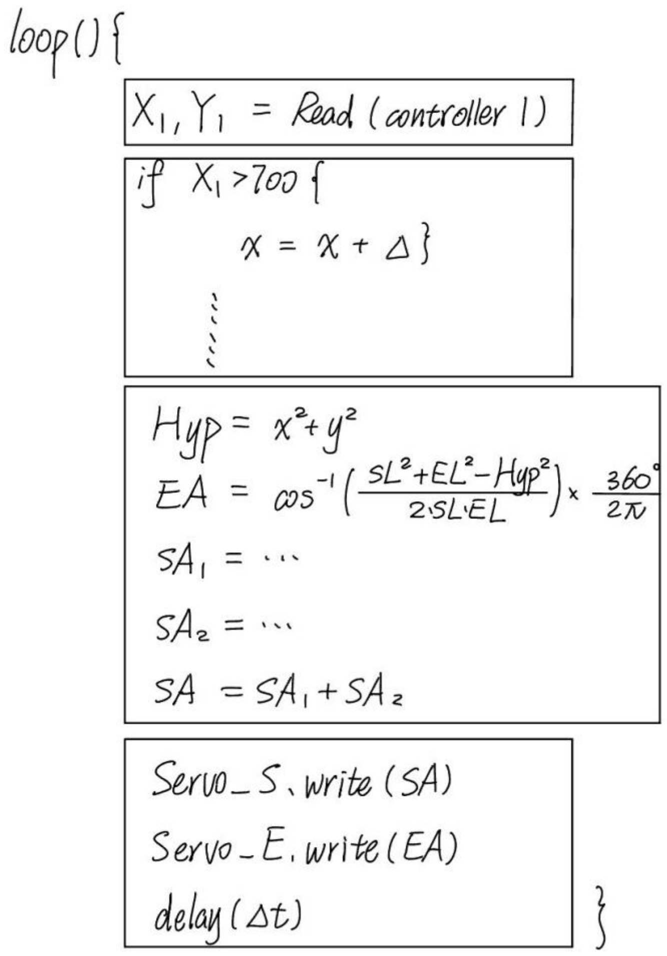
Figure 7 Geometry Calculation Demo

As shown in figure above, the servo at (0,0) was named Shoulder Servo, the servo in the middle was named Elbow Servo and the last servo was named Wrist Servo. Thus, SL and EL represented the length of upper arm and forearm. We assumed the position of the Wrist was (x, y). Thus, it was easy to calculate the resulting servo angles with the cosine rule.

The rule implies that it was possible to change the value of x or y directly:

$$if\ (X > 700)\{x = x + \Delta\}$$

In this case, Δ stands for the distance that the Wrist Servo was changed horizontally. Thus, for each loop in the loop() function, we first changed the (x, y) position of the wrist according to joysticks output (X, Y), then calculated the new angles that needed to be updated into the servo motors based on the algorithm introduced above, and finally wrote the new angles into the servos. So, the new coding structure is shown below in Figure 8:



```
loop() {
    X_1, Y_1 = Read (controller 1)

    if X_1 >700 {
        x = x + Δ }
    ⋮

    Hyp = $x^2 + y^2$
    EA = $\cos^{-1}\left(\frac{SL^2+EL^2-Hyp^2}{2 \cdot SL \cdot EL}\right) \times \frac{360°}{2\pi}$
    SA_1 = ...
    SA_2 = ...
    SA = SA_1 + SA_2

    Servo_S.write(SA)
    Servo_E.write(EA)
    delay(Δt)
}
```

Figure 8 Final Servo Controlling System

The last adjustment was the balance between the moving distance Δd , each loop and delay time Δt . As the Figure 9 below shows, to achieve a smoother moving system, $\Delta d = 0.1cm$ and $\Delta t = 10ms$ is chosen to provide a buffer after each move.

$$t_{loop} = t_{calculation} + t_{delay}$$

one loop $\Rightarrow$ one movement $\Rightarrow$ (x, y) change once Δd

$$\text{distance moved/second} = \text{move times} \times \Delta d$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$v = \text{loops/second} \times \Delta d$$

$$\Downarrow$$

$$v = f_{loop} \times \Delta d$$

$$f_{loop} = \frac{1}{t_{loop}} \propto \frac{1}{t_{delay}}$$

Keep v a medium speed (constant):

$$(\text{delay time})^{-1} \times \Delta d = \text{Constant}$$

Figure 9 Smooth Movement Balancing

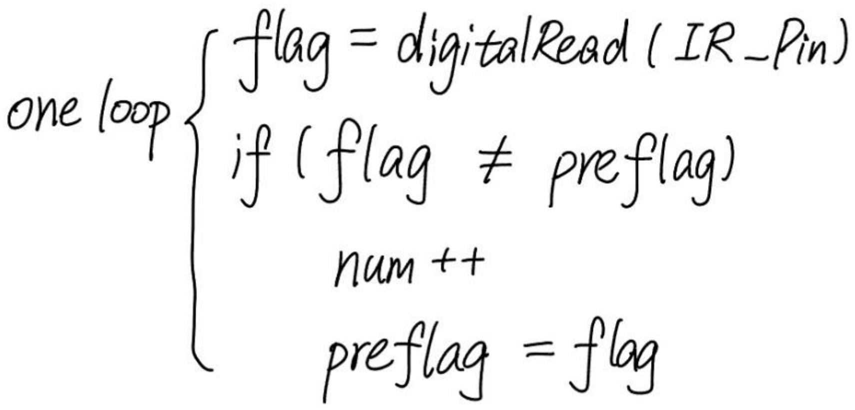
Game Design

After deciding the algorithm for arm controlling, the rest of the work to be done was on the game design. To build a basic structure for the game part, it was important to decide which device we needed to receive the input from the real world, in this case the object entering the hoop. Reflecting on the contactless sensing method, IR sensor was chosen to implement in the hoop design. First, starting from the essential syntax of IR sensor:

$$output = digitalRead(IRPin)$$

As shown in the code above, the output of IR sensor is a digital signal read from the IR pin, which means there is just two kinds of output 0 or 1. For each kind of output, there is a corresponding situation, 0 meaning nothing detected and 1 meaning something detected. Thus, the main purpose of the code was to record the changes in the IR sensor output as the number of balls into the basket.

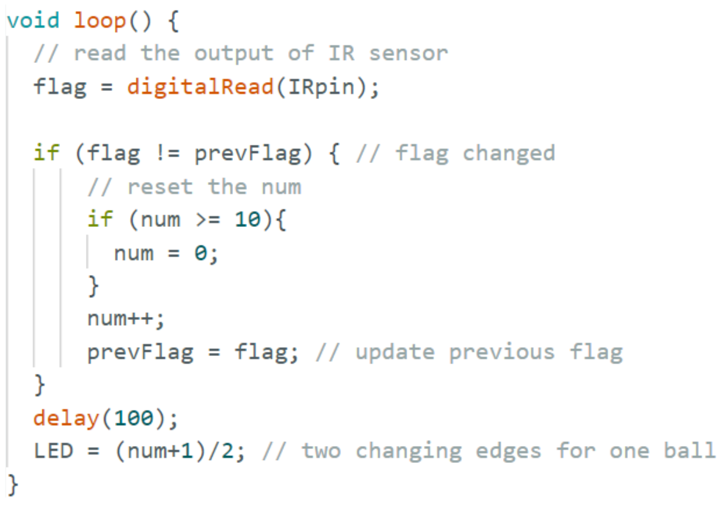
Hence, the coding was reasonably straightforward. One variable in the software recorded the previous state of the IR sensor in the last loop, stayed during the current loop, then changed at the end of current loop. The other recorded the current IR sensor output read in the beginning of the current loop. Next, if function was adopted to detect the change. This is displayed in Figure 10.



```
one loop {
    flag = digitalRead(IR_Pin)
    if (flag ≠ preflag)
        num ++
        preflag = flag
}
```

Figure 10 Basic Structure for IR sensor Code

To provide deep understanding of the code, it is necessary to now discuss the other facts found in later testing. Although the code design above seemed promising, the fact is that due to the reading frequency of IR sensor (>30Hz) and the rough surface of the ball, it was a possibility for the IR sensor to output more than one change during the detection process. Thus, it is important noting that improvement for this code was needed, which resulted in the code shown below (Figure 11):



```
void loop() {
  // read the output of IR sensor
  flag = digitalRead(IRpin);

  if (flag != prevFlag) { // flag changed
    // reset the num
    if (num >= 10){
      num = 0;
    }
    num++;
    prevFlag = flag; // update previous flag
  }
  delay(100);
  LED = (num+1)/2; // two changing edges for one ball
}
```

Figure 11 Codes for IR sensor

As shown above in the Fig. 6, the key design was the delay time and LED calculation. In the code it is evident that there were two changes in one falling process, which meant the true number of the balls needed to be divided by 2. And according to what was stated earlier, the reading frequency is too high for this system, which may cause the system to unstable after experiencing too many changes. Thus, the delay time was introduced to avoid this phenomenon.

After finishing the IR sensor, the rest of the work was to figure out a method to use these variables to control the 7-segment display (Figure 12). It was straightforward to design the coding with the aid of inner circuit shown below:

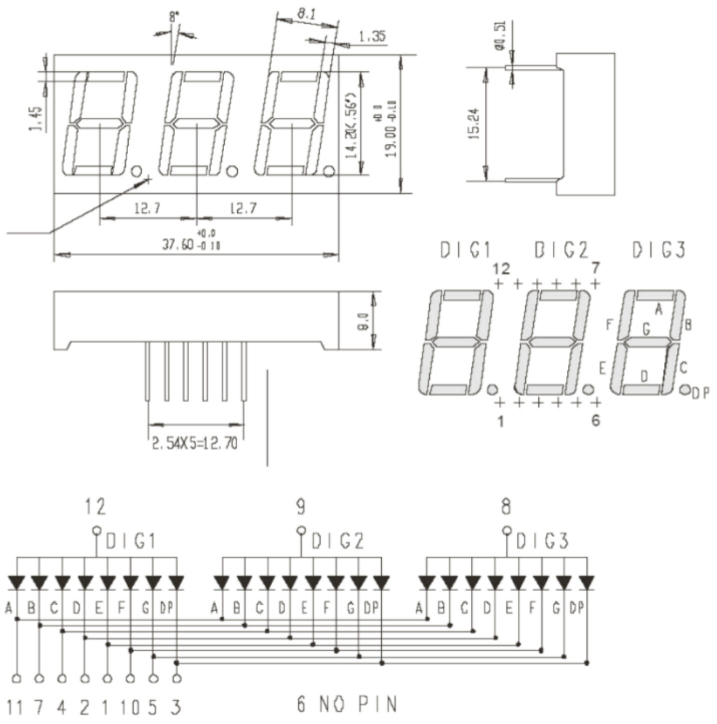


Figure 12 LED Display Circuit Diagram [3]

The method was to pick up the corresponding diodes and make the pins "LOW". Thus, the gaming section of the project was completed.

Hardware Implementation

During the process of completing the hardware section of the project, it was divided into three parts.

The first part was the work on the joystick control. The joysticks had five pin feet that controlled horizontal (x), and vertical (y) movement, intermediate buttons, voltage input, and ground. However, the central button was not used in the design process. To control the arm, x and y were connected to the Arduino board and controlled through the computer program presented earlier. In summary, when the user input signals through the joystick, it controlled the motors via the Arduino board.

The second part was the robotic arm. Firstly, two types of motors were used; the base rotary motor and the bottom motor used large motors with a torque of 30N/CM, and the other parts were all smaller motors with a torque of 9N/CM. Each motor was connected by three wires, namely signal input, power input, and ground. In the signal input part, the rotation speed and angle of the motor were controlled by the program, and the motor was adjusted. During testing, we discovered that the motor could not rotate 360 degrees, it could only rotate less than 180 degrees, so when installing the motor, it was necessary to determine the position of the initial motor. Secondly, in terms of structure, due to the imprecision of the 3D-printed pieces, adhesive was our only option when it came to connecting them together and supporting the structure.

Our third task was the game part, the duties were divided into working on the structure and the circuit. The components of the game were a paper ball and cardboard bucket, with a door at the bottom of the bucket for the ball to pass through. By adding an IR sensor at the entrance of the bucket, the ball could be detected when it passed through the sensor. Through the detection of the ball from the sensor, a signal triggered the number on the electronic screen to increase by one. However, the IR sensor also caused some issues with the accuracy of the detection. For example, when it detected an object in the detection interval, it always output a signal of 1, which led to signal superposition, and the number on the electronic screen increased by more than 1, an issue that only came to our attention after our project was already complete.

Connection:

Picture of Arduino mega as Shown in Figure 13:

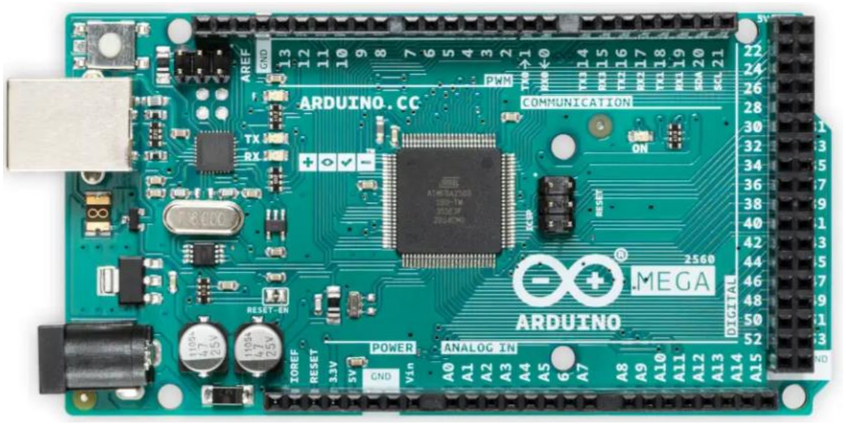


Figure 13 Arduino Mega

- For controller 1, the x output was connected to the A0 pin of the Arduino, and the y output was connected to the A1 pin.
- For controller 2, the x output was connected to the A2 pin of the Arduino, and the y output was connected to the A3 pin.
- For controller 3, the x output of controller 3 was connected to the A4 pin of the Arduino, and the y output was unused.
- The shoulder motor was connected to pin A8.

- The elbow motor was connected to pin A9.
- The wrist motor was connected to pin A7.
- The claw motor was connected to pin A6.
- The base motor was connected to pin A5.

For the game half of the project, a second Arduino Mega was used:

The IR sensor was connected to pin 2 of the Arduino board.

For the 7 segment display (Figure 14):

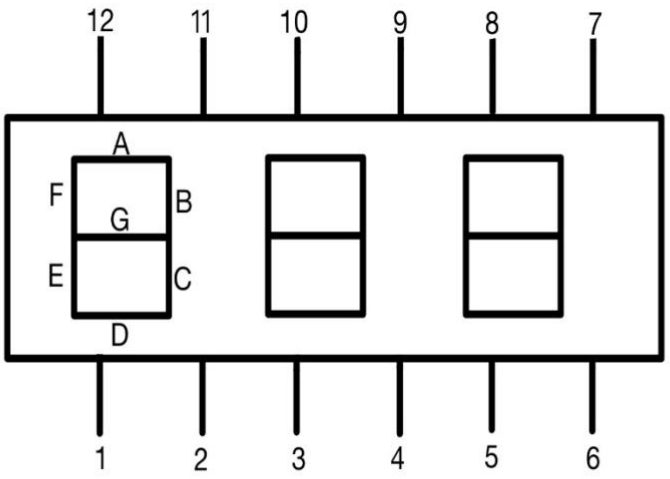


Figure 14 Seven segment display

- When the power supply was connected to 12, the signal could control the first digit;
- When the power supply was connected to 9, the signal could control the second digit;
- When the power supply was connected to 8, the signal could control the third digit;
- When connecting 11 pins, digital tube A lit up;
- When connecting 7 pins, digital tube B lit up;
- When connecting 4 pins, digital tube C lit up;
- When connecting 2 pins, digital tube D lit up;
- When connecting 1 pin, digital tube E lit up;
- When connecting 10 pins, digital tube F lit up;
- When connecting 5 pins, digital tube G lit up;
- When connecting 3 pins, digital tube dot lit up;
-

Results

A 3D model of the final overall robotic arm is shown in figure 15.

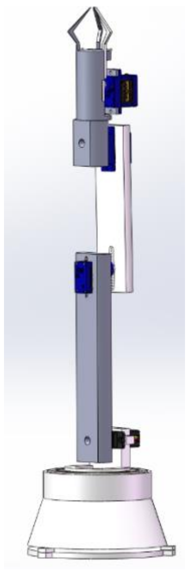


Figure 15 Final design of the arm structure

The final product has met its functional requirements: the arm was functional, capable of picking up small objects as intended by design, the control was responsive and intuitive, with each input on the controller joysticks, causing the arm to move in an expected and intended way. The inverse kinematic control of the arm was unique and improved the overall handling of the arm, its implementation was successful with all the servo angles calculated and output correctly. The game part was functional, with the sensor being capable of detecting the objects falling through and consequently increasing the point counter as intended.

Despite this, the final product was not without flaws, specifically a glitch of the system occurring occasionally causing the arm to rotate to the left. Some testing was done as a result and we were able to locate the origin of the problem, but further testing and improvements could not be carried out due to time constraints. One gear within the elbow servo was suspected to be broken, which hindered the smoothness of the arm operation, yet we did not have a replacement part for it. Though the effect was small, this may affect the accuracy of the arm over long term use. As well as this, most of 3D-printed parts were printed smaller than expected, which led in the components not being able to fit properly, leaving us no option but to sand and glue the parts together, which had a minor impact on the overall aesthetic of the product.

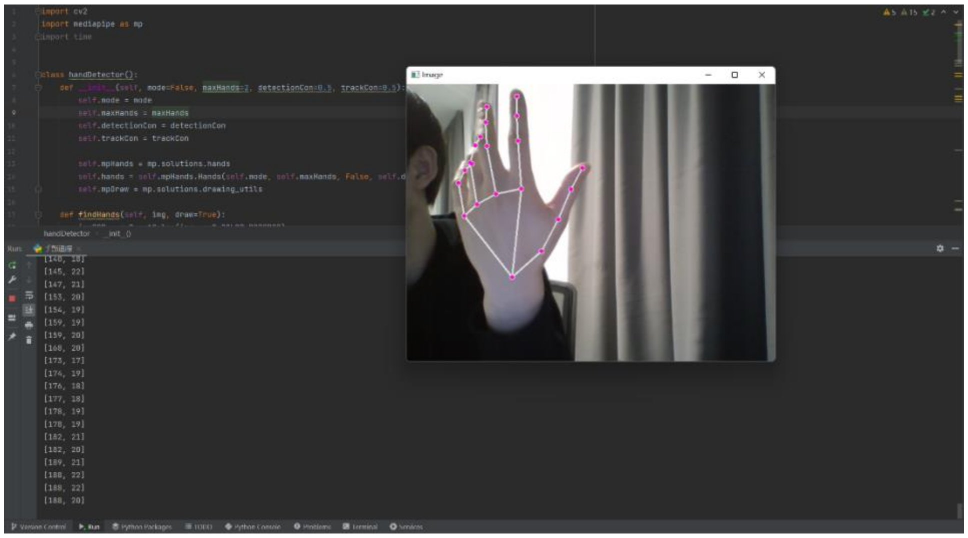
Discussion and Conclusions

There were several problems in the final product, and as mentioned above, we were not able to fix the problems due to the time constraint and due to the fact that the problems occurred mainly during the later stages of the project. Despite this, we do have ideas on how to fix the problems if given more time.

During the final test, we noticed that the connection of the joysticks was not perfect. Specifically, the output signal of each joystick was not stable at 512 (1024/2). Instead of an ideal flat signal, the real output signal contained some glitches caused by the bad connection of the controller itself. In some instances, the signal jumped to 0 from 512, which led to the bad performance of the robotic arm. For example, the base motor sometimes rotated to initial stage despite having no user command.

The most efficient way to fix this problem would be to change the controller to a new one. However, since it was not possible to change the device at the end of the project, the only way to avoid this was to hold the controller firmly against the table during use.

There were also several ideas that were not finished due to limited time and materials. The first one was a hand tracking system to control the arm. As shown below in Figure 16, we used Python package PySerial and OpenCV to send the position of detected hand to the Arduino.



```
import cv2
import mediapipe as mp
import time


class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands, False, self.d
        self.mpDraw = mp.solutions.drawing_utils

    def findHands(self, img, draw=True):
```

```
[140, 18]
[145, 22]
[147, 21]
[153, 20]
[154, 19]
[159, 19]
[159, 20]
[168, 20]
[173, 17]
[174, 19]
[176, 18]
[177, 18]
[178, 19]
[178, 19]
[182, 21]
[182, 20]
[189, 21]
[188, 22]
[188, 22]
[188, 20]
```

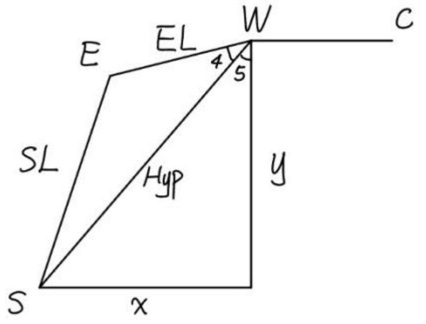
Figure 16 Hand Tracking Demo

We used the position of the hand to determine the direction we wanted the robotic arm to move. Unfortunately, this idea was dismissed to avoid the whole system crashing. The reason for this is that the processing time for pictures taken in 1 second determined the FPS of this system, which is shown as:

$$FPS = \frac{1}{t_{per\ picture}}$$

FPS determines the number of signals transmitted to Arduino in one second. As mentioned before, the delay time was necessary to make the operate system smoothly. In this case the number of signals transmitted to Arduino per second determined the delay time in each loop. Thus, this system would have become more dependent on the PC performance, which would not have been successful in this project.

Another idea we decided against is a self-stabilizing claw (Figure 17). The purpose of this design is to make sure that the claw does not move around whilst the position of wrist changes. The design is simple and the figure below shows the geometry behind it.



$$WA = 90^\circ + \angle 4^\circ + \angle 5^\circ$$

$$\angle 4 = \cos^{-1}\left(\frac{EL^2 + Hyp^2 - SL^2}{2 \cdot Hyp \cdot EL}\right)$$

$$\angle 5 = \cos^{-1}\left(\frac{y}{Hyp}\right)$$

Figure 17 Self-stabilizing System

However, after testing the entire robotic arm, we found that whilst this system had the potential to make our project more sophisticated, it ultimately made the robotic arm harder to control. This is because the stabilizing claw may have led to users forgetting which part of robotic arm they were controlling. Although it was appealing to give our robotic arm an edge through applying these ambitious ideas, our main focus was on the functionality of the arm, and we felt that it was highly important to create and present a good finished product rather than have nothing but great ideas in the end.

If we had more time and financial support during this process, we would be able to improve the overall mechanism of robotic arm. In terms of components, we would be able to choose more sophisticated motors, such as the 3040 model, to control the rotation speed and angle of each node of the robotic arm, as well as the entire base in a more accurate way. In addition, we would no longer be confined to the connection method we previously used between the motor and each branch arm; we would use a more suitable component, and a more common screw to connect the two, to improve the stability of the robotic arm.

In terms of 3D-printing, if higher precision printing could be used, then our designs may have turned out to be more aesthetically pleasing. In addition to this, it would be possible to print with stronger materials such as nylon, which have higher strength and lighter weight, yet it may take longer to print. Furthermore, we could also use CNC technology instead of 3D printing technology and replace the entire robotic arm structure with metal to reduce the structural errors and make the mechanical structure of the entire robotic arm more stable.

Longer project development time also means we would be able to find problems with the existing robotic arm and improve it ourselves, then move on to working the next, newly improved version of the arm. However, since we only had 4 weeks and could only do one session of 3D printing, we couldn't possibly solve the problems that occurred closer to our deadline.

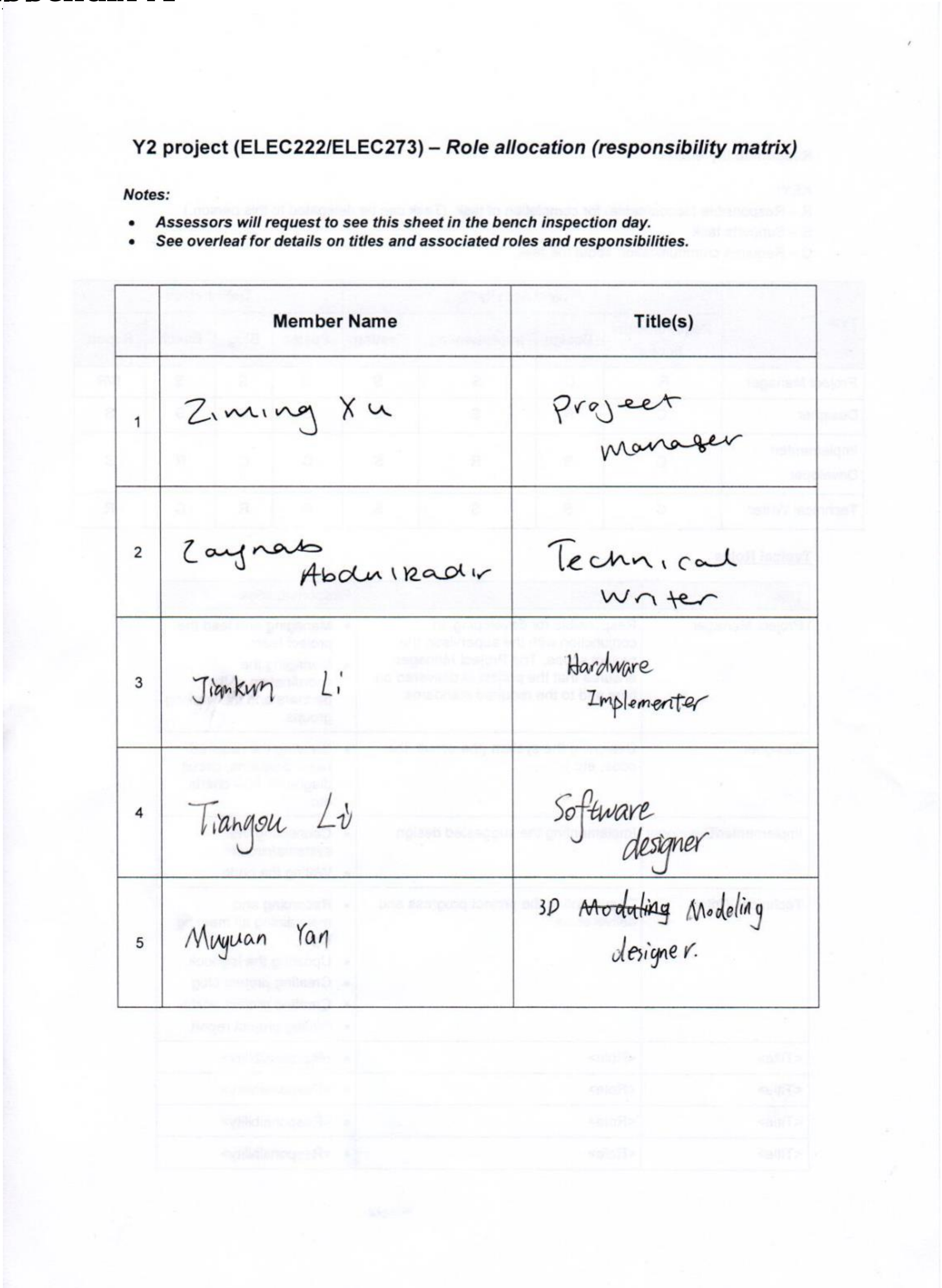
References

[1] B.Earl, “Multi-tasking the Arduino - Part 1,”,” Adafruit Learning System, 03 November 2014. [Online]. [Accessed 13 March 2023].

[2] “What Is Inverse Kinematics?.,” Mathwroks.com, 2023. [Online]. Available: https://uk.mathworks.com/discovery/inverse-kinematics.html. [Accessed 13 March 2023].

[3] Rapidonline.com, “TruOpto OSL30561-IB 14.2mm 3 Digit Display Anode 50MCD,” [Online]. Available: https://static.rapidonline.com/pdf/200091_v1.pdf. [Accessed 13 March 2023].

Appendix A

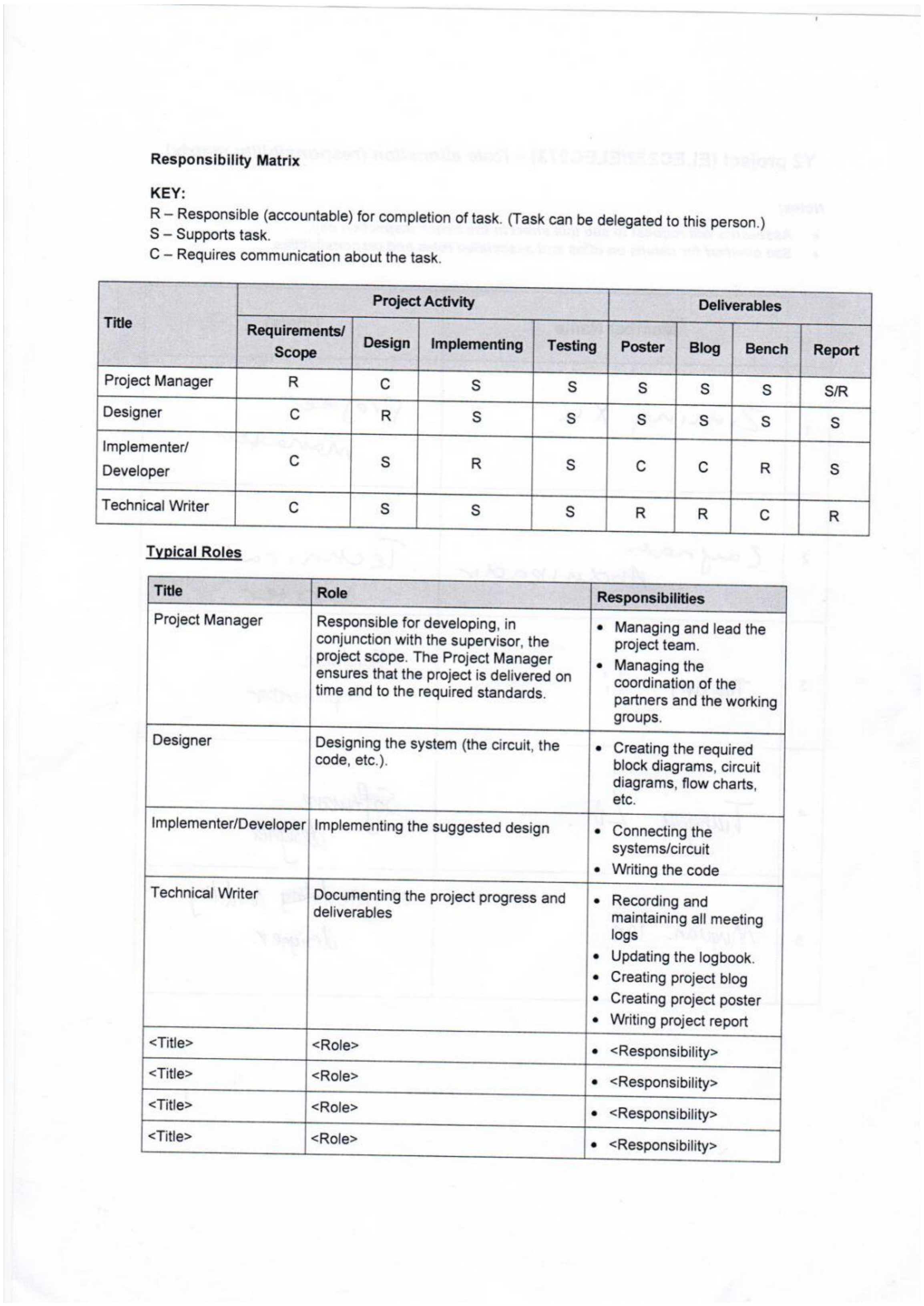


Y2 project (ELEC222/ELEC273) – *Role allocation (responsibility matrix)*

Notes:

- ***Assessors will request to see this sheet in the bench inspection day.***
- ***See overleaf for details on titles and associated roles and responsibilities.***

	Member Name	Title(s)
1	Ziming Xu	Project manager
2	Zaynab Abdulkadir	Technical Writer
3	Jiankun Li	Hardware Implementer
4	Tiangou Li	Software designer
5	Muyuan Yan	3D ~~Moduling~~ Modeling designer.



Responsibility Matrix

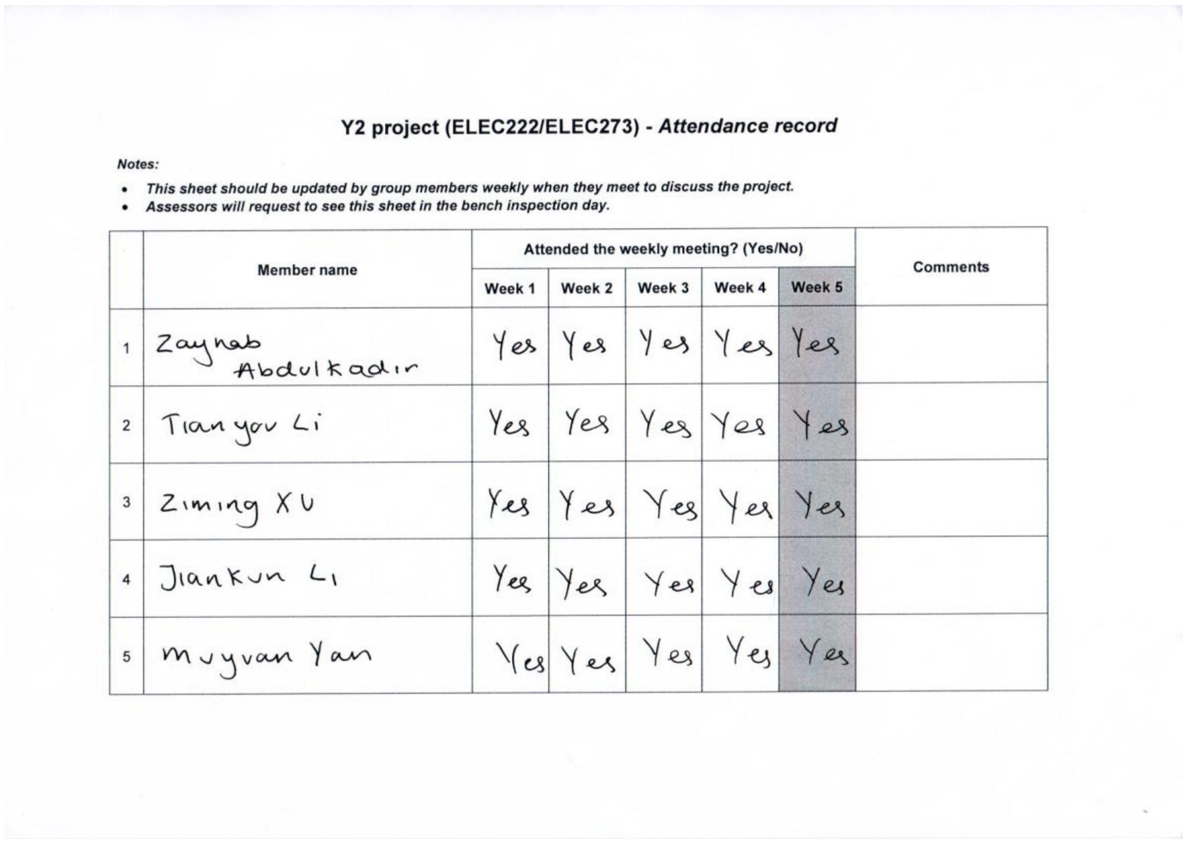
KEY:

R – Responsible (accountable) for completion of task. (Task can be delegated to this person.)
S – Supports task.
C – Requires communication about the task.

Title	**Project Activity**				**Deliverables**			
	Requirements/ Scope	**Design**	**Implementing**	**Testing**	**Poster**	**Blog**	**Bench**	**Report**
Project Manager	R	C	S	S	S	S	S	S/R
Designer	C	R	S	S	S	S	S	S
Implementer/ Developer	C	S	R	S	C	C	R	S
Technical Writer	C	S	S	S	R	R	C	R

Typical Roles

Title	**Role**	**Responsibilities**
Project Manager	Responsible for developing, in conjunction with the supervisor, the project scope. The Project Manager ensures that the project is delivered on time and to the required standards.	• Managing and lead the project team. • Managing the coordination of the partners and the working groups.
Designer	Designing the system (the circuit, the code, etc.).	• Creating the required block diagrams, circuit diagrams, flow charts, etc.
Implementer/Developer	Implementing the suggested design	• Connecting the systems/circuit • Writing the code
Technical Writer	Documenting the project progress and deliverables	• Recording and maintaining all meeting logs • Updating the logbook. • Creating project blog • Creating project poster • Writing project report
<Title>	<Role>	• <Responsibility>
<Title>	<Role>	• <Responsibility>
<Title>	<Role>	• <Responsibility>
<Title>	<Role>	• <Responsibility>

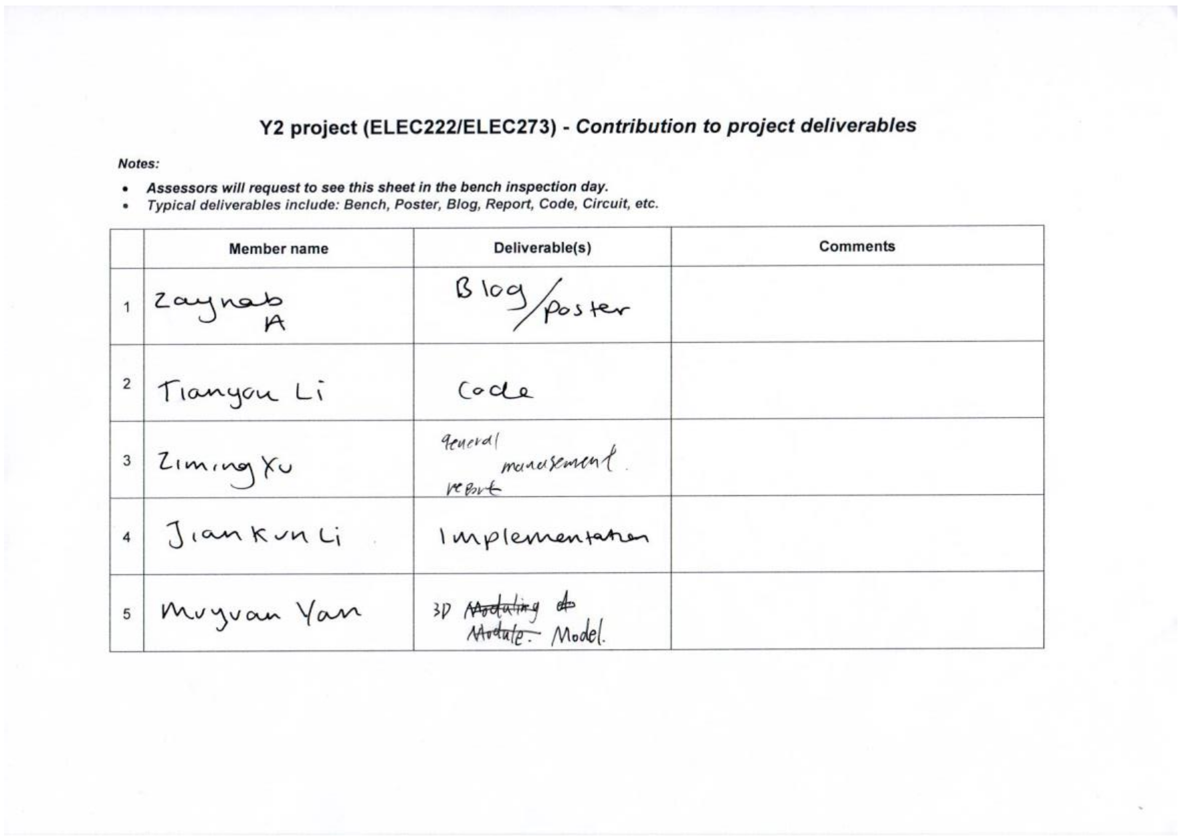


Y2 project (ELEC222/ELEC273) - *Attendance record*

Notes:

- ***This sheet should be updated by group members weekly when they meet to discuss the project.***
- ***Assessors will request to see this sheet in the bench inspection day.***

	Member name	Attended the weekly meeting? (Yes/No)					Comments
		Week 1	Week 2	Week 3	Week 4	Week 5	
1	Zaynab Abdulkadir	Yes	Yes	Yes	Yes	Yes	
2	Tianyou Li	Yes	Yes	Yes	Yes	Yes	
3	Ziming XU	Yes	Yes	Yes	Yes	Yes	
4	Jiankun Li	Yes	Yes	Yes	Yes	Yes	
5	Muyuan Yan	Yes	Yes	Yes	Yes	Yes	

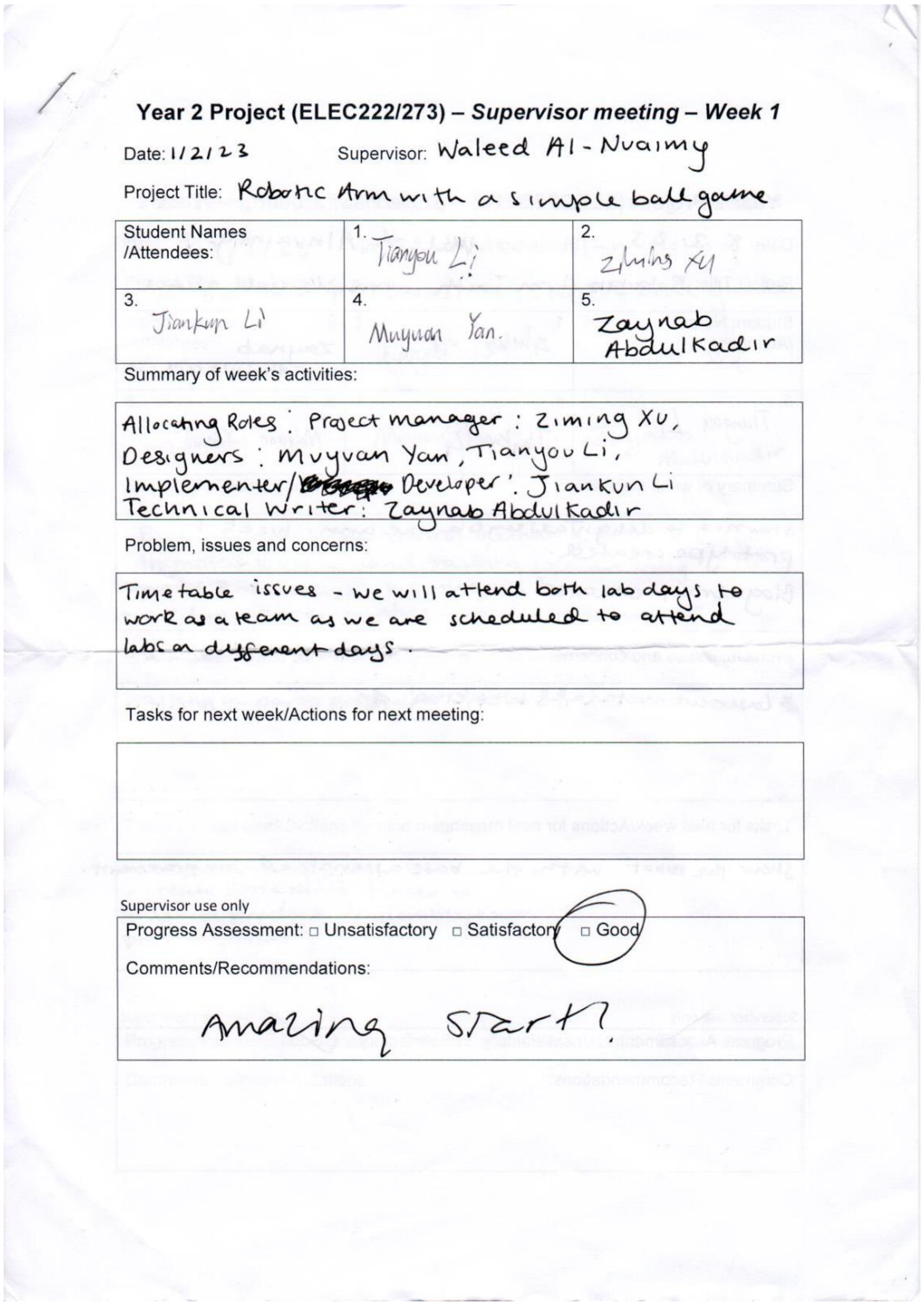


Y2 project (ELEC222/ELEC273) - *Contribution to project deliverables*

Notes:

- ***Assessors will request to see this sheet in the bench inspection day.***
- *Typical deliverables include: Bench, Poster, Blog, Report, Code, Circuit, etc.*

	Member name	Deliverable(s)	Comments
1	Zaynab A	Blog/poster	
2	Tianyou Li	Code	
3	Ziming Xu	general management report	
4	Jiankun Li	Implementation	
5	Muyuan Yan	3D ~~Modeling~~ ~~do~~ ~~Module.~~ Model.	



Year 2 Project (ELEC222/273) – *Supervisor meeting – Week 1*

Date: 1/2/23 Supervisor: Waleed Al-Nuaimy

Project Title: Robotic Arm with a simple ball game

Student Names /Attendees:	1. Tianyou Li	2. [illegible]
3. Jiankun Li	4. Muyuan Yan.	5. Zaynab AbdulKadir

Summary of week's activities:

Allocating Roles: Project manager: Ziming XU,
Designers: Muyuan Yan, Tianyou Li,
Implementer/[illegible] Developer: Jiankun Li
Technical Writer: Zaynab AbdulKadir

Problem, issues and concerns:

Timetable issues - we will attend both lab days to work as a team as we are scheduled to attend labs on different days.

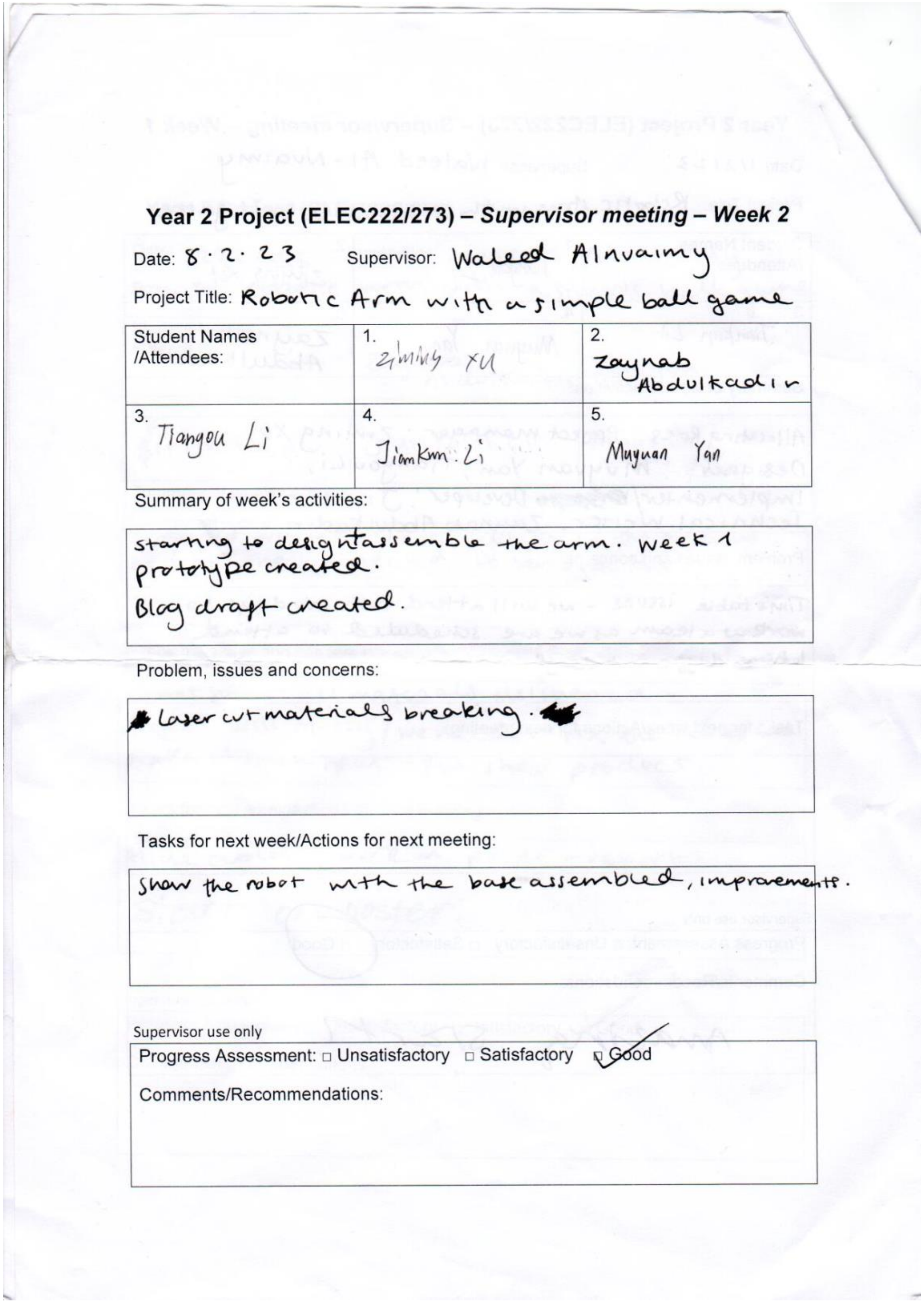
Tasks for next week/Actions for next meeting:

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory [✓ □ Good]

Comments/Recommendations:

Amazing start!



Year 2 Project (ELEC222/273) – *Supervisor meeting* – Week 2

Date: 8 · 2 · 23 Supervisor: Waleed Alnuaimy

Project Title: Robotic Arm with a simple ball game

Student Names /Attendees:	1. Ziming XU	2. Zaynab Abdulkadir
3. Tiangou Li	4. Jiankun Li	5. Muyuan Yan

Summary of week's activities:

starting to design+assemble the arm. week 1 prototype created.
Blog draft created.

Problem, issues and concerns:

Laser cut materials breaking.

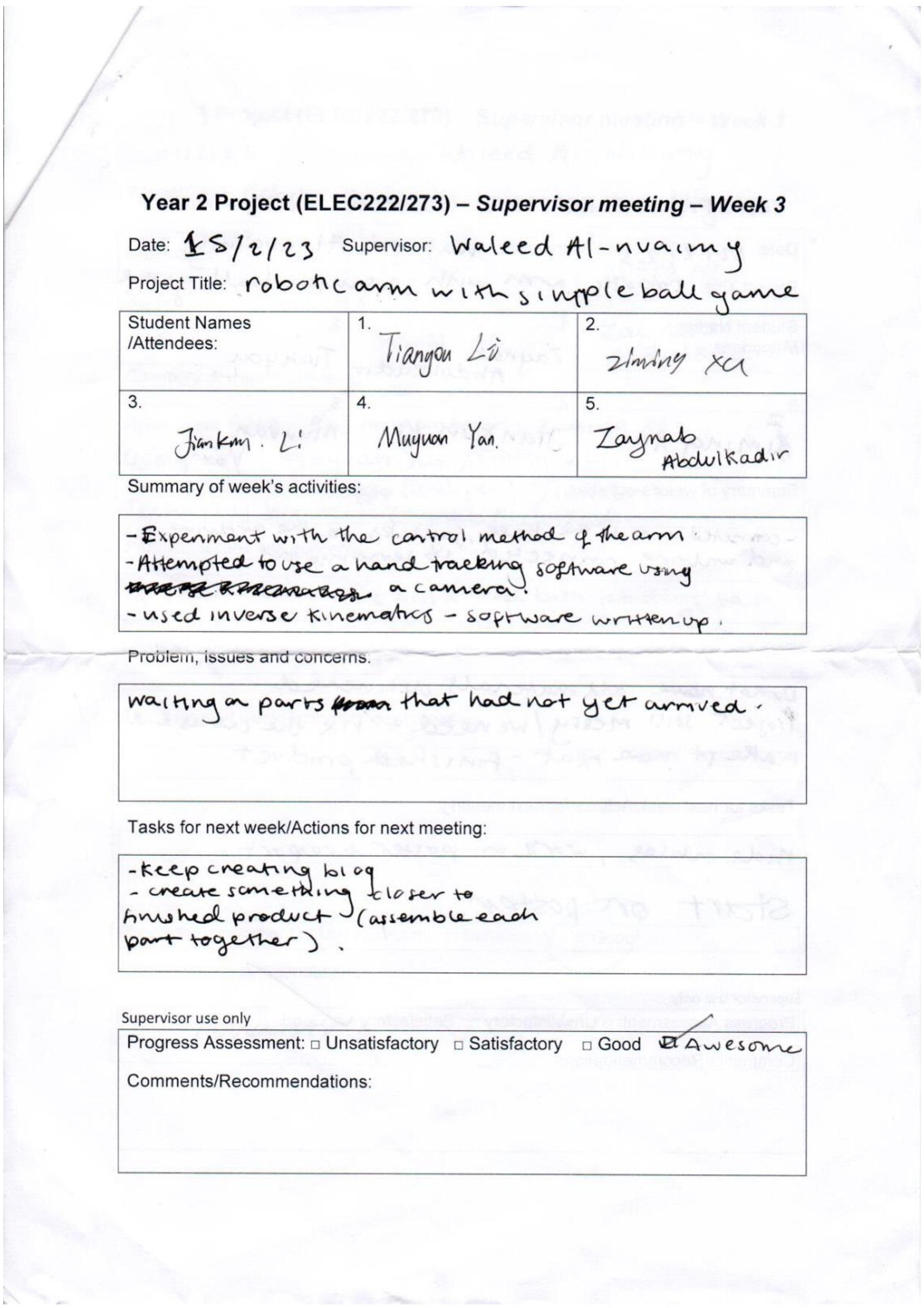
Tasks for next week/Actions for next meeting:

show the robot with the base assembled, improvements.

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory [✓ Good]

Comments/Recommendations:



Year 2 Project (ELEC222/273) – *Supervisor meeting – Week 3*

Date: 15/2/23 Supervisor: Waleed Al-nuaimy

Project Title: robotic arm with simple ball game

Student Names /Attendees:	1. Tiangou Li	2. Ziming Xu
3. Jiankun. Li	4. Muyuan Yan.	5. Zaynab Abdulkadir

Summary of week's activities:

- Experiment with the control method of the arm
- Attempted to use a hand tracking software using a camera.
- used inverse kinematics - software written up.

Problem, issues and concerns:

waiting on parts that had not yet arrived.

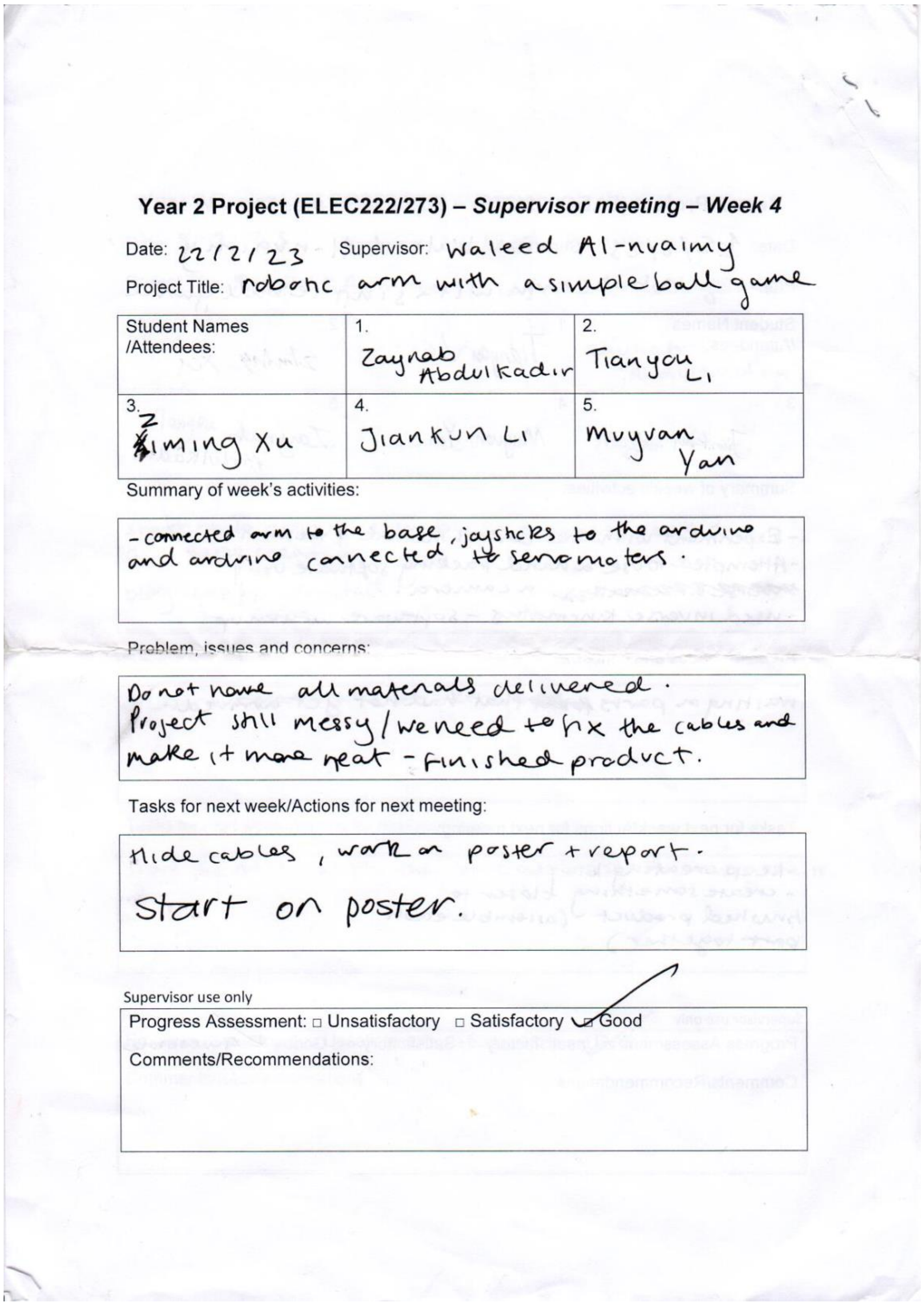
Tasks for next week/Actions for next meeting:

- keep creating blog
- create something closer to finished product (assemble each part together).

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory □ Good ☑ Awesome

Comments/Recommendations:


Year 2 Project (ELEC222/273) – *Supervisor meeting – Week 4*

Date: 22/2/23 Supervisor: Waleed Al-nuaimy

Project Title: robotic arm with a simple ball game

Student Names /Attendees:	1. Zaynab Abdulkadir	2. Tianyou Li
3. Ziming Xu	4. Jiankun Li	5. Muyuan Yan

Summary of week's activities:

- connected arm to the base, joysticks to the arduino and arduino connected to servo motors.

Problem, issues and concerns:

Do not have all materials delivered.
Project still messy / we need to fix the cables and make it more neat - finished product.

Tasks for next week/Actions for next meeting:

Hide cables, work on poster + report.

Start on poster.

Supervisor use only

Progress Assessment: □ Unsatisfactory □ Satisfactory [✓ Good]

Comments/Recommendations:

Appendix B

Individual contribution breakdown:

Name	Breakdown	Percentage of Total Contribution
Ziming Xu (Project Management)	Mechanical design (80%) Project assembly (33%) Project Report (20%) Testing (30%) Poster (10%)	20%
Muyuan Yan (3D Modelling)	3D Modelling (100%) Project Assembly (33%) Project Report (20%)	20%
Tianyou Li (Software Design)	Code Design (80%) Mechanical design (20%) Testing (70%) Project Report (20%) Poster (10%)	20%
Jiankun Li (Implementation)	Hardware Implementation (60%) Project Assembly (33%) Project Report (20%) Poster (10%)	20%
Zaynab Abdulkadir (Technical Writer)	Blog (100%) Project Report (20%) Poster (70%) Document Management (100%)	20%

Appendix C

Arm Source Code (C.1) and Game Source Code (C.2)

C.1 Arm Source Code

```
#include <Servo.h>


// define servos

Servo Servo_S;

Servo Servo_E;

Servo Servo_W;

Servo Servo_C;

Servo Servo_B;




int Controller_1_X = A0;  // controlls Cord_X

int Controller_1_Y = A1;  // controlls Cord_Y

int Controller_2_X = A2;  // controlls base rotation

int Controller_2_Y = A3;  // controlls wrist

int Controller_3_X = A4;  // controlls claw



// define variables for current position

double CP_1_X;

double CP_1_Y;

double CP_2_X;

double CP_2_Y;

double CP_3_X;
```

```
// define servo pins on Arduino
int Servo_S_pin = A8;
int Servo_E_pin = A9;
int Servo_W_pin = A7;
int Servo_C_pin = A6;
int Servo_B_pin = A5;


// length for shoulder and elbow used in calculation
double SL = 12.5; //cm
double EL = 9.5;  //cm


// angle for five servos
double SA;
double EA;
double WA;
double CA;
double BA;


// location of wrist used in calculation
double Cord_X;   // cm
double Cord_Y;  // cm
double Cord_move = 0.1;   //ammount to move with each cycle in cm


//math variables
double Torad = PI/180;  // change degree to rad
double Derad = 180/PI;  // change rad to degree
double Hyp; // hypotenuse
double AS_Part1;  // shoulder angle 1
double AS_Part2;  // shoulder angle 2


void setup() {
  Serial.begin (9600);
  // attach all the servos to pins
  Servo_S.attach (Servo_S_pin);
  Servo_E.attach (Servo_E_pin);
  Servo_W.attach (Servo_W_pin);
  Servo_C.attach (Servo_C_pin);
  Servo_B.attach (Servo_B_pin);


  pinMode (Controller_1_X,INPUT);
  pinMode (Controller_1_Y,INPUT);
  pinMode (Controller_2_X,INPUT);
  pinMode (Controller_2_Y,INPUT);
  pinMode (Controller_3_X,INPUT);


  Cord_X = EL;
  Cord_Y = SL;
  CA = 150; // initial state for claw
  BA = 130; // initial state for base
}


void loop() {
```

```
// read the inputs from joysticks
  CP_1_X = analogRead (Controller_1_X); // A0 A1
  CP_1_Y = analogRead (Controller_1_Y);
  CP_2_X = analogRead (Controller_2_X); // A2 A3
  CP_2_Y = analogRead (Controller_2_Y);
  CP_3_X = analogRead (Controller_3_X);


  if (CP_1_X < 300){
    if (Cord_X > 2){
      Cord_X = Cord_X - Cord_move;}}


  if (CP_1_X > 700){
    if (Cord_X < 15){
      Cord_X = Cord_X + Cord_move;}}


  if (CP_1_Y < 300){
    if (Cord_Y > 1){
      Cord_Y = Cord_Y - Cord_move;}}


  if (CP_1_Y > 700){
    if (Cord_Y < 15){
      Cord_Y = Cord_Y + Cord_move;}}


  if (CP_2_Y > 700){
    if (WA < 170){
      WA = WA + 1;}}
```

```
if(CP_2_Y < 300){
  if (WA > 10){
    WA = WA - 1;}}


if (CP_2_X > 750){
  if (BA < 150){
    BA = BA + 1;}}


if(CP_2_X < 250){
  if (BA > 10){
    BA = BA - 1;}}


if (CP_3_X > 700){
    CA = 110;}


if(CP_3_X < 300){
    CA = 160;}

// calculate the angles
Hyp = sqrt((Cord_X * Cord_X) + (Cord_Y * Cord_Y));
EA =  180-(acos(((SL*SL)+(EL*EL)-(Hyp*Hyp))/(2*SL*EL)) * Derad);
AS_Part1 = acos(Cord_X/Hyp) * Derad;
AS_Part2 = acos(((SL*SL)+(Hyp*Hyp)-(EL*EL))/(2*SL*Hyp)) * Derad;
SA = 180-(AS_Part1 + AS_Part2);
```

```
  Servo_S.write(SA);

  Servo_E.write(EA);

  Servo_W.write(WA);

  Servo_C.write(CA);

  Servo_B.write(BA);


  delay(10); // smooth movement adjustment parameter

}
```

C.2 Game Source Code

```
// define variables for IR sensor

int IRpin = 2;

int flag = 0;

int prevFlag = 0;



// numbers for counting scores

int num = 0;



// define pins for 7 segment display

int a = 13;

int b = 12;

int c = 11;

int d = 10;

int e = 9;

int f = 8;

int g = 7;
```

```
// use to determine what num should be display
int LED;


void setup() {
  Serial.begin(9600);
  // prepare the pins mode
  pinMode(a, OUTPUT);      pinMode(b, OUTPUT);     pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);      pinMode(e, OUTPUT);     pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
}


void loop() {
  // read the output of IR sensor
  flag = digitalRead(IRpin);


  if (flag != prevFlag) { // flag changed
      // reset the num
      if (num >= 10){
        num = 0;
      }
      num++;
      prevFlag = flag; // update previous flag
      Serial.println(num);
  }


  delay(100);
```

```
LED = (num+1)/2; // two changing edges for one ball


// define the LED patterns
if (LED == 0){
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, LOW);
}


if (LED == 1){
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
}


if (LED == 2){
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, LOW);
}


if (LED == 3){
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, LOW);
}


if (LED == 4){
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
}


if (LED == 5){
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
}


if (LED == 6){
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
}


if (LED == 7){
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
```

```
  if (LED == 8){
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
  }


  if (LED == 9){
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, HIGH);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
  }
}
```