SCHULE SARNEN

PROJEKTARBEIT

EINGEREICHT BEI: MARGRIT WIRZ

WANDDRUCKER

4. MAI 2016

DONATO WOLFISBERG
IOS.3D
Brunnmattweg 12 , 6060 Sarnen

Inhalt

1 Einleitung

1.1 Thema meiner Arbeit

Ich habe als Abschlussarbeit selbst einen Wanddrucker gebaut und die nötigen Programme dafür geschrieben.

1.2 Motivation

Ich arbeite gerne am PC. Ich mache am PC nicht nur Spiele, sondern mache auch gerne kleine Programme, um so Probleme zu lösen. Mit dieser Abschlussarbeit konnte ich mehr über das Programmieren und die Programmiersprache Python lernen.

Im Sommer fange ich eine Lehre als Applikationsentwickler bei der CSS an. Mit diesem Projekt konnte ich mir bestimmt Wissen aneignen, das mir in der Lehre nützlich sein wird.

1.3 Ziele der Arbeit

Ich habe mir die folgenden Ziele gesetzt:

1. Ich baue ein einfaches GUI[1], mit dem man alles steuern kann und das einfach bedienbar ist.
2. Das hergestellte Programm kann die Pixel aus einem Bild herauslesen.
3. Ich baue eine Konstruktion mit 2 Motoren und einem Druckkopf auf einem Holzbrett.
4. Mein Programm sollte ein Bild möglichst genau auf das Blatt zeichnen können.

1.4 Voraussetzungen

1.4.1 Wissen

Das Projekt setzt ein Grundwissen in der Programmierung voraus. Bis anhin habe ich mit folgenden Programmiersprachen gearbeitet: LUA, Python, HTML, C#. Ich habe

[1] GUI = graphisches Benutzerinterface

mir Python, HTML und CSS auf der Webseite codecademy[2] angeeignet. Auf dieser Webseite kann man schrittweisse viele Programmiersprachen wie z.B. Python, SQL, Java lernen und sich Wissen über APIs[3] aneignen.
Meine Motivation, Python zu lernen, war das Geschenk zu meinem 15. Geburtstag: ich erhielt einen raspberry pi. Das ist ein Kleinstcomputer von der Grösse einer Kreditkarte, auf dem ein Unix läuft. Dieser besitzt zur Steuerung viele Anschlüsse.

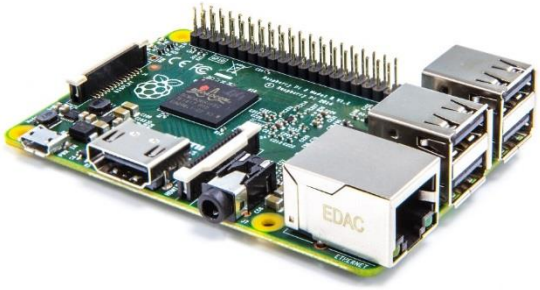


Abbildung 1: Raspberry Pi 2

Ich habe mit dem raspberry pi und DC-Motoren sowie einem Ultraschallsensor ein Auto gebaut, das selbst fahren kann und Hindernissen ausweicht. Mit diesem Projekt konnte ich mein Grundwissen in der Praxis anwenden und vertiefen.

1.4.2 Material

Ich brauchte in materieller Hinsicht einen raspberry pi, 2 Stepper-Motoren und ein Elektromagnet für den Druckkopf. Zudem brauchte ich eine Holzplatte, um die ganze Konstruktion darauf zu befestigen.

1.5 Vorgehen

Ich wollte mein Programm objektorientiert mit Klassen und Methoden aufbauen. Jede Klasse besteht aus verschiedenen Methoden und kann sich eigene Variablen merken, wenn sie erzeugt worden ist. Eine Klasse kann auch mehrmals erzeugt

[2] https://www.codecademy.com

[3] APIs: Anwendungsprogrammierschnittstellen

werden, zum Beispiel die Klasse für den Motor, welche einmal für den linken und einmal für den rechten Motor erzeugt wurde. Jeder Motor weiss nun zum Beispiel, um wieviel Grad er sich bereits gedreht hat. Die Methoden dienen dazu, Informationen einer Klasse abzurufen wie z.B. *linkerMotor→zeigeGrad(x)* oder etwas zu machen wie z.B. *rechterMotor→dreheGrad(180).*

Zuerst habe ich überlegt, welche Struktur das Programm haben muss und wie die Klassen miteinander kommunizieren müssen. Viele Funktionen, wie die grafische Oberfläche (GUI[4]) sind für Python als Libraries verfügbar und müssen nicht selbst programmiert werden.

Als ich das gemacht hatte, musste ich schauen, was ich alles an Material brauche. Ich habe zuerst ein einfaches Gerüst für die Motoren gebaut, wo ich die Steuerung der Motoren testen konnte. Nachher habe ich sie auf einem Brett montiert.

1.6 Dank

Ich danke meiner Klassenlehrerin Frau Margit Wirz, dass sie das Projekt bewilligt hat und meine Arbeit begleitete.
Ich danke meinem Vater, dass er mich mit seinem Fachwissen unterstützt und gecoacht hat.
Ich danke meiner Mutter fürs Korrigieren dieses Textes.

2 Beruf Informatiker

Ich mache im August eine Lehre als Informatiker, das heisst als Applikations-entwickler bei der CSS Krankenkasse in Luzern. Ich mache gerne kleinere Projekte mit dem Computer und bin auch sonst sehr technikbegeistert. Daher war es für mich auch schon schnell klar, dass ich diese Lehre machen will.

[4] GUI: Graphical User Interface - Grafische Benutzeroberfläche

3 Herstellung meines Produktes

3.1 Planung

Ich habe zuerst geschaut, was mit meinem Wissen und in der vorgegebenen Zeit etwa machbar wäre. Dies war ziemlich schwierig für mich, da ich noch nie ein solches Projekt gemacht hatte. Als ich wusste, was und wie ich es machen will, fing ich an zu schauen, was ich genau brauchen würde und was ich noch kaufen muss.

3.2 Material beschaffen

Den raspberry pi, mit dem ich das Projekt machte, hatte ich bereits. Ich hatte ihn schon vorbereitet: Das Betriebssystem raspian war installiert und alle notwendigen Programme zum Entwickeln heruntergeladen.

Am Anfang wollte ich den Druckkopf mit kleinen 5V 4-Phasenmotoren heben und senken. Aber dann hatte ich gemerkt, dass diese zu langsam und zu schwach sind. Deshalb hatte ich 5V 2-Phasenmotoren gekauft. Diese sind schneller und stärker, aber weniger genau. Die 4-Phasenmotoren haben 512 Schritte pro Umdrehung, das heisst, ich konnte sie auf 0.7 Grad genau steuern. Die 2-Phasenmotoren haben jedoch nur 212 Schritte pro Umdrehung. Darum kann ich sie nur auf 1.7 Grad genau steuern. Dies ist aber immer noch ziemlich genau.

Abbildung 2: Alter und neuer Steppermotor

Abbildung 3: Servomotor

3.3 Kosten

Den raspberry pi hatte ich bereits zuhause. Ebenso hatte ich bereits die nötigen Kabel, die kleinen Steppermotoren und Breadboards.

Die beiden grösseren 2-Phasen-Motoren und die zugehörige Steuerungsplatine, welche ich kaufen musste, da meine bestehenden zu schwach waren, kosteten zusammen 80 Franken.

3.4 Programmieren

3.4.1 Planung der Klassen und Funktionen

Am Anfang habe ich die Abhängigkeit der Programme in einer PowerPoint-Präsentation festgelegt. Dies half mir sehr.

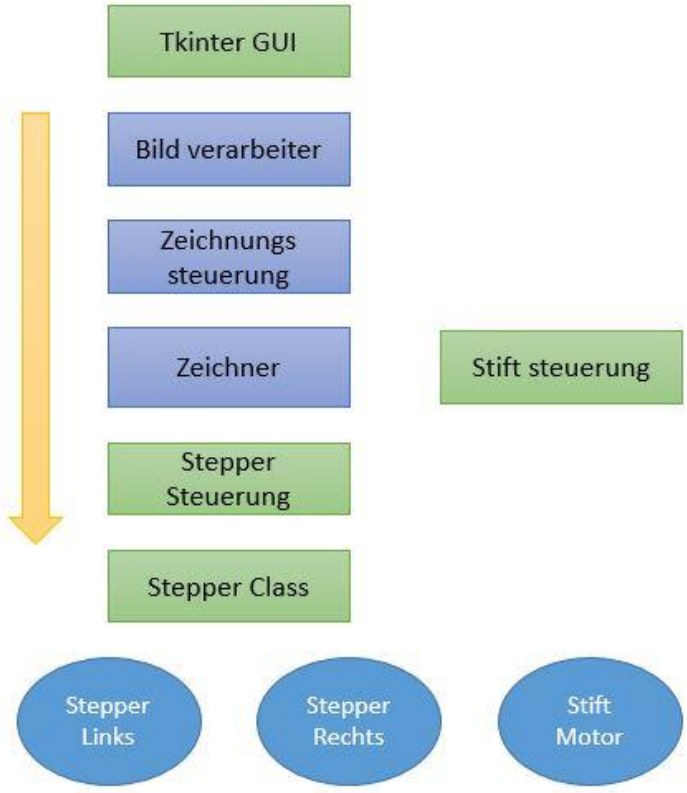


Abbildung 4: Programm Kommunikation Verbildlichung

3.4.2 Erstellung GUI-Klasse (Tkinter GUI)

Die GUI-Klasse öffnet ein Windows-Fenster (GUI) mit dem man ein Bild laden und den Zeichner starten kann.

Als ich wusste, wie die Programme miteinander kommunizieren sollen, machte ich als erstes die *Stepper-Klasse*, welche beide Motoren um 360° dreht. Dann habe ich

ein kleines, einfaches GUI gemacht, wo man eingeben kann, wie viel die Motoren drehen sollen. Das GUI machte ich mit der Python Library namens Tkinter.

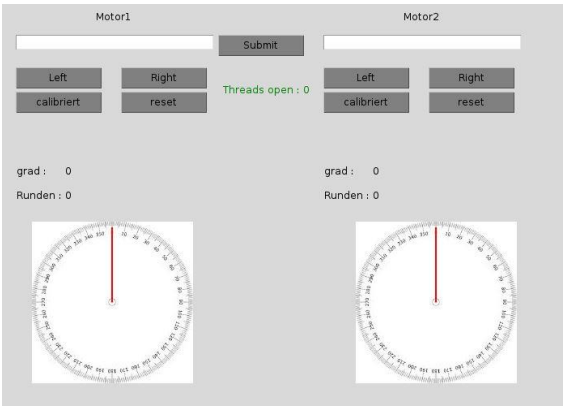


Abbildung 5: einfaches GUI

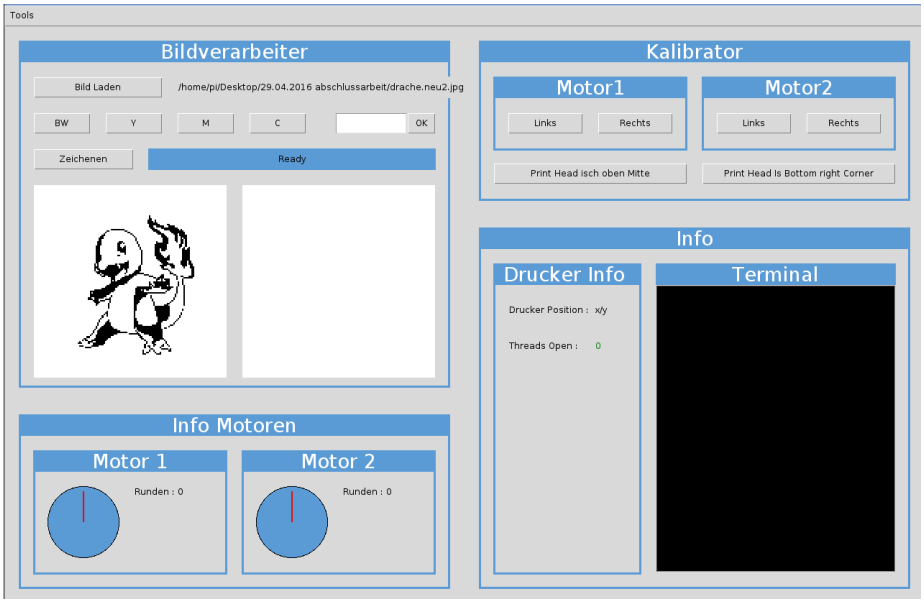


Abbildung 6: Fertiges GUI

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *  #Lberary für GUI
from stepper import steppermotor #Class steppermotor für einzelne Steuerung der Motore
import stepper_steuerungen #
from threading import Timer
from threading import Thread
from threading import activeCount
from thread import start_new_thread
import sys
from time import sleep
from math import *
from Bildverarbeiter import Bildverarbeitung
from Zeichnungs_steuerung import Zeichnungssteuerung
import MotorCalibrator
from MotorCalibrator import Farbenundschriften
import os
import datetime
import atexit
from servo import servoClass
from StaticParameters import statischeParameter
```

```
def Motordreh(pMotor1dreh, pMotor2dreh):...

def stepperreset(pStepper):...

def linemover(pline, pcan, pgrad):...

def changetext():...

def Imgpathopen():...

def imgdrawcanvasx():...

def ZeichneBild():...

def setbwscale(pbwscale):...

def setCOL(pCOL):...

#~ starte MotorCalibratorWindow und gibt schrift und farbe
def MotorCalibratorWinMain():...

def MotorCalibratorWinFarbe():...

#~ Returns schrift und farbe wen extern gestarted
def FarbenundschriftenMain():...


def IstKalibriert():...

def savefile():...
```

Abbildung 7: Struktur der GUI-Klasse

3.4.3 Stepper-Steuerungsklasse

Die Motoren-Steuerungsklasse ist dafür zuständig, dass sich die beiden Motoren um jeweils eine bestimmte Anzahl Grad drehen. Sie muss auch dafür sorgen, dass beide Motoren etwa gleichzeitig fertig sind, auch wenn sie verschiedene Gradzahlen haben. Sonst gäbe es keine gerade Linie zwischen zwei Punkten, sondern eine „eckige Linie“.

Sie kontrolliert die beiden Steppermotoren via die Stepper-Klasse und bewegt diese unterschiedlich schnell, sodass am Ende beide gleichzeitig fertig sind.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from stepper import steppermotor
from time import sleep ## Import 'time' library. Allows us to use 'sleep'
#from threading import Thread
#import thread
from threading import *
from StaticParameters import statischeParameter


def machMotor1():...

def machMotor2():...

def stepperclean(pMotor):...

Maxspeed = statischeParameter.Speed * 360 / 60 #grad in Minute
AnzStepps = 200

def drehe2(pMotor1,pGradM1, pMotor2, pGradM2):...
```

Abbildung 8: Struktur der Stepper-Steuerungsklasse

3.4.4 Stepper-Klasse

Die Stepper-Klasse steuert einen Motor und bekommt die Anzahl Grad und die Geschwindigkeit. Sie gibt die Befehle an den Motor-Hat, der dann die Impulse für den Motor erzeugt. Ein Motor-Hat ist eine Erweiterungsplatine, welche die Pulsweitensignale für die Steuerung der Motoren viel genauer als der raspberry pi und auch mit 5V erzeugen kann.

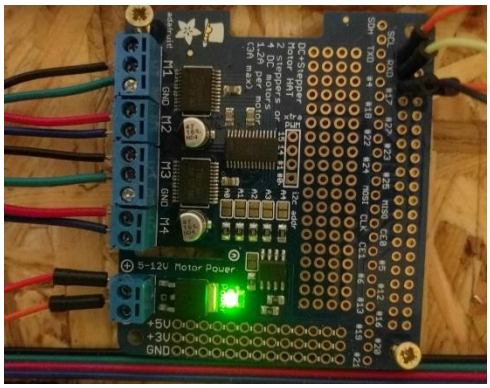


Abbildung 9: Adafruit DC & Stepper Motor HAT

Ich hatte zwei Motoren: links und rechts. Ein Vorteil der Arbeit mit Klassen ist, dass ich die Klasse nur einmal schreiben und dann für beide Motoren brauchen kann. Man nennt das „eine Instanz anlegen“. Jede Instanz kennt seinen Motor und hat auch ein Gedächtnis und weiss zum Beispiel, wo er steht.

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO ## Import GPIO library
from time import sleep ## Import 'time' library. Allows us to use 'sleep'
from Adafruit_MotorHAT import * ## import Shield Liberary
import atexit
from StaticParameters import statischeParameter

anzStepper = -1
Speed = statischeParameter.Speed       # 30 RPM

class steppermotor:
    # # A=0
    # # B=0
    # # C=0
    # # D=0

    def __init__(self,pMNum):...


    # grad = raw_input("wie viel grad: ")
    def nullstellung(self):...

    def stepperinfoGrad(self):...

    def stepperinfoRunden(self):...

    def stepperreset(self):...

    def drehe(self, grad, pSpeed):...

    def stepperclean(self):...

if __name__ == "__main__":
    stepper1 = steppermotor(1)
    stepper2 = steppermotor(2)
    #~ atexit.register(stepper1.stepperclean)
    #~ atexit.register(stepper2.stepperclean)

    stepper1.drehe(360)
    stepper1.drehe(-360)
    stepper2.drehe(360)
    stepper2.drehe(-360)
    stepper1.stepperclean()
    stepper2.stepperclean()
```

Abbildung 10: Struktur der Stepper.Klasse

3.4.5 Verschönerung und Verbesserung des GUIs

Dann verbesserte ich das GUI, um die aktuellen Grad-Werte der Motoren als zwei Tachos anzuzeigen. Ich versuchte es mit einer Funktion aus der Tkinter Library namens canvas. Unter einem canvas kann man sich eine Leinwand vorstellen, auf die man malen kann. Ich habe in diesem GUI getestet, wie sie funktionieren. Die Aktualisierung der canvases und der Texte, Gradwerte und Runden verarbeitete ich in einem Thread. So kann ich sie gleichzeitig laufen lassen.

3.4.6 Bildverarbeiter-Klasse

Die Bildverarbeiter-Klasse kann ein Bild aus der SD-Karte einlesen und analysieren. Sie schreibt das Resultat in einen Array, das ist wie ein Koordinatensystem mit x und y. Jedes Kästchen entspricht einem Pixel, das gezeichnet werden muss.

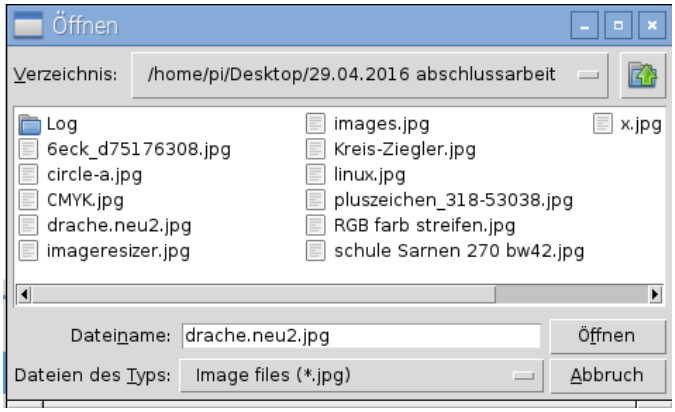


Abbildung 11: Dialog zum File auswählen

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from PIL import Image
import PIL
import Tkinter
import tkFileDialog
from StaticParameters import statischeParameter


breite = 128


class Bildverarbeitung:
    #~256
    def __init__(self):...

    def ImgOpen(self):...

    def Imgconverter(self):...

    def infoImgArray(self):...

    def infoblackwhite(self):...

    def setblackwhite(self, pblackwhite):...

    def infoFilePath(self):...

    def infoBildgeladen(self):...

    def infoBasewidht(self):...

    def infoBaseheight(self):...

    def infoColor(self):...

    def setColor(self, pCOL):...

    #Img in konsole schreiben test
    def imgtestprint(self):...
```

Abbildung 12: Struktur der Bildverarbeiter-Klasse

3.4.7 Zeichnungssteuerung-Klasse

Die Zeichensteuerung-Klasse erhält das Array mit dem Koordinatensystem vom Bildverarbeiter und muss zuerst einen Weg finden, wie er alle Pixel zeichnen kann, dies möglichst an einem Stück und ohne zu grosse leere Umwege zu machen. Er macht das in Runden. Er fängt mit eins an und beim Kreisbild, das unten abgebildet ist, hat er nach 74 Runden den Punkt in der Mitte gemalt.

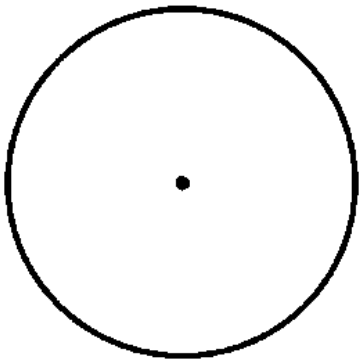


Abbildung 13: Kreis Originalbild

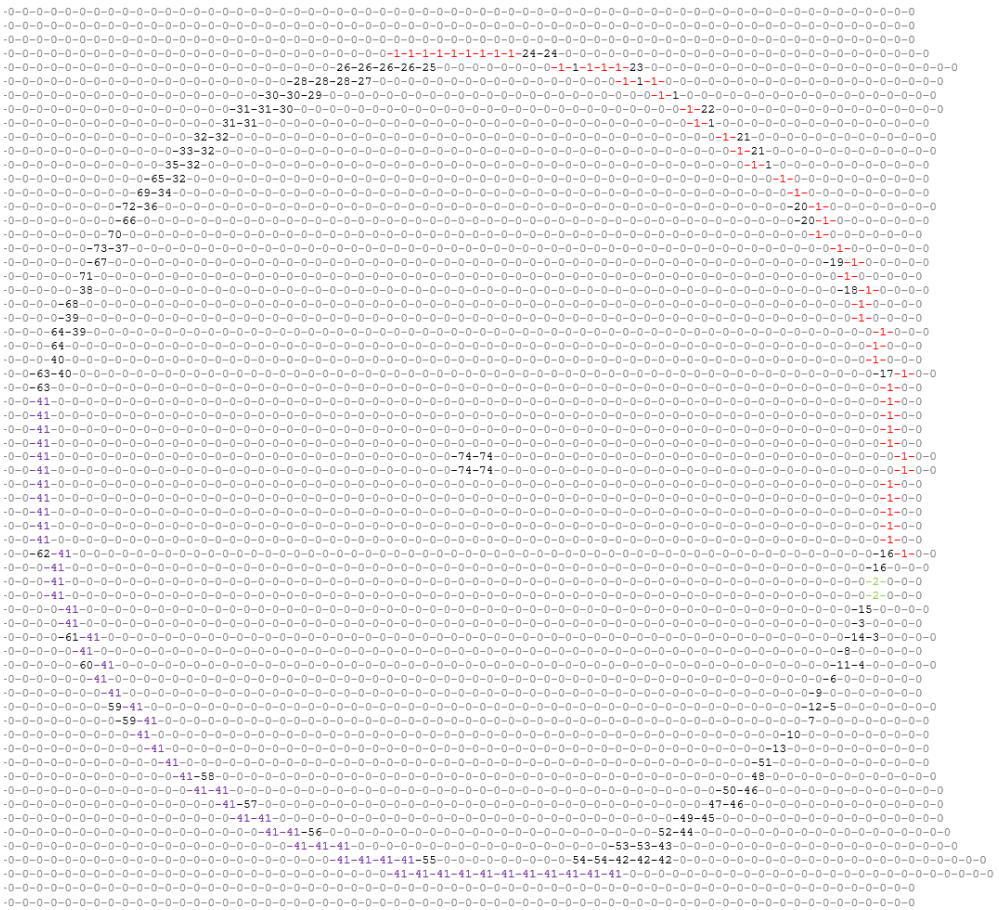


Abbildung 14: Auszug aus Log-File mit Runden

Etwas Spezielles, was ich hier gebrauchen konnte, war Rekursion. Das heisst, eine Methode ruft sich selbst immer wieder auf. Ich brauchte sie, um immer wieder einen Nachbarn zum Zeichnen zu finden, bis es keinen mehr gab.

```
def FindeNachbar(pix,piy):

    #~ print(str(runde) + " " + str(pix) + " " + str(piy))          [self.instrZeile]

    def PixelZuZeichen(pix,piy):
        if pix < 0 or piy < 0 or piy > im_y -1 or pix > im_x - 1: #ist pixel auserhalb des bildes
            return False
        elif im_array[pix][piy] == 9999:
            return True
        else:
            return False

    for i in range(1,5):
        for ix in range(2*i+1):
            for iy in range(2*i+1):
                if abs(ix-i) == i or abs(iy-i) == i:
                    istneu = PixelZuZeichen(pix+ix-i,piy+iy-i)

                    if istneu:
                        fillinstr(pix+ix-i, piy+iy-i)
                        FindeNachbar(pix+ix-i, piy+iy-i)
```

Abbildung 15: FindeNachbar Rekursion

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Bildverarbeiter import Bildverarbeitung
from Zeichner import ZeichnerClass
from time import sleep
from servo import servoClass
from StaticParameters import statischeParameter
import sys

sys.setrecursionlimit(10000) # 10000 is an example, try with different values

class Zeichnungssteuerung:
    def __init__(self,pstepper1,pstepper2,pservo):...

    def MacheInstruktionsTabelle(self,pbildverarbeiter,IstKalibriert):...


    def ZeicheBild(self,pIstKalibriert):...

    def GeheZuNull(self):...
```

Abbildung 16: Struktur der Zeichensteuerungsklasse

3.4.8 Zeichner-Klasse

Die Zeichner-Klasse muss den Druckkopf zu einer bestimmten Koordinaten bewegen können. Er muss am Anfang kalibriert werden, damit er weiss, wie lange die beiden Schnüre zum Motor1 und Motor2 sind. An diesem Punkt ist a und b immer gleich und er kann die Länge von c mit dem Pythagoras ausrechnen.

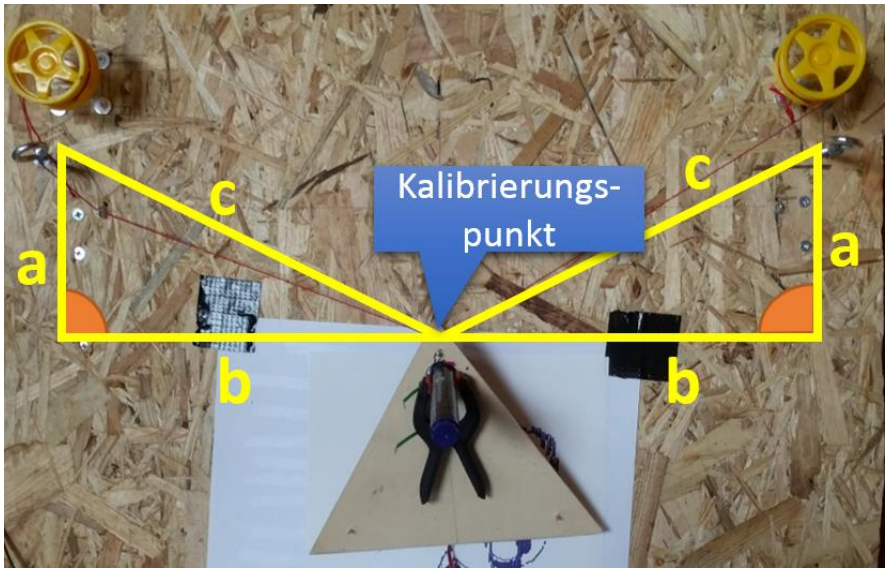


Abbildung 17: Druckkopf mit Motoren und Pythagoras-Formel

Wenn er den Befehl erhält, zur Koordinate x und y zu gehen, dann rechnet er mit dem Pythagoras die neue Länge c der Schnüre aus. A und b kennt er aus den Konstanten und den Koordinaten. Der Umfang der Spulen ist definiert und er kann nun ausrechnen, um wieviel Grad er jeden Motor drehen muss.

Die beiden Gradwerte übergibt er der Stepper-Steuerung.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import stepper_steuerungen
import sys
from time import sleep
from math import *
from servo import servoClass
from StaticParameters import statischeParameter


class ZeichnerClass:

    def __init__(self, pMotor1 , pMotor2,pservo):...


    def RechePyti(self,pA,pB):...

    def StiftHebenSenken(self,pZ):...

    def Nullpunkt(self):...

    def GeheZu(self,pX,pY,pZ):...
#if __name__ == "__main__":
```

Abbildung 18: Struktur der Zeichner-Klasse

3.4.9 Stiftheberklasse

Diese Klasse muss den kleinen Servomotor, der unter dem Druckkopf angemacht ist, steuern und den Stift von der Wand wegheben, wenn er eine neue Linie anfangen muss.

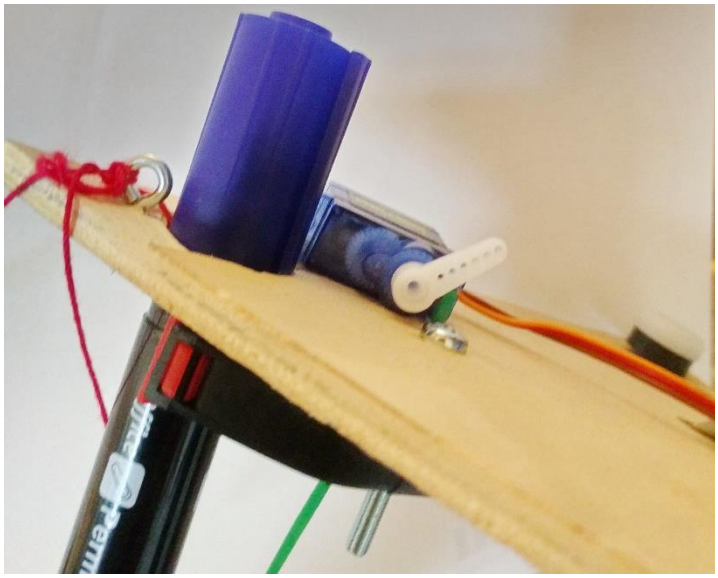


Abbildung 19: Druckkopf von unten mit Servomotor für Stiftheber

Das wollte ich zuerst mit einem Elektro-Magnet machen, aber dieses war zu schwer und zu langsam.

Auch der Servo ist nicht optimal, da er genaue Impulse braucht und der Raspberry keine sehr genaue Uhr hat. Dadurch zitterte er manchmal. Hier könnte ich es sicher noch verbessern, aber ich hatte keine Zeit mehr.

3.4.10 Motorkalibrator

Die Motorkalibrator erstellt ein kleines GUI, bei dem man eingeben kann, wieviel die Motoren drehen sollen. Dies war am Anfang sehr nützlich, um den Druckkopf schnell zu positionieren.

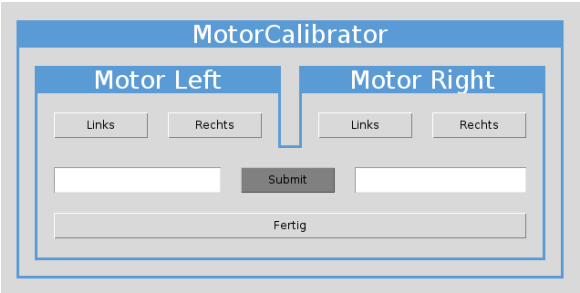


Abbildung 20: Motorkalibrator-GUI

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
import stepper_steuerungen
from stepper import steppermotor
from threading import Timer
from threading import Thread
from threading import activeCount


def Motordreh(pMotor1dreh, pMotor2dreh):...


def MotorCalibratorWin():...

#~ holt die farben
def Farbenundschriften(pBackGroundColor,pTitlefont,pstepper1,pstepper2,*args):...


if __name__ == "__main__":
    import Schoenes_GUI_teil_2
    BackGroundColor, Titlefont, stepper1, stepper2 = Schoenes_GUI_teil_2.FarbenundschriftenMain()
    MotorCalibratorWin()
```

Abbildung 21: Struktur der Motorkalibrator-Klasse

3.4.11 Testen

Als ich das erste Mal das ganze Programm mit Druckkopf laufen liess, war ich etwas irritiert, denn das Resultat sah nicht ganz so aus, wie das Vorbild. Es hätte ein Sechseck sein sollen.

Abbildung 22: Mein erster Ausdruck

Erst nach langem Suchen im Code fand ich die Fehler: An verschiedenen Orten passierten Rundungsfehler beim Berechnen der Koordinaten und Längen. Denn beim Entwickeln hatte ich immer mit ganzen Zahlen getestet. Beim Bildzeichnen gibt es aber Werte mit Nachkommastellen. Was es noch schlimmer machte war, dass die

Berechnungen mehrere hundert Mal aufgerufen wurden und sich die kleinen Fehler zu grossen Fehlern entwickelten. Das nächste Mal würde ich genauer mit dem Runden aufpassen.

Der Fehler ist mir so passiert:

Beim Programmieren gibt es zwei grundlegende Zahlentypen:

- Integer, die haben keine Kommastellen
- Floate, die haben Kommastellen.

Wenn man in python3 die Zahl 1 durch die Zahl 2 teilt, dann gibt es das, was man sich wahrscheinlich denken würde, nämlich 0.5. Aber wenn man es in python2.7, in der Version, in der ich mein Projekt gemacht habe, so macht, gibt dass 0. Dies ist so, weil es zwei Integer-Zahlen sind. Wenn man es weiss, ist dies aber ganz einfach zu beheben, indem man die hintere Zahl in einen Floate verwandelt. Dies würde in Python so aussehen.

```
X = Dividend / float(Divisor)
```

Nun sah das Resultat schon etwas besser aus.

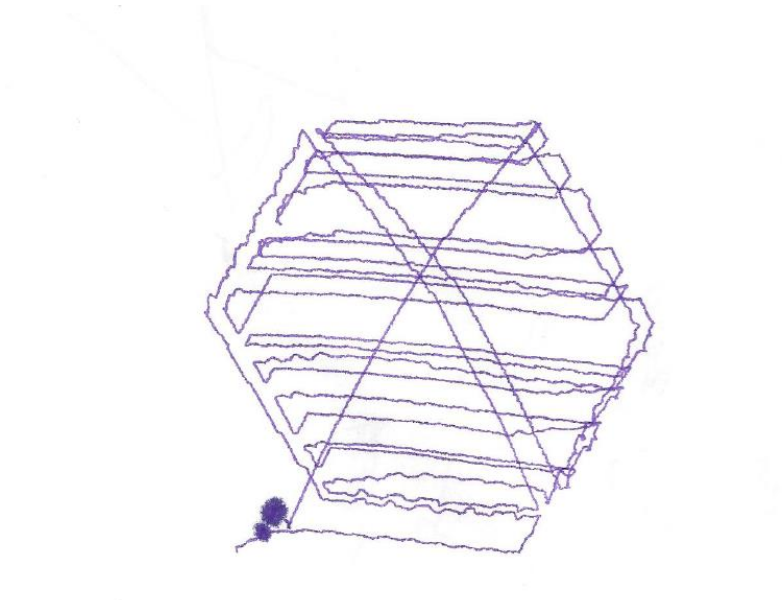


Abbildung 23: Ausdruck ohne Rundungsfehler

Der Stiftheber war abgeschaltet, damit man sah, auf welchem Weg er das Bild gemalt hat und ich hatte einen feinen Stift genommen, damit man es besser sieht.

3.4.12 Verbesserung und Bug fixes

Nachdem das Programm grundsätzlich funktioniert, ging es darum, es zu verbessern und schneller zu machen.

- Die Zeichensteuerung macht zu viele Umwege. Ich änderte sie, dass sie nicht immer von oben wieder anfing, um nach dem nächsten Pixel zu suchen, sondern zuerst in der Umgebung von 8x8 Pixeln. Erst wenn er dort nichts mehr fand, machte er oben weiter. Dadurch wurde der Druck viel schneller.
- Der Druckkopf machte ich jetzt aus Holz und nicht mehr aus Karton, damit er stabiler war.
- Die Abstände zwischen dem Druckkopf, den Schnüren und den Gleitstiften des Druckkopfs musste ich durch Ausprobieren optimal einstellen.

3.5 Zusammenbauen

Der Zusammenbau des Holzgerüstes konnte ich an einem Tag fertig machen. Bei Hornbach haben ich und mein Vater eine Holzplatte gekauft, die ich zuschneiden konnte. Ich bohrte die Löcher und baute hinten noch ein kleines Gestell für die Elektronik.

Abbildung 24: Wanddrucker von vorne

Abbildung 25: Wanddrucker von hinten

4 Produkt

Hier sind ein paar Beispiele, wie der Wanddrucker zeichnen kann.

Abbildung 26: Beispiel Drache

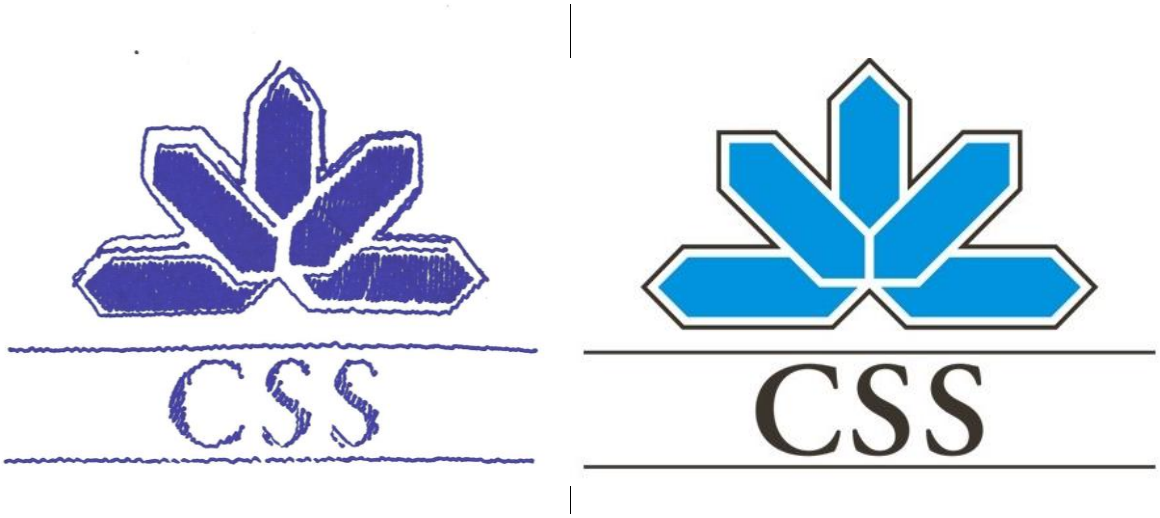


Abbildung 27: Beispiel CSS-Logo

5 Schlusswort

Es war eine sehr spannende Herausforderung. Ich konnte erstmals an einem etwas grösseren und komplizierteren Projekt über mehrere Wochen arbeiten. Manchmal hatte ich das Gefühl, dass es doch ein wenig zu gross ist.

Ich habe gelernt, wie man Klassen und Methoden richtig einsetzt und sie miteinander verbindet. Ich war überrascht, dass man beim Testen noch so viele Fehler finden würde. Ich musste sehen, dass man nicht so einfach einen perfekten Drucker bauen kann. Beim Zeichen eines Bildes wickelt jeder Motor 5m Schnur auf und ab. Beim Bild kommt es auf jeden Millimeter drauf an und da können nur schon durch kleine Ungenauigkeiten Bildverschiebungen passieren.

Der Zeitplanung konnte ich nicht ganz einhalten.
Die Planung hätte ich besser machen können. Es wäre besser gewesen, hätte ich aufgeschrieben, welche Klassen und Funktionen ich brauche und wie sie miteinander kommunizieren. Ich habe erst beim Arbeiten am Projekt gelernt, wie wichtig es gewesen wäre, eine gute Planung zu machen.

6 Quellenangaben

https://www.codecademy.com

7 Medien

7.1 Abbildungsverzeichnis

7.2 Bilderverzeichnis

Die folgenden Abbildungen sind aus dem Internet, alle anderen Bilder habe ich selbst fotografiert.

Abbildung 1: https://www.raspberrypi.org/blog/raspberry-pi-2

Abbildung 2: https://www.adafruit.com/product/324

Abbildung 3: http://www.play-zone.ch/de/tower-pro-micro-servo-9g-sg90.html

8 Anhang

8.1 Eingabe Abschlussarbeit

Projektidee

Name, Vorname	**Coach:**
Wolfisberg Donato	**Margrit Wirz**

Prov. TITEL des Vorhabens: (kann sich im Verlauf der Arbeit noch verändern)
Wand-Drucker mit raspberry PI

Worum es im Vorhaben geht: (kurze Beschreibung)
Ich will ein Wand-Drucker bauen, der von einem raspberry PI gesteuert wird. Die Schwierigkeit bei meinem Projekt wird das Programmieren sein, dies ist auch wo ich mein Focus drauf setzten möchte.
Der Druckkopf hängt an zwei Drähten die von je von einem Motor auf und abgewickelt Werden können. Je nach Länge der Drähte kann jeder Punkt auf dem Blatt erreicht werden.
Die Länge der Drähte kann man mit dem Pythagoras berechnen.
Der Stift auf dem Druckkopf wird mit einem Elektromagnet gehoben und gesenkt.

Vorgesehenes Endergebnis: (kann sich im Verlauf der Arbeit noch verändern)

- Eine GUI(Grafisches User Interface) mit dem man alles steuern kann.
- Das man aus einem Bild die Pixel rauslesen kann.
- Konstruktion aus Zwei Motoren und einem Druckkopf aus Holz.
- Das Zeichnen des eingegebenen Bildes soll möglichst genau sein.

Die **Schülerin** bzw. der **Schüler** verpflichtet sich, sich ernsthaft mit dem oben umschriebenen persönlichen Vorhaben auseinanderzusetzen. Das bedeutet: Verantwortung für das eigene Lernen zu übernehmen und das selbstgewählte Vorhaben – unter Einhaltung der für alle gültigen Abmachungen – zu Ende zu führen.

Der **Coach** verpflichtet sich, die Schülerin bzw. den Schüler während der ganzen Dauer der Arbeit zu begleiten, mit dem Ziel, dass das Vorhaben erfolgreich und termingerecht zu Ende geführt werden kann. Dazu gehören – wenn nötig – auch kritische Rückmeldungen und gegebenenfalls Ermahnungen.

Spezielle individuelle Abmachungen:

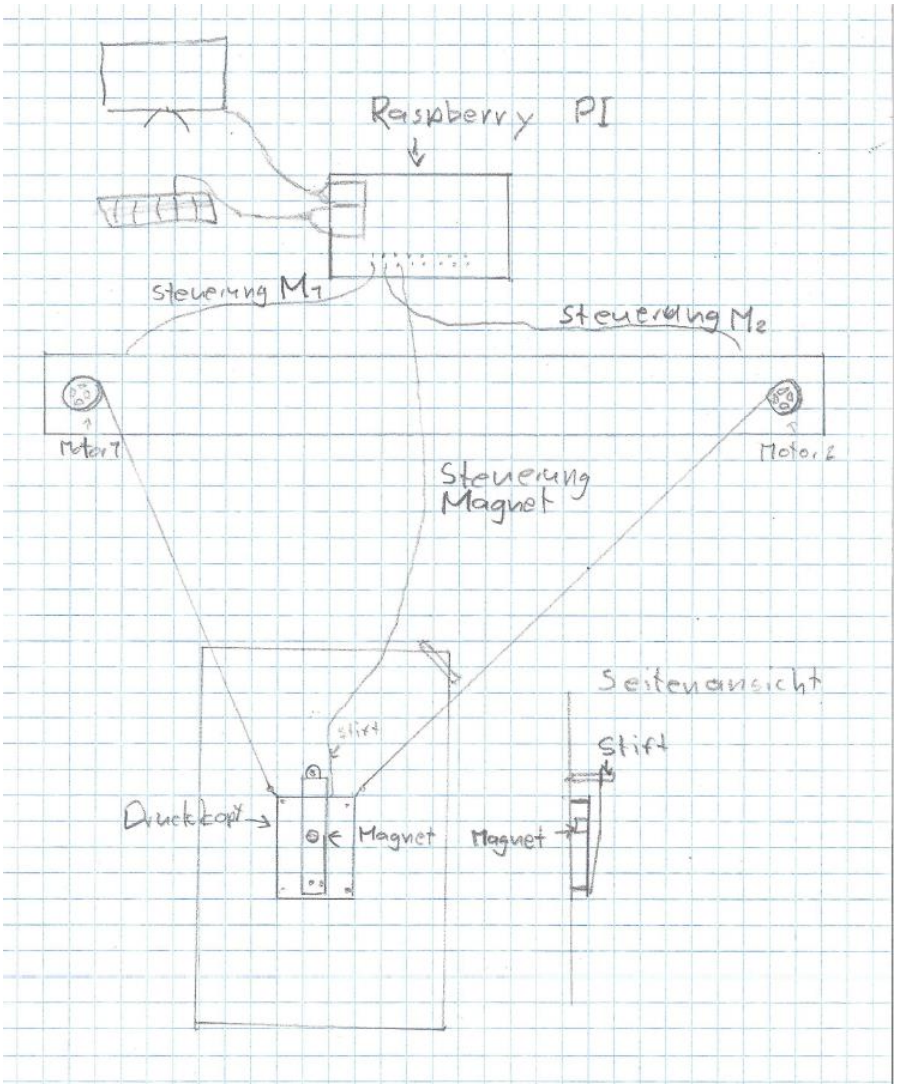
Die Unterzeichnenden bestätigen, die gemeinsamen Abmachungen zu kennen. Dazu gehören insbesondere auch die Rahmenbedingungen.

Datum, Unterschrift Schüler/in	Datum, Unterschrift Eltern	Datum, Unterschrift Coach
20. Jan. 2016	20. Jan 2016	M. Wirz
Donato		

8.2 Budget

Gegenstand	Preis in Franken
Raspberry pi	40
2 Phasenmotoren mit HAT	90
Servomotor	3
Kleine 4-Phasenmotoren, Kleinteile (Kabel, Breakoutboard)	20
Holzbrett	10
Total	**163**

8.3 Erste Projektskizze



8.4 Das Projektjournal

Name: Donato Wolfisberg

Projekt: Wanddrucker

Datum	**Arbeitsprotokoll** ▪ Tätigkeiten (Was? Wo? Mit wem?) ▪ Ergebnisse (Menge? Qualität? Erfolge? Probleme?)	**Dauer**	**Lerntagebuch** ▪ Einsichten ▪ Nächste Arbeitsschritte
1.2.2016 bis 5.2.2016	Informationen sammeln	3h	Ich weiß jetzt, wie ich es machen will.
7.2.2016	Schauen welche Motoren und Magnete ich brauche und wo ich sie kaufen kann.	1.5h	Ich habe gemerkt, dass die kleinen Motoren zu schwach sind und darum habe ich neue 2-Phasen Motoren gekauft.
10.2.2016 Bis 19.2.2016	Kleines GUI schreiben	4h	Ich habe ein kleines TestGUI mit der Python Klasse Tkinter programmiert.
22.2.2016 bis 5.3.2016	• Motoren befestigen • Motoren-Steuerung schreiben	6h	Beim Testen und schreiben der Programme hatte ich keine größere Probleme.

Datum	**Arbeitsprotokoll** ▪ Tätigkeiten (Was? Wo? Mit wem?) ▪ Ergebnisse (Menge? Qualität? Erfolge? Probleme?)	**Dauer**	**Lerntagebuch** ▪ Einsichten ▪ Nächste Arbeitsschritte
15.3.2016 Bis 20.3.2016	• Bildverarbeiter Schreiben • Canvase in GUI einfügen	8h	Ich war überrascht, wie einfach dass es war, die Pixel aus einem Bild herauszulesen.
6.4.2016 Bis 16.4.2016	• Druckkopf machen • Drucker-Steuerung	9h	Ich hatte gemerkt dass der Elektromagnet für den Stiftheber zu langsam und zu schwer ist, darum habe ich einen Servomotor verwendet. Ich hatte ein Testdruckkopf aus Karton gemacht.
23.4.2016 Bis 1.5.2016	• Programm verbessern • Programm testen • Neue Halterung machen	13h	Es gab viele Fehler, die mir erst beim ersten Zeichnen aufgefallen waren. Diese waren schwer zu finden und mühsam zu lösen. Ich habe den Druckknopf aus Holz gemacht und ich habe eine neue Halterung aus Holz gemacht. Ich hatte zuerst geglaubt, dass ich fertig bin, aber das Testen hat dann sehr viel Zeit verbraucht. Aber schliesslich funktionierte das Programm.

8.5 Meine nächsten Schritte planen

Meine nächsten Schritte planen

Donato Wolfisberg

Wanddrucker

Was?	Wo?	Bis wann?
Kleines GUI Schreiben	Zuhause	19.2.2016
Motoren Befestigen Motoren Steuerung Schreiben	Zuhause	2.3.2016
Bild Verarbeiter Programm Schreiben	Zuhause	15.3.2016
Druckkopf machen	Zuhause	28.3.2016
Drucker Steuerung	Zuhause	6.4.2016
Programm verbessern	Zuhause	1.5.2016
Programm Abgeben	Schule	4.5.2016

8.6 Meine Überprüfbaren Zielsetzungen

Meine Überprüfbaren Zielsetzungen

Donato Wolfisberg 29.1.2016

Was?	Überprüfbare Ziele
Kleines GUI Schreiben.	Auf dem Pi startet ein Programm mit dem ich Sachen eingeben und anzeigen kann.
Motoren Befestigen Motoren Steuerung Schreiben	Eine Befestigung für beide Motoren Ist gemacht. Über das GUI kann ich die Motoren einfach steuern.
Bild Verarbeiter Programm Schreiben	Ich kann über das GUI ein Bild einlesen und die Pixel auswerten.
Druckkopf machen	An den beiden Motoren hängt ein kleiner Wagen mit einem Stift. Mit einem Magnet oder Stepper Motor kann ich den Stift heben und senken.
Drucker Steuerung	Der Wanddrucker kann eine Diagonale auf ein Blatt zeichnen.
Programm verbessern	Der Wanddrucker kann auch kompliziertere formen aufs Blatt zeichen.