

```
1 import tkinter
2 import sys
3 import KlasseRoboter
4 #import GUI_tkinter
5 class MyApp(tkinter.Frame):
6     programm_laeuft="nein"
7     def __init__(self, master=None):
8         super().__init__(master)
9         self.pack()
10        self.createWidgets()
11        self.my_roboter=KlasseRoboter.Roboter()
12
13
14
15    def createWidgets(self):
16        self.howtoLabel_1 = tkinter.Label(self)
17        self.howtoLabel_1.pack()
18        self.text_1 = tkinter.StringVar()
19        self.text_1.set(\
20            "Diese Programm steuert den Roboter.\n \
21            Er muss dazu wissen, wie viele Scheiben der Turm hat. \n \
22            Bitte geben Sie die Anzahl als Zahl (z. Bsp 4) ein:")
23        self.howtoLabel_1["textvariable"] = self.text_1
24
25
26
27        self.nameEntry_1 = tkinter.Entry(self,width=3)
28        self.nameEntry_1.pack()
29        self.name_1 = tkinter.StringVar()
30        self.name_1.set(1)
31        self.nameEntry_1["textvariable"] = self.name_1
32
33        self.howtoLabel_2 = tkinter.Label(self)
34        self.howtoLabel_2.pack()
35        self.text_2 = tkinter.StringVar()
36        self.text_2.set(\
37            "Der Turm wird verschoben von Position :")
38        self.howtoLabel_2["textvariable"] = self.text_2
39
40        self.nameEntry_2 = tkinter.Entry(self,width=3)
41        self.nameEntry_2.pack()
42        self.name_2 = tkinter.StringVar()
43        self.name_2.set("A")
44        self.nameEntry_2["textvariable"] = self.name_2
45
46        self.howtoLabel_3 = tkinter.Label(self)
47        self.howtoLabel_3.pack()
48        self.text_3 = tkinter.StringVar()
49        self.text_3.set(\
50            "nach Position :")
51        self.howtoLabel_3["textvariable"] = self.text_3
52
53        self.nameEntry_3 = tkinter.Entry(self,width=3)
54        self.nameEntry_3.pack()
55        self.name_3 = tkinter.StringVar()
56        self.name_3.set("B")
57        self.nameEntry_3["textvariable"] = self.name_3
58
59        self.nameEntry_4 = tkinter.Entry(self,width=50)
60        self.nameEntry_4.pack()
61        self.name_4 = tkinter.StringVar()
62        self.name_4.set(self.name_1.get()+self.name_2.get()+self.name_3.get())
63        self.nameEntry_4["textvariable"] = self.name_4
64
65        #self.ok = tkinter.Button(self)
66        #self.ok["text"] = "Ok"
67        #self.ok["command"] = self.onReverse()
68        #self.ok.pack()
69
```

```

70     self.rev_1 = tkinter.Button(self)
71     self.rev_1["text"] = "Start"
72     self.rev_1["command"] = self.onReverse
73     self.rev_1.pack()
74
75
76
77
78     self.rev_2 = tkinter.Button(self)
79     self.rev_2["text"] = "Not aus!"
80     self.rev_2["command"] = self.onNotAus
81     self.rev_2.pack()
82
83
84     def onReverse(self):
85         erlaubt1="A B C"
86         erlaubt2="1 2 3 4"
87
88         if (self.name_1.get() in erlaubt2 and self.name_2.get() \
89             in erlaubt1 and self.name_3.get() in erlaubt1 and \
90             MyApp.programm_laeuft=="nein"):
91             self.name_4.set(self.name_1.get()+" "+ \
92                 self.name_2.get()+" "+self.name_3.get()+ \
93                 " ist als Eingabe erlaubt. Das Programm wird gestartet.")
94             if self.name_1.get()=="1":
95                 anzahl=1
96             elif self.name_1.get()=="2":
97                 anzahl=2
98             elif self.name_1.get()=="3":
99                 anzahl=3
100            else :
101                anzahl=4
102
103            if self.name_2.get()=="A":
104                von=1
105            elif self.name_2.get()=="B":
106                von=2
107            else :
108                von=3
109
110            if self.name_3.get()=="A":
111                nach=1
112            elif self.name_3.get()=="B":
113                nach=2
114            else :
115                nach=3
116            print ((anzahl),(von),(nach))
117            print ("OK!")
118
119
120            self.my_roboter.turmVersetzen1(anzahl,von,nach)
121            MyApp.programm_laeuft="ja"
122            #Programmaufruf
123
124            elif (MyApp.programm_laeuft=="ja"):
125                self.name_4.set("Das Programm läuft bereits!")
126                print ("Fehler!")
127            else:
128                self.name_4.set(self.name_1.get()+" "+ \
129                    self.name_2.get()+" "+self.name_3.get()+ \
130                    " ist als Eingabe nicht erlaubt.Bitte korrigieren!")
131                print ("Fehler!")
132
133            def onNotAus(self):
134                # Stepper nach oben
135                # Schlitten nach links
136                sys.exit()
137
138            #and self.name_2.get() in erlaubt1 and self.name_3.get() in erlaubt2:

```

```
139         #else:
140         #     self.name_4.set(self.name_1.get()+self.name_2.get()+self.name_3.get()+\
141         #         " sind fehlerhaft. Bitte korrigiere die Eingaben oben")
142
143     root=tkinter.Tk()
144     app=MyApp(root)
145     app.mainloop()
146
147
148     # if __name__ == "__main__":
149
150     #     my_gui=GUI_tkinter.MyApp()
151     #     #my_greifer.aufnehmen()
152     #     #time.sleep(0.2)
153     #     #my_greifer.ablegen()
154
```

```

1 import RPi.GPIO as gpio
2 gpio.setmode(gpio.BCM)
3 import time
4 import KlasseSchlitten
5 import KlasseGreifer
6 import KlasseAblage
7 import signal
8 import sys
9 gpio.setmode(gpio.BCM)
10
11 class Roboter:
12     def __init__(self,gpio_step=(12,20,16,21),gpio_mot=(27,17),\
13                 gpio_slt=(13,26,5),gpio_mag=(22,4),winkel_step=430,sleep_step=0.005):
14         self.gpio_step=gpio_step
15         self.gpio_mot=gpio_mot
16         self.gpio_slt=gpio_slt
17         self.gpio_mag=gpio_mag
18         self.winkel_step=winkel_step
19         self.sleep_step=sleep_step
20
21         self.my_schlitten_1=KlasseSchlitten.Schlitten(self.gpio_mot,self.gpio_slt)
22
23         self.my_ablage_1=KlasseAblage.Ablage()
24         self.my_greifer_1=KlasseGreifer.Greifer(self.gpio_step,self.winkel_step, \
25         self.sleep_step,self.gpio_mag,self.my_ablage_1)
26     def turmVersetzen(self,anzahl,von,ueber,nach):
27         print(anzahl,von,ueber,nach)
28
29         if anzahl > 1:
30
31             self.turmVersetzen(anzahl-1,von,nach,ueber)
32             #Lege die letzte Scheibe von "von" nach "nach"
33             print ("im
turmversetzen",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anzah
l(3))
34             print ("im turmversetzen",von,nach,ueber)
35             self.my_schlitten_1.zu_position(von)
36             self.my_greifer_1.aufnehmen(von)
37             self.my_ablage_1.verringere_um_1(von)
38             print ("im
turmversetzen",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anzah
l(3))
39             self.my_schlitten_1.zu_position(nach)
40             self.my_greifer_1.ablegen(nach)
41             self.my_ablage_1.erhoehe_um_1(nach)
42             print ("im
turmversetzen",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anzah
l(3))
43             if anzahl > 1:
44
45                 self.turmVersetzen(anzahl-1,ueber,von,nach)
46
47             def turmVersetzen1(self,anzahl,von,nach):
48                 self.my_ablage_1.setze_anzahl(von,anzahl)
49                 self.my_ablage_1.setze_anzahl(nach,0)
50                 ueber=6-von-nach
51                 self.my_ablage_1.setze_anzahl(ueber,0)
52                 print ("im
turmversetzen1",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anza
hl(3))
53                 self.turmVersetzen(anzahl,von,ueber,nach)
54
55
56
57
58 if __name__ == "__main__":
59
60     my_roboter=Roboter()
61     my_roboter.turmVersetzen1(4,1,3)

```

```
62 | #time.sleep(0.2)  
63 | #my_greifer.ablegen()  
64 |  
65 |  
66 |
```

```
1 import RPi.GPIO as gpio
2 import time
3 import KlasseStepper
4 import KlasseMagnet
5 import KlasseAblage
6 import signal
7 import sys
8 gpio.setmode(gpio.BCM)
9 class Greifer():
10     def __init__(self,tup4_step,wink_step,slep_step,tup2_mag,ablage):
11
12         self.steppertupel=tup4_step
13         self.stepperwinkel=wink_step
14         self.stepsleep=slep_step
15
16         self.magnettupel=tup2_mag
17         self.my_magnet=KlasseMagnet.Magnet(self.magnettupel)
18         self.my_ablage=ablage
19         self.my_stepper=KlasseStepper.Stepper(self.steppertupel)
20
21     def aufnehmen(self,position):
22         print ("Greiferaufnehmen",self.my_ablage.gib_anzahl(position))
23         winkel=self.stepperwinkel-self.my_ablage.gib_anzahl(position)*90
24         print ("Greiferaufnehmen runter",winkel,position)
25         self.my_stepper.drehe_um_winkel(+winkel,self.stepsleep)
26         self.my_magnet.ein()
27         print ("Greiferaufnehmen hoch",winkel)
28         self.my_stepper.drehe_um_winkel(-winkel,self.stepsleep)
29
30     def ablegen(self,position):
31         print ("Greiferablegen",self.my_ablage.gib_anzahl(position))
32         winkel=self.stepperwinkel-(self.my_ablage.gib_anzahl(position)+1)*90
33         print ("Greiferablegen",winkel)
34         self.my_stepper.drehe_um_winkel(winkel,self.stepsleep)
35         self.my_magnet.aus()
36         self.my_stepper.drehe_um_winkel(-winkel,self.stepsleep)
37
38 if __name__ == "__main__":
39     #my_ablage.erhoehe_um_1(3)
40     my_ablage=KlasseAblage.Ablage()
41     print ("Ausgabe1",my_ablage.gib_anzahl(3))
42     my_ablage.erhoehe_um_1(3)
43     my_ablage.erhoehe_um_1(3)
44     print ("Ausgabe2",my_ablage.gib_anzahl(3))
45
46     my_greifer=Greifer((12,20,16,21),360,0.009,(22,4),my_ablage)
47     my_greifer.aufnehmen(3)
48     #time.sleep(0.2)
49     my_greifer.ablegen(3)
50
51     gpio.cleanup()
52
53
54
```

```
1 import RPi.GPIO as gpio
2 import time
3 import KlasseSchalterreihe
4 import KlasseMotor
5
6 gpio.setmode(gpio.BCM)
7
8 class Schlitten():
9     def __init__(self,tup2_mot,tup3_Slt):
10
11         self.motortupel=tup2_mot
12         #Schalterleiste
13
14         self.Schaltertupel=tup3_Slt
15
16         self.my_motor=KlasseMotor.Motor(self.motortupel)
17
18         self.my_schalterreihe=KlasseSchalterreihe.Schalterreihe(self.Schaltertupel)
19     def nach_eins(self):
20         if not self.my_schalterreihe.position()==1:
21             self.my_motor.links()
22             self.warten_auf_ankunft_an(1)
23             self.my_motor.stop()
24
25     def nach_drei(self):
26         if not self.my_schalterreihe.position()==3:
27             self.my_motor.rechts()
28             self.warten_auf_ankunft_an(3)
29             self.my_motor.stop()
30
31     def nach_zwei(self):
32         if not self.my_schalterreihe.position()==2:
33             if self.my_schalterreihe.position()==1:
34                 self.nach_drei()
35             if self.my_schalterreihe.position()==3:
36                 self.my_motor.links()
37                 self.warten_auf_ankunft_an(2)
38                 self.my_motor.stop()
39
40     def zu_position(self,s_position):
41         if s_position==1:
42             self.nach_eins()
43         if s_position==2:
44             self.nach_zwei()
45         if s_position==3:
46             self.nach_drei()
47
48
49
50     def warten_auf_ankunft_an(self,position):
51         while not self.my_schalterreihe.position()==position:
52             time.sleep(0.1)
53
54 if __name__ == "__main__":
55
56     my_schlitten=Schlitten((27,17),(13,26,5))
57     #my_motor=KlasseMotor.Motor((27,17))
58     #my_schlitten.nach_drei()
59
60     my_schlitten.zu_position(2)
61     #my_schlitten.warten_auf_ankunft_an(3)
62     #self.my_motor.links()
63     #time.sleep(5)
64     #my_motor.stop()
65     #self.my_motor.stop()
66     gpio.cleanup()
67
68
69
```

70 |
71 |
72 |
73 |
74 |


```
1 import RPi.GPIO as gpio
2 import time
3 import KlasseStepper
4 import KlasseMagnet
5 gpio.setmode(gpio.BCM)
6 gpio.setwarnings(False)
7 class Ablage():
8     anzahl_an_1=0
9     anzahl_an_2=0
10    anzahl_an_3=0
11
12    def __init__(self):
13
14        self.anzahl_an_x=0
15    @staticmethod
16    def verringere_um_1(position):
17        if position==1:
18            Ablage.anzahl_an_1=Ablage.anzahl_an_1 - 1
19        if position==2:
20            Ablage.anzahl_an_2=Ablage.anzahl_an_2 - 1
21        if position==3:
22            Ablage.anzahl_an_3=Ablage.anzahl_an_3 - 1
23    @staticmethod
24    def erhoehe_um_1(position):
25        if position==1:
26            Ablage.anzahl_an_1=Ablage.anzahl_an_1 + 1
27        if position==2:
28            Ablage.anzahl_an_2=Ablage.anzahl_an_2 + 1
29        if position==3:
30            Ablage.anzahl_an_3=Ablage.anzahl_an_3 + 1
31    @staticmethod
32    def setze_anzahl(position,zahl):
33        if position==1:
34            Ablage.anzahl_an_1=zahl
35        if position==2:
36            Ablage.anzahl_an_2=zahl
37        if position==3:
38            Ablage.anzahl_an_3=zahl
39    @staticmethod
40    def gib_anzahl(position):
41        if position==1:
42            return Ablage.anzahl_an_1
43        if position==2:
44            return Ablage.anzahl_an_2
45        if position==3:
46            return Ablage.anzahl_an_3
47
48
49
50
51
52
53 if __name__ == "__main__":
54
55     Ablage.erhoehe_um_1(3)
56
57     my_ablage=Ablage()
58
59     #my_ablage.verringere_um_1(3)
60
61     my_ablage.setze_anzahl(3,8)
62     print (my_ablage.gib_anzahl(3))
63     my_ablage.erhoehe_um_1(3)
64     print (my_ablage.gib_anzahl(3))
65     my_ablage.verringere_um_1(3)
66     print (my_ablage.gib_anzahl(3))
```

```
1 import RPi.GPIO as gpio
2 import time
3 gpio.setmode(gpio.BCM)
4 class Magnet():
5     def __init__(self,tup2_mag):
6         self.gpio1=tup2_mag[0]
7         self.gpio2=tup2_mag[1]
8         gpio.setup(self.gpio1,gpio.OUT)
9         gpio.setup(self.gpio2,gpio.OUT)
10        gpio.output(self.gpio1,gpio.LOW)
11        gpio.output(self.gpio2,gpio.LOW)
12
13    def ein(self):
14        gpio.output(self.gpio1,gpio.HIGH)
15        print (gpio.input(self.gpio1))
16        print (gpio.input(self.gpio2))
17
18    def aus(self):
19        gpio.output(self.gpio1,gpio.LOW)
20        print (gpio.input(self.gpio1))
21        print (gpio.input(self.gpio2))
22
23 if __name__ == "__main__":
24
25     my_magnet=Magnet((22,4))
26
27     my_magnet.ein()
28     time.sleep(50)
29     #my_magnet.aus()
30
31     gpio.cleanup()
32
33
34
```

```
1 import RPi.GPIO as gpio
2 import time
3 gpio.setmode(gpio.BCM)
4
5 class Motor():
6     def __init__(self,tup2_mot):
7         self.gpio1=tup2_mot[0]
8         self.gpio2=tup2_mot[1]
9
10        gpio.setup(self.gpio1,gpio.OUT)
11        gpio.setup(self.gpio2,gpio.OUT)
12
13        gpio.output(self.gpio1,gpio.LOW)
14        gpio.output(self.gpio2,gpio.LOW)
15
16        #print (self.gpio1,self.gpio2)
17        #Schalterleiste
18        #self.Schalter1=tup3Slt[0]
19        #self.Schalter2=tup3Slt[1]
20        #self.Schalter3=tup3Slt[2]
21        #self.Schaltertupel=tup3Slt
22
23
24
25    def rechts(self):
26        #Bewegung nach rechts
27        gpio.output(self.gpio1,gpio.HIGH)
28        gpio.output(self.gpio2,gpio.LOW)
29
30    def links(self):
31        #Bewegung nach links
32        gpio.output(self.gpio1,gpio.LOW)
33        gpio.output(self.gpio2,gpio.HIGH)
34
35    def stop(self):
36        #Bewegung stoppen
37        gpio.output(self.gpio1,gpio.LOW)
38        gpio.output(self.gpio2,gpio.LOW)
39
40    def ausgabe_gpio(self):
41        print(gpio.input(self.gpio1))
42        print(gpio.input(self.gpio2))
43
44
45 if __name__ == "__main__":
46
47     my_motor=Motor((27,17))
48
49     my_motor.ausgabe_gpio()
50     print("links")
51     my_motor.links()
52     time.sleep(0.5)
53     my_motor.ausgabe_gpio()
54     my_motor.stop()
55     print("stop")
56     time.sleep(0.5)
57     print("rechts")
58     my_motor.rechts()
59     time.sleep(0.5)
60     my_motor.stop()
61     my_motor.ausgabe_gpio()
62
63
64     gpio.cleanup()
65     gpio.setwarnings(False)
66
67
68
69
```

70 |
71 |
72 |
73 |

```
1 import RPi.GPIO as gpio
2 import time
3 gpio.setmode(gpio.BCM)
4 gpio.setwarnings(False)
5 class Stepper():
6
7
8     def __init__(self,tup4_step):
9
10         self.steppertupel=tup4_step
11         self.in1=tup4_step[0]
12         self.in2=tup4_step[1]
13         self.in3=tup4_step[2]
14         self.in4=tup4_step[3]
15         gpio.setup(self.in1,gpio.OUT)
16         gpio.setup(self.in2,gpio.OUT)
17         gpio.setup(self.in3,gpio.OUT)
18         gpio.setup(self.in4,gpio.OUT)
19
20
21
22     def einstellung_stepper(self,schalterGpio,asleep):
23         gpio.setmode(gpio.BCM)
24
25         schalter=schalterGpio
26
27         print (gpio.input(schalter))
28         while (gpio.input(schalter)==0):
29             gpio.output(self.in1,gpio.HIGH)
30             time.sleep(asleep)
31             gpio.output(self.in1,gpio.LOW)
32             gpio.output(self.in2,gpio.HIGH)
33             time.sleep(asleep)
34             gpio.output(self.in2,gpio.LOW)
35             gpio.output(self.in3,gpio.HIGH)
36             time.sleep(asleep)
37             gpio.output(self.in3,gpio.LOW)
38             gpio.output(self.in4,gpio.HIGH)
39             time.sleep(asleep)
40             gpio.output(self.in4,gpio.LOW)
41
42
43
44
45     def drehe_um_winkel(self,winkel,asleep):
46
47         if winkel > 0:
48             drehungen=int(round((winkel*512/360)))
49             i=1
50
51             while i in range (0,drehungen):
52                 gpio.output(self.in1,gpio.HIGH)
53                 time.sleep(asleep)
54                 gpio.output(self.in1,gpio.LOW)
55                 gpio.output(self.in2,gpio.HIGH)
56                 time.sleep(asleep)
57                 gpio.output(self.in2,gpio.LOW)
58                 gpio.output(self.in3,gpio.HIGH)
59                 time.sleep(asleep)
60                 gpio.output(self.in3,gpio.LOW)
61                 gpio.output(self.in4,gpio.HIGH)
62                 time.sleep(asleep)
63                 gpio.output(self.in4,gpio.LOW)
64                 i=i+1
65         else:
66             drehungen=int(round((winkel*512/360*(-1))))
67             i=1
68             while i in range (0,drehungen):
69                 gpio.output(self.in4,gpio.HIGH)
```

```
70         time.sleep(asleep)
71         gpio.output(self.in4,gpio.LOW)
72         gpio.output(self.in3,gpio.HIGH)
73         time.sleep(asleep)
74         gpio.output(self.in3,gpio.LOW)
75         gpio.output(self.in2,gpio.HIGH)
76         time.sleep(asleep)
77         gpio.output(self.in2,gpio.LOW)
78         gpio.output(self.in1,gpio.HIGH)
79         time.sleep(asleep)
80         gpio.output(self.in1,gpio.LOW)
81         i=i+1
82
83
84 if __name__ == "__main__":
85
86     my_stepper=Stepper((12,20,16,21))
87
88     my_stepper.drehe_um_winkel(+30,0.009)
89     #my_stepper.drehe_um_winkel(+100,0.009)
90
91     gpio.cleanup()
92
93
```

```
1 import RPi.GPIO as gpio
2 import time
3 gpio.setmode(gpio.BCM)
4
5 class Schalterreihe(object):
6
7     def __init__(self,tup3S1t):
8
9         self.gpioS1=tup3S1t[0]
10        self.gpioS2=tup3S1t[1]
11        self.gpioS3=tup3S1t[2]
12        gpio.setup(self.gpioS1,gpio.IN,pull_up_down=gpio.PUD_UP)
13        print(gpio.input(self.gpioS1))
14        gpio.setup(self.gpioS2,gpio.IN,pull_up_down=gpio.PUD_UP)
15        print(gpio.input(self.gpioS2))
16        gpio.setup(self.gpioS3,gpio.IN,pull_up_down=gpio.PUD_UP)
17        print(gpio.input(self.gpioS3))
18
19
20
21    def position(self):
22        position="X"
23
24        if gpio.input(self.gpioS1)==0:
25            position=1
26        if gpio.input(self.gpioS2)==0:
27            position=2
28        if gpio.input(self.gpioS3)==0:
29            position=3
30        return position
31
32
33
34
35
36
37
38 if __name__ == "__main__":
39
40     my_schalterreihe=Schalterreihe((13,26,5))
41     gpio.setup(13,gpio.IN,pull_up_down=gpio.PUD_UP)
42
43     gpio.setup(26,gpio.IN,pull_up_down=gpio.PUD_UP)
44
45     gpio.setup(5,gpio.IN,pull_up_down=gpio.PUD_UP)
46
47     #Die Schalter müssen gedrückt werden
48     for i in range (10):#my_schlitten.nach_eins()
49         print(my_schalterreihe.position())
50         time.sleep(2)
51
52
53
54     gpio.cleanup()
55
56
57
```