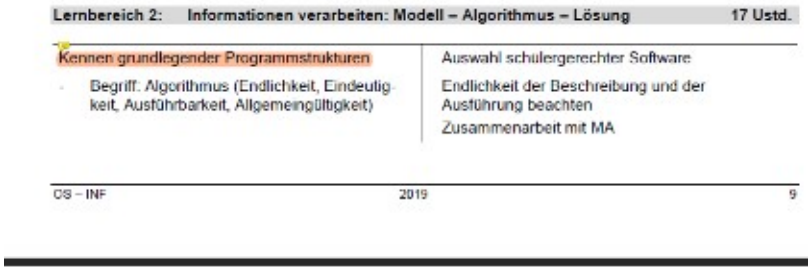
Handreichung PhC

1. Kurzvorstellung

- Bau und Programmierung eines Roboters
- Steuerung über einen Raspberry Pi
- SuS programmieren Roboter über vorgegeben Befehle

2. Einordnung in die Lehrpläne

OS Klasse 8 LB 2 Modell – Algorithmus – Lösung

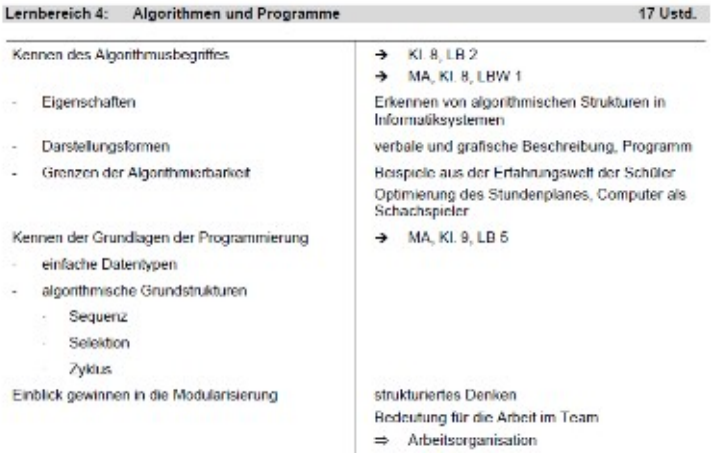


Lernbereich 2: Informationen verarbeiten: Modell – Algorithmus – Lösung		17 Ustd.
Kennen grundlegender Programmstrukturen	Auswahl schülergerechter Software	
- Begriff: Algorithmus (Endlichkeit, Eindeutigkeit, Ausführbarkeit, Allgemeingültigkeit)	Endlichkeit der Beschreibung und der Ausführung beachten Zusammenarbeit mit MA	

OS – INF 2019 9

Oberschule	
- Programmstrukturen: Folge, Wiederholung, Verzweigung	Ausgangspunkt: umgangssprachliche Handlungsbeschreibung
- eine Darstellungsform	Struktogramm, Ablaufplan, Zustandsdiagramm

Gy Klasse 9/10 LB 4 Algorithmen und Programme



Lernbereich 4: Algorithmen und Programme		17 Ustd.
Kennen des Algorithmusbegriffes	→ Kl. 8, LB 2 → MA, Kl. 8, LBW 1	
- Eigenschaften	Erkennen von algorithmischen Strukturen in Informatiksystemen	
- Darstellungsformen	verbale und grafische Beschreibung, Programm	
- Grenzen der Algorithmierbarkeit	Beispiele aus der Erfahrungswelt der Schüler Optimierung des Stundenplanes, Computer als Schachspieler	
Kennen der Grundlagen der Programmierung	→ MA, Kl. 9, LB 5	
- einfache Datentypen		
- algorithmische Grundstrukturen		
- Sequenz		
- Selektion		
- Zyklus		
Einblick gewinnen in die Modularisierung	strukturiertes Denken Bedeutung für die Arbeit im Team ⇒ Arbeitsorganisation	

<Autor>

3. Lernziele

SuS lernen Algorithmen zum steuern eines Roboters kennen. (Kognitiv)

SuS modellieren eigene Bewegungsabfolgen des Roboters. (Kognitiv)

SuS schreiben Algorithmen in Python. (Kognitiv)

SuS werden Aufmerksam auf Bauteile des Roboters. (Affektiv)

SuS wenden das EVA-Prinzip auf Roboter an. (Kognitiv)

SuS manipulieren das Verhalten des Roboters über eingaben im Programmcode. (psychomotorisch)

4. Voraussetzungen

Fachliche Voraussetzungen sind:

- SuS können am Computer Textdateien bearbeiten.
- SuS kennen die Begriffe Klasse, Objekt, Methode und Attribut und sind in der Lage diese zu verbinden.
- SuS kennen das EVA-Prinzip.

Technische Voraussetzungen sind:

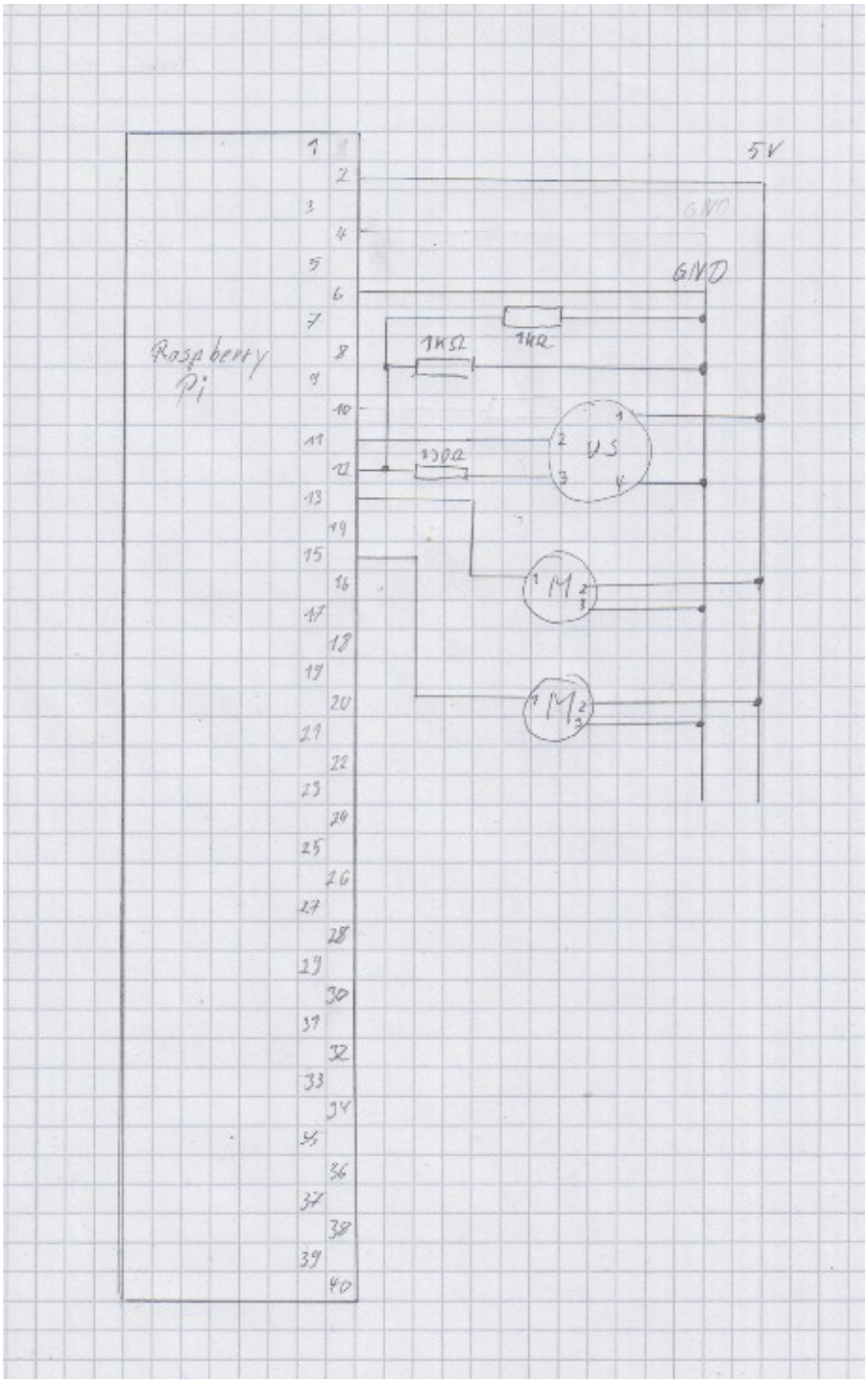
- ein Texteditor

Materielle Voraussetzungen sind:

- ein RasberryPi
- ein Ultraschallentfernungssensor
- zwei Servomotoren
- eine Steckplatine
- mehrere Verbindungsdrähte
- verschiedene Widerstände (330Ω und 500Ω)
- zwei Räder, welche auf die Servomotoren passen
- ein frei drehbares Rad
- eine Platte (Holz oder Kunststoff)

5. Kurzdarstellung

RaspberryPi wie im folgenden Schaltplan verdrahten.



Eine Platte so zuschneiden, das Raspberry, Steckplatine und die Servomotoren darauf passen und sie anschließend dort befestigen.

Die Räder an die Servomotoren Steckern und das frei drehbare an der Unterseite befestigen.

Die SD-Karte des Raspberry m,it dem Betriebssystem bespielen.

Nun einen Bildschirm und eine Tastatur anschließen um den Roboter mit den Grundlegenden Klassen zu beschreiben.

Vorgegebene Klassen des Roboters:

```
import RPi.GPIO as GPIO
import time

class robo_technik:

    def __init__(self, pinStart, pinAugeIn, pinAugeOut, pinR, pinL):
        self.pinS = pinStart
        self.pinIn = pinAugeIn
        self.pinOut = pinAugeOut
        self.pinR = pinR
        self.pinL = pinL

    def start(self):
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(self.pinS, GPIO.IN)
        GPIO.setup(self.pinIn, GPIO.IN)
        GPIO.setup(self.pinOut, GPIO.OUT)
        GPIO.setup(self.pinR, GPIO.OUT)
        GPIO.setup(self.pinL, GPIO.OUT)

    def ende(self):
        GPIO.cleanup()

    def sehen(self):
        time.sleep(0.02)
        GPIO.output(self.pinOut, True)
        time.sleep(0.00001)
        GPIO.output(self.pinOut, False)
        tSart= time.time()
        tEnd =time.time()

        while GPIO.input(self.pinIn) == 0:
            tStart = time.time()

        while GPIO.input(self.pinIn) == 1:
            tEnd = time.time()

        tWert = tEnd - tStart

        sWert = (tWert * 34300) / 2

        print(sWert)

        if sWert > 10 and sWert < 50:
            print("frei")
            return True
        else:
            print("stopp")
            return False
```

```
def vor(self):
    GPIO.output(self.pinL, True)
    time.sleep(0.003)

    GPIO.output(self.pinL, False)
    time.sleep(0.017)

    GPIO.output(self.pinR, True)
    time.sleep(0.000001)

    GPIO.output(self.pinR, False)
    time.sleep(0.02)

def zurueck(self):
    GPIO.output(self.pinL, True)
    time.sleep(0.000001)

    GPIO.output(self.pinL, False)
    time.sleep(0.02)

    GPIO.output(self.pinR, True)
    time.sleep(0.003)

    GPIO.output(self.pinR, False)
    time.sleep(0.017)

def rechts(self):
    GPIO.output(self.pinL, True)
    time.sleep(0.003)

    GPIO.output(self.pinL, False)
    time.sleep(0.017)

    GPIO.output(self.pinR, True)
    time.sleep(0.003)

    GPIO.output(self.pinR, False)
    time.sleep(0.017)

def links(self):
    GPIO.output(self.pinL, True)
    time.sleep(0.000001)

    GPIO.output(self.pinL, False)
    time.sleep(0.02)

    GPIO.output(self.pinR, True)
    time.sleep(0.000001)

    GPIO.output(self.pinR, False)
    time.sleep(0.02)
```

In einer zweiten Datei wird der Roboter programmiert. Zum starten muss folgender Code genutzt werden:

```
import robo_technik

name = robo_technik.robo_technik(16, 12, 11, 13, 15)
name.start()
```

Dabei wird zuerst die Klasse robo_technik importiert. Anschließend wird sie initialisiert mit den nötigen Pins des Raspberry Pi. Der erste Befehlö der Programmierung ist .start(), welche die Pins aktiviert. Der letzte Befehl muss ende() sein, der die Pins deaktiviert.