Westfälische Hochschule
Gelsenkirchen Bocholt Recklinghausen
University of Applied Sciences

ENTWICKLUNG EINE DREHVORRICHTUNG FÜR EINEN LABORAUFBAU

Projekt Arbeit

Prof. Dr. Martin Pollakowski

Hasan Beqai 202020400
Saief Elddine Fattoum 202020286
Master Elektrotechnik – Automatisierungs- und Leittechnik

Inhalt

Abbildungen:

Ziel des Projekts

Das Ziel des Projekts ist ein Programm auf das Raspberry Pi zu schreiben, damit kann eine Person per Mausklick, das Schrittmotor bewegen lässt. Also, eine Schnittstelle zwischen einem Menschen und dem Schrittmotor designt werden muss. Durch diese Schnittstelle kann man auf einen Winkel oder auf einen Punkt in einem Kreis klicken, dann wird der Motor gemäß diesem Winkel oder Punkt bewegt.

Python und Tkinter

Python

Python, die höhere Open-Source Programmiersprache, wird wegen unterschiedlicher Gründe überall benutzt. Zum ersten hat Python eine einfache und lesbaren Syntax. Python unterstützt auch viele Programmierparadigmen, wie zum Beispiel objektorientierte, aspektorientierte und funktionale Programmierung. Zur Veranschaulichung ist Python reich an vielen Bibliotheken, die verschiedene Aspekte der Softwareentwicklung wie www, html, CGI, xml, Datenbanken, GUI usw. unterstützen. Diese Eigenschaft lässt Python stark im Bereich der Verarbeitung von Daten, Big Data und des maschinellen Lernens. Außerdem kann Python mit unterschiedlichen Datenbanken interagieren, wie zum Beispiel MySQL und SQL-lite.

Tkinter

Zu diesem Projekt werden unterschiedlichen Bibliotheken von Python benutzt, aber die Hauptbibliothek ist Tkinter. Tkinter ist eine GUI-Bibliothek [Graphical User Interface-Bibliothek, oder auf Deutsche grafische Benutzerschnittstelle Bibliothek]. GUI verbindet Software und Technik mit dem Menschen durch ein grafisches Element, die auf einem Bildschirm erscheinen und werden durch Berühren von einer Menschenhand oder durch einen Computerklick gesteuert. Das Ziel dieses GUI in diesem Projekt ist die Bewegung eines Schrittmotors per Mausklick.

Das Ziel kann durch unterschiedlichen GUI-Python Bibliotheken erreichen, aber der besten Bibliothek ist Tkinter wegen der verschiedenen Eigenschaften, die Tkinter hat. Tkinter ist in der Lage, mit gleichem Programm auf unterschiedlichen Betreibsystems wie zum Beispiel Linux, Windows und MacOS funktioniert, weil er einen cross-platform [plattformübergreifend] ist. Außerdem wird Tkinter als ein leichtgewichtiges Framework betrachtet und im Vergleich zu anderen Frameworks relativ schmerzlos zu verwenden.

Installieren von Python und Tkinter

Installieren Python auf Windows und MacOS Betriebssystemen:

Den folgenden Schritten erklären, wie man Python auf Windows Betriebssystem Python installieren kann:

- Herunterladen Sie den Python Entwicklungsumgebungen (Python-IDLE) und Python Bibliotheken entweder von Microsofte-Store oder von offiziellen Python Webseite und es ist vorzuziehen, dass Sie die Version 3.8 herunterladen.

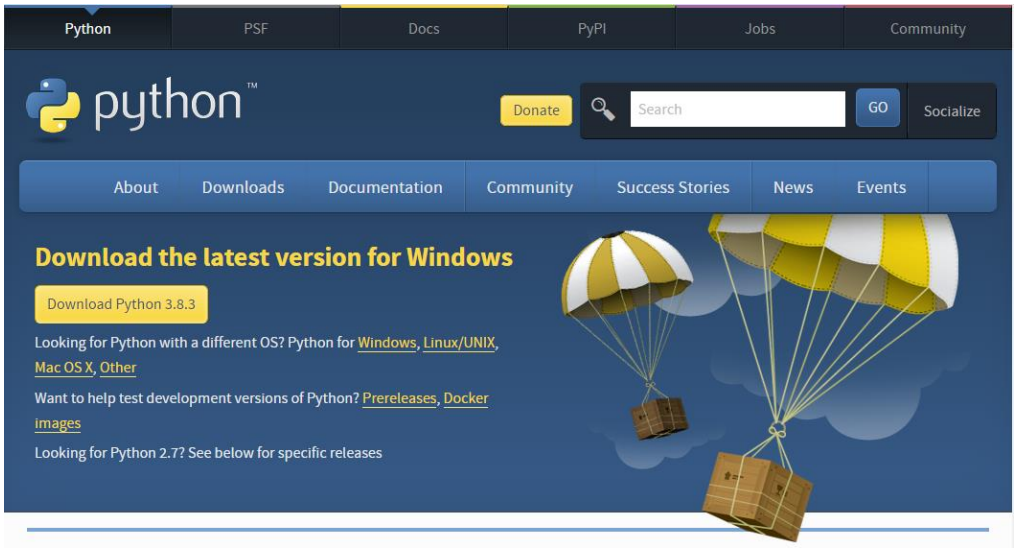


Abbildung 1: Python Offiziellen Webseite

- Führen Sie die Python-exe aus. Datei und während der Ausführung stellen Sie sicher, dass das Kontrollkästchen neben „Python 3.8 to PATH“ hinzufügen aktiviert ist.

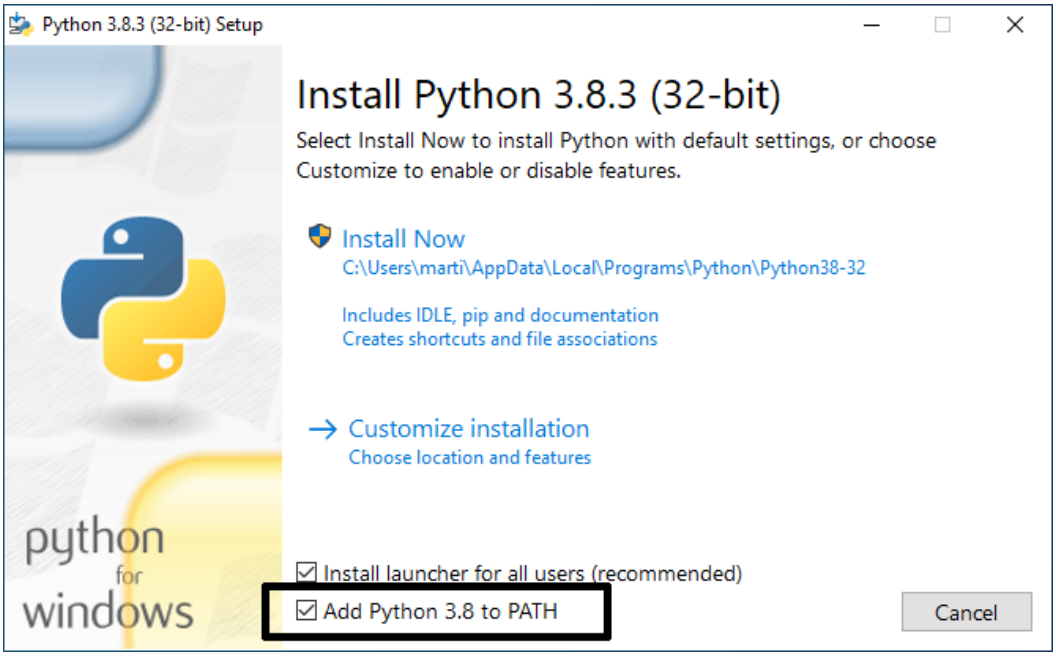


Abbildung 2: Python Ausführen

- Das Unterschied zwischen dem Windows und MacOS ist, dass die Zweite kann nur von der offiziellen Webseite der Python herunterladen kann.

Installieren Python auf Raspberry Pi OS and Linux Betriebssystemen:

- Mann kann durch Ausführen der untere Befehl in dem Terminal sicher stellen, dass Python breit zu herunterladen ist:

```
sudo apt update
sudo apt upgrade
```

und hier bedeute „sudo“:“Super User tue“

- Danach kann man auf das Terminal den folgenden Befehl ausführen, um das Python Version 3 zu installieren:

```
sudo apt install python3
```

- Auf das Raspberry Pi kann man die Thonny Python Entwicklungsumgebungen nutzen, um das Programm auszuführen.

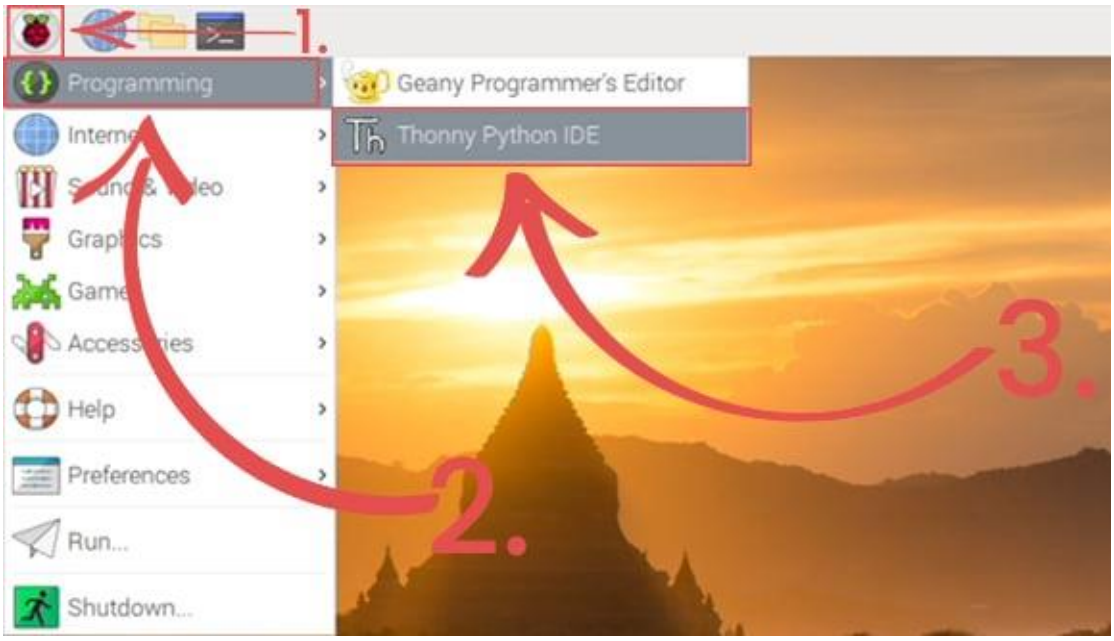


Abbildung 3: Thonny Entwicklungsumgebungen auf Raspberry Pi nutzen

- Oder kann man das Terminal benutzen, um das Programm auszuführen wie das folgende Beispiel:

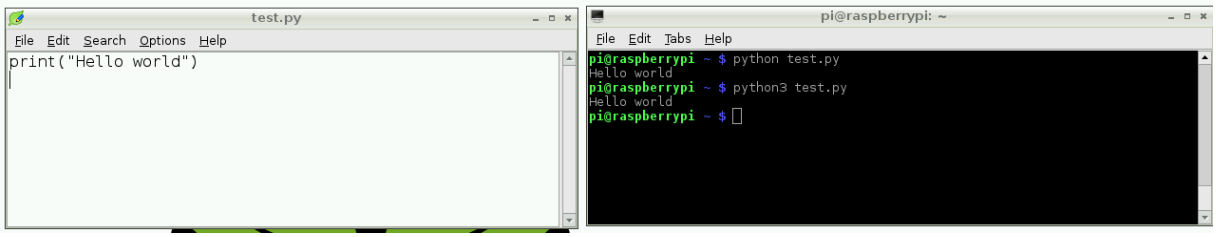


Abbildung 4: Ausführen Python Programm auf Terminal

Benutzung von Tkinter Bibliothek in dem Programm

- Tkinter ist ein Hauptbibliothek in Python, so sie hat schon mit Python installieret. Um diese Bibliothek zu benutzen, rufen man diese Bibliothek in seinem Programm mit dem folgenden Befehl:

```
from tkinter import *
```

und diese Sterne bedeutet, dass ich alle Methoden und Klassen in der Bibliothek Tkinter importiert habe.

Aufbau der Struktur der Schnittstelle

Der Aufbau diese Schnittstelle kann durch die Verbindung von unterschiedlichen Python Bibliotheken erklären werden. Die benutzenden Bibliotheken sind:

- Tkinter
- Datetime
- Webbrowser
- Turtle
- RPi.GPIO
- time

Jede obere Bibliothek wird gebracht, um eine eignete Ziel zu erreichen und zusammen sind mit der Hauptbibliothek Tkinter verbinden, um eine Schnittstelle zu erstellen.

Tkinter:

Dem nachfolgenden Programm ist in der Lage, ein Menü mit Registerkarten zu erstellen

```
from tkinter import *
import tkinter  as tk
from tkinter import ttk

# Ganz Fenster erstellen:

app= tk.Tk()
app.title("Schrittmotor Steuerung mit GUI - Menü")
app.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend der Anzahl der Spalten
app.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend der Anzahl der Zeilen

#Tabs erstellen:

tabControl = ttk.Notebook(app)
tab1 = ttk.Frame(tabControl)
tab2 = ttk.Frame(tabControl)
tab3= ttk.Frame(tabControl)

tabControl.add(tab1, text ='Home')
tabControl.add(tab2, text ='Run Motor')
tabControl.add(tab3, text ='Motor Status')

tabControl.pack(fill ="both")


tab1.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend der Anzahl der Spalten
tab1.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend der Anzahl der Zeilen
tab2.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend der Anzahl der Spalten
tab2.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend der Anzahl der Zeilen
tab3.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend der Anzahl der Spalten
tab3.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend der Anzahl der Zeilen

#Run main application:

app.mainloop()
```

Abbildung 5: Menü Erstellen mit Tkinter

- Am Anfang wird das Tkinter Bibliothek importiert mit diesem Befehl:

```
from tkinter import *
```

dann wird das Wort „Tkinter“ nur in diesem Programm verkürzt bei der Verwaltung von den Python Schlüsselwort „as“. Um ein Fenster, mit der Titel "Schrittmotor Steuerung mit GUI - Menü", zu erstellen, wird das Objekt “app“ verwandet. Es ist wichtig zu bemerken, dass diese Fenster hat keine feste Fläche, sondern seine Fläche hängt von den Objekten, die in diesem Fenster sind.

- Wenn man dir Registerkarten in das Menü zeigen möchte, soll man das Klasse ttk.Notebook anrufen mit dem folgenden Befehl:

```
from tkinter import ttk
```

Wenn man auf eine dieser Registerkarten klickt, zeigt das Notizbuch-Widget einen untergeordneten Bereich an, der der ausgewählten Registerkarte zugeordnet ist.

- Der Rest von oberem Programm erklärt, wie man diese registerkarten Namen geben kann und erläutert die Eigenschaften diesem Registerkarten.

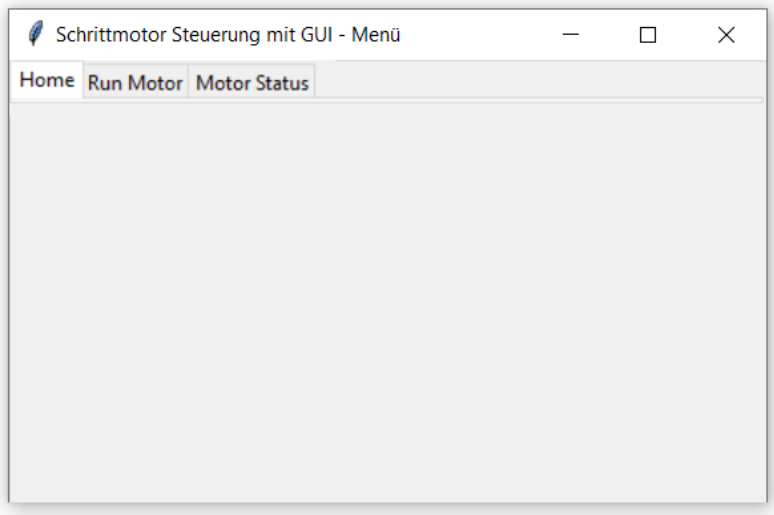


Abbildung 6: Menü und Registerkarten

- Der nächste Schritt ist der Bearbeitung von Home Registerkarte, wo sie wie die untere Bild aussieht:

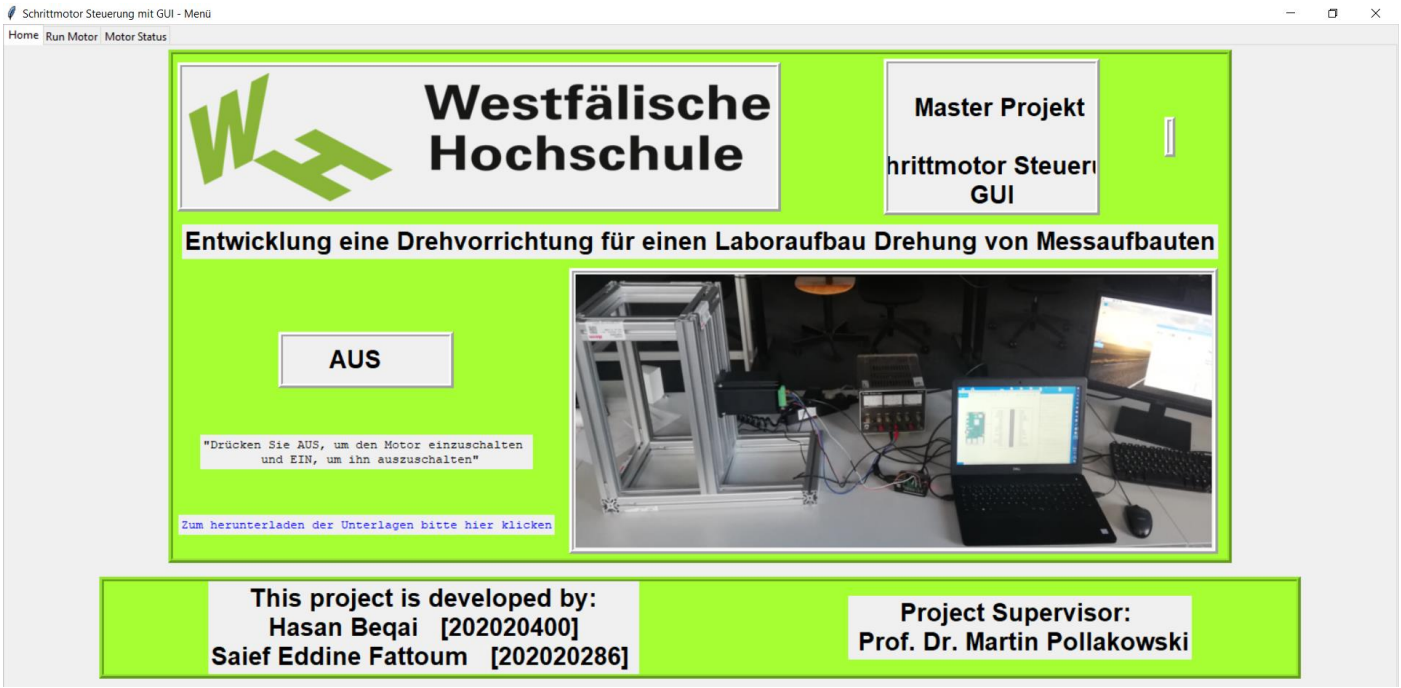


Abbildung 7: Home Seite erste Entwurf

- Um dieser Schritt zu schaffen, wird einen Font definiert, zwei Bilder importieren und Ein Methode, um der Motor ein oder auszuschalten.
  a. Der Font: die hellgrüne Farbe in dem Hintergrund wird durch die folgende Variable

```
fnt = font.Font(family = "helvetica",size=20, weight ="bold")#diese Schriftart
wird in der App verwendet.
```

  definiert und diese Variable Eigenschaften werden zu jedem Widget [Label, Frame, Button…usw.] erbt zum Beispiel, die folgenden Label erbt die hellgrüne Hintergrund Farbe gemäß dem folgenden Befehl:

```
lb1=tk.Label(content, borderwidth=5, relief="ridge",text="\n  Master Projekt" +
     "\n\n Schrittmotor Steuerung\nGUI",font = fnt)
```

  Der Ursprung von diese Font Klass ist das Tkinter Bibliothek, die wird auch mit dem folgenden Befehl angerufen:

```
from tkinter import font
```

  b. Die Bilder werden als Framen betrachtet und in diesem Framen werden diese Bilder gepackt, wie in dem folgenden Beispiel:

```
frame1=tk.Frame(content,borderwidth=5, relief="ridge")
```

```
img1= PhotoImage(file='img1.PNG')
Label(frame1,image=img1).pack()
```

werden das Fram mit dem Rahmenbreite 5 Pixels erstellt und es enthalt das Bild von Westfälsche Hochschule Logo. Hinweise: dieses Fram ist auch ein Inhalt von die größer Fram, das als „Content“ genannt wird.

c. Methoden:

In der Button Widget wird eine Methode mit dem Befehl „Command“ angerufen:

```
b1 = Button(content, text = " AUS ",borderwidth=5,  relief="ridge",font = fnt,  command=Simpletoggle)
```

Diese Methode heißt “Simpletoggle” wird als If-else Bedingte Anweisung beschreiben.

```
def Simpletoggle():
  if b1.config('text')[-1] == '       EIN       ':
     b1.config(text='       AUS       ')
     print("Der Motor wird gestoppet")
  else:
     b1.config(text='       EIN       ')
     print("Der Motor wird gestartet")
```

Dies ist eine Umschaltmethode zwischen Ein und Aus. So wird der Motor ein- und ausgeschaltet und die folgenden Sätze geschrieben.

Abbildung 8: Motor Ein- und Ausschaltung

- Die Struktur der gesamten Widgets im Bild der Startseite wird durch das Rasterlayout organisiert. Das Rasterlayout hängt von der Anzahl der Spalten und der Anzahl der

Zeilen ab, aus denen das Raster besteht. Das weitere Beispiel kann die Idee von Rasterlayout erklären.

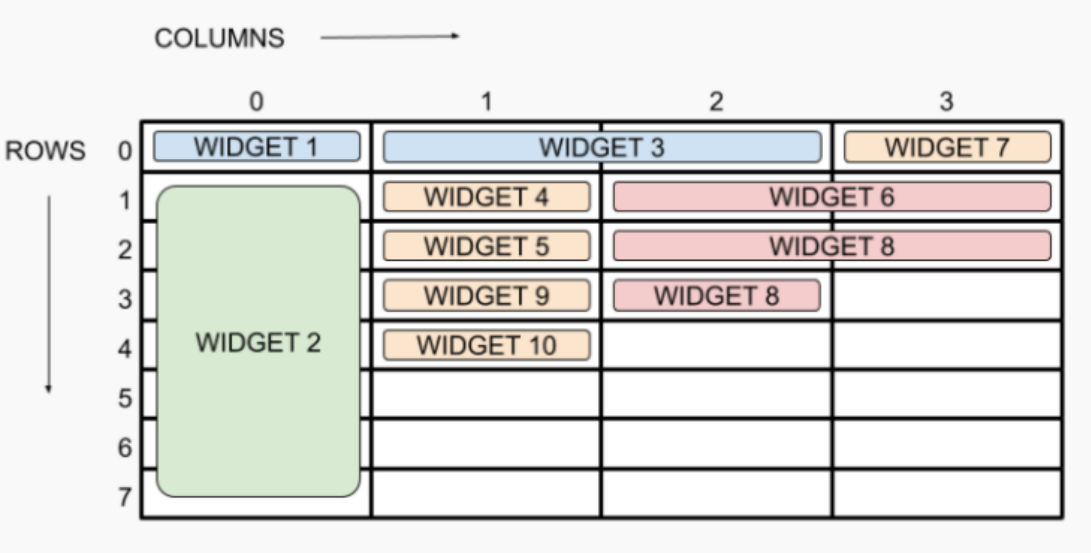


Abbildung 9: Rasterlayout Erklärung

- Gemäß das Rasterlayout wird den „Run Motor“ und „Motor Staus“ Registerkarten organisiert.
- In der zweite Registerkarte gibt es ein Methode heißt „run_motor()“:

```
def run_motor():
    global angle
    global tetaOld
    tetaOld = float(angle)
    angle = winkel.get()
    return print("der Motor läuft mit dem Winkel " + str(angle))
```

Diese Methode nimmt den Winkel vom Benutzer und druckt ihn aus. Sie benutzt zwei globalen Variablen, um zwischen den alten Winkel und neuen Winkel zu wechseln.

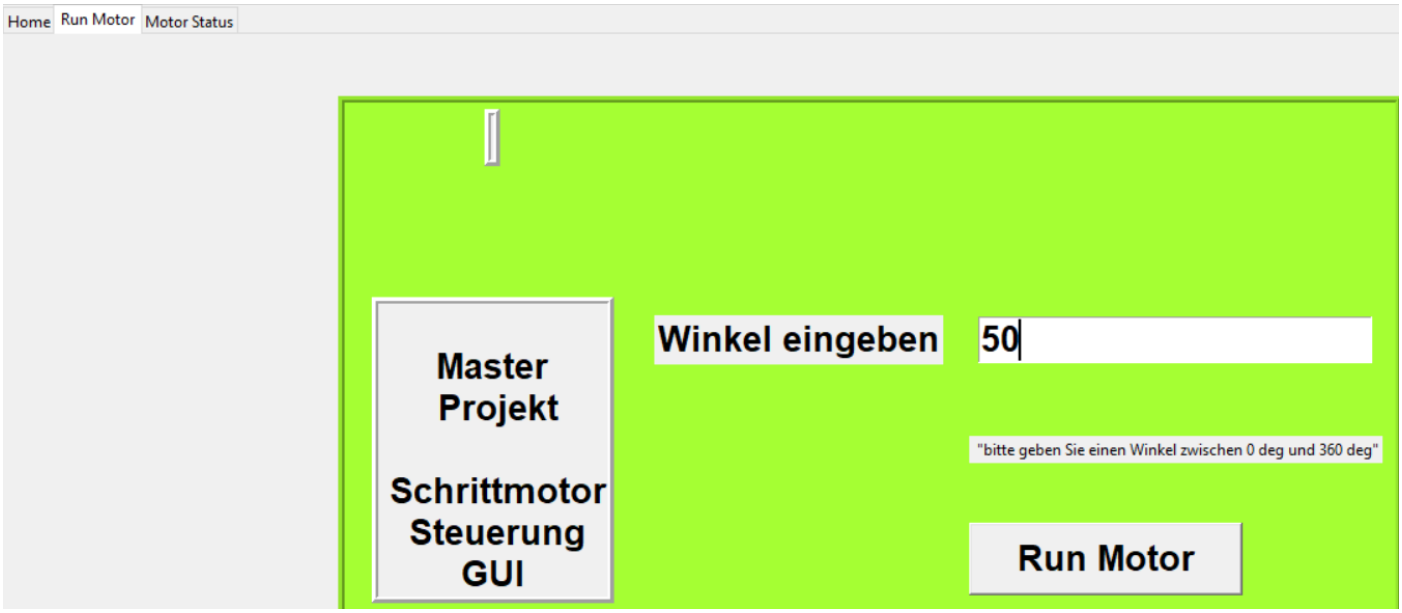


Abbildung 10: Run-Motor Seite Entwurf

- In „Motor Status“ Seite wird das neu Bibliothek „turtle“ benutzt, die wird in kommenden Punkten erklärt.

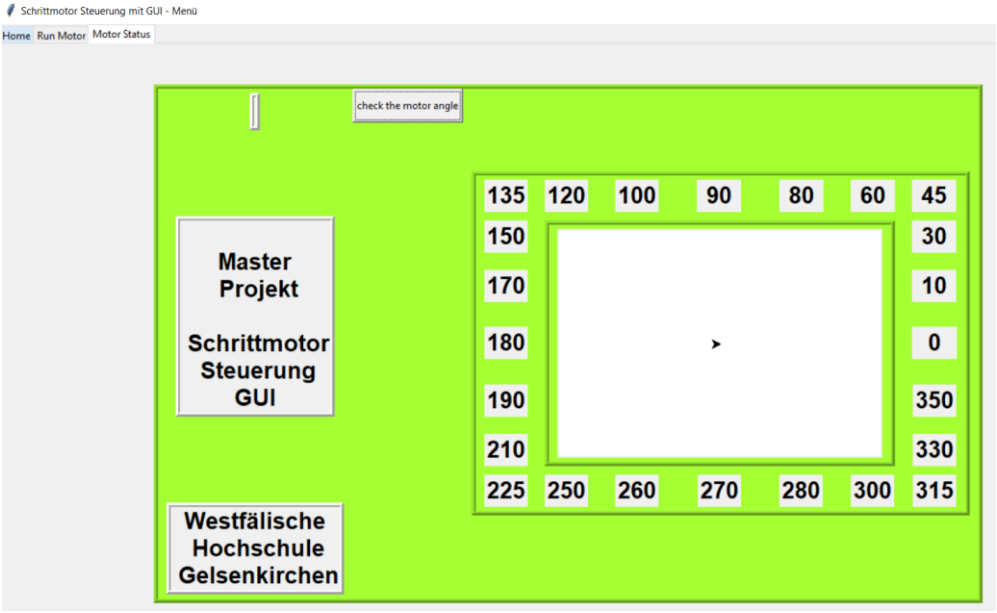


Abbildung 11: Motor-Status Seite Entwurf

RPi.GPIO:

- Dieser Bibliothek bietet ein Python-Modul zur Steuerung des GPIO auf einem Raspberry Pi.
- Die rechte Darstellung kann das Raspberry Pi GPIOs erklären:
  - ✓ Es betrachtet als kleine Computer, wo es viele Eigenschaften von Computer hat, wie zum Beispiel Raspberry Pi hat USB-Anschluss, Strom- und Ethernet-Kabel, HDMI, RAM, Micro-SD-Karte, Kamera Anschluss, Anzeige Port und es ist in der Lage, über WLAN und Bluetooth mit anderen Geräten zu kommunizieren.
  - ✓ In Vergleich zum Computer, Raspberry Pi hat extra Pins, die als GPIOs genannt werden. Mit diesen GPIOs kann das Raspberry Pi mit Aktoren und Sensoren auch kommunizieren.

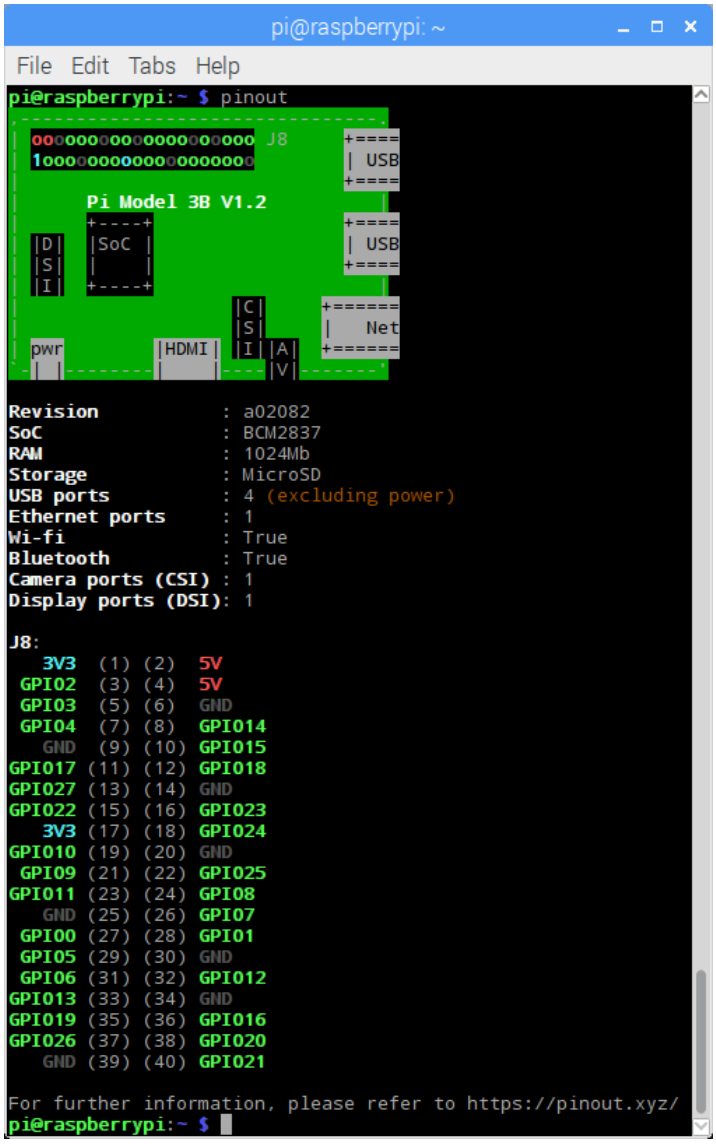


Abbildung 12: GPIOs von Raspberry Pi

- Das Raspberry Pi wird mit dem
  Motor durch GIPOs verbindet. So kann man mit der Software, die auf diese Raspberry Pi geschrieben werden, das Motor steuern kann.

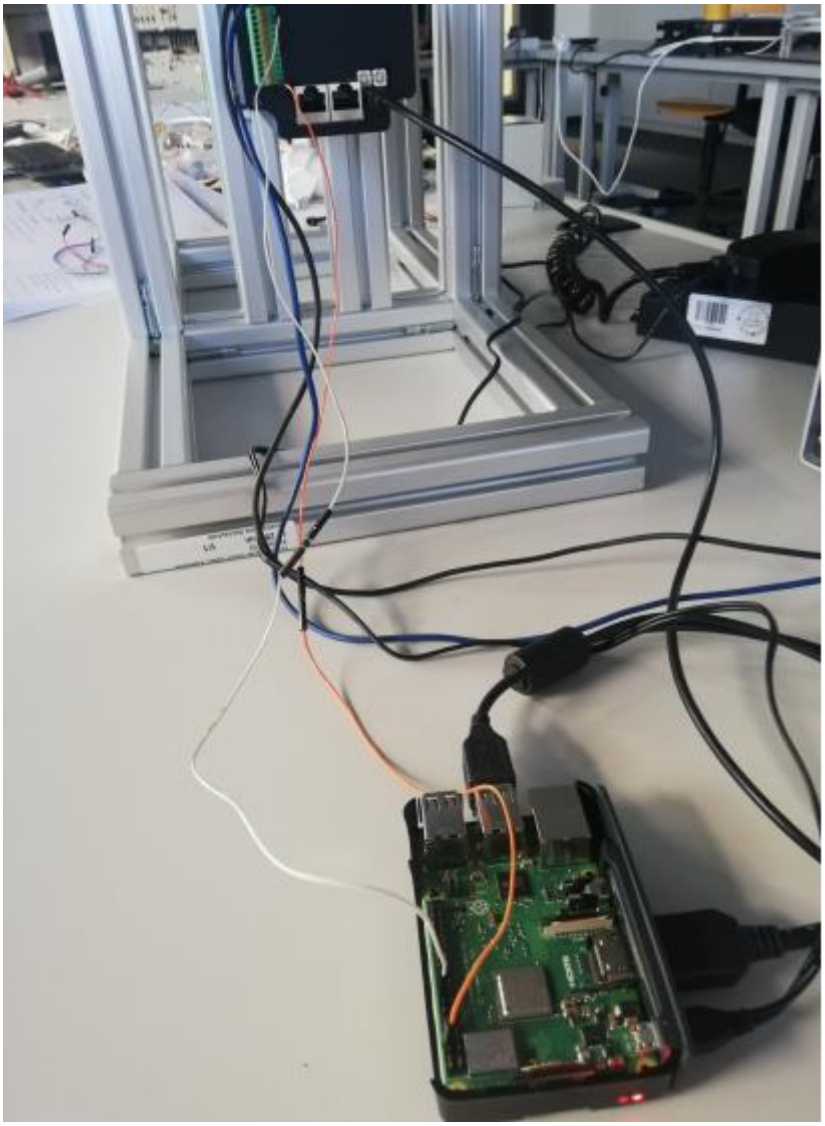


Abbildung 13: Raspberry Pi mit Schrittmotor

- Die drei Drähte, die zwischen dem Motor und dem Pi verbinden, sind die 5-V-Versorgungsspannung, die Masse und das Datenkabel. Die Datenleitung überträgt die Befehle vom Raspberry Pi zum Motor.
- Zu der Software können wir mit dem folgenden Befehl das RPi.GPIO Bibliothek importieren:

```
import RPi.GPIO as gpio
```

- Mit dem folgenden Befehl wird eingestellt, dass der GPIO über seine GPIO Nummer angesprochen wird:

```
gpio.setmode(gpio.BOARD)
```

- Der obere Befehl kann auch sein so:

```
gpio.setmode(gpio.BCM)
```

- Das Unterschied zwischen den BOARD und BCM ist Weise, wie den GPIOs zu Nummern zugewiesen werden. Das untere Abbild kann die Nummerung Methoden erklären.

Abbildung 14: Unterschied zwischen BOARD und BCM

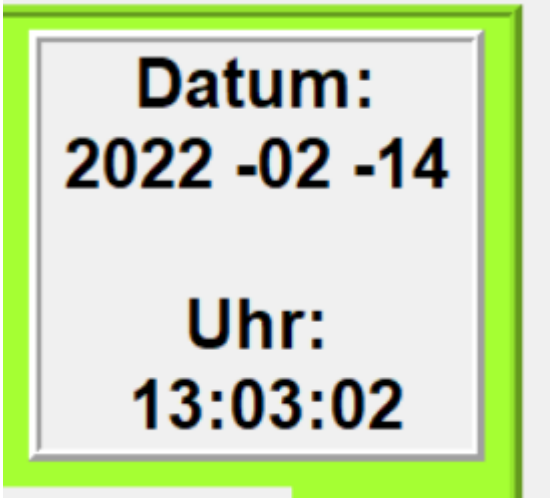
- Danach wird das gewaltete GPIO als Variable betrachtet, und diese Variable ist der Board Nummer der Raspberry Pi. Es wird auch als Ausgang zugeordnet.

```
x =23
gpio.setmode(gpio.BOARD)
gpio.setup(x,gpio.OUT)
```

- So, dieser Ausgang wird alle Informationen, die durch die grafische Schnittstelle bei einem Menschen gibt, zum Schrittmotor transformieren.

Datetime:

- Es ist eine Python-Bibliothek, die Datum und Uhrzeit in einem gut verständlichen Format verwaltet. Eine Methode namens clock_time() wird definiert, um in jeder Registerkarte das aktuelle Datum und die genaue Uhrzeit anzuzeigen, abhängig von der Zeitzone, in der der Benutzer verfügbar ist.




```
def clock_time():
    time= dt.datetime.now()
    time= (time.strftime("Datum:\n %Y -%m -%d
\n \n Uhr: \n %H:%M:%S"))
    txt.set(time)
    app.after(1000,clock_time)
#Diese Methode wird verwendet, um Datum und
Uhrzeit einzustellen und zu aktualisieren.
app.after(100,clock_time)#Rufen Sie diese Methode nach 100 ms nach dem Erstellen
der App auf.
```

Abbildung 15: Datum Entwurf

dieser Methode wird in einem Label eingepackt, wo sie das aktuelle Datum und die Uhrzeit zeigt.

Turtle:

- Turtle ist eine berühmte Python-Bibliothek, die entwickelt wurde, um Kindern beizubringen, damit sie Spaß am Programmieren haben. Es ist eine Grafikbibliothek, in der man mit Python-Syntax und -Codierung auf einem Zeichenbrett zeichnen kann.

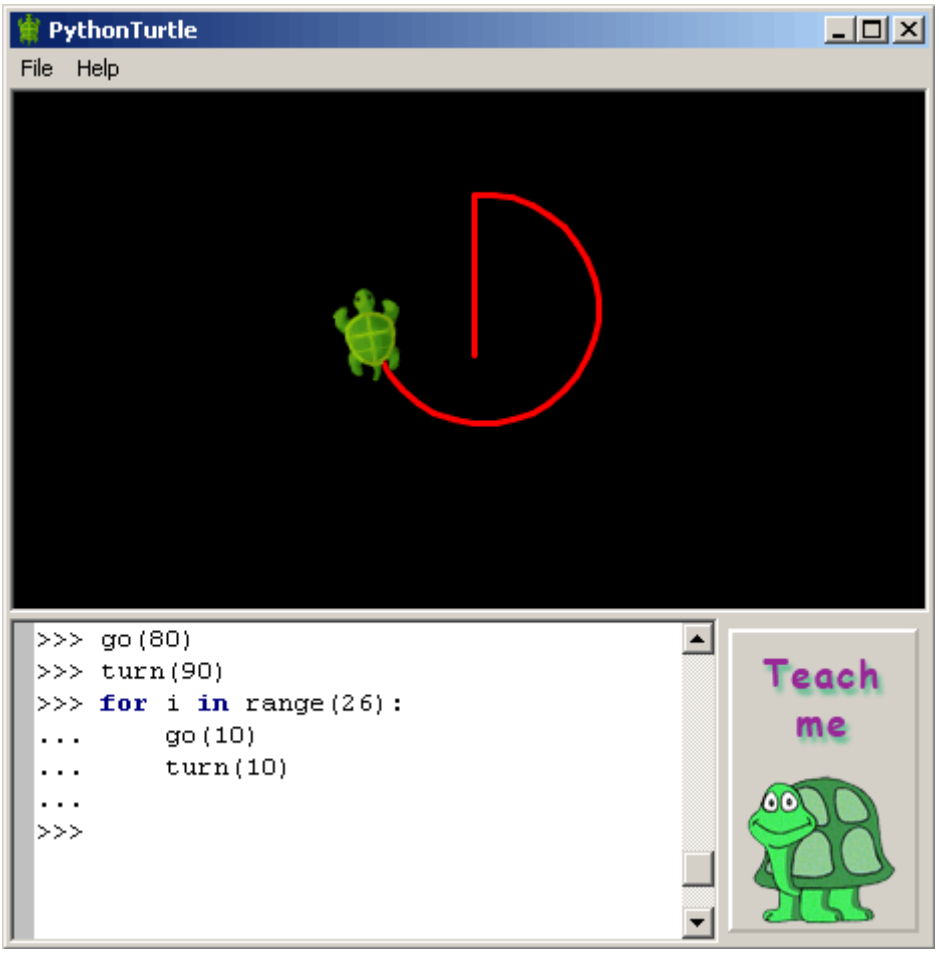


Abbildung 16: Turtle Bibliothek

- Mit Turtle-Python kann jeder zeichnen, was er will, wenn er es in einer bekannten geometrischen Form beschreiben kann, geometrischen Formen wie ein gerade Linie, Quadrat, Rechteck, Kreis und mehr. Diese Bibliothek hat so viel Funktionen, die mit Zeichnen helfen können. Aber die berühmteste Funktion ist der „forward“ Funktion, die in ein Methode heißt fw() in unseren Programm benutzt werden.

```
def fw():
    teta = tetaOld+ float(angle)
    draw.left(teta)
    draw.forward(100)
    time.sleep(2)
    draw.reset()
```

- Mit turtle Bibliothek man kann zuerst eine Schildkröte, die in Ursprung des Fensters legt, dann kann man diese Schildkröte bewegt gemäß den Funktionen, die man in seinem Programm geschrieben hat.
- In das Programm haben wir die Schildkröte Form um einen Pfeil gewechselt. Dann wird das Pfeil eine Richtung von einem Winkel nehmen. Dieser Winkel wird von dem Benutzer durch die „Run Motor“ Seite in der Schnittstelle spazieret. Danach wird das Pfeil nach Recht [„forward“] mit ein 100 Pixels Abstand bewegt, um den geeigneten Winkel auf der Seite „Motor-Status“ zu repräsentieren. Zu Ende wird der Pfeil ins Uhrsprung Punkt zurück gegangen. Zwischen das „forward“ und „reset“ Funktionen gibt es heißt time. sleep Funktion, die in dem nächsten Punkt erläutern wird.

time:

- Das time. sleep Funktion gehört zum Bibliothek Time, die mit dem folgenden Befehl importiert wird:

- ```
  import time
  ```

- Dann wird die Funktion:

  ```
  time.sleep(2)
  ```

  für nur zwei Sekunden angerufen. Diese Funktion. Diese Funktion stoppt das Programm dort, wo es sich für zwei Sekunden befindet, um die Darstellung des Pfeils für zwei Sekunden auf dem angegebenen Winkel zu halten.

Webbrowser:

- Diese Bibliothek wird durch den folgenden Befehl importiert:

  ```
  import webbrowser
  ```

- Um alle dokumentierten Informationen zu diesem Projekt herunterzuladen, wird eine Webbrowser-Bibliothek verwendet. Diese Bibliothek hilft uns, einen herunterladbaren Link zu erstellen, dieser Link enthält eine PDF-Datei, die die technischen Informationen zu diesem Projekt enthält.
- Dieser Download-Prozess kann durch eine Methode, die “callback“ heißt, erreicht werden

  ```
  def callback(url):
      webbrowser.open_new(url)
  ```
- Mit dem oberen Befehl kann das Dokument eröffnet wird.
  ```

Abgeschlossener Code und Ergebnis

Programm

```
#Bibliotheken
from tkinter import *
import tkinter  as tk
from tkinter import ttk
from tkinter import font
import turtle
import datetime as dt
import RPi.GPIO as gpio
import time
import webbrowser
# Ganz Fenster erstellen:
app= tk.Tk()
app.title("Schrittmotor Steuerung mit GUI - Menü")
app.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend der
Anzahl der Spalten
app.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend
der Anzahl der Zeilen
#Tabs erstellen:
tabControl = ttk.Notebook(app)
tab1 = ttk.Frame(tabControl)
tab2 = ttk.Frame(tabControl)
tab3= ttk.Frame(tabControl)
tabControl.add(tab1, text ='Home')
tabControl.add(tab2, text ='Run Motor')
tabControl.add(tab3, text ='Motor Status')
tabControl.pack(fill ="both")
tab1.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend
der Anzahl der Spalten
tab1.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend
der Anzahl der Zeilen
tab2.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend
der Anzahl der Spalten
tab2.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend
der Anzahl der Zeilen
tab3.grid_rowconfigure(0, weight=1) #Konfigurieren Sie die Seitenbreite entsprechend
der Anzahl der Spalten
tab3.grid_columnconfigure(0, weight=1)#Konfigurieren Sie die Seitenlänge entsprechend
der Anzahl der Zeilen
#Variablen:

fnt = font.Font(family = "helvetica",size=20, weight ="bold")#diese Schriftart wird in
der App verwendet.
txt = StringVar()#diese Variable wird in der Methode "clock_time()" verwendet, um immer
Datum und Uhrzeit zu aktualisieren
angle =0 # global varibale
tetaOld=0 # global variable
x =23
gpio.setmode(gpio.BOARD)
gpio.setup(x,gpio.OUT)
#Methoden:
def Simpletoggle():

    if b1.config('text')[-1] == '       EIN       ':
        b1.config(text='       AUS       ')
        print("Der Motor wird gestoppet")
    else:
        b1.config(text='       EIN       ')
        print("Der Motor wird gestartet")
#Dies ist eine Umschaltmethode zwischen Ein und Aus. So wird der Motor ein- und
ausgeschaltet.
```

```
def run_motor():
    global angle
    global tetaOld
    tetaOld = float(angle)
    angle = winkel.get()
    return print("der Motor läuft mit dem Winkel " + str(angle))
def clock_time():
    time= dt.datetime.now()
    time= (time.strftime("Datum:\n %Y -%m -%d \n \n Uhr: \n %H:%M:%S"))
    txt.set(time)
    app.after(1000,clock_time)
#Diese Methode wird verwendet, um Datum und Uhrzeit einzustellen und zu aktualisieren.
app.after(100,clock_time)#Rufen Sie diese Methode nach 100 ms nach dem Erstellen der
App auf.
def fw():
    teta = tetaOld+ float(angle)
    draw.left(teta)
    draw.forward(100)
    time.sleep(2)
    draw.reset()
def callback(url):
    webbrowser.open_new(url)

#in Tab 1: Home

content = tk.Frame(tab1,borderwidth=5, relief="ridge", bg='#A5FF33')

frame = tk.Frame(content,borderwidth=5, relief="ridge")
img=PhotoImage(file='whlogo3.png')
Label(frame,image=img).pack()

lb1=tk.Label(content, borderwidth=5, relief="ridge",text="\n  Master Projekt" +
             "\n\n Schrittmotor Steuerung\nGUI",font = fnt)
lb1.config(anchor=CENTER)

lb2 = tk.Label(content,borderwidth=5, relief="ridge",textvariable=txt, font = fnt,
               foreground= "black")
lb2.config(anchor=CENTER)

lb3=tk.Label(content, text="Entwicklung eine Drehvorrichtung für einen Laboraufbau"+
             " Drehung von Messaufbauten",font = fnt, foreground= "black")

frame1=tk.Frame(content,borderwidth=5, relief="ridge")
img1= PhotoImage(file='img1.PNG')
Label(frame1,image=img1).pack()


b1 = Button(content, text = "        AUS      ",borderwidth=5, relief="ridge",font =
fnt, command=Simpletoggle)
b1.config(anchor=W)
lb4 =tk.Label(content,text="\"Drücken Sie AUS, um den Motor einzuschalten \n und EIN,
um ihn auszuschalten\"")
lb4.config(font=("Courier", 10))
note = Label(content, text="Zum herunterladen der Unterlagen bitte hier klicken",
fg="blue", cursor="hand2")
note.config(font= ("Courier", 10))
note.bind("<Button-1>", lambda e: callback("/home/pi/Desktop/Projekt Beqai-
Fattoum/Projekt/Bachelorarbeit_Kuruyamac.pdf"))
content1 = tk.Frame(tab1,borderwidth=5, relief="ridge", bg='#A5FF33')
lb7=tk.Label(content1,text="This project is developed by:\nHasan Beqai
[202020400]\n"+
             "Saief Eddine Fattoum   [202020286] ",font = fnt, foreground= "black")
lb8=tk.Label(content1, text="Project Supervisor: \n Prof. Dr. Martin Pollakowski",font
= fnt, foreground= "black")
```

```
#Grid und Layouts organisieren:

content.grid(column=0, row=0, padx=10, pady=5)
frame.grid(column=0, row=0, columnspan=4, rowspan=2,padx=5, pady=5)
lb1.grid(column= 5, row=0, columnspan=2, rowspan=2, padx=15, pady=5)
lb2.grid(column=7, row=0, columnspan=2, rowspan=2, padx=10, pady=5)
lb3.grid(column=0, row=2,columnspan=8, rowspan=1, padx=10, pady=5)
b1.grid(column=1, row= 4,columnspan=1, rowspan=1)
lb4.grid(column=1, row=5,columnspan=1, rowspan=1)
note.grid(column=1, row=6,columnspan=1, rowspan=1)
frame1.grid(column=3, row= 3,columnspan=6, rowspan=4, padx=10, pady=5)
content1.grid(column=0, row=6,columnspan=2, rowspan=2, padx=10, pady=10)
lb7.grid(column=0, row=6,padx=115)
lb8.grid(column=3, row=6, columnspan=2, padx=115)

#*************************************************************************************
******************#
#in Tab 2: Motor Status
#*************************************************************************************
******************#
fram3 =tk.Frame(tab2,borderwidth=5, relief="ridge", bg='#A5FF33')
lb11=tk.Label(fram3, borderwidth=5, relief="ridge",text="\nMaster\n Projekt" +
         "\n\n Schrittmotor\n Steuerung\nGUI",font = fnt)
lb11.config(anchor=CENTER)
lb21 = tk.Label(fram3,borderwidth=5, relief="ridge",textvariable=txt, font = fnt,
           foreground= "black")
lb21.config(anchor=CENTER)
lb31= tk.Label(fram3,borderwidth=5, relief="ridge",text="Westfälische\n Hochschule\n
Gelsenkirchen", font = fnt,
           foreground= "black")
lb31.config(anchor=CENTER)
lb41=tk.Label(fram3,text="\n\n \n\n ", bg='#A5FF33')
lb51=tk.Label(fram3,text="\n\n \n\n ", bg='#A5FF33')


winkel_Label=tk.Label(fram3,text="Winkel eingeben",  font = fnt)
winkel=tk.Entry(fram3, font = fnt)
winkel.focus()
run =tk.Button(fram3, text= "   Run Motor   ", command= run_motor,  font = fnt)
note = tk.Label(fram3, text=" \"bitte geben Sie einen Winkel zwischen 0 deg und 360
deg\"")

#Grid und Layouts organisieren in Tab1:
fram3.grid(column=0, row=0, padx=10, pady=5)
lb21.grid(column= 0, row=0, columnspan=2, rowspan=2, padx=15, pady=5)
lb51.grid(column= 0, row=2, columnspan=2, rowspan=2, padx=15, pady=5)
lb11.grid(column=0, row=4, columnspan=2, rowspan=2, padx=10, pady=5)
lb41.grid(column= 0, row=6, columnspan=2, rowspan=2, padx=15, pady=5)
lb31.grid(column =0, row =8, columnspan=2, rowspan=2, padx=10, pady=5)

winkel_Label.grid(column =4, row =3, columnspan=4, rowspan=2, padx=10, pady=5)
winkel.grid(column =8, row =3, columnspan=6, rowspan=2, padx=10, pady=5)
run.grid(column =8, row =5, columnspan=4, rowspan=2, padx=10, pady=5)
note.grid(column =8, row =4, columnspan=6, rowspan=2, padx=10, pady=5)
#*************************************************************************************
******************#
#in Tab 3: Run Motor
#*************************************************************************************
******************#
fram2 =tk.Frame(tab3,borderwidth=5, relief="ridge", bg='#A5FF33')
lb11=tk.Label(fram2, borderwidth=5, relief="ridge",text="\nMaster\n Projekt" +
         "\n\n Schrittmotor\n Steuerung\nGUI",font = fnt)
lb11.config(anchor=CENTER)
```

```
lb21 = tk.Label(fram2,borderwidth=5, relief="ridge",textvariable=txt, font = fnt,
          foreground= "black")
lb21.config(anchor=CENTER)
lb31= tk.Label(fram2,borderwidth=5, relief="ridge",text="Westfälische\n Hochschule\n
Gelsenkirchen", font = fnt,
          foreground= "black")
lb31.config(anchor=CENTER)
lb41=tk.Label(fram2,text="\n\n \n\n ", bg='#A5FF33')
lb51=tk.Label(fram2,text="\n\n \n\n ", bg='#A5FF33')


fram4 = tk.Frame(fram2,borderwidth=5, relief="ridge", bg='#A5FF33')
flb1= tk.Label(fram4, text= "120", font = fnt,foreground= "black")
flb2= tk.Label(fram4, text= "100", font = fnt,foreground= "black")
flb3= tk.Label(fram4, text= " 90 ", font = fnt,foreground= "black")
flb4= tk.Label(fram4, text= " 80 ", font = fnt,foreground= "black")
flb5= tk.Label(fram4, text= " 60 ", font = fnt,foreground= "black")
flb0=tk.Label(fram4, text="135", font = fnt,foreground= "black")
flb01=tk.Label(fram4, text=" 45 ", font = fnt,foreground= "black")
flb6= tk.Label(fram4, text= "150", font = fnt,foreground= "black")
flb7= tk.Label(fram4, text= "170", font = fnt,foreground= "black")
flb8= tk.Label(fram4, text= "180", font = fnt,foreground= "black")
flb9= tk.Label(fram4, text= "190", font = fnt,foreground= "black")
flb10= tk.Label(fram4, text= "210", font = fnt,foreground= "black")
flb11= tk.Label(fram4, text= "225", font = fnt,foreground= "black")
flb12=tk.Label(fram4, text="250", font = fnt,foreground= "black")
flb13= tk.Label(fram4, text= "260", font = fnt,foreground= "black")
flb14= tk.Label(fram4, text= "270", font = fnt,foreground= "black")
flb15= tk.Label(fram4, text= "280", font = fnt,foreground= "black")
flb16= tk.Label(fram4, text= "300", font = fnt,foreground= "black")
flb17= tk.Label(fram4, text= "315", font = fnt,foreground= "black")
flb18= tk.Label(fram4, text= "330", font = fnt,foreground= "black")
flb19= tk.Label(fram4, text= "350", font = fnt,foreground= "black")
flb20= tk.Label(fram4, text= "  0  ", font = fnt,foreground= "black")
flb21= tk.Label(fram4, text= " 10 ", font = fnt,foreground= "black")
flb22= tk.Label(fram4, text= " 30 ", font = fnt,foreground= "black")


fram5= tk.Frame(fram4,borderwidth=5, relief="ridge", bg='#A5FF33')
canvas = tk.Canvas(fram5)
draw = turtle.RawTurtle(canvas)
Board_Button = tk.Button(fram2, text ="check the motor angle", command = fw)



#Grid und Layouts organisieren in Tab1:
fram2.grid(column=0, row=0, padx=10, pady=5)
lb21.grid(column= 0, row=0, columnspan=2, rowspan=2, padx=15, pady=5)
lb51.grid(column= 0, row=2, columnspan=2, rowspan=2, padx=15, pady=5)
lb11.grid(column=0, row=4, columnspan=2, rowspan=2, padx=10, pady=5)
lb41.grid(column= 0, row=6, columnspan=2, rowspan=2, padx=15, pady=5)
lb31.grid(column =0, row =8, columnspan=2, rowspan=2, padx=10, pady=5)
fram4.grid(column = 4, row = 0,columnspan=10, rowspan=10, padx=10, pady=5)
flb1.grid(column = 6, row = 0,columnspan=2, rowspan=2, padx=10, pady=5)
flb0.grid(column = 4, row = 0,columnspan=2, rowspan=2, padx=10, pady=5)
flb2.grid(column = 8, row = 0,columnspan=2, rowspan=2, padx=10, pady=5)
flb3.grid(column = 10, row = 0,columnspan=2, rowspan=2, padx=10, pady=5)
flb4.grid(column = 12, row = 0,columnspan=2, rowspan=2, padx=10, pady=5)
flb5.grid(column = 14, row = 0,columnspan=2, rowspan=2, padx=10, pady=5)
flb01.grid(column = 16, row = 0,columnspan=2, rowspan=2, padx=10, pady=5)
flb6.grid(column = 4, row = 2,columnspan=2, rowspan=2, padx=10, pady=5)
flb7.grid(column = 4, row = 4,columnspan=2, rowspan=2, padx=10, pady=5)
flb8.grid(column = 4, row = 6,columnspan=2, rowspan=2, padx=10, pady=5)
flb9.grid(column = 4, row = 8,columnspan=2, rowspan=2, padx=10, pady=5)
flb10.grid(column = 4, row = 10,columnspan=2, rowspan=2, padx=10, pady=5)
```

```
flb11.grid(column = 4, row = 12,columnspan=2, rowspan=2, padx=10, pady=5)
flb12.grid(column = 6, row = 12,columnspan=2, rowspan=2, padx=10, pady=5)
flb13.grid(column = 8, row = 12,columnspan=2, rowspan=2, padx=10, pady=5)
flb14.grid(column = 10, row = 12,columnspan=2, rowspan=2, padx=10, pady=5)
flb15.grid(column = 12, row = 12,columnspan=2, rowspan=2, padx=10, pady=5)
flb16.grid(column = 14, row = 12,columnspan=2, rowspan=2, padx=10, pady=5)
flb17.grid(column = 16, row = 12,columnspan=2, rowspan=2, padx=10, pady=5)
flb18.grid(column = 16, row = 10,columnspan=2, rowspan=2, padx=10, pady=5)
flb19.grid(column = 16, row = 8,columnspan=2, rowspan=2, padx=10, pady=5)
flb20.grid(column = 16, row = 6,columnspan=2, rowspan=2, padx=10, pady=5)
flb21.grid(column = 16, row = 4,columnspan=2, rowspan=2, padx=10, pady=5)
flb22.grid(column = 16, row = 2,columnspan=2, rowspan=2, padx=10, pady=5)
fram5.grid(column =7, row= 3, columnspan=8, rowspan=8, padx=10, pady=5)

canvas.grid(column =7, row= 3, columnspan=8, rowspan=8, padx=10, pady=5)

Board_Button.grid( row=0, column=2, sticky='nsew')

#Run main application:

app.mainloop()
```

Ergebnis

Abbildung 17: Home Seite

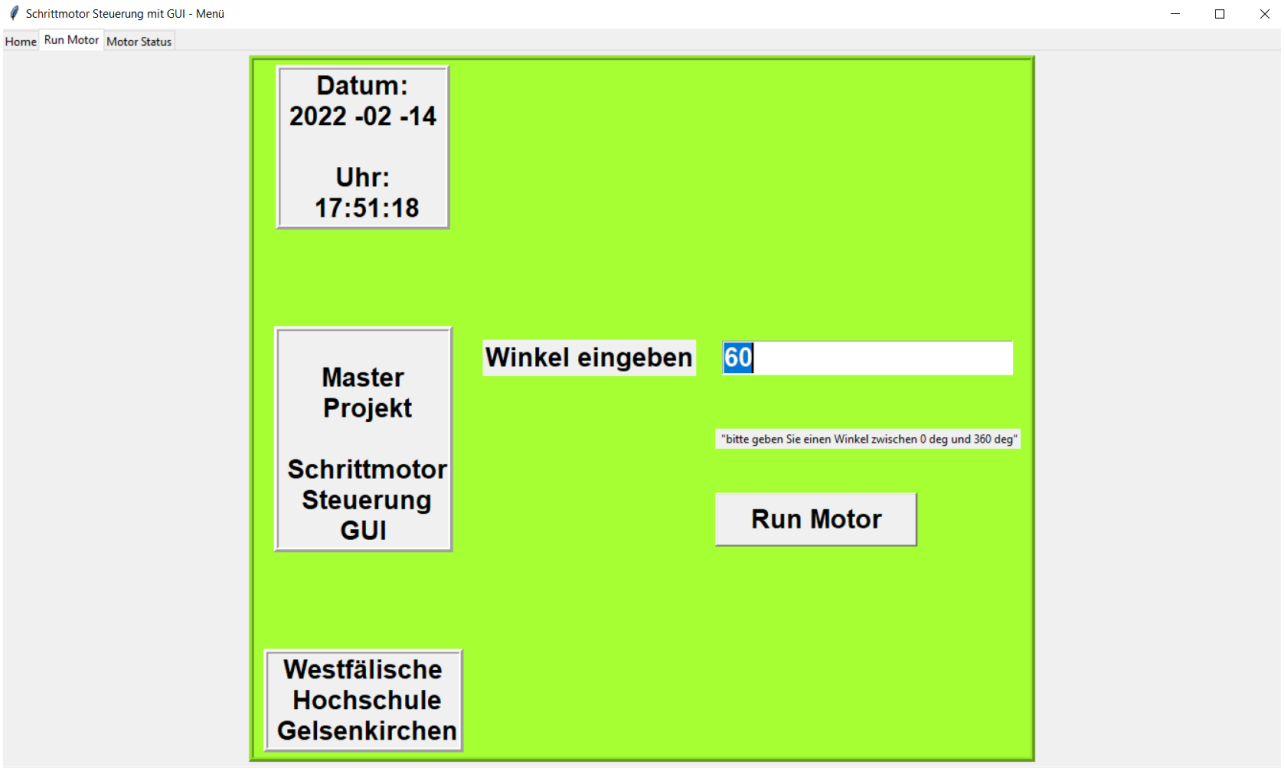


Abbildung 18: Rin-Motor Seite

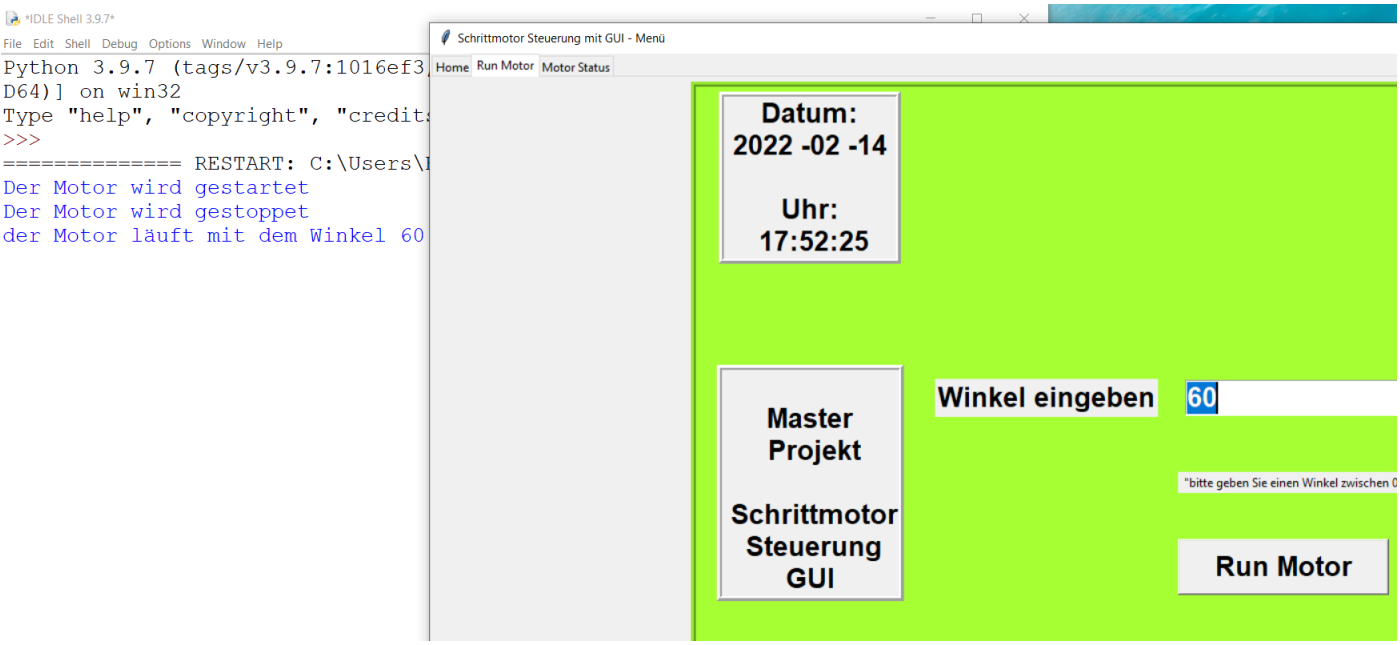


Abbildung 19: Testen von Methoden

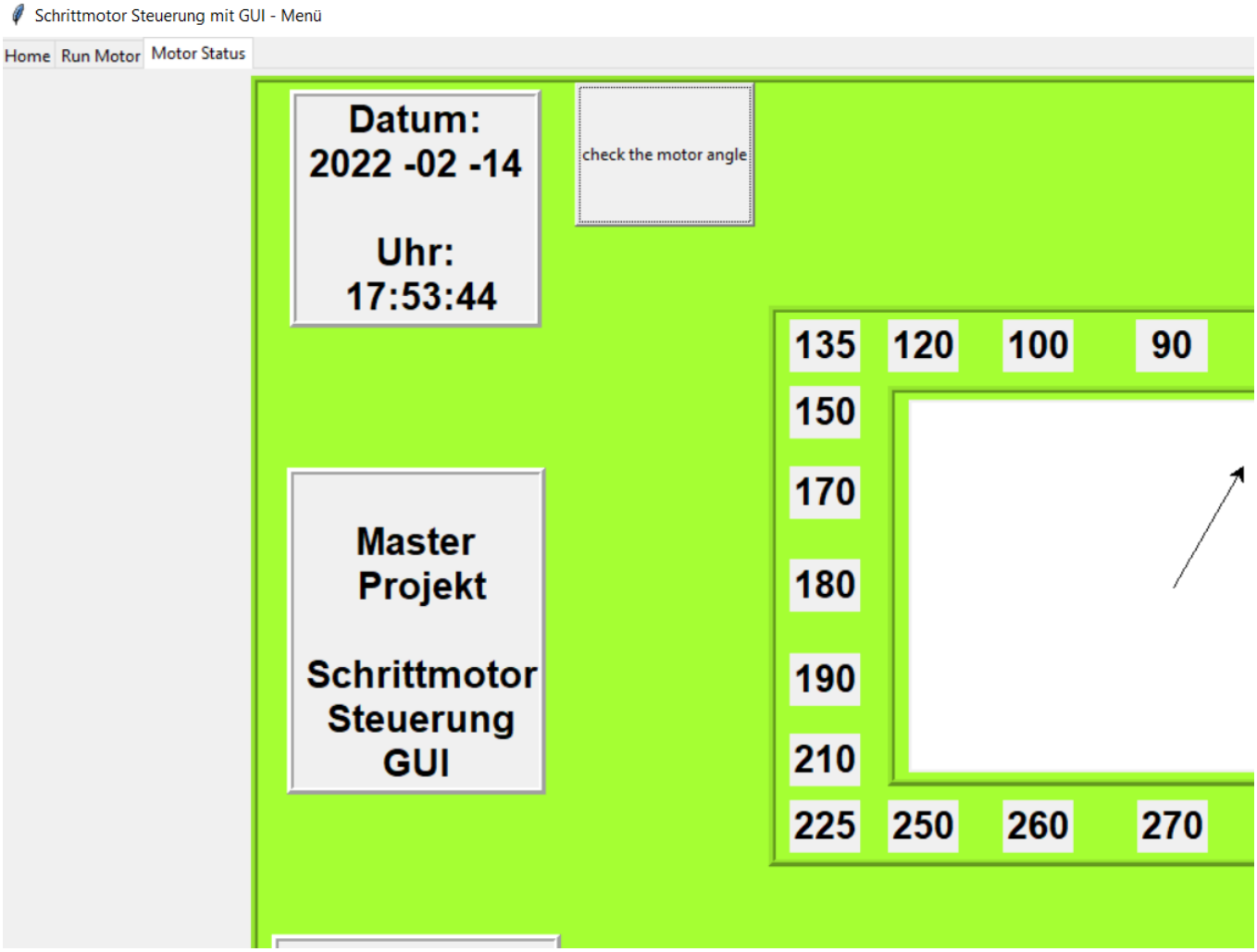


Abbildung 20: Motor Status Seite