

Projektdokumentation: Roboterarm

Grundlagen der Informatik II SOSE 2017

23.06.2017

Name	Matrikelnummer
Fany Bowt	4345894
Jisoo Choi	3103933
Michael Vogt	4391254

Inhaltsverzeichnis

1	Beschreibung des Projekts und Funktionsweise	2
2	Der Aufbau	2
3	Arduino/C Code-Beschreibung	3
3.1	Setup	3
3.2	Steuerungsfunktionen	4
3.3	Die Loop	5
3.4	WLAN-Empfang	5
4	Python Code und GUI	6
4.1	GUI	6
4.2	Python Code	7

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.

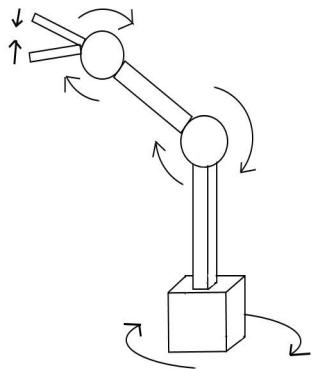


Abbildung 1: Skizze vom Arm

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, wird im späteren Teil, beim Beschreiben des Codes genauer erläutert. 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 Der Aufbau

Ein Arduino, welcher mit dem WLAN Shield gekoppelt ist, steuert den Roboterarm indem er Kommandos via WLAN an die einzelnen Servomotoren sendet. Um einen Servo anzusteuern sind drei Verbindungen mit dem Arduino nötig: VCC, GND und DATA. Mittels einer speziellen Bibliothek lassen sich dann die Servos auf den 1 Grad genau ansteuern. Dabei sind die Servomotoren jedoch auf 180° mechanisch begrenzt. Es werden schließlich 4 Servomotoren durch den DATA-Pin von einem Arduino kontrolliert (siehe Abbildung 2). Ein weiterer Arduino dient zur Stromversorgung der Servomotoren

(VCC und GND), da ansonsten das WLAN Shield nicht einwandfrei funktioniert hat, auf diesem Arduino ist kein Code hochgeladen worden.

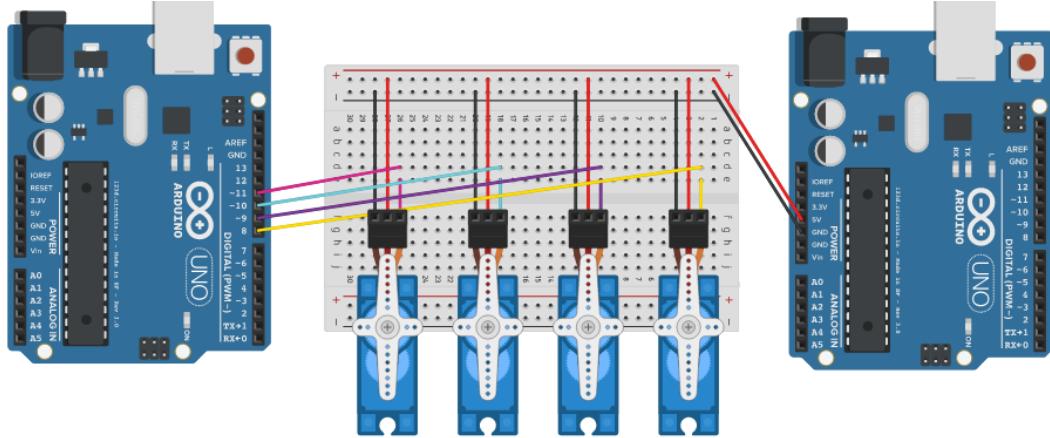


Abbildung 2: Schaltbild

Die Arme wurden aus Holz ausgeschnitten. Anschließend konnten die Servos mit kleinen Schrauben am Holz befestigt werden. Eine größere Holzplatte dient als Unterlage. Sehr wichtig war es, dass der Arm ein Ausgleichsgewicht besitzt da ansonsten der Servo zu stark belastet wird, sodass der Arm nicht hochgehoben werden kann.

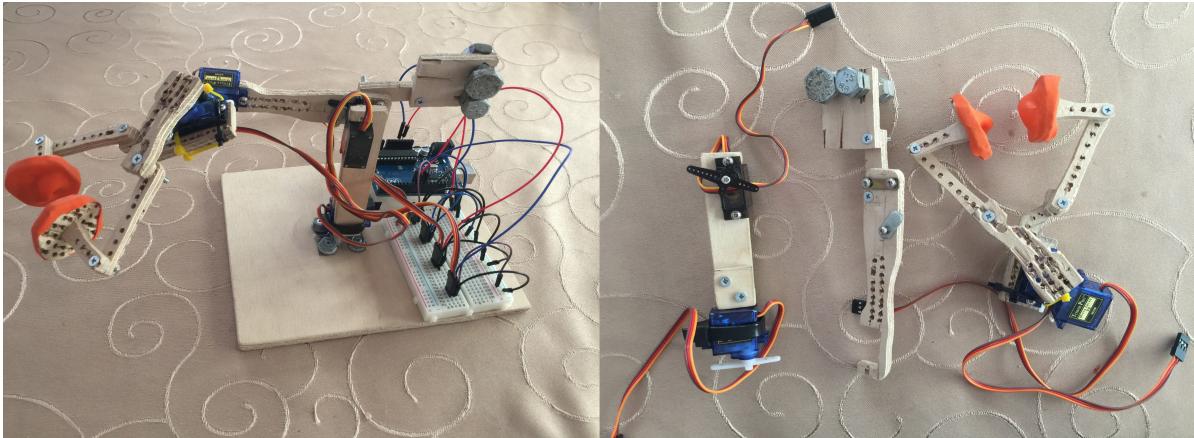


Abbildung 3: Aufbau

3 Arduino/C Code-Beschreibung

3.1 Setup

Mittels der Bibliothek servo.h lassen sich die Servos sehr simpel ansteuern. Dazu müssen zunächst Variablen als Servos deklariert werden (siehe Abbildung 3 Zeile 7-10) und

daraufhin auf welchen Pin (Zeile 11-14). Diese Pins sind dann an DATA anzuschließen. Im Setup wird dann mit dem Befehl servo.attach(*servo_pin*) der jeweilige Servo einem Pin zugewiesen.

```

1 #include <Servo.h> //Servo-Bibliothek
2 #include <SoftwareSerial.h> //SoftwareSerial-Bibliothek
3 #include "DumbServer.h" //Verknüpfung mit DumbServer.h
4 SoftwareSerial esp_serial(3, 2);
5 EspServer esp_server;
6
7 Servo servol;
8 Servo servo2;
9 Servo servo3;
10 Servo servo4;
11 int servo_lpin = 8;
12 int servo_2pin = 9;
13 int servo_3pin = 10;
14 int servo_4pin = 11;
15
16 void setup() {
17   Serial.begin(9600);
18   esp_serial.begin(9600);
19
20   servol.attach(servo_lpin);
21   servo2.attach(servo_2pin);
22   servo3.attach(servo_3pin);
23   servo4.attach(servo_4pin);

```

Abbildung 4: If-funktion mit dem Auslesen der Kommandos via WLAN

3.2 Steuerungsfunktionen

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 Spezial-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);
  }
}

```

Abbildung 5: Bewegungsfunktionen

3.3 Die Loop

Damit der Arduino sein empfangenes Kommando ausführt, wird mit der Variable `python_button_var` ein Switch-Cases integriert. Switch-Cases funktioniert nur mit Zahlen bzw. Integern. Darum müssen die in Python gesendeten Buchstaben zunächst 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 6: 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 Abbilung 6, 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;
}
```

Abbildung 7: Die switch-case

3.4 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

Verbindung 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 Arduinoprogramm, 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 6.

4 Python Code und GUI

4.1 GUI

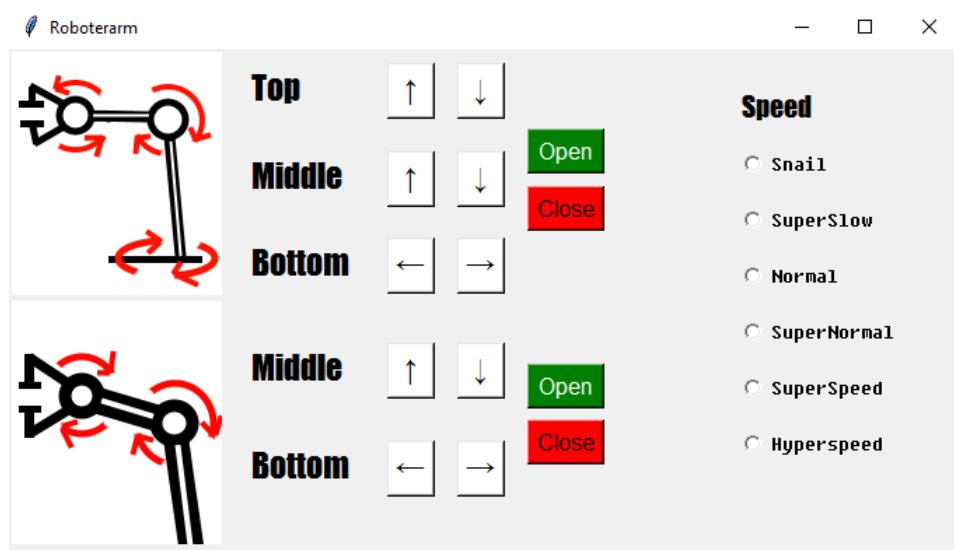


Abbildung 8: Die mit Tkinter erstellte GUI

Die GUI wurde mit Tkinter erstellt und diese haben wir in drei Sektionen eingeteilt. Die erste Sektion ist für das Bewegen der einzelnen Servomotoren in die jeweils angezeigte Richtung auf dem Button. Die Open und Close Buttons dienen dazu den oberen Servo (Nr.4) anzusteuern, damit sich die Zange öffnet und schließt.

Die zweite Sektion, die Untere, ist dafür da um die Spezialfunktion zu nutzen. Damit das Ganze übersichtlich bleibt, haben wir auch die Buttons für den unteren und den oberen Servomotoren hinzugefügt, so teilt es sich in die zwei verschiedenen Ansteuerungs Möglichkeiten auf, sodass besser nachvollziehbar sind.

Die dritte Sektion ist die Geschwindigkeit, mit dieser können wir zwischen sechs Radio buttons auswählen. Alle Servos bekommen die selbe Geschwindigkeit, die Funktion ist nicht für einzelne Servomotoren gedacht.

4.2 Python Code

Jeder Button bekommt eine eigene Funktion. Diese Funktionen bewirken, dass die Buttons einen String an das WLAN Shield senden. In Abbildung 6 ist dann zu sehen, wie die Strings zu Integern umgeändert werden, damit die Cases funktionieren.

```
def case_3(self,event):
    self.arduino.send_command('c') """Case 99"""

def case_4(self,event):
    self.arduino.send_command('d') """Case 100"""

def case_5(self,event):
    self.arduino.send_command('e') """Case 101"""

def case_6(self,event):
    self.arduino.send_command('f') """Case 102"""

def case_7(self,event):
    self.arduino.send_command('g') """Case 103"""

def case_8(self,event):
    self.arduino.send_command('h') """Case 104"""
```

Abbildung 9: Button Funktionen

Während der jeweilige Buttons gedrückt wird, wird der String gesendet, sodass wenn wir den Button gedrückt halten der Servo durchgehend gesteuert wird. Sollte der Button losgelassen werden, wird das Case 107 ausgeführt, diese bewirkt das der jeweilige Servo sich nicht mehr bewegt.

Das Kommando wird mit der *send_command* Funktion übermittelt. Diese sendet an unseren Socket Server den String, welcher gerade übermittelt wurde durch den Button und der Arduino mit WLAN Shield wandelt ihn in ein Integer um, um die Case auszuführen.

```
def send_command(self, command):
    self.socket.send(command.encode('utf-8') + b'\n')
```

Abbildung 10: Kommando Übermittlung an den Arduino mit WLAN Shield