Projektdokumentation: Roboterarm

Grundlagen der Informatik II SOSE 2017

Jisoo Choi (3103933), Michael Vogt (4391254), Fany Bowt (4345894)

23.06.2017

Inhaltsverzeichnis

1 Beschreibung des Projekts und Funktionsweise **2**

2 Aufbau **2**

3 Arduino/C Code-Beschreibung **3**

3.1 WLAN-Empfang 4

4 Python Code und GUI **5**

1 Beschreibung des Projekts und Funktionsweise

Ein Arduino sollte anfänglich 4 Mikro Servomotoren steuern, dabei befindet sich ein Servomotor bei der Konstruktion am unteren Teil des Greifarms, dieser lässt den Arm nach rechts oder links lenken. Ein weiterer Servomotor befindet sich in der Mitte des Armes, dieser lässt den mittleren Teils des Armes nach oben und unten steuern. Zwei Servomotoren befinden sich im oberen Teil der Konstruktion, einer der Servomotoren platziert am Greifer, um den Greifer nach oben oder unten zu bewegen und ein weiterer um die Zange zu öffnen und zu schließen.

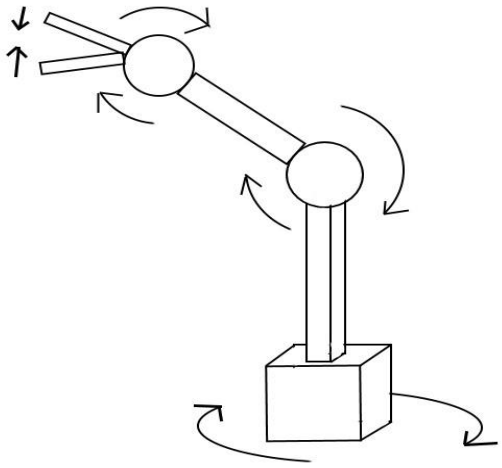

Abbildung 1: Skizze vom Arm

Über die grafische Benutzeroberfläche (GUI) sollte der Arm gesteuert werden. Dabei sollte mit Buttons, der jeweilige Servo und die jeweilige Richtung auszuwählen sein. Des Weiteren sollte die Geschwindigkeit der Servos veränderbar sein.
Wie wir mit dem Arduino kommunizieren und den Servos die Kommandos geschickt haben, darauf werden wir im späteren Teil, beim Beschreiben des Codes genauer darauf eingehen. Die gesamte Kommunikation mit dem Arduino, sollte im Anschluss dann auch ohne USB Kabel funktionieren, und so wurden die gesamten Kommandos via WLAN an den Arduino gesendet.
Die Funktion hinter dem ganzen Projekt war, einen leicht steuerbaren Arm zu konstruieren, welcher Sachen greifen und bewegen kann.

2 Aufbau

Ein Arduino, welcher mit dem WLAN Shield gekoppelt ist, steuert den Roboterarm indem er Kommandos via WLAN an die einzelnen Servomotoren sendet. Ein weiterer Arduino dient zur Stromversorgung der Servomotoren, da ansonsten das WLAN Shield nicht einwandfrei funktioniert hat, auf diesem Arduino ist kein Code hochgeladen worden.

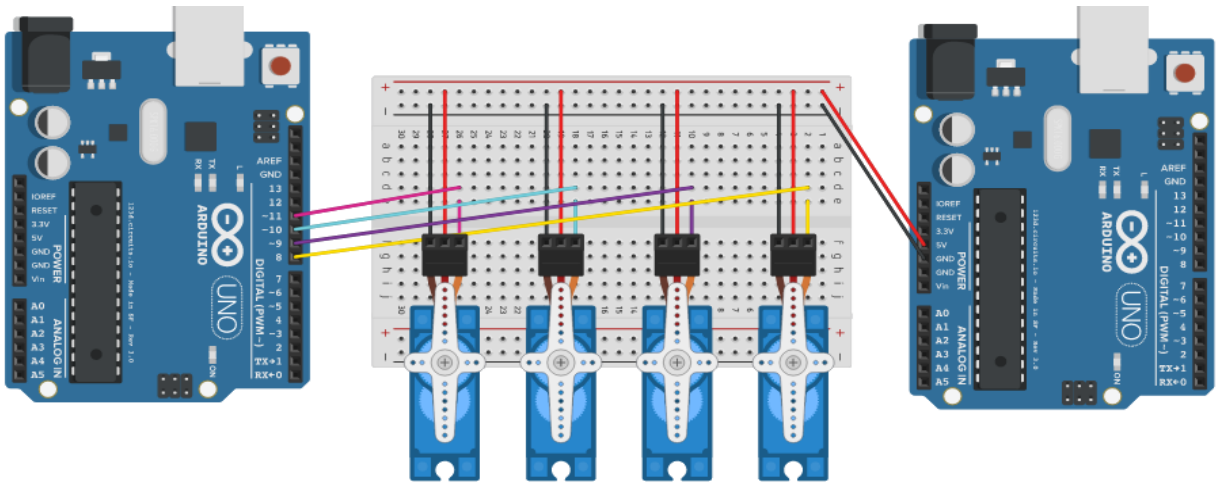

Abbildung 2: Schaltbild

3 Arduino/C Code-Beschreibung

In unserem C Code haben wir zuerst vier Funktionen erstellt. Eine dient dazu den jeweiligen Servomotor nach links zu bewegen und eine Andere, um ihn nach rechts zu bewegen. Die anderen beiden Funktionen, die sogenannten SSpezial-Funktionen", bewirken, dass Servo Nr. 2 und Nr. 3 immer die Summe aus 180° bilden. Dadurch bleibt der mittlere Teil des Armes, siehe Abbildung 1, immer Senkrecht stehen, dies lässt sich nach oben sowie nach unten steuern.

```
void spezial_funktion_hoch (int maximalwinkel, Servo *servoname, Servo *servoname2) {
  int var = servoname->read();
  int var2 = servoname2->read();

  if (var < maximalwinkel) {
    servoname->write(var + 1);
    servoname2->write(var2 + 1);
    delay(speed_var);
  }
}

void spezial_funktion_runter (int minimalwinkel, Servo *servoname, Servo *servoname2) {
  int var = servoname->read();
  int var2 = servoname2->read();

  if (var > minimalwinkel) {
    servoname->write(var - 1);
    servoname2->write(var2 - 1);
    delay(speed_var);
  }
}
```

Damit der Arduino sein empfangenes Kommando ausführt, mit der Variable: *python_button_var*, haben wir Switch Cases integriert. Switch-Cases funktioniert nur mit Zahlen bzw. Integern. Darum müssen die in Python gesendeten Buchstaben erstmal in den If-Funktionen zu Integern umgeändert werden, damit anschließend die Cases benutzt werden können.

```
void loop()
{
  // Check if the python program sent commands
  if (esp_server.available()) {
    // Read one line of commands
    String command = esp_server.readStringUntil('\n');
    Serial.println(command);

    if (command == "a") {
      python_button_var = 97;
    }
    else if (command == "b") {
      python_button_var = 98;
    }
    else if (command == "c") {
      python_button_var = 99;
    }
```

Abbildung 3: If-funktion mit dem Auslesen der Kommandos via WLAN

Neben den Richtungen können wir mit Switch Cases auch die Geschwindigkeit der Servomotoren verändern, indem das Delay größer oder kleiner gesetzt wird für die Variable *speed_var*. (Siehe Case 49-54)

```
switch (python_button_var) {

   case 97: arm_hoch(180, &servo1);   break; //a
   case 98: arm_runter(11, &servo1);  break; //b

   case 99: arm_hoch(180, &servo2);   break; //c
   case 100: arm_runter(11, &servo2);  break; //d

   case 101: arm_hoch(180, &servo3);   break; //e
   case 102: arm_runter(11, &servo3);  break; //f

   case 103: arm_hoch(180, &servo4);   break; //g
   case 104: arm_runter(11, &servo4);  break; //h

   case 105: spezial_funktion_hoch(180, &servo2, &servo3); break; //i
   case 106: spezial_funktion_runter(11, &servo2, &servo3); break; //j

   case 107: break; //default case //k

   case 49: speed_var = 5; break;
   case 50: speed_var = 8;  break;
   case 51: speed_var = 11; break;
   case 52: speed_var = 14; break;
   case 53: speed_var = 17; break;
   case 54: speed_var = 20; break;
 }
```

3.1 WLAN-Empfang

In unserem Code lassen wir den Arduino etwas über das WLAN empfangen, indem wir die DumbServer.h, sowie DumbServer.cpp eingebaut haben. Diese erlauben uns eine Verbidnung mit dem WLAN Shield ESP8266 herzustellen.
Um mit dem Python Programm in Verbindung zu treten, benötigen wir die IP Adresse und den richtigen Port. Den Port teilen wir dem Python und dem Arduino Programm selber zu, die IP Adresse erfahren wir durch unser Arduino Programm, sobald wir den Seriellen Monitor öffnen und der Server gestartet ist.
Das genaue Kommando welches wir empfangen und auslesen, tun wir mit dem oberen Teil des Codes in Abbildung 3.

4 Python Code und GUI

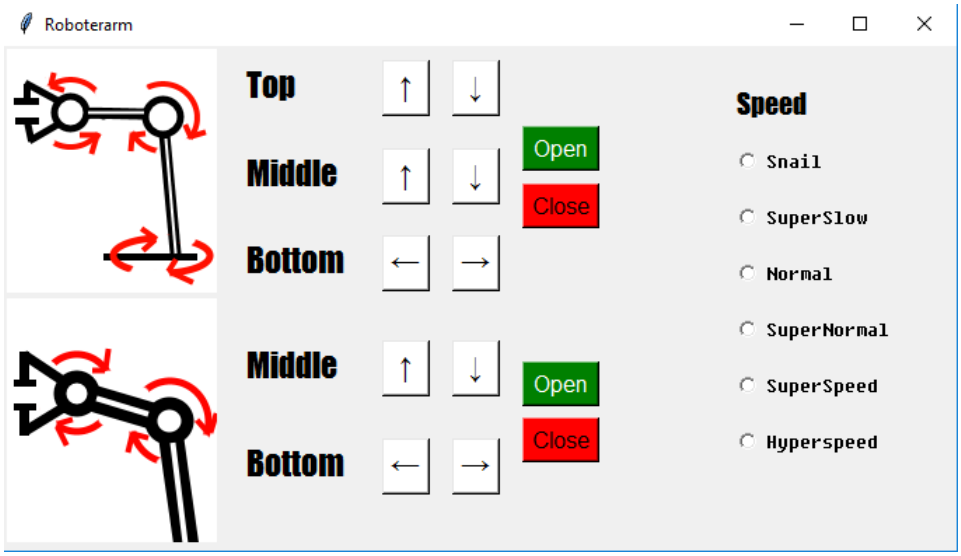

Abbildung 4: Mit Tkinter erstellte GUI