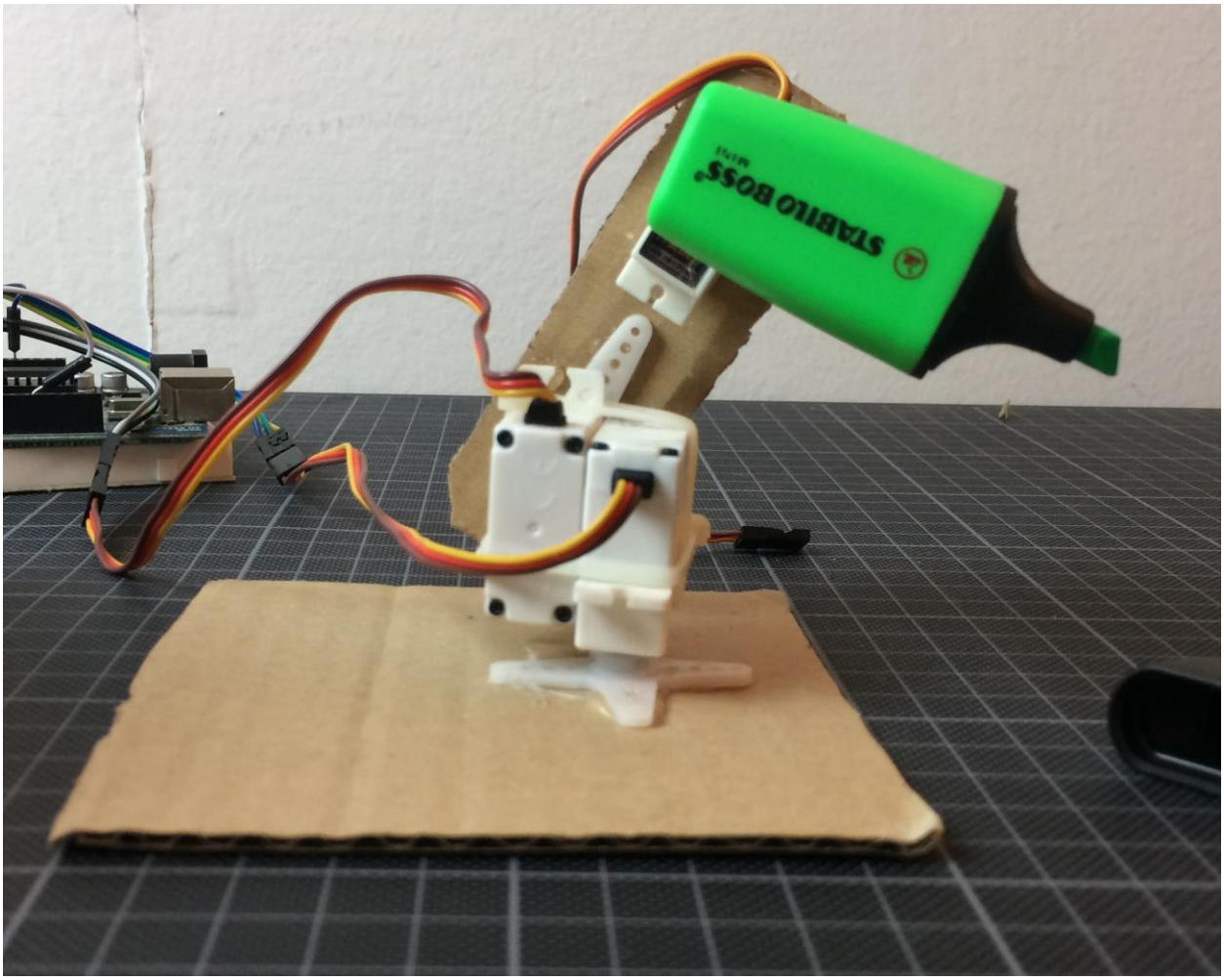
3DOF Mal-Roboter

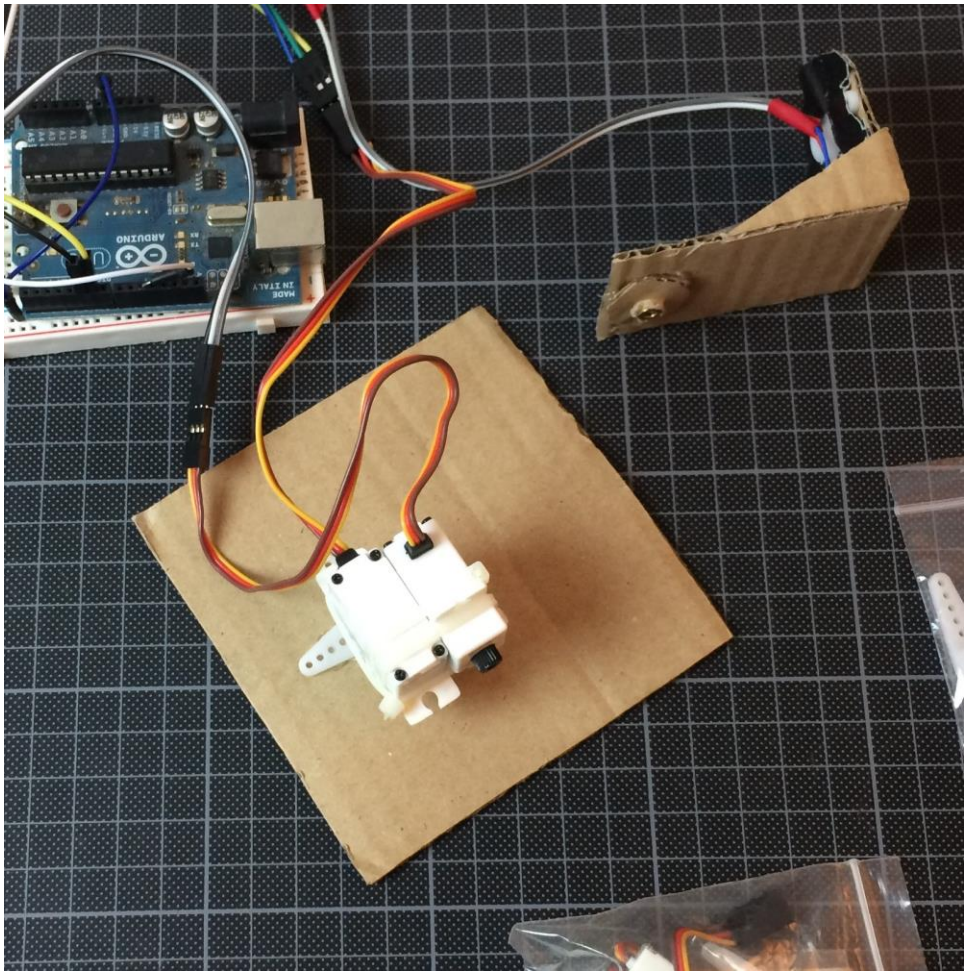
In dieser Einheit werden wir unseren 2DOF Servo-Roboter um einen Servo erweitern und mit einem Text-Marker ausstatten.



Beschreibung der Übung:

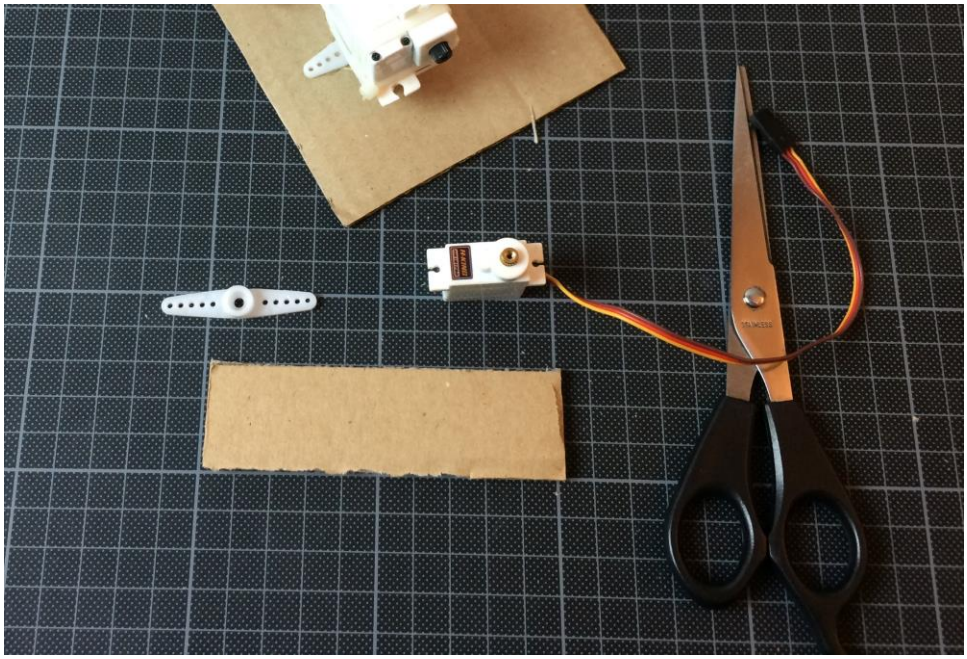
Ziel der Übungen ist es den Roboter manuell und automatisch zu steuern. Am Ende wollen wir Figuren und Buchstaben mit dem Roboter Zeichnen.

Bauanleitung:

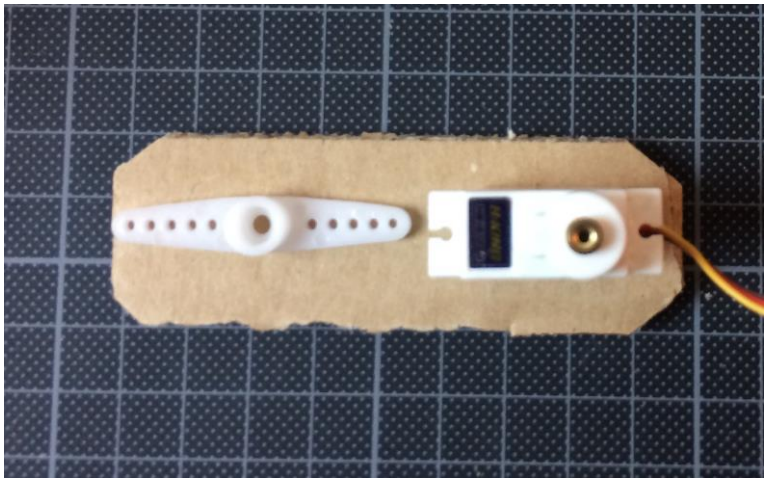


Nehmt die Sensor-Halterung des 2DOF Roboters ab.

Schneidet ein etwa 3 x 10cm Stueck aus Karton aus.

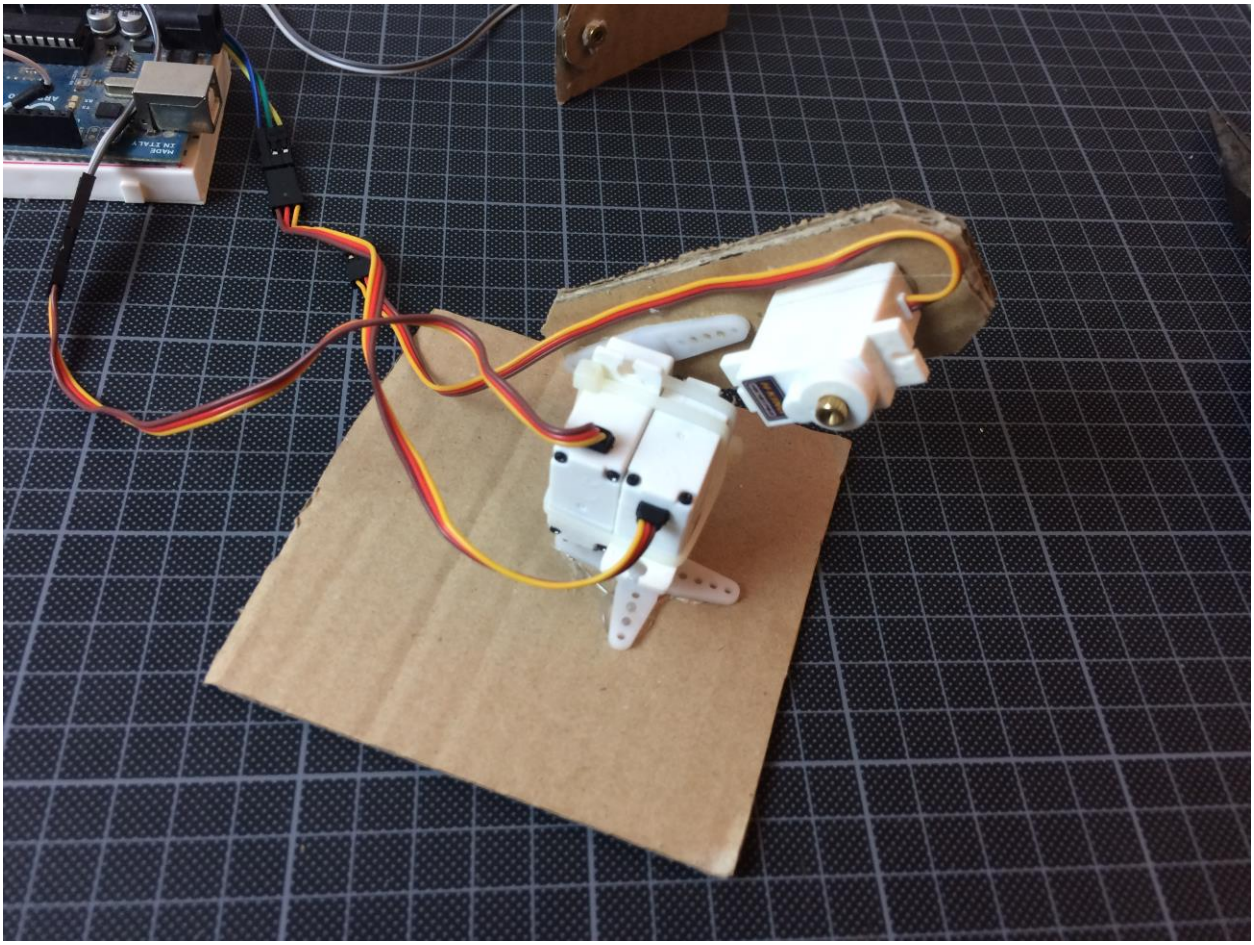


Ihr könnt die Ecken abschrägen.



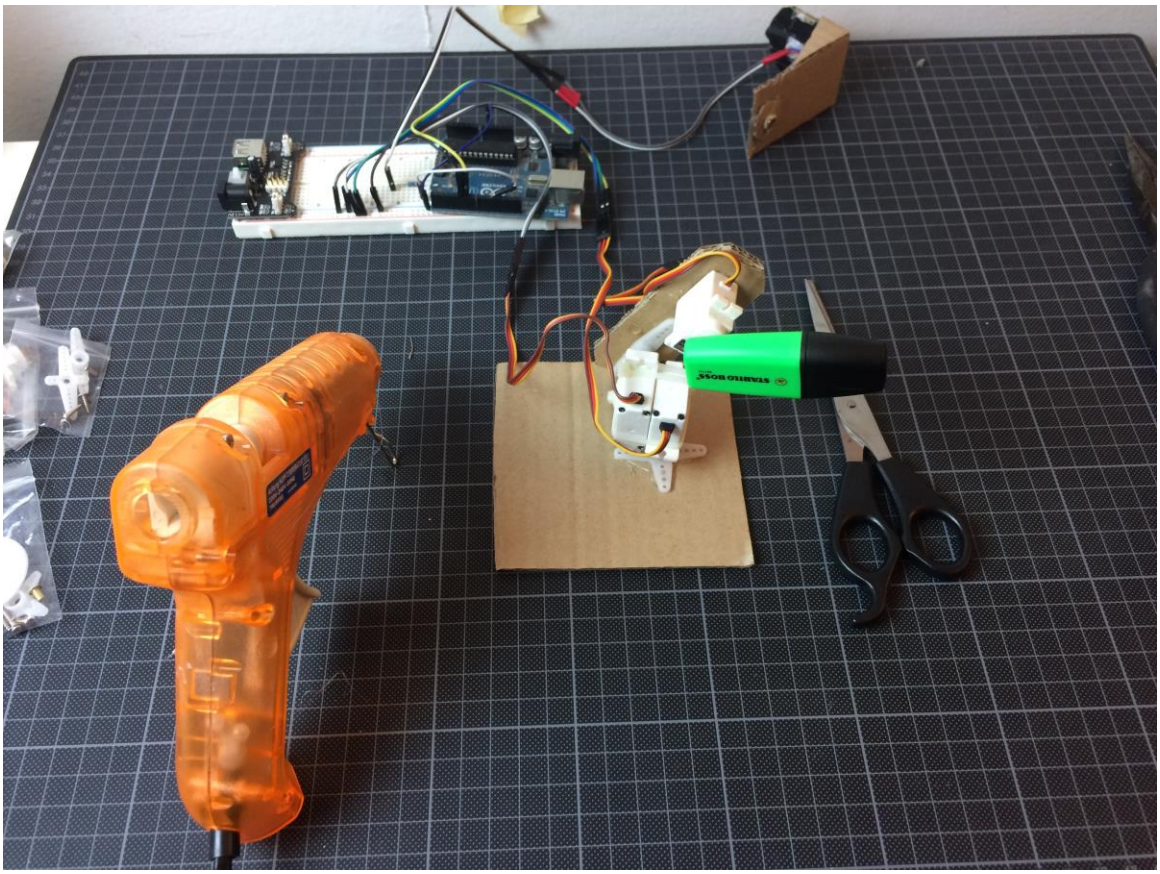
Befestigt ein Servo-Horn und einen Servo sowie abgebildet mit Heißkleber auf den Karton.

Steck das Servo-Horn an den zweiten Servo der Basis.



Anschliessend Befestigt eine Servo-Horn des dritten Servos an den Text-Marker mit Heisskleber.

Steck das Horn mit Marker an den dritten Servo.



Fertig! Der Roboter sollte wie wie in der Abbildung oben aussehen.

Verkabelung:

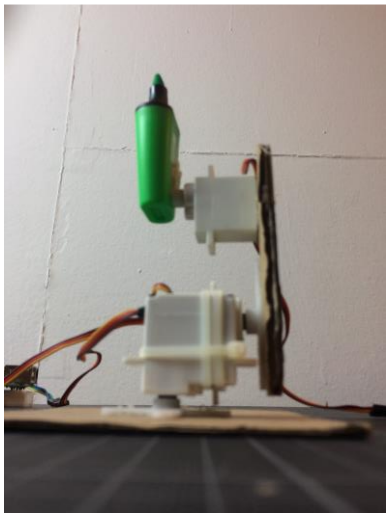
Verbinden sie Servos, Lichtsensor und optional einen analogen Joystick mit dem Arduino und der Stromversorgung. Benutzen sie dazu Male-to-Male und Female-to-Male Jumper-Kabel.

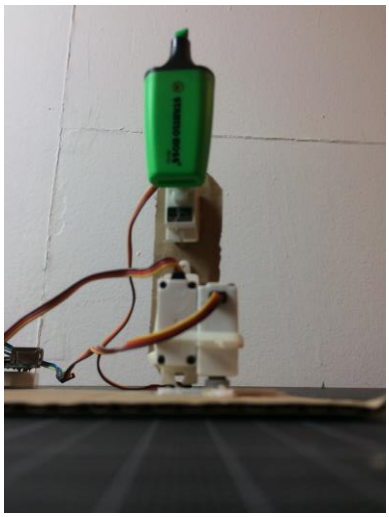
Komponente	**Verbindung**
1. Servo (vertikale)	Servo Power- (Braun/Schwarz) -> Ground am Steckbrett Servo Power+(Orange/Rot) -> +5Volt am Steckbrett Servo Signal (Gelb/Weiss) -> Arduino Pin 5
2. Servo (horizontal)	Servo Power- (Braun/Schwarz) -> Ground am Steckbrett Servo Power+(Orange/Rot) -> +5Volt am Steckbrett Servo Signal (Gelb/Weiss) -> Arduino Pin 6
3. Servo (horizontal oben/ende)	Servo Power- (Braun/Schwarz) -> Ground am Steckbrett Servo Power+(Orange/Rot) -> +5Volt am Steckbrett Servo Signal (Gelb/Weiss) -> Arduino Pin 7
Arduino Ground	Arduino Ground Pin -> Ground am Steckbrett
Stromversorgung (5Volt)	Stromversorgung minus (Leitung) -> Ground am Steckbrett Stromversorgung plus (Leitung) -> +5V am Steckbrett

Übung 1: Manuelle Ausgangs-Position

Bewegt manuell alle Servos in die Ausgangs Position und merkt euch die Winkelwerte.

Erweitere den Code um die `switch case` statements ‚r' und ‚f' um den neuen Servo nach oben und unten zu Bewegen.





Setup Code:

```
/* Inkludieren der Servo.h library fuer die Servo-Steuerung */
#include<Servo.h>

/* Definiere die minimal und maximal Position der Servos */
#define SVO_POS_MIN     500
#define SVO_POS_MAX     2500
#define SVO_POS_MID     1500

/* Deklarieren der drei Servos */
Servo myServoX;
Servo myServoY;
Servo myServoY_End;

int myServoPosX = SVO_POS_MID;
int myServoPosY = SVO_POS_MID;
int myServoPosY_End = SVO_POS_MID;

void setup() {
       /* Initialize Serial with 115200 BaudRate */
       Serial.begin(115200);

       /* Setzen der Servo-Pins */
       myServoX.attach(5);     /* Attach servo PWM signal to pin 5 */
       myServoY.attach(6);     /* Attach servo PWM signal to pin 6 */
       myServoY_End.attach(7); /* Attach servo PWM signal to pin 7 */
}
```

```
void loop() {
	myServoX.write(myServoPosX); /* Update position of servo 1 */
	myServoY.write(myServoPosY); /* Update position of servo 2 */
	myServoY_End.write(myServoPosY_End); /* Update position of servo 3 */

	while (Serial.available() > 0) /* while there is data available repeat reading */
	{
		char data = Serial.read(); /* read one byte from the received data */

		switch (data)
		{
		case 'a':
			myServoPosX = myServoPosX + 1;
			break;

		case 'd':
			myServoPosX = myServoPosX - 1;
			break;

		case 'w':
			myServoPosY = myServoPosY + 1;
			break;

		case 's':
			myServoPosY = myServoPosY - 1;
			break;

		case 'r':

/****************** Ergaenze Code hier (myServoPosY_End + 1) ********************/

			break;

		case 'f':

/****************** Ergaenze Code hier (myServoPosY_End – 1) ********************/

			break;

		default:
			break;
		}

		/* Beschraencken auf min und max Grenzen */
		if (myServoPosX < SVO_POS_MIN) myServoPosX = SVO_POS_MIN;
		if (myServoPosX > SVO_POS_MAX) myServoPosX = SVO_POS_MAX;
		if (myServoPosY < SVO_POS_MIN) myServoPosY = SVO_POS_MIN;
		if (myServoPosY > SVO_POS_MAX) myServoPosY = SVO_POS_MAX;
		if (myServoPosY_End < SVO_POS_MIN) myServoPosY_End = SVO_POS_MIN;
		if (myServoPosY_End > SVO_POS_MAX) myServoPosY_End = SVO_POS_MAX;

		Serial.print("ServoX: "); Serial.print(myServoPosX);
		Serial.print("\tServoY: "); Serial.print(myServoPosY);
		Serial.print("\tServoY End: "); Serial.println(myServoPosY_End);
	}
}
```

Übung 2: Automatische Ausgangs-Position

Erweitere den Code über der setup Funktion um drei weiter integer Variablen um die Ausgangs-Position zu speichern.

```
/* Ausgangs Winkel */
int myServoPosX_Zero = 1500; /* Fuege hier deinen Ausgangs Wert fuer Servo 1 ein */
int myServoPosY_Zero = 1500; /* Fuege hier deinen Ausgangs Wert fuer Servo 2 ein */
int myServoPosY_End_Zero = 1500; /* Fuege hier deinen Ausgangs Wert fuer Servo 3 ein */
```

Füge einen weiteren `switch case` hinzu um die aktuelle Position als Ausgangs-Position zu Speichern.

```
            case 'Z':
/******** Ergaenze Code hier (weise aktuelle Position der Zero Position zu) ********/
                    break;
            case 'z':
/******** Ergaenze Code hier (weise Zero-Position der aktuellen Position zu) ********/
                    break;
```

Übung 3: Line Interpolieren

In unserer letzten Einheit haben wir eine Funktion erstellt die zwischen zwei Winkel-Werten interpoliert.

In dieser Übung werden wir die Funktion `DriveLinear` um einen dritten Winkel erweitern.

```
void DriveLinear(int TargetPosX, int TargetPosY, int TargetPosY_End, int TimeMS)
{
	int timePast = 0;
	int tmpStartedX = myServoPosX;
	int tmpStartedY = myServoPosY;
	int tmpStartedY_End = myServoPosY_End;

	while (timePast < TimeMS)
	{

		myServoPosX = tmpStartedX + ((float)(TargetPosX - tmpStartedX) *
			      (float)timePast / (float)TimeMS);
		myServoPosY = tmpStartedY + ((float)(TargetPosY - tmpStartedY) *
			      (float)timePast / (float)TimeMS);

/******** Ergaenze Code hier (weise myServoPosY_End den interpolierten Wert zu) ********/


		myServoX.writeMicroseconds(myServoPosX); /* Update position of servo X */
		myServoY.writeMicroseconds(myServoPosY); /* Update position of servo Y */

/******** Ergaenze Code hier (rufe myServoY_End.writeMicroseconds auf) ********/

		delay(1);
		timePast += 1;
	}
}
```

Und ergänze den Kommando `switch case` um drei weitere cases (‘1‘,‘2‘,‘3‘):

```
case '1':
	DriveLinear(1100,1700,1300, 2000);
	break;
case '2':
	DriveLinear(1600,1300,1600, 2000);
	break;
case '3':
	DriveLinear(1100,1852,804, 1500);
	DriveLinear(1400,1800,1752, 1500);
	DriveLinear(1800,2100,1652, 1500);

	DriveLinear(myServoPosX_Zero, myServoPosY_Zero, myServoPosY_End_Zero, 1500);
	break;
```

Übung 4: Winkel von 3D Koordinaten

Nun erstellen wir eine Funktion um 3D Koordinaten in Winkel für unsere drei Servos auszurechen.

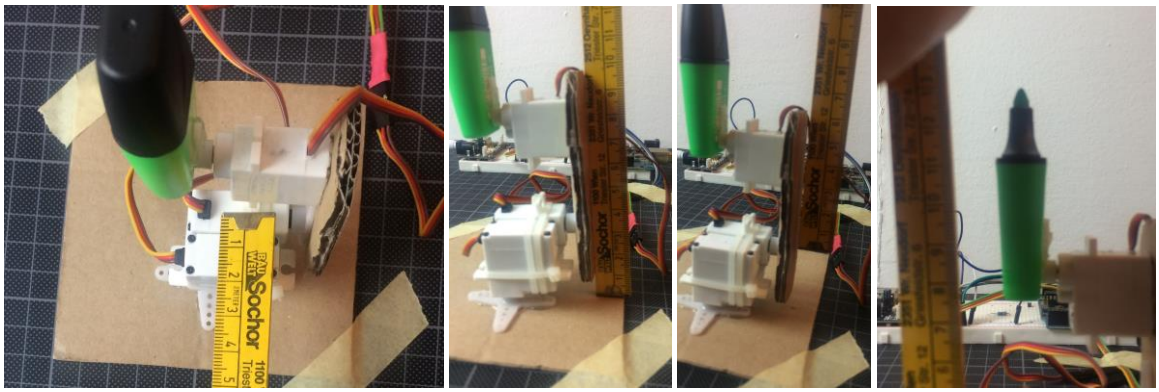
Aber dafür muss unser Roboter bzw. unser Programm wissen wie groß der Roboter ist und wo seine Servo Dreh-Punkte (Achsen) sind.

Dafür müssen wir jedes Segment unseres Roboters abmessen. Ein Segment geht von einer Servo Drehpunkt zum nächsten Servo-Drehpunkt.

Unser Null-Punkt unseres Koordinatensystems befindet sich im Drehpunkt unseres ersten Servos.

Vermesst euren Roboter und notiert euch die Werte in Millimeter.

- Versatz Servo1-Drehachse zu Servo2-Drehachse
- Abstand Bodenplatte zu Servo2-Drehachse
- Abstand Servo2 zu Servo3
- Abstand Servo3 zu Spitze des Stiftes
- Versatz Stiftes zu Servo1 (optional)



Bei meinem Roboter sind diese Werte: 14mm, 30mm, 50mm, 54mm, 10mm

Speichert eure Werte wie folgt (global) ab:

```
float dist_Servo1To2_Back = 14.0f;
float dist_BaseTo2_Up = 30.0f;
float dist_Servo2To3 = 50.0f;
float dist_Servo3ToTip = 54.0f;
float dist_Servo1To_ToLeft = 10.0f;

Legt auch 180 Grad PWM/Winkel Skalierungs-Variablen an (global):

int PWM_Servo1_180 = (900 * 2);
int PWM_Servo2_180 = (900 * 2);
int PWM_Servo3_180 = (900 * 2);
```

In der Funktion Cal_3D_To_ServoPWM findet ihr die ganze Mathematik um aus 3D Koordinaten Winkelwerte zu errechnen.

```
////////////////////////////////////////////////////////////////////////////////
bool Cal_3D_To_ServoPWM(float X, float Y, float Z,
                        int &Servo1PWM, int &Servo2PWM, int &Servo3PWM)
{
	Z *= -1;

	float dist1 = sqrt((X * X) + (Y * Y));
	float alpha1 = asin(X / dist1);

	dist1 += dist_Servo1To2_Back;

	Z += dist_BaseTo2_Up;

	float tmpAlphaTarget = atan2(Z, dist1);
	float lengthToTarget = sqrt((Z * Z) + (dist1 * dist1));
	float tmpAlphaBase1 = acos( (dist_Servo3ToTip * dist_Servo3ToTip
	                            - dist_Servo2To3 * dist_Servo2To3
	                            - lengthToTarget * lengthToTarget)
	                            / (lengthToTarget * dist_Servo2To3 * (-2.0f)));

	float alpha2 = tmpAlphaBase1 - tmpAlphaTarget;

	float tmpAlpha1 = asin((dist1 - cos(alpha2) * dist_Servo2To3) / dist_Servo3ToTip);
	float tmpAlpha2 = PI / 2 - alpha2;

	float alpha3 = tmpAlpha1 + tmpAlpha2;

	Servo1PWM = myServoPosX_Zero + (alpha1 / PI) * (float)PWM_Servo1_180;
	Servo2PWM = myServoPosY_Zero + ((PI / 2 - alpha2) / PI) * (float)PWM_Servo2_180;
	Servo3PWM = myServoPosY_End_Zero - ((PI - alpha3) / PI) * (float)PWM_Servo3_180;
	return true;
}
////////////////////////////////////////////////////////////////////////////////
```

Erweitert euren switch case um folgende (Test) cases:

```
case '0':
	Cal_3D_To_ServoPWM(30, 30, 30, myServoPosX, myServoPosY, myServoPosY_End);
	break;
case '9':
	Cal_3D_To_ServoPWM(-30, 30, 30, myServoPosX, myServoPosY, myServoPosY_End);
	break;
case '8':
	Cal_3D_To_ServoPWM(0, 30, 30, myServoPosX, myServoPosY, myServoPosY_End);
	break;
```

Übung 5: 3D Linien

Nun erstellen wir eine Funktion um zwischen 3D Punkten linear zu interpolieren.

Die Funktion ist sehr ähnlich mit unseres 2D Linien Zeichen Funktion.

```
float PosNow_X = 0;
float PosNow_Y = 30;
float PosNow_Z = 40;

void DriveLinear3D(int TargetPosX, int TargetPosY, int TargetPosZ, int TimeMS)
{
	int timePast = 0;
	int tmpStartedX = PosNow_X;
	int tmpStartedY = PosNow_Y;
	int tmpStartedZ = PosNow_Z;

	while (timePast < TimeMS)
	{
		PosNow_X = tmpStartedX + ((float)(TargetPosX - tmpStartedX) *
			(float)timePast / (float)TimeMS);
		PosNow_Y = tmpStartedY + ((float)(TargetPosY - tmpStartedY) *
			(float)timePast / (float)TimeMS);
		PosNow_Z = tmpStartedZ + ((float)(TargetPosZ - tmpStartedZ) *
			(float)timePast / (float)TimeMS);

		Cal_3D_To_ServoPWM(PosNow_X, PosNow_Y, PosNow_Z,
				myServoPosX, myServoPosY, myServoPosY_End);

		myServoX.writeMicroseconds(myServoPosX); /* Update position of servo X */
		myServoY.writeMicroseconds(myServoPosY); /* Update position of servo Y */
		myServoY_End.writeMicroseconds(myServoPosY_End);/* Update position servo */

		delay(1);
		timePast += 1;
	}
}

////////////////////////////////////////////////////////////////////////////////
```

Erweitert euren switch case um folgende (Test) cases:

```
case '7':
	DriveLinear3D(30, 30, 30, 1000);

	DriveLinear3D(30, 30, 0, 1000);

	DriveLinear3D(30, 50, 0, 1000);
	DriveLinear3D(10, 50, 0, 1000);
	DriveLinear3D(10, 30, 0, 1000);
	DriveLinear3D(30, 30, 0, 1000);

	DriveLinear3D(30, 30, 30, 1000);
	break;
```

Übung 5: A-Linien

Erweitert euren Code um ein große A mit dem Roboter zu zeichnen.

☺