Final Project Report

106062322 江岷錡

GOAL

利用 Servo Motor 建置四軸（一軸底座馬達、兩軸前臂伸屈、一軸前爪）的機械手臂，並搭配 Joystick 、UltraSonic 進行機械手臂的操控，同時，連接上藍芽模組，讓手機也得以遠端遙控機械手臂。

詳細各零件操控如下：

1. 基礎操作：搭配 JoyStick 操控 ，使其能左右轉動約 180 度距離，且手臂向前及向後伸展。
2. 手控遠近操作：配合超聲波，可進行遠距操控，讓手接近及遠離時調整前爪。
3. 藍芽操控：使用藍芽，遠端操控手臂，進行任何操控。

並會遠端連線到 MSP 430 ，MSP 430 的按鈕得以調整目前馬達轉速，並透過亮紅、綠燈的形式，提醒目前的轉速設定。

COMPONENTS

本次實作使用 Arduino 作為 MotherBoard ，串接 Servo Motor / Joystick / UltraSonic / Bluetooth (HC-06) / MSP430 各類機件，以下將分別描述各自的實作：

Servo Motor

在上方定義時，便先統一化各個常數的設定，以便各參數調用時較為方便。

```
Servo myservo[4];
/// motor
// 0 -> 底座向前
// 1 -> 爪
// 2 -> 上座向前
// 3 -> 底部旋轉
const int servoPort[4] = { 6, 7, 8, 9 };
int currentAngle[4] = { 0 };
const int UPPER_BOUND = 2400;
const int LOWER_BOUND = 500;
int perMovement = 100;
int moveForwardRunningMotor = 0; // 確定向前伸展的馬達，目前呼叫是哪一個
```

而在 **Setup** Function 中，由於 Servo PWM 脈衝範圍，與原先 Arduino 預設的不相同，若未進行脈衝調整會讓轉角小於 180 度。

故此處我們先進行脈衝範圍的修正，並預設每個 Servo Motor 轉角的初始值。

```
myservo[i].attach(servoPort[i], 500, 2400); // 修正脈衝寬度範圍
myservo[i].write(90); // 一開始先置中90度
currentAngle[i]= (UPPER_BOUND + LOWER_BOUND) / 2;
```

接著，模組化對馬達的操控，所有進行馬達正轉與反轉的操作，皆可透過下方定義的 **increaseMotor** 及 **decreaseMotor** Functions 進行馬達的正反轉。

```
void increaseMotor(int motorNum) {
    if (currentAngle[motorNum] + perMovement <= UPPER_BOUND) {
      currentAngle[motorNum] += perMovement;
    } else {
      currentAngle[motorNum] = UPPER_BOUND;
    }
    myservo[motorNum].writeMicroseconds(currentAngle[motorNum]);
}

void decreaseMotor(int motorNum) {
    if (currentAngle[motorNum] - perMovement >= LOWER_BOUND) {
      currentAngle[motorNum] -= perMovement;
    } else {
      currentAngle[motorNum] = LOWER_BOUND;
    }
     myservo[motorNum].writeMicroseconds(currentAngle[motorNum]);
}
```

由於前軸有兩個 Servo Motor ，在操作向前伸展時，必須先行指定馬達才得以進行操作。此處我們也獨立出 Function 供外部呼叫。

```
void moveFrontBack(int pos) {
  // pos: 0 -> forward, 1 -> backward
  // Note: motor 2 need to reverse direction
  bool shouldIncrease = (moveForwardRunningMotor == 0 && pos == 0)
|| (moveForwardRunningMotor == 2 && pos == 1);
  if (shouldIncrease) increaseMotor(moveForwardRunningMotor);
  else decreaseMotor(moveForwardRunningMotor);
}
```

Joystick

在 Joystick Function 中，除了將 x / y value 提取出，也針對 Joystick 壓下按鈕進行 Debounce 的機制設定。

```
void joystickDetect() {
  int xVal = analogRead(xAxis);
  int yVal = analogRead(yAxis);

  if (yVal <= 10) {
```

```
    moveFrontBack(0);
  } else if (yVal >= 1020) {
    moveFrontBack(1);
  }

  if (xVal <= 10) {
    increaseMotor(3);
  } else if (xVal >= 1020) {
    decreaseMotor(3);
  }

  int isPress = digitalRead(joyStickButton);
  if (isPress != joyLastButtonState) {
    joyLastDebounceTime = millis();
  }
  if ((millis() - joyLastDebounceTime) > debounceDelay) {
    if (isPress != joyButtonState) {
      joyButtonState = isPress;
      if (joyButtonState == HIGH) {
        changeMovingForwardMotor();
      }
    }
  }
  joyLastButtonState = isPress;
}
```

UltraSonic

由於當距離無限遠時，所偵測出的 distance 將會呈現不穩定的高值，此處我們限制只有當 distance 在 **AVAILABLE_DISTANCE_LOW(0)** 及 **AVAILABLE_DISTANCE_HIGH(40)** 的區間時，才會針對前爪馬達進行操作。

前爪的放開與收縮的判斷原則，則是以手貼近或遠離感測器作為標準。

當手貼近時，前爪會收縮，而當手遠離時，前爪會放開。

```
void ultraSensor() {
   digitalWrite(trigPin, LOW); // Clears the trigPin
   delayMicroseconds(2);
   /* Sets the trigPin on HIGH state for 10 ms */
   digitalWrite(trigPin, HIGH);    delayMicroseconds(10);
   digitalWrite(trigPin, LOW);
   /* Reads Echo pin, returns sound travel time in ms */
   duration = pulseIn(echoPin, HIGH);
   /* Calculating the distance */
```

```
   distance = duration*0.034/2;

   if (distance >= AVAILABLE_DISTANCE_LOW && distance <=
AVAILABLE_DISTANCE_HIGH) {
    // Only distance when it's reasonable
    if (lastDistance - distance < 0) {
      // which means become closer
      decreaseMotor(1);
    } else {
      // which means become farer
      increaseMotor(1);
    }
    lastDistance = distance;
   }
}
```

Bluetooth

本次藍芽模組是使用 HC-06 ，並搭配 Android 手機 Arduino Bluetooth APP 進行整體操作。簡單而言，HC-06 與 Arduino 溝通仍是使用 SoftwareSerial UART Interface ，在使用前必須先進入 AT Mode ，設定基本參數後，才得以使用。

HC-06 我們設定 Baudrate 為 9600。

而當藍芽模組收到參數後，會再進行馬達的操控。

```
const int RX_PIN = 10;
const int TX_PIN = 11;
SoftwareSerial BTSerial(RX_PIN, TX_PIN); // RX | TX

BTSerial.begin(9600);

void bluetoothControl() {
  // Keep reading from HC-06 and send to Arduino Serial Monitor
  if (BTSerial.available()) {
    int value = BTSerial.read();
    switch(value) {
      case 49:
        // left
        increaseMotor(3);
        break;
      case 50:
        // up
        moveFrontBack(0);
        break;
      case 51:
```

```
        // right
        decreaseMotor(3);
        break;
      case 52:
        // down
        moveFrontBack(1);
        break;
      case 53:
        increaseMotor(1);
        break;
      case 54:
        changeMovingForwardMotor();
        break;
      case 55:
        decreaseMotor(1);
        break;
      case 56:
        break;
    }
  }
}
```

MSP430

MSP430 我們用以操控馬達轉速。

當 MSP430 按鈕按下時，會傳送訊號至 Arduino ，我們便可以進行馬達轉速的調整，並依據目前的轉速傳送訊號給 MSP430 應顯示的狀態。

而詳細的 MSP430 code 也有附載在繳交區中。

```
  if (mspSerial.available()) {
    char serialData = mspSerial.read();
    Serial.print(serialData);
    if (serialData == 'b') {
      perMovement = (perMovement == 100) ? 200 : 100;
      mspSerial.print("o");
      mspSerial.println();

      if (perMovement == 100) {
        mspSerial.print("c");
        mspSerial.println();
      } else {
        mspSerial.print("w");
        mspSerial.println();
```

```
      }
    }
  }
```

DIFFICULTIES

1. Software Serial 同時只能監聽一項裝置

在串接 MSP430 與 Bluetooth 時，發現兩者裝置不得同時並存，只能同步接收到其一來源的訊號。經過查詢，發現 SoftwareSerial 有限制同時僅能接收一個裝置的來源訊號。解決方式則是新增一個按鈕，切換目前的訊號來源。

```
void serialButtonDetect() {
  int reading = digitalRead(buttonPins[1]);
  if (reading != lastButtonStates[1]) {
    lastDebounceTimes[1] = millis();
  }
  if ((millis() - lastDebounceTimes[1]) > debounceDelay) {
    if (reading != buttonStates[1]) {
      buttonStates[1] = reading;

      if (buttonStates[1]) {
        if (isBluetoothOpen) {
          mspSerial.listen();
          isBluetoothOpen = false;
        } else {
          BTSerial.listen();
          isBluetoothOpen = true;
        }
      }
    }
  }
  lastButtonStates[1] = reading;
  analogWrite(ledPins[2], isBluetoothOpen ? 100 : 0);
}
```

2. Arduino 供電不足

由於本次串接 4 顆 Servo Motor，與其餘 Sensors ，原先皆透過單一 5V 腳進行供電，但整體驅動的電力極為不足。後期我們便調整為，4 顆 Servo Motor 透過外接電源模組進行供電，其餘透過 Arduino 供電。

3. HC-06 藍芽模組

由於 HC-06 需要透過 UART 進行傳輸，也必須進入 AT Mode 操作部分參數，以便調用其 Sensor 服務。同時， iPhone 不能直接連接到藍芽模組，因此在藍芽模組的串接上便花費不少心力。

PROGRAM FLOW CHART