SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Prize Winner

Programming, Apps & Robotics

Year 5-6

Nivaan Sardana

St Peter's College

Mimic Articulated Robot Arm

What is an Articulated Robot:

An articulated robot is a robot with rotary joints that are built using several motors to allow movement or rotation in different directions. Most robot arms are usually wired or are remote controlled robots, but mine is different, as I have explained below.

Where the idea came from:

A few months ago my mother had to undergo a surgery and I wondered what if the doctor was in a different city and could the doctor somehow operate on her from far away. So, I searched it and I found that it was possible with the help of robot arms. Then I began to observe the use of robotic arms all around me - in industries, car companies, construction and even medical use! I watched several videos on this. My robot arm is a bit different to a usual robot arm because it uses mimicking technology - this means it mimics your hand actions. It is different to other normal robot arms that are programmed to repeat specific tasks.

My Research: Bibliography

I did some research about how to build a robotic arm that was controlled by hand movement and found the below very useful links. I also acknowledge that the project uses several open source information such as STL files for 3D printing and codes and libraries for Arduino components.

- https://www.thingiverse.com/thing:1838120
- https://www.thingiverse.com/thing:1750025
- https://www.thingiverse.com/thing:1748596
- https://smartbuilds.io/diy-robot-arm-arduino-hand-gestures/
- https://www.youtube.com/watch?v=SWEDCZSa3ow
- https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/
- https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/
- https://www.learnrobotics.org/blog/map-potentiometer-servo-arduino/
- https://learn.sparkfun.com/tutorials/basic-servo-control-for-beginners/all
- https://learn.adafruit.com/16-channel-pwm-servo-driver?view=all
- https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/

System and Science explanation

The project consists of two parts- a 3-D printed robotic arm and a hand glove, both with their own Arduino boards that are connected via Bluetooth mechanism. The glove has sensors on it such as accelerometers, gyroscopes and other mechanical instruments which map the direction, movement, location and speed of movement of my hand. This then sends signals to the robotic arm via Bluetooth that make the motors in arm rotate in a particular manner/ direction.

What is a Gyroscope

A gyroscope works by using the piezo effect, i.e. it has a small ball inside of it surrounded by walls in cuboid shape made of electric piezo crystals. Whenever you tilt the sensor, due to the force of gravity the ball has to move towards one of the walls. When the ball collides with a wall it creates miniscule piezo currents. There are three pairs of parallel walls in a cuboid. Each wall has its own axis X, Y and Z. X is left to right, Y is forwards and backwards and Z is up and down. The amount of current generated by the piezo crystals in walls helps to determine the angle of inclination.

How a Flex sensor works

A flex sensor works by using resistance. The more you bend it the more the resistance changes, but when it is at its normal position the resistance is even. So, when the resistance changes, a signal gets triggered, and is sent to the Arduino to change a part of the circuit effectively acting like switch.

What is 6DOF

DOF is short form for degrees of freedom. DOF means how many axes an articulated robot can rotate around. So 6DOF means that a particular object or machine has six degrees of freedom. Just like my robot arm which is not 4DOF but 6DOF which means it could be more complicated than a 4 DOF as it has more degrees of freedom.

What is torque

Torque is the capability of a shaft to make rotation happen. For example, you have a nut and a bolt and you want to get the nut off the bolt with a wrench and you start loosening it by using a lot of force. In this example torque is the amount of force you have to apply for it to start unscrewing.

How servos work

A servo motor is a closed-loop system which integrates positional output to take over the rotational or linear speed and position. It also uses PWM (pulse with modulation) to control the shaft and the direction in which the servo spins based on the width of the pulse signal. For example, a pulse width of 1ms would rotate it clockwise, but a signal width of 2ms would rotate it anticlockwise.

What is payload

Payload is the amount of weight that a machine can carry. Payload is very useful in many large-scale projects not just in robot arms but everything, like rockets. For example scientists need to see the amount of payload rocket can carry so they know how many passengers and things can come aboard it. But for robot arms this is very important because you have to know the maximum weight it can carry as overloading a robot would almost be like destroying it.

What is a planetary gearbox

A planetary gearbox is a complex system of gears which are all driven by the middle gear, the sun gear. They help transfer the rotation movement from one point to another, and also in some cases, can increase or decrease the speed of rotation. The setup is, the sun gear in the middle then 3 normal gears surrounding it which rotate a gear track, thus making the gear track move and also move whatever the gear track is connected to.

How does Bluetooth work

Bluetooth is a way in which different devices communicate with each other wirelessly at super high frequencies and radio waves. These waves are electromagnetic and travel with frequencies around 2.4 gigahertz or 2.4 billion waves per second. Bluetooth is used in many ways and in many things such as, GPSs and sending things wirelessly from one device to another.

How do wires work

Wires are used in everyday objects. In fact, our whole house works on them because electricity that we have is transmitted through wires. Wires work by using small strips of highly conductive material usually wrapped inside a flexible rubber tube. The super conductive material passes the electricity through it as it is conductive. It is wrapped with rubber so the user doesn't get hurt or get electrocuted.

Building it and Challenges

Although I could buy an arm from the market and program it, I decided to 3D print the parts. I used this project as a good opportunity to transition to build my own components with a 3D printer, and then assemble and program them. As I do not know how to 3D model a complex arm yet, I went to **Thingiverse** to select the right model and print it. Assembling and programming were full of several challenges:

- Locating the right components
- 3D printing
- Assembly
- Programming
- Batteries

How I overcame the challenges

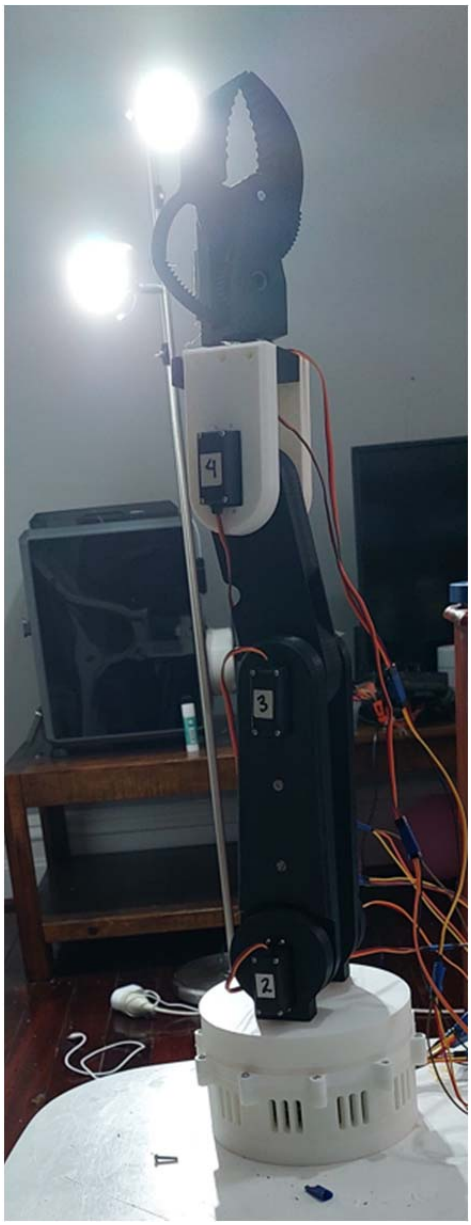
Components
The components were the most difficult part of this project as most of the things I needed were out of stock or were not readily available in Australia. After days of searching Amazon, electronic shops and Bunnings with my mom, I was able to collect most of the materials. Another major issue was that I just couldn't find the right type of battery to power the motors because either it wasn't enough power or it would be very powerful and could destroy my robot. After searching the web with the help of my mom I could finally find the batteries that were needed. Even though I found the parts I still had to assemble them, which was also not easy.

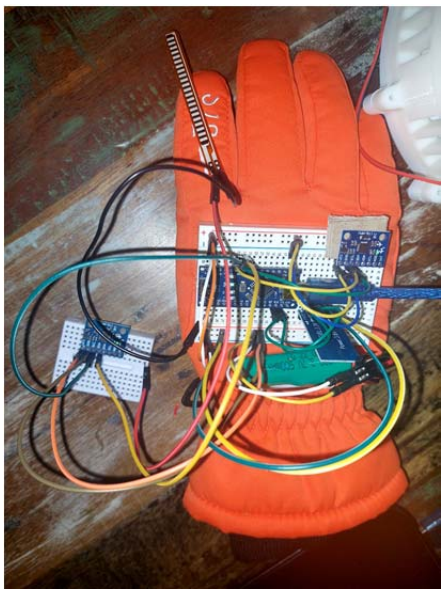
3D printing
3D printing was also difficult. I had recently got 3D printer which meant I had no idea on how to use it. After a few hours of struggling, I finally got it working. But there were a few problems that I had to face. The printer kept on printing in mid-air somehow and will not print the component as I had hoped. This happened many times so every time I fiddled with some settings, and it started working again. Finally, after a lot of effort and trying many times, I had printed all the components required for my robot.

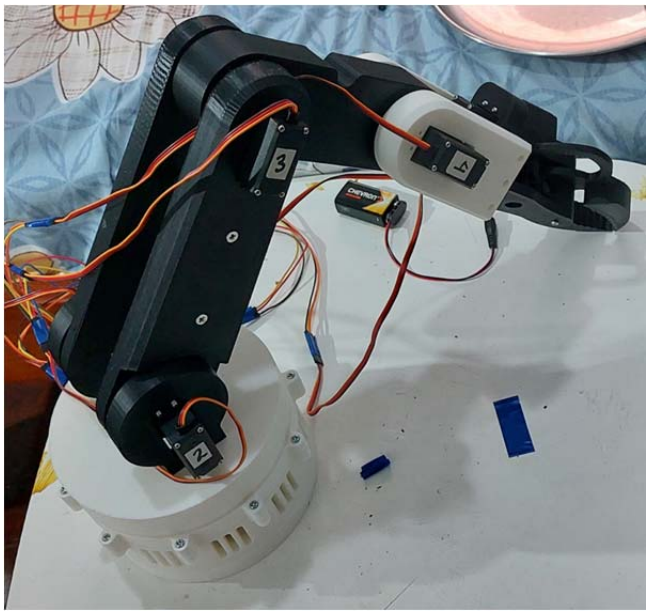
Assembly
The assembly was very hard as lots of the wires kept on disconnecting, so I had to tape most of them. But the main challenge was the screws as some screw holes were too small or too big for the fit. I realized that 3-D printer does not do such great job yet and it becomes very difficult to screw. I had to take help of my parents to screw some parts as they had more strength to drive parts in. I also had to superglue some parts because they kept disconnecting.

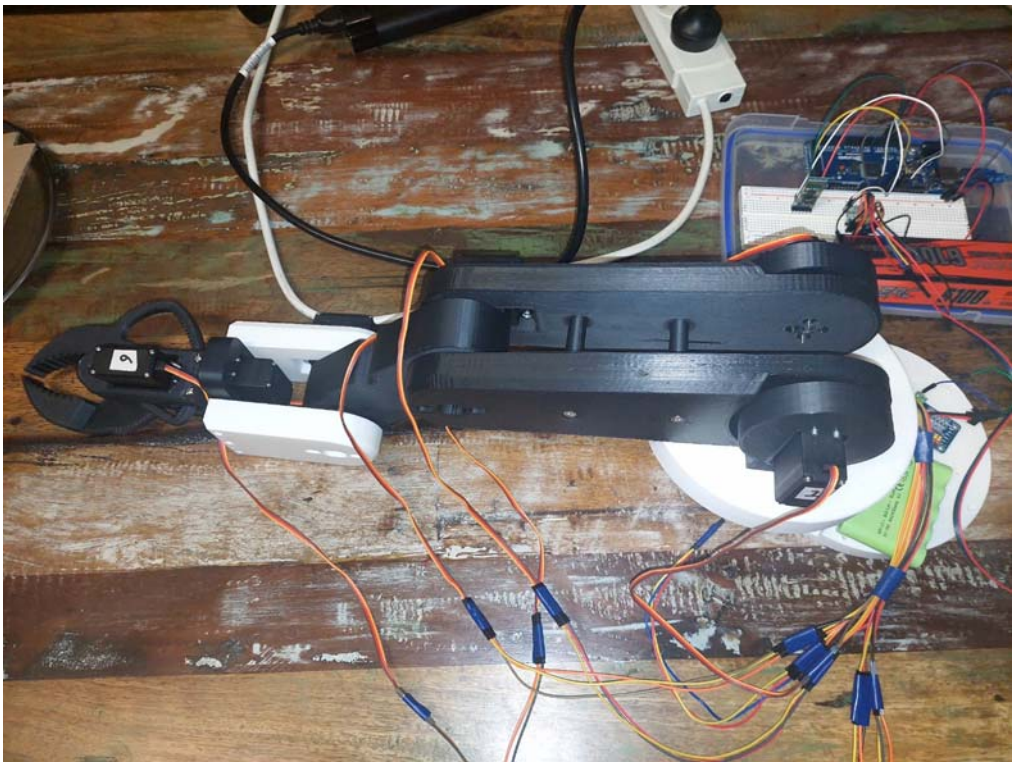
Programming

The program had 2 tricky bits but the first was the hardest. The first tricky part was to get the 2 Bluetooth modules to communicate and connect with each other. This part should've taken me 5minutes but because I was still learning how to get them to connect, and it was my first time with Bluetooth modules it took me a whole day. The second part was to get the motors working, but because I was using such powerful motors, I could not find batteries that would provide suitable amount of current or amperes to drive them. It also became tricky to get them to move in sync and finding out where to connect them.









Applications of mimic robot arm

- Robot arms would have lots of applications in many industries across the globe. Robot arms could be used for going into small crevasses and dangerous places where humans or other animals can't go to without the risk of losing a life.
- Robot arms can also be used for helping elderly people or disabled people doing daily tasks which otherwise would hurt them. For example an elderly person will like to make coffee but then is unable to because he/she could hurt herself/himself so she uses the robot arm to make coffee.
- Industrial robot arms on the other hand could be used in construction sites to stop workers from having back problems later on by using it's strength to carry heavy objects for them.
- It could also be used for mining because normally miners would get lung damage from large amounts of smoke but if the controller stood outside they could autonomously control the robot to mine for them.
- This same technology could be made smaller so it can enter our bodies for medical purposes say for example you have a blood clot in a region of your body which could become deadly, the doctor would make you drink a nano robot arm so it will enter your vein and clear the blood clot and remove it.
- Bigger versions could also be used for surgical uses as the doctor could be in a different place remotely control the robot arm and make the robot arm operate on you instead.

Steps ahead

This robot is just a prototype and like any experiment, it can be developed further a lot. Cameras can be added so it can have its own AI or the controller can see where it is going. Its remote range can be enhanced with powerful communication devices, so it could be truly long-distance and go on rescue operations and missions. It would be useful to add wheels to allow mobility.

Program and code pics

The code is split into 2 important parts- the master and the slave. The master is the glove which sends the signals to the robot, while the operator is the robot arm which receives the signals and does what it is told to do. The code works by the master sending all the outputs of the gyros and flex sensors to the operator where it sees what has been activated and what has changed. Then, if the outputs are different to what it was sent before, it maps the changes to motors creating changes in their angles and direction.

Like a true scientist, I will keep developing the program code further so that it is more efficient and mimics hand movements more accurately.

Glove Code:


```
//Glove Master Code for Gyro and Flex sensor data transferred to slave arm via bluetooth.
// Compiled by Nivaan Sardana using open source codes for separate components

#include <Wire.h>
#include <MPU6050.h>
#include <SoftwareSerial.h>

MPU6050 mpu1;
MPU6050 mpu2;
SoftwareSerial bluetooth(10, 11); // RX, TX
const int MPU6050_CLOCK_PLL_XGYRO;
const int MPU6050_GYRO_FS_250;
const int MPU6050_ACCEL_FS_2;
const int flexPin = A0;

void setup() {
  Wire.begin();
  Serial.begin(9600);
  bluetooth.begin(9600);

  mpu1.setClockSource(MPU6050_CLOCK_PLL_XGYRO);
  mpu1.setFullScaleGyroRange(MPU6050_GYRO_FS_250);
  mpu1.setFullScaleAccelRange(MPU6050_ACCEL_FS_2);

  mpu2.setClockSource(MPU6050_CLOCK_PLL_XGYRO);
  mpu2.setFullScaleGyroRange(MPU6050_GYRO_FS_250);
  mpu2.setFullScaleAccelRange(MPU6050_ACCEL_FS_2);
}
```

```
void loop() {
  int flexValue = analogRead(flexPin);

  int16_t gyroX1, gyroY1, gyroZ1;
  int16_t accelX1, accelY1, accelZ1;

  int16_t gyroX2, gyroY2, gyroZ2;
  int16_t accelX2, accelY2, accelZ2;

  mpu1.getMotion6(&accelX1, &accelY1, &accelZ1, &gyroX1, &gyroY1, &gyroZ1);
  mpu2.getMotion6(&accelX2, &accelY2, &accelZ2, &gyroX2, &gyroY2, &gyroZ2);

  // Sending data via Bluetooth
  bluetooth.print("Flex: ");
  bluetooth.print(flexValue);
  bluetooth.print(" MPU1 - AccelX: ");
  bluetooth.print(accelX1);
  bluetooth.print(" AccelY: ");
  bluetooth.print(accelY1);
  bluetooth.print(" AccelZ: ");
  bluetooth.print(accelZ1);
  bluetooth.print(" GyroX: ");
  bluetooth.print(gyroX1);
  bluetooth.print(" GyroY: ");
  bluetooth.print(gyroY1);
  bluetooth.print(" GyroZ: ");
  bluetooth.print(gyroZ1);
  bluetooth.print(" MPU2 - AccelX: ");
  bluetooth.print(accelX2);
  bluetooth.print(" AccelY: ");
  bluetooth.print(accelY2);
```

```
  bluetooth.print(" AccelZ: ");
  bluetooth.print(accelZ2);
  bluetooth.print(" GyroX: ");
  bluetooth.print(gyroX2);
  bluetooth.print(" GyroY: ");
  bluetooth.print(gyroY2);
  bluetooth.print(" GyroZ: ");
  bluetooth.println(gyroZ2);

  delayMicroseconds(1000); // Delay for 1 second before sending the next data
}
```

Servo and Gyro Code


```
#include <Wire.h>
#include <Servo.h>

#define MPU6050_ADDR 0x68 // I2C address of the MPU6050

Servo servo1;
Servo servo2;

int servo1Pin = 10; // Connect servo1 to digital pin 3
int servo2Pin = 9; // Connect servo2 to digital pin 5

int servo1Angle = 90; // Initial angle for servo1
int servo2Angle = 90; // Initial angle for servo2

int servo1MinAngle = 0;   // Minimum angle for servo1
int servo1MaxAngle = 180; // Maximum angle for servo1
int servo2MinAngle = 0;   // Minimum angle for servo2
int servo2MaxAngle = 180; // Maximum angle for servo2

int accX, accY, accZ;

void setup() {
  Wire.begin();

servo1.attach(servo1Pin);
  servo1.write(servo1Angle);

servo2.attach(servo2Pin);
  servo2.write(servo2Angle);

  // Wait for the servo to reach the start angle
```

```
  // Initialize MPU6050
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0);    // Wake up the MPU6050 (0 = out of sleep mode)
  Wire.endTransmission();

  servo1.attach(servo1Pin);
  servo2.attach(servo2Pin);

  servo1.write(servo1Angle);
  servo2.write(servo2Angle);

  delay(300); // Delay for servo initialization
}

void loop() {
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x3B); // Starting register for accelerometer data
  Wire.endTransmission(false);

  Wire.requestFrom(MPU6050_ADDR, 6, true); // Read 6 bytes

  accX = Wire.read() << 8 | Wire.read();
  accY = Wire.read() << 8 | Wire.read();
  accZ = Wire.read() << 8 | Wire.read();

  // Map the accelerometer data to servo angles
  servo1Angle = map(accY, -16384, 16384, servo1MinAngle, servo1MaxAngle);
  servo2Angle = map(accY, -16384, 16384, servo2MinAngle, servo2MaxAngle);

  // Constrain the servo angles within the allowed range
  servo1Angle = constrain(servo1Angle, servo1MinAngle, servo1MaxAngle);
  servo2Angle = constrain(servo2Angle, servo2MinAngle, servo2MaxAngle);

  // Move the servos
  servo1.write(servo1Angle);
  servo2.write(servo2Angle);

  delay(0);
}
```

Robot Arm Code:


```
/*********************************************************************************
//Articulated Robot Arm Code by Nivaan Sardana
// Using Arduino Mega2560  and including some codes for components available in open source

/* Include the HCPCA9685 library */
#include "HCPCA9685.h"

/* I2C slave address is 0x40 */
#define  I2CAdd 0x40

/* Create an instance of the library */
HCPCA9685 HCPCA9685(I2CAdd);

//initial parking position of the motor
const int base_1_parking_pos = 60;
const int base_2_parking_pos = 60;
const int servo_joint_1_parking_pos = 70;
const int servo_joint_2_parking_pos = 47;
const int servo_joint_3_parking_pos = 63;
const int servo_joint_4_parking_pos = 63;

//Degree of robot servo sensitivity - Intervals
int base_1_pos_increment = 20;
int base_2_pos_increment = 20;
int servo_joint_1_pos_increment = 20;
int servo_joint_2_pos_increment = 50;
int servo_joint_3_pos_increment = 60;
int servo_joint_4_pos_increment = 40;
```

```
int servo_joint_4_pos_increment = 40;

// motor positions
int base_1_parking_pos_i = base_1_parking_pos;
int base_2_parking_pos_i = base_2_parking_pos;
int servo_joint_1_parking_pos_i = servo_joint_1_parking_pos;
int servo_joint_2_parking_pos_i = servo_joint_2_parking_pos;
int servo_joint_3_parking_pos_i = servo_joint_3_parking_pos;
int servo_joint_4_parking_pos_i = servo_joint_4_parking_pos;


//Minimum and maximum angle of servo motor
int base_1_min_pos = 10;
int base_1_max_pos = 180;
int base_2_min_pos = 10;
int base_2_max_pos = 180;
int servo_joint_1_min_pos = 10;
int servo_joint_1_max_pos = 400;
int servo_joint_2_min_pos = 10;
int servo_joint_2_max_pos = 380;
int servo_joint_3_min_pos = 10;
int servo_joint_3_max_pos = 380;
int servo_joint_4_min_pos = 10;
int servo_joint_4_max_pos = 120;
int servo_L_pos = 0;
int servo_R_pos = 0;
int servo_joint_1_pos = 0;
int servo_joint_2_pos = 0;
int servo_joint_3_pos = 0;
int servo_joint_4_pos = 0;
```

```
  delay(3000);
  //wakeUp();
  //flexMotors();

}

void loop() {

  if (Serial.available() > 0) { // Checks whether data is coming from the serial port
    state = Serial.read(); // Reads the data from the serial port
    Serial.print(state); // Prints out the value sent

       //Move (Base Rotation) Stepper Motor Left
    if (state == 'S') {
      baseRotateLeft();
      delay(response_time);
    }

    //Move (Base Rotation) Stepper Motor Right
    if (state == 'O') {
      baseRotateRight();
      delay(response_time);
    }

    //Move Shoulder Down
    if (state == 'c') {
      shoulderServoForward();
      delay(response_time);
    }
```