

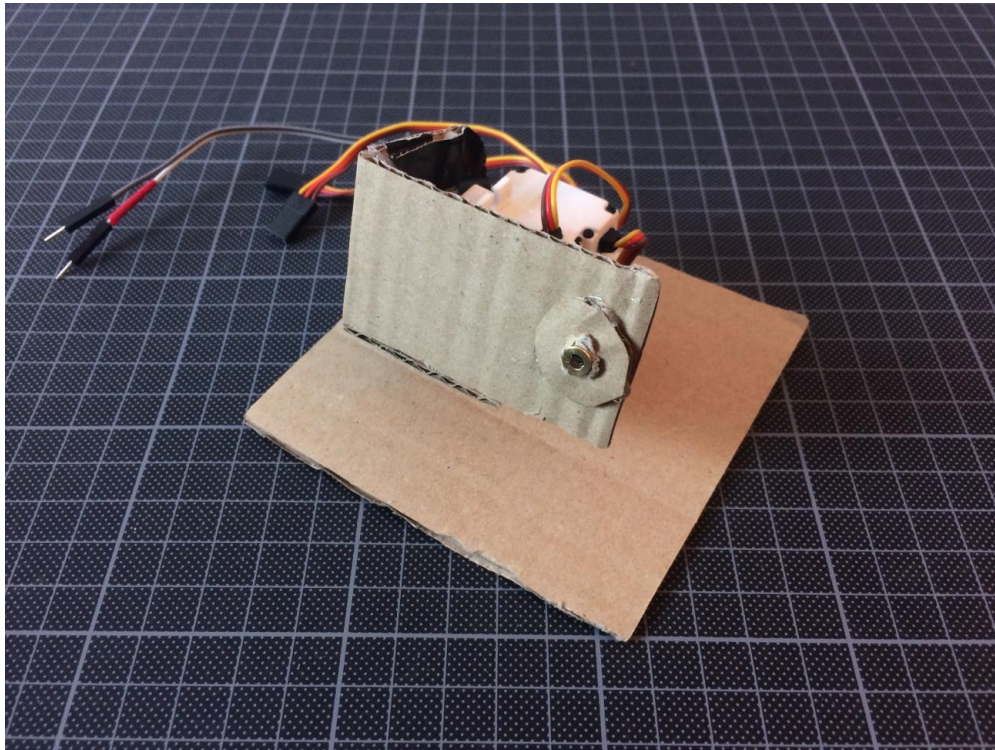
2DOF Laser Robot

Beschreibung:

In der letzten Lehreinheit haben wir einen 2 „Degree of Freedom“ (DOF) Light Scan Roboter gebaut. In dieser Einheit nützen wir diesen Roboter und wollen damit Folgendes umsetzen:

- Datenkommunikation zwischen Computer (PC/Laptop) und Arduino
- Steuerung beider Servos per Computer (PC/Laptop)
- Erweitern des Roboters um eine Laserdiode
- Steuern des Lasers per Computer
- Erstellen einer Funktion um mit dem Laser Linien zu „Zeichnen“
- Abfahren von mehreren Linien
- Umrechnen von 2D Koordinaten einer Ebene auf Servo-Winkel-Werte
- Abfahren von Buchstabe ,A' ,M' ,N' etc...

Der Roboter besteht dann aus zwei Servos und einer Laserdiode.



Datenkommunikation zwischen Computer (PC/Laptop) und Arduino

Um einen Roboter (oder jegliche andere Applikation) per PC, Laptop oder Smartphone zu steuern ist es wichtig Daten richtig zu empfangen bzw. richtig zu interpretieren.

Solange man mit nur einem Byte zur Steuerung seines Systems auskommt gestaltet sich der Kommunikation-Mechanismus einfach.

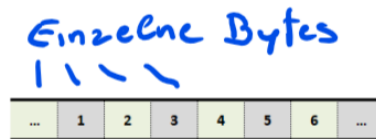


Figure 1 Datenstrom von einzelne Bytes

Ein empfangenes Byte kann man mit `Serial.read()` lesen und den gelesenen Wert einer Variablen zuweisen oder in einem Buffer speichern.

Zb: `char data = Serial.read(); /* read one byte from the received data */`

Mit der Methode `Serial.available()` kann man abfragen wie viele empfangen Bytes im Empfangsbuffer liegen.

Sobald man mehr als ein Byte an Daten zur Steuerung benötigt muss man darauf achten den Beginn und das Ende eines einzelnen Datensatzes richtig zu erkennen.

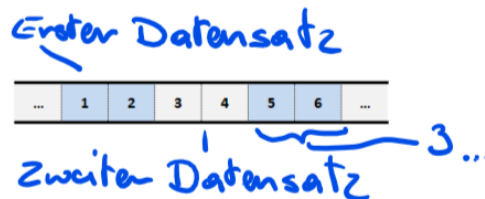


Figure 2 Datensätze mit fixer Größe

Zwei Bytes kann man zum Beispiel mit zwei `Serial.read()` einlesen.

```
if (Serial.available() >= 2)
{
    char lowByte = Serial.read(); /* read the first byte from the received data */
    char highByte = Serial.read(); /* read the second byte from the received data */

    int data = lowByte + (highByte * 256); /* Add this value together
                                           each highByte bit worth 256 lowByte bit
                                           or you can bit shift the (highByte << 8) */
}
```

Bei fixer Daten-Größe muss man „nur“ drauf achten das man den ersten Anfang des Datensatzes richtig erwischt und dann auf das dass man über die Dauer beim Lesen synchron bleibt.

Man kann mit einer Daten-Sequenz (MagicWord) den Anfang „Signalisieren“ bzw. synchronisieren.

Hat man es mit variabler Datensatz-Größe zutun muss man die Daten in einem Empfangs-Buffer zwischen speichern bis man sich sicher ist das man ein gesamtes Paket empfangen hat bevor man dieses Verarbeitet.

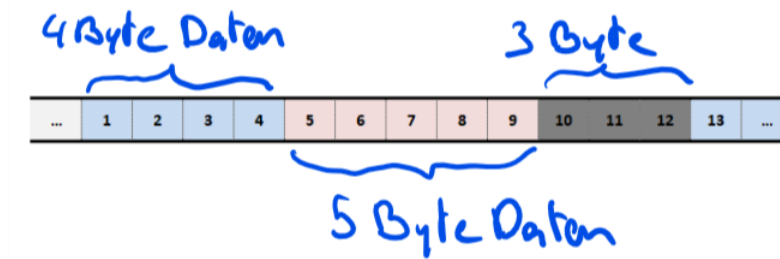


Figure 3 Datensätze mit variabler Größe

Bei Text-Daten kann man die String-Terminierung (Byte Wert 0) nützen um das Ende eines Datensatzes festzustellen.

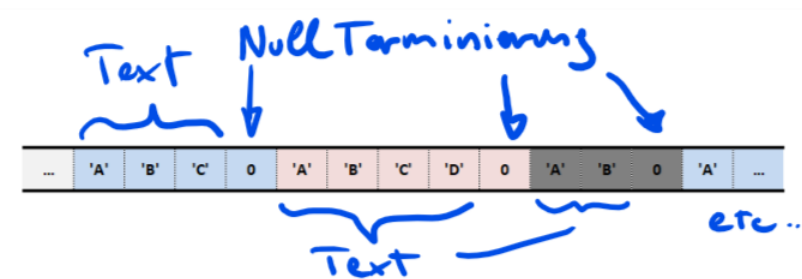


Figure 4 Text Daten mit Null-Terminierung

Das funktioniert nur deswegen zuverlässig weil der Byte Wert 0 im ASCII Zeichensatz keinem Zeichen entspricht. Mit folgendem Code kann man bequem Text Daten aus dem empfangsBuffer lesen.

```
char empfangsBuffer[256]; /* Zwischen Speicher mit Groesse 256 */
int curserIndex = 0;      /* Cursor Index (startet bei 0) */

bool LeseTextDaten() /* Returned true wenn man ein ganze Paket im Buffer hat */
{
    while (Serial.available() > 0) /* while there is data available repeat */
    {
        char data = Serial.read(); /* read one byte from the received data */

        empfangsBuffer[curserIndex] = data; /* Speichere Daten in den Buffer */
        curserIndex = (curserIndex + 1) % 256; /* Erhoehe den curser */

        if (data == 0) {
            curserIndex = 0; /* Ruecksetzen des Curser */
            return true;    /* Volles Paket empfangen */
        }
    }
    return false; /* Return mit false wenn kein volles Paket empfangen wurde */
}
```

Hat man kein „Sonderzeichen/Wert“ das/der nicht in den Nutz-Daten vorkommt dann kann man die Nutz-Daten in einem Frame verpacken.

Ein einfaches Frameformat besteht aus einer Längenangabe, den Nutz-Daten und einer Checksumme.

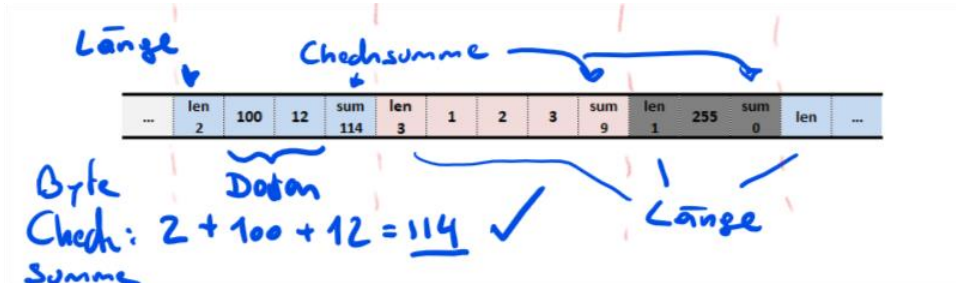


Figure 5 Daten-Pakete mit Länge, Inhalt und Checksumme

```
byte empfangsBuffer[256]; /* Zwischen Speicher mit Groesse 256 */
int curserIndex = 0; /* Cursor Index */
int startIndex = 0; /* Index des Starts */
int lenDaten = -1; /* Anzahl der erwarteten Daten */
byte checksum = 0; /* Checksumme */

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

        if (lenDaten == 0) /* Alle Daten gelesen */
        {
            if (checksum == data){ /* Ueberprueffe ob Checksumme korrekt */
                checksum = 0; lenDaten = -1; /* curserIndex = 0;
                return true; /* Yeah korrektes Paket */
            }
            else { /* Frame wurde nicht Richtig empfangen */
                /* Reparse from first byte startIndex + 1 or simple reset */
                checksum = 0; lenDaten = -1; /* curserIndex = 0;
            }
        }
        /* Start Frame */
        if(lenDaten == -1){checksum=data; lenDaten=data; startIndex=curserIndex;}
        else
        {
            lenDaten = lenDaten - 1; /* Decrease Length Counter */
            checksum += data; /* Add Up Checksum */
        }

        empfangsBuffer[curserIndex] = data; /* Speichere Daten in den Buffer */
        curserIndex = (curserIndex + 1) % 256; /* Erhoehe den curser */
    }
    return false; /* Kein volles Paket empfangen */
}
```

Hat man ein Paket nicht am Anfang erwischt so sollte man die bereits Empfangenen Daten erneut auf Länge + Daten + Checksumme überprüfen/re-parsen.

Folgender Code re-parsed die bereits Empfangenen Daten im Fall das ein Paket mit un-korrektter Checksumme empfangen wurde:

```
byte empfangsBuffer[256]; /* Zwischen Speicher mit Groesse 256 */
int curserIndex = 0;      /* Cursor Index */
int startIndex = 0, endIndex = -1; /* Index des Starts und Endes*/
int lenDaten = -1;        /* Anzahl der erwarteten Daten */
byte checksum = 0;        /* Checksumme*/
bool reparse = false;

bool LeseBinaereDaten()
{
    while(Serial.available() > 0 || reparse) /* while there is data available repeat*/
    {
        byte data;
        if (reparse)
            data = empfangsBuffer[curserIndex];
        else
            data = Serial.read(); /* read one byte from the received data */

        if (lenDaten == 0) /* Alle Daten gelesen */
        {
            if (checksum == data) /* Ueberprueffe ob Checksumme korrekt */
            {
                lenDaten = -1;
                if(endIndex == curserIndex){ endIndex = -1; reparse = false; }
                return true; /* Yeah korrektes Paket */
            }
            else
            {
                /* Frame wurde nicht Richtig empfangen */
                /* Re-parse from first byte startIndex + 1 */
                empfangsBuffer[curserIndex] = data;

                reparse = true;
                if (endIndex == -1) endIndex = curserIndex;

                lenDaten = -1;/**/ curserIndex = (startIndex + 1) % 256;
                continue;
            }
        }
        /* Start Frame */
        if(lenDaten == -1){checksum=data;lenDaten= data;startIndex= curserIndex;}
        else
        {
            lenDaten = lenDaten - 1; /* Decrease Length Counter */
            checksum += data; /* Add Up Checksum */
        }

        empfangsBuffer[curserIndex] = data; /* Speichere Daten in den Buffer */
        curserIndex = (curserIndex + 1) % 256; /* Erhoehe den curser */
    }
    return false; /* Kein volles Paket empfangen */
}
```

Übung 1: Datenkommunikation Computer <-> Arduino

Erstelle ein Programm das Daten von deinem Laptop empfängt und darauf antwortet bzw. reagiert.
Benütze den Serial-Monitor um mit dem Arduino zu Kommunizieren.

Programm Serial HelloWorld:

```
////////////////////////////////////  
void setup() {  
    /* put your setup code here, to run once: */  
  
    Serial.begin(115200); /* Initialize Serial with 115200 BoudRate */  
}  
  
void loop() {  
    /* put your main code here, to run repeatedly: */  
  
    while (Serial.available() > 0) /* while there is data available repeat reading */  
    {  
        char data = Serial.read(); /* read one byte from the received data */  
  
        if (data == 'a') /* if it is the character a */  
        {  
            Serial.println("Hey abcdefgh...");  
        }  
  
        if (data == '1') /* if it is the character 1 */  
        {  
            Serial.println("Hey 1234567...");  
        }  
    }  
}  
////////////////////////////////////
```

Man kann die `if` Bedienung durch eine `switch case` Anweisung ersetzen

```
...  
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':  
            Serial.println("Hey abcdefgh...");  
            break;  
  
        case '1':  
            Serial.println("Hey 1234567...");  
            break;  
  
        default:  
            break;  
    }  
}  
...
```

Übung 2: Steuerung beider Servos per Computer (PC/Laptop)

Erstelle ein Programm das Daten vom PC/Laptop empfängt und damit die Winkel-Position der Servos steuert.

Sende die aktuellen Servo-Werte nach empfangen eines Steuer-Befehles an den PC zurück.

Folgende Steuerzeichen sollen verwendet werden:

,w' ... Bewege um 1 nach oben
,s' ... Bewege um 1 nach unten
,a' ... Bewege um 1 nach links
,d' ... Bewege um 1 nach rechts

Achte darauf das der Bereich 0 bis 180 beider Servos nicht unter oder überschritten wird.

Setup Funktion:

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

/* Definiere die minimal und maximal Position der Servos */
#define SVO_POS_MIN    0
#define SVO_POS_MAX    180
#define SVO_POS_MID    90

/* Deklarieren der Servos als globale Objekte/Variablen mit Namen myServoX und myServoY*/

Servo myServoX;
Servo myServoY;

int myServoPosX = SVO_POS_MID;
int myServoPosY = SVO_POS_MID;

void setup() {
    /* put your setup code here, to run once: */

    Serial.begin(115200); /* Initialize Serial with 115200 BoudRate */

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

Main Loop Funktion:

Ergänze den fehlenden Code!

```
void loop() {
    /* put your main code here, to run repeatedly: */

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

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

        switch (data)
        {
            case 'a':
                /* Erhoehe myServoPosX */
                /****** SCHREIBE CODE HIER *****/

                break;

            case 'd':
                /* Verringere myServoPoX */
                /****** SCHREIBE CODE HIER *****/

                break;

            case 'w':
                /* Erhoehe myServoPosY */
                /****** SCHREIBE CODE HIER *****/

                break;

            case 's':
                /* Verringere myServoPoY * /
                /****** SCHREIBE CODE HIER *****/

                break;

            default:
                break;
        }

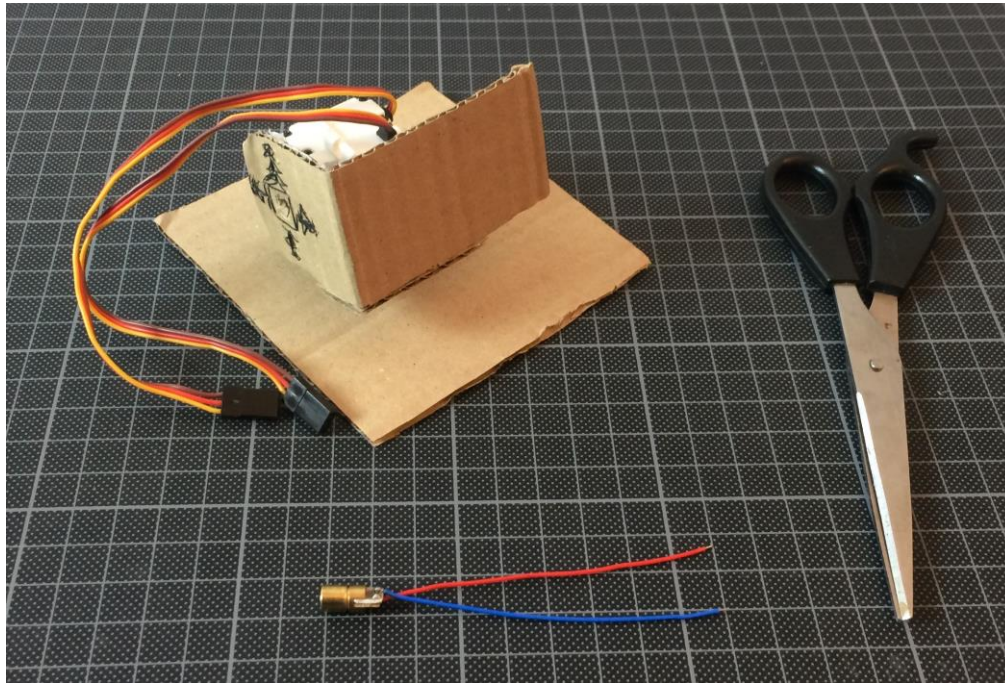
        /* Beschraenken auf min und max Grenzen */
        if ( myServoPosX < SVO_POS_MIN ) myServoPosX = SVO_POS_MIN;
        if ( myServoPosX > SVO_POS_MAX ) myServoPosX = SVO_POS_MAX;
        if ( myServoPosY < SVO_POS_MIN ) myServoPosY = SVO_POS_MIN;
        if ( myServoPosY > SVO_POS_MAX ) myServoPosY =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

        /* Sende aktuellen myServoPoX und myServoPoY Wert an den PC/Laptop */

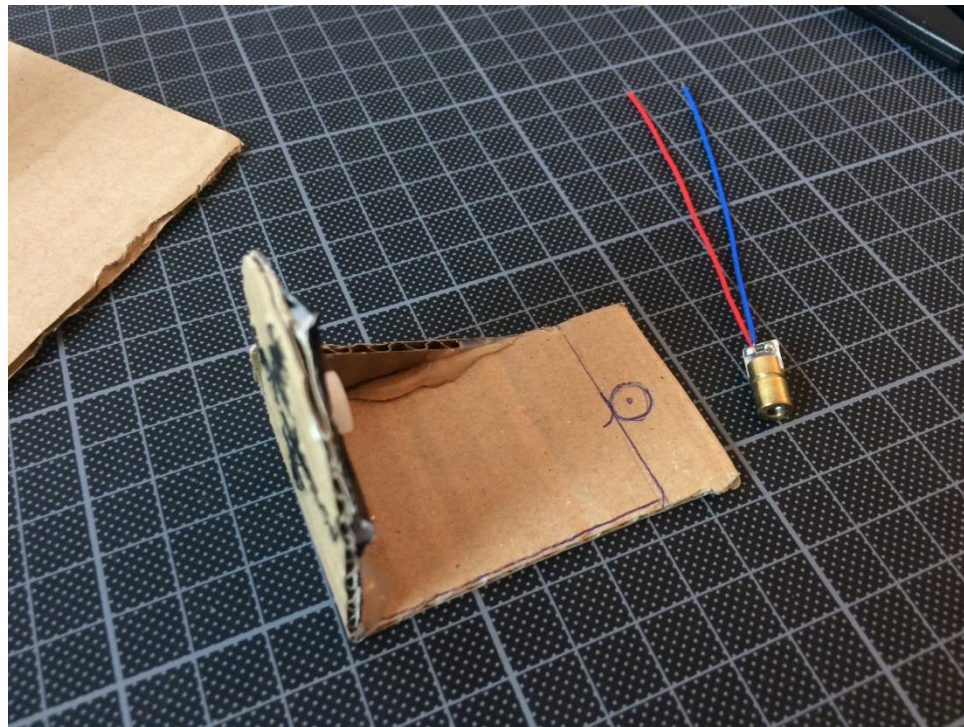
        /****** SCHREIBE CODE HIER *****/

    }
}
```


Bauanleitung: Erweitern des Roboters um eine Laserdiode



1.) Nimm die Sensor-Halterung ab. Und mach eine Markierung für das Laserdioden-Loch.



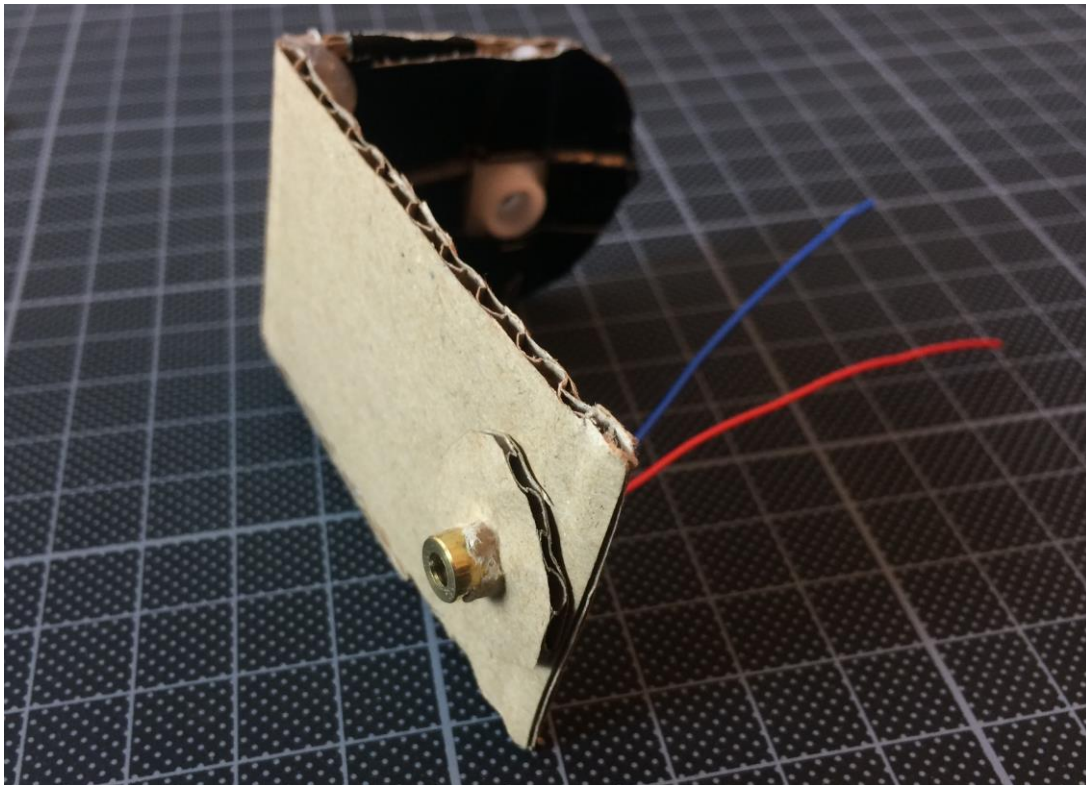
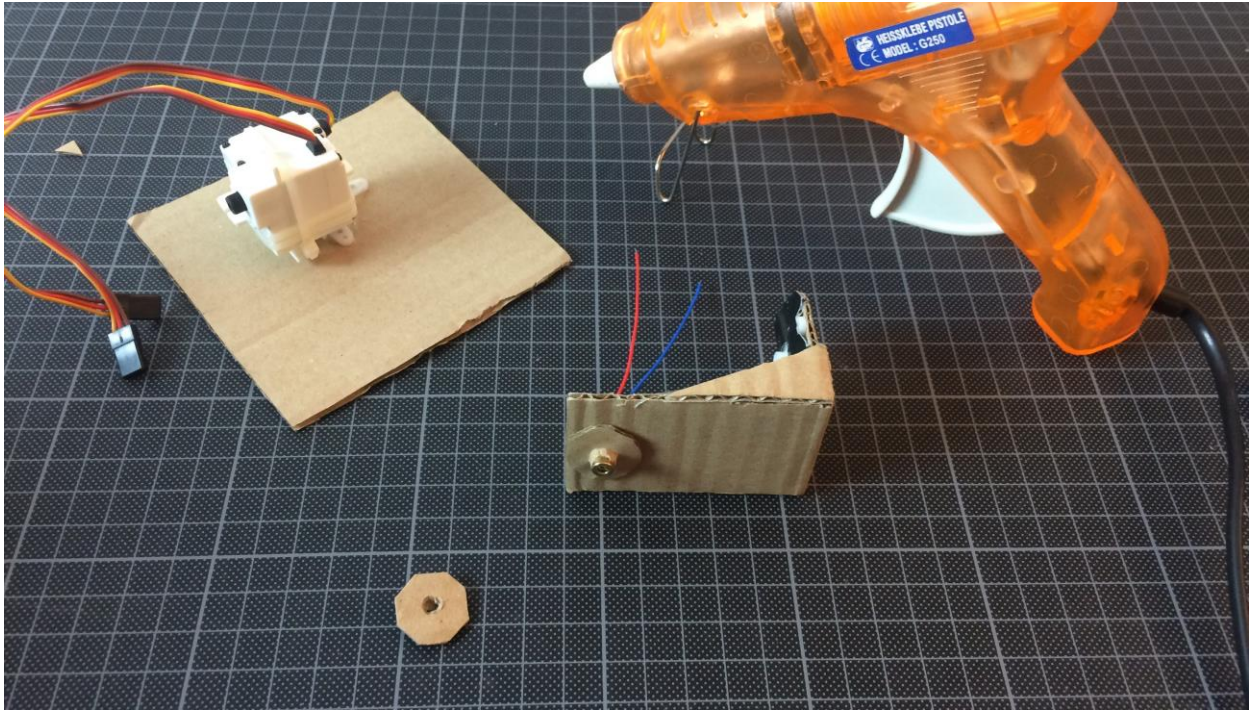
2.) Mach vorsichtig ein Loch an der Stelle der Markierung.



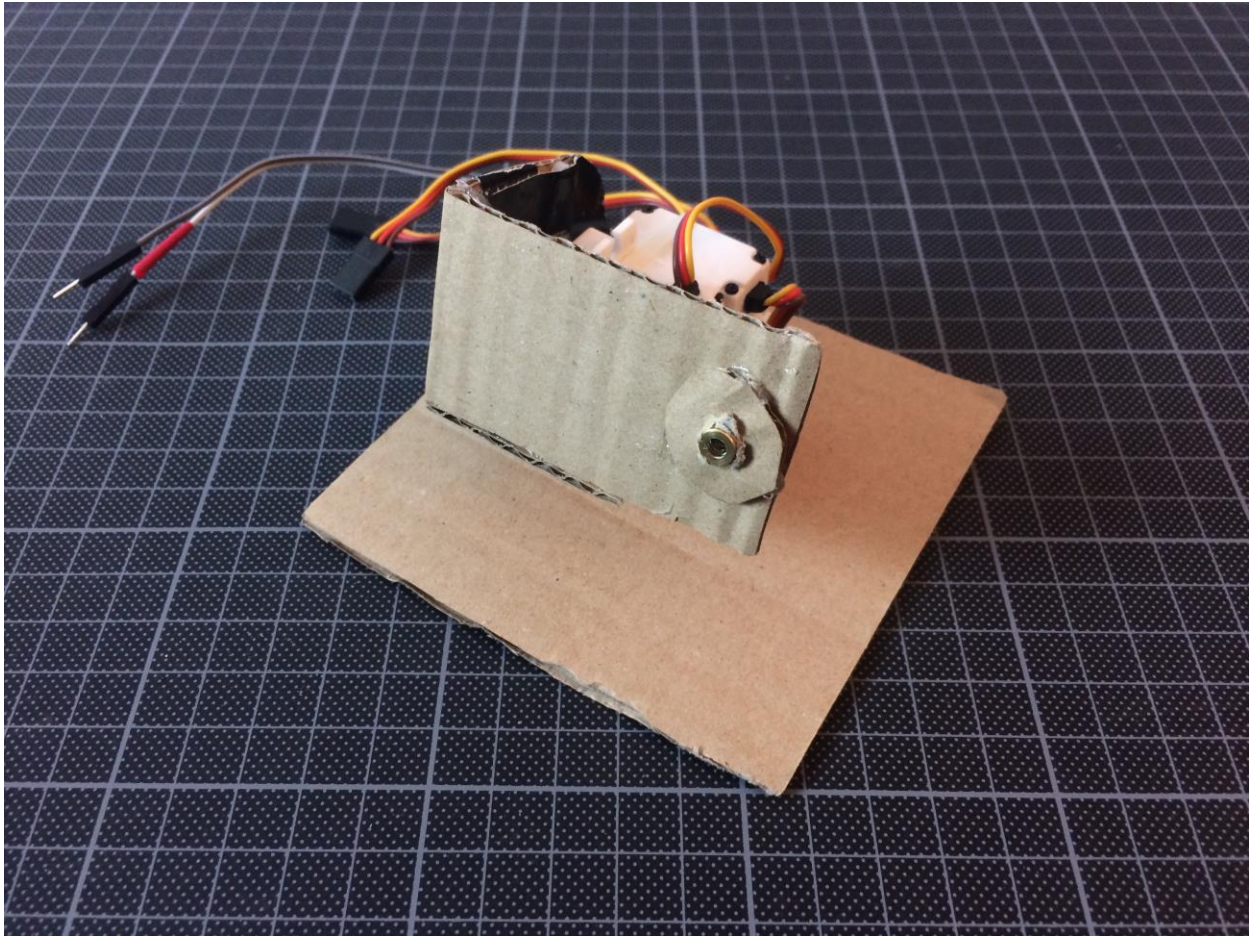
3.) Mach ein weiteres Loch in ein Stueck Karton und schneide Rundes Stueck um das Loch.



- 4.) Stecke den Laser durch die Sensor-Halterung. Und befestige das runde Kartonstück mit wenig Heißkleber an der Sensor-Halterung.



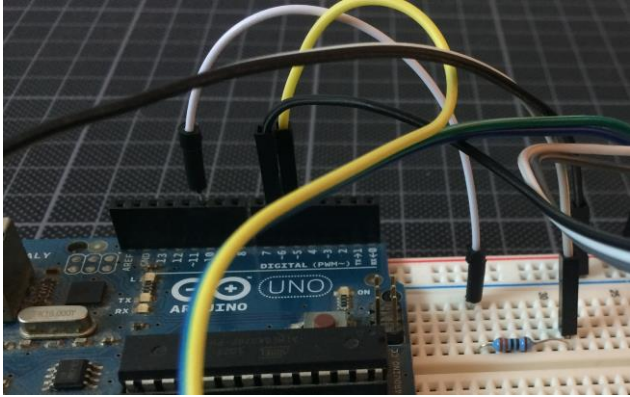
5.) Zusammenstecken der Teile



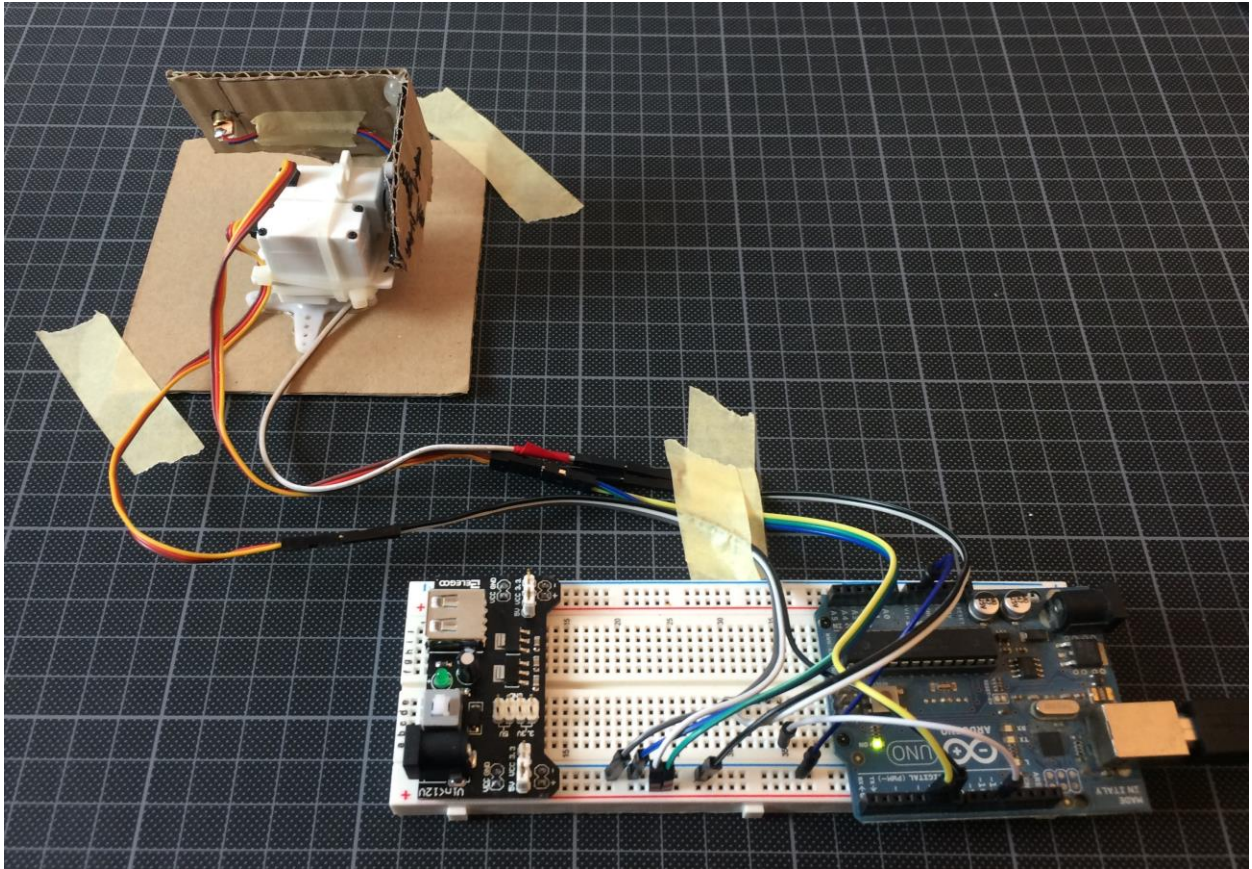
Zu guter Letzt muss nur noch alles an den Arduino angeschlossen werden!

Verkabelung:

Verbinde Servos, Laserdiode 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
Arduino Ground	Arduino Ground Pin -> Ground (blau) am Steckbrett
Laser-Diode	+ Kabel mit roter Markierung -> über Steckbrett mit einem ~330 Ohm Widerstand an den Arduino Pin 10  - Kabel ohne roter Markierung -> Steckbrett/Arduino Ground
Optionaler Joystick	Joystick-GND -> Arduino Ground Joystick-+5V -> Arduino +5Volt Joystick-VRx -> Arduino analog Pin A1 Joystick-VRy -> Arduino analog Pin A2
Stromversorgung (5Volt)	Stromversorgung minus (Leitung) -> Ground am Steckbrett Stromversorgung plus (Leitung) -> +5V am Steckbrett

Fertig Verkabelt t sollte es ungefähr so aussehen:



Befestige die Bodenplatte und Kabel mit etwas Maler-Klebeband damit es beim Bewegen nicht rutschen kann und von einem Weg zeigt.

ACHTUNG: Achte drauf das der Laserstrahl niemals ins Auge gelangen kann. Und richte ihn nicht auf andere Personen. Laserlicht kann auch bei geringer Stärke Augen schädigend sein.

Übung 3: Steuerung des Lasers per Computer (PC/Laptop)

Erweitere dein Programm so dass man den Laser Ein- und Ausschalten kann.

,1' ... Laser Ausschalten

,2' ... Laser Einschalten

ACHTUNG: Achte drauf das der Laserstrahl niemals ins Auge gelangen kann. Und richte ihn nicht auf andere Personen. Laserlicht kann auch bei geringer Stärke Augen schädigend sein.

Setup Funktion:

Erweitere die **Setup Funktion** und setze pinMode von Pin 10 auf Output und benutze digitalWrite um den Laserpointer auszuschalten.

```
/* Setzen des Laser-Pins */  
pinMode( 10, OUTPUT);  
digitalWrite( 10, LOW);
```

Loop Funktion:

Erweitere den **switch case** der loop Funktion um folgende cases:

```
case '1':  
    digitalWrite(10, LOW); /* Laser ausschalten */  
  
    break;  
case '2':  
    digitalWrite(10, HIGH); /* Laser einschalten */  
    break;
```

Übung 4: Umstellen von Servo.write() auf Servo.writeMicroSeconds()

Erweitere dein Programm so dass statt der Servo.write Funktion die Servo.writeMicroseconds Funktion benutzt wird.

```
myServoX.writeMicroseconds(myServoPosX); /* Update position of servo X */
```

Die writeMicroseconds ermöglicht es in feineren Schritten zu fahren und hat eine höhere Auflösung.

Der Bereich von writeMicroseconds 500 bis 2500 statt bei write von 0 bis 180.

Setup Funktion:

Ändere die Makros SVO_POS_MIN, SVO_POS_MAX, SVO_POS_MID auf:

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

Loop Funktion:

Ändere die Servo.write Funktionen in der loop Funktion auf writeMicroseconds.

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

Nun kann man genauer den Laser steuern.

Übung 5: Erstelle eine Funktion um linear von A nach B zu fahren.

Erweitere dein Programm um eine Funktion um möglichst linear von A nach B zu Fahren.

Die Funktion bekommt die Zielwerte der Servos und Zeit in Millisekunden als Parameter.

Erweitere dann dein Programm um Befehle 0,9,8,7 um zwischen Verschieden Punkten zu fahren.

Neue Funktion:

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

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

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

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

Loop Funktion:

Erweiter den `switch case` um Folgende Cases:

```
case '0':
    DriveLinear(1500, 1500, 1000);
    break;
case '9':
    DriveLinear(1300, 650, 1000);
    break;
case '8':
    DriveLinear(1300, 800, 1000);
    break;
case '7':
    DriveLinear(1700, 800, 1000);
    break;
case '6':
    DriveLinear(1700, 650, 1000);
    break;
case '5':
    DriveLinear(1300, 650, 1000);
    break;
```

Nun kann man linear zwischen Punkten fahren.

Übung 6: Erweitern der Cases um Buchstaben zu „Zeichen“

Erstelle einen case um den Buchstaben A zu Zeichen.

Dabei soll der Laser nur dann aktiv sein wenn er eine Linie des Buchstabens abfährt.

Loop Funktion:

Erweiter den `switch case` um Folgende Cases:

```
        case 'A':  
/***** MODIFIZIERE CODE HIER *****/  
        digitalWrite(10, LOW);  
        DriveLinear(1500, 1500, 500);  
        digitalWrite(10, HIGH);  
        DriveLinear(1300, 650, 500);  
        digitalWrite(10, LOW);  
        DriveLinear(1300, 800, 500);  
        digitalWrite(10, HIGH);  
        DriveLinear(1700, 800, 500);  
        digitalWrite(10, LOW);  
        DriveLinear(1700, 650, 500);  
        digitalWrite(10, HIGH);  
        DriveLinear(1300, 650, 500);  
        digitalWrite(10, LOW);  
        break;
```

Nun kann man per Befehl ein Muster abfahren.

Übung 7: Umrechnen von Position auf Ebene nach Servo-Winkel-Werte

Mit folgenden Änderungen und Funktionen können wir Koordinaten einer 2D Ebene auf die entsprechenden Servo-Winkel-Werte zurück rechnen. Ein bisschen **Sin/Cos Trigonometrie** reicht für die Mathematik. Vervollständig dann den `case` um den Buchstaben M zu zeichnen.

```
/* ////////// FUEGE DIESE ZEILEN GLOBAL AM ANFANG DES PROGRAMM CODES EIN ////////// */

float distPlane = 195; // Distance der Flaeche (mit Rechtenwinkel X Y)

int PWM_Servo1_ZERO = 1500;
int PWM_Servo2_ZERO = 660;
int PWM_180 = (900 * 2);

float XOffset = 0;
float YOffset = 0;

/* ////////// MODIFIZIERE initial Wert von myServoPosX und myServoPosY ////////// */

int myServoPosX = PWM_Servo1_ZERO;
int myServoPosY = PWM_Servo2_ZERO;

/***** Compute Angels From Positions *****/
void ComputeAngles(int &servo1, int &servo2, float X, float Y)
{
    X += XOffset;
    Y += YOffset;

    float distX = sqrt((X * X) + (distPlane * distPlane));
    float alphaX = asin(X / distX);

    float distY = sqrt((Y * Y) + (distX * distX));
    float alphaY = asin(Y / distY);

    float distTotal = distY;

    int PWM_Servo1 = PWM_Servo1_ZERO + (alphaX / PI) * (float)PWM_180;
    int PWM_Servo2 = PWM_Servo2_ZERO + (alphaY / PI) * (float)PWM_180;

    servo1 = PWM_Servo1;
    servo2 = PWM_Servo2;
}

/***** Compute Position by Angles *****/
void ComputePosByAngle(float &PosX, float &PosY)
{
    float Angle1 = ((float)myServoPosX - (float)PWM_Servo1_ZERO) / (float)PWM_180 * PI;

    float Angle2 = ((float)myServoPosY - (float)PWM_Servo2_ZERO) / (float)PWM_180 * PI;

    PosX = tan(Angle1) * distPlane - XOffset;

    PosY = tan(Angle2) * sqrt(PosX * PosX + distPlane * distPlane) - YOffset;
}
```

```

/***** Draw Functions *****/
void DriveLinearXY(float TargetPosX, float TargetPosY, int TimeMS)
{
    int timePast = 0;

    float tmpStartedX = 0;
    float tmpStartedY = 0;

    ComputePosByAngle(tmpStartedX, tmpStartedY);

    while (timePast < TimeMS)
    {
        float tmpPosX = tmpStartedX + ((float)(TargetPosX - tmpStartedX) *
                                         (float)timePast / (float)TimeMS);
        float tmpPosY = tmpStartedY + ((float)(TargetPosY - tmpStartedY) *
                                         (float)timePast / (float)TimeMS);

        ComputeAngles(myServoPosX, myServoPosY, tmpPosX, tmpPosY);

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

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

void DrawCircel(float PosX, float PosY, float Radius, int Time)
{
    int timePast = 0;

    for (int i = 0; i < Time; i++)
    {
        float tmpPosX = sin(2 * PI / (float)Time * (float)i) * Radius + PosX;
        float tmpPosY = cos(2 * PI / (float)Time * (float)i) * Radius + PosY;

        ComputeAngles(myServoPosX, myServoPosY, tmpPosX, tmpPosY);
        myServoX.writeMicroseconds(myServoPosX); /* Update position of servo X */
        myServoY.writeMicroseconds(myServoPosY); /* Update position of servo Y */
        delay(1);
    }
}

```

```
/****** ERWEITERE DEN KOMMANDO SWITCH CASE IN DER LOOP FUNCTION *****/
```

```
case 'q':
    digitalWrite(10, LOW);
    DriveLinearXY(0, 0, 500);
    digitalWrite(10, HIGH);
    DriveLinearXY(20, 0, 2000);
    DriveLinearXY(20, 20, 2000);
    DriveLinearXY(0, 20, 2000);
    DriveLinearXY(0, 0, 2000);
    break;

case 'o':
    XOffset = 0; YOffset = 0;
    ComputePosByAngle(XOffset, YOffset);
    break;

case 'z':
    PWM_Servo1_ZERO = myServoPosX;
    PWM_Servo2_ZERO = myServoPosY;
    break;

case 'x':
    digitalWrite(10, LOW);
    DriveLinearXY(0, 0, 100);
    digitalWrite(10, HIGH);
    DriveLinearXY(20, 20, 500);
    digitalWrite(10, LOW);
    DriveLinearXY(0, 20, 100);
    digitalWrite(10, HIGH);
    DriveLinearXY(20, 0, 500);
    break;

case 'k':
    DrawCircel(20, 20, 20, 1000);
    break;

case 'K':
    DrawCircel(20, 20, 20, 4000);
    break;

case 'M':

/***** SCHREIBE CODE HIER UM DEN BUSCHSTABEN M ZU ZEICHEN *****/
    break;
```