

Unterrichtsbaustein für eine Automatisierte Belichtungsanlage mit dem Raspberry Pi

Projektbeschreibung

Bei der Auswahl unserer Projektidee war uns wichtig, dass unser Projekt einen Lebensweltbezug hat und die Schüler/innen ebenso praktisch damit arbeiten können. Durch die praktische Anwendung sollen die Schüler/innen Grundlagen zur Programmiersprache erlernen und erste Erfahrungen zu Informatiksystemen sammeln. Die Projektstage ermöglichen einen intensiven Einblick und eine praktische Auseinandersetzung. In diesem Unterrichtsbaustein erlernen Sie schrittweise die Programmierung einer LED-Leiste für ein Mini-Gewächshaus mittels eines Raspberry Pi und der Programmiersprache Python. Das Ziel besteht darin, dass die LED-Leiste bei unterschrittenem Helligkeitswert aktiviert wird. So erhalten die Pflanzen im Gewächshaus auch bei Nacht genug Licht. In einer Gruppengröße von fünf SuS sollen sie unter Anleitung der Lehrkraft die Programmierung und den Zusammenbau der Hardware innerhalb von 5 Unterrichtstagen fertigstellen. Eine Vorerfahrung mit dem Raspberry Pi kann helfen, ist aber nicht zwingend notwendig, da mithilfe dieser Anleitung jeder Schritt detailliert erklärt wird.

Hinweise zur Durchführung in einer Schulklasse:

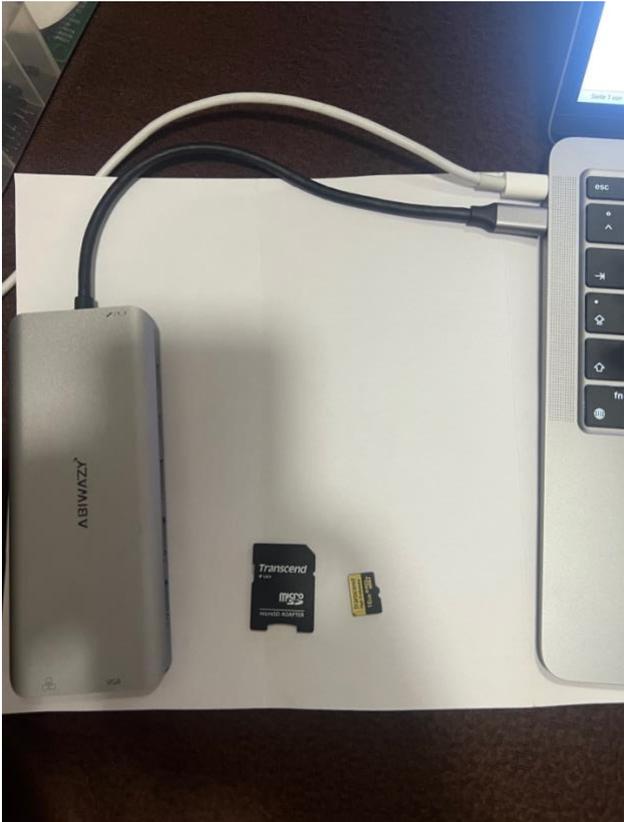
Bei der Durchführung in einer Schulklasse kann es hilfreich sein, wenn die Lehrkraft einige Vorerfahrungen mit dem Raspberry Pi oder Programmiersprache Python hat. Zudem sollte die Lehrkraft vor den Unterrichtseinheiten die Anleitung selbst einige Male durchführen, um Fehlerquellen bei der Durchführung in der Klasse schnell zu erkennen und Hilfe zu leisten. Außerdem muss die Anleitung je nach Klassenstufe und Leistungsniveau der Schüler:innen angepasst werden, so können beispielsweise einige Schritte vorab erledigt werden wie Schritt 1, um zügiger das Projekt zu starten oder zu vereinfachen.

Schritt 1

Material / Medien

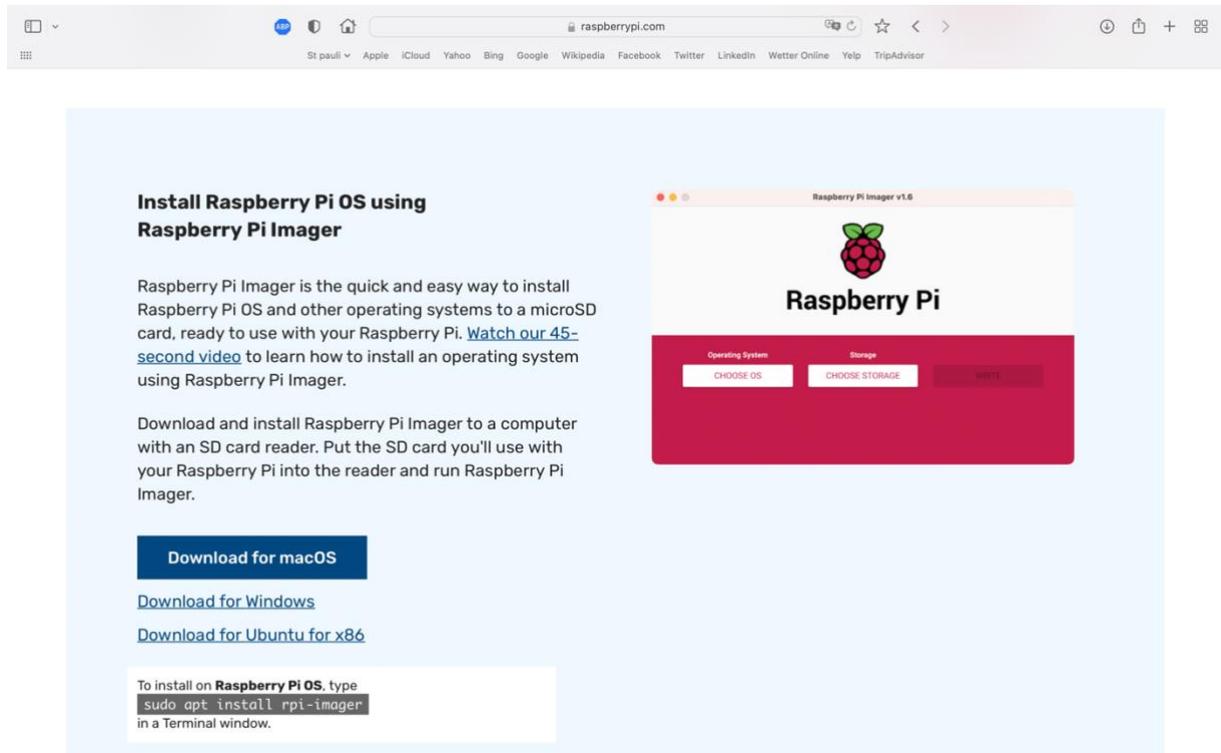
- Laptop
- microSD Karte mit Adapter
- Je nach Kompatibilität Adapter für SD-Karte an Laptop notwendig

In der ersten Lektion wird das Betriebssystem des Raspberry Pi (AB1) welcher ein Einplatinencomputer ist, installiert. Um das Betriebssystem zu installieren, ist eine microSD Karte notwendig, die in einen Adapter hineingefügt wird. Anschließend wird die SD-Karte an den Laptop oder Computer angeschlossen. Je nach Endgerät kann auch wie bei unserem Fall ein zusätzlicher Adapter notwendig sein, um die SD-Karte anzuschließen.



(Abb. 1, 2, 3)

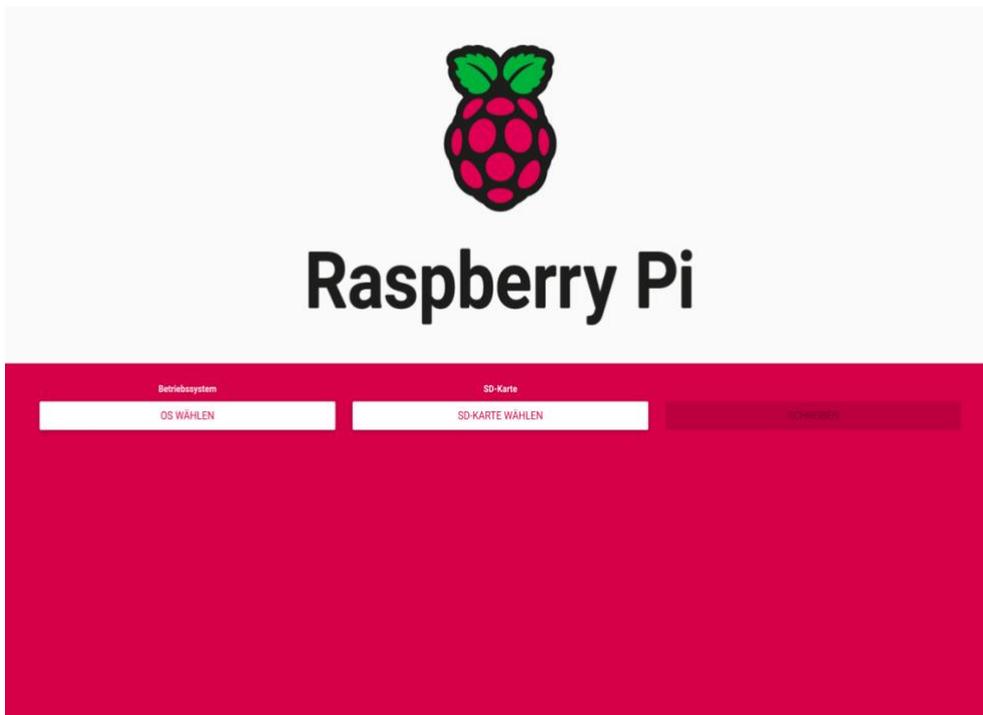
Um das Betriebssystem auf der SD-Karte zu installieren, muss zuerst von der folgenden Internetseite <https://www.raspberrypi.com/software/> der „Imager“ heruntergeladen und installiert werden auf dem Laptop. (Abb. 3)



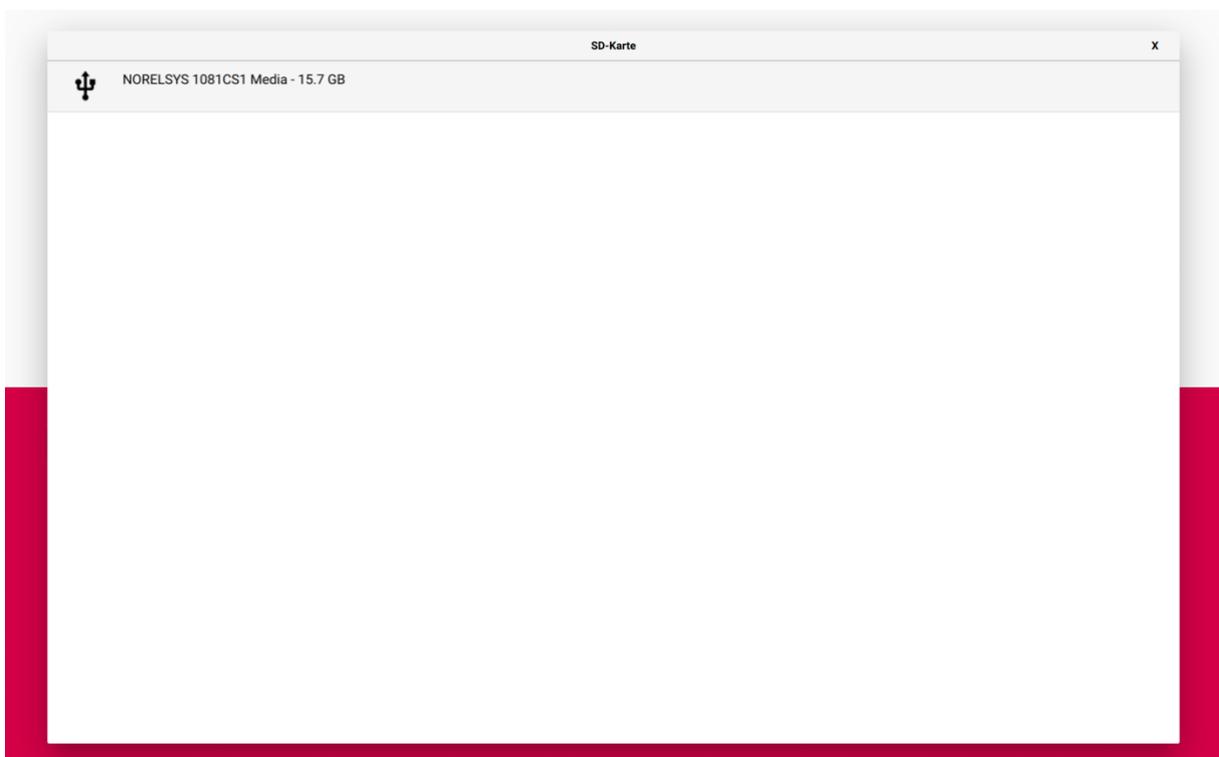
(Abb. 4)

Als nächster Schritt wird das Programm gestartet. Im Programm wird nun die SD-Karte ausgewählt, wie auf Abb. 6 anschließend das zu installierende Betriebssystem. Es wird das Betriebssystem „RASPBERRY PI OS 32-Bit“ (Abb. 7) gewählt und darauffolgend auf

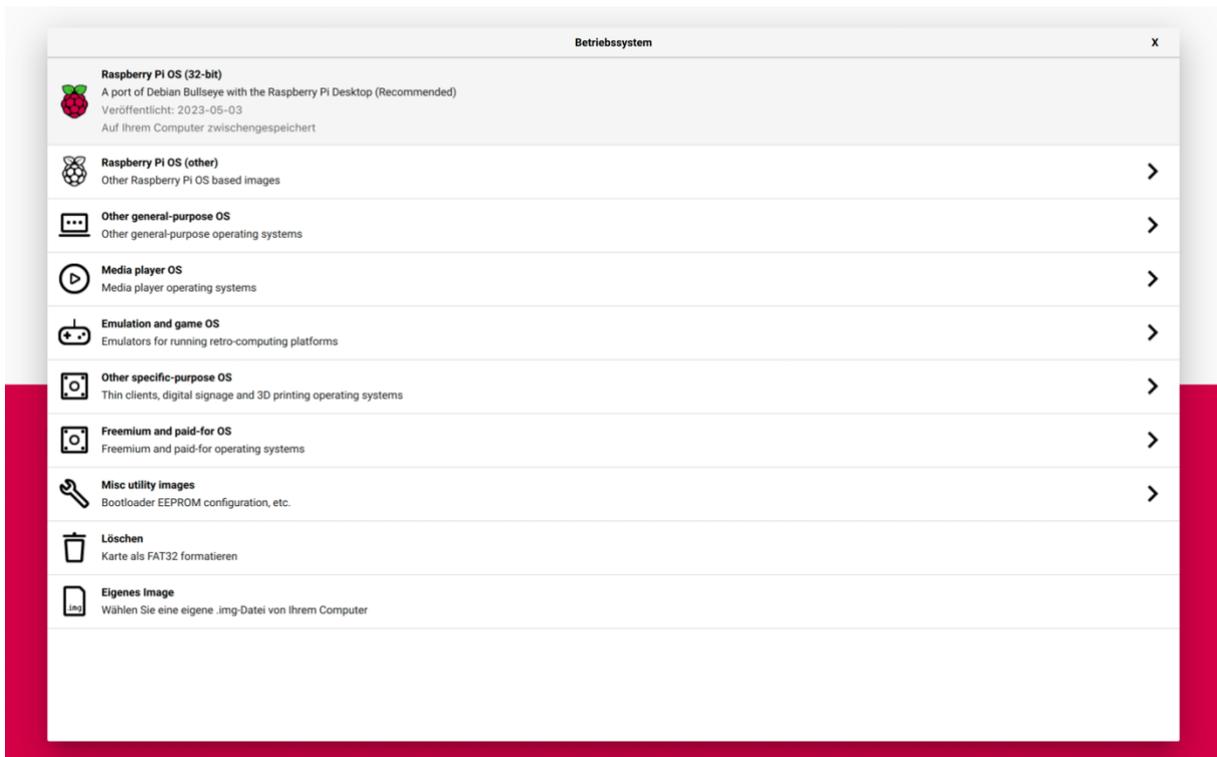
Schreiben geklickt. Daraufhin soll es, wie auf Abb. 8 aussehen.



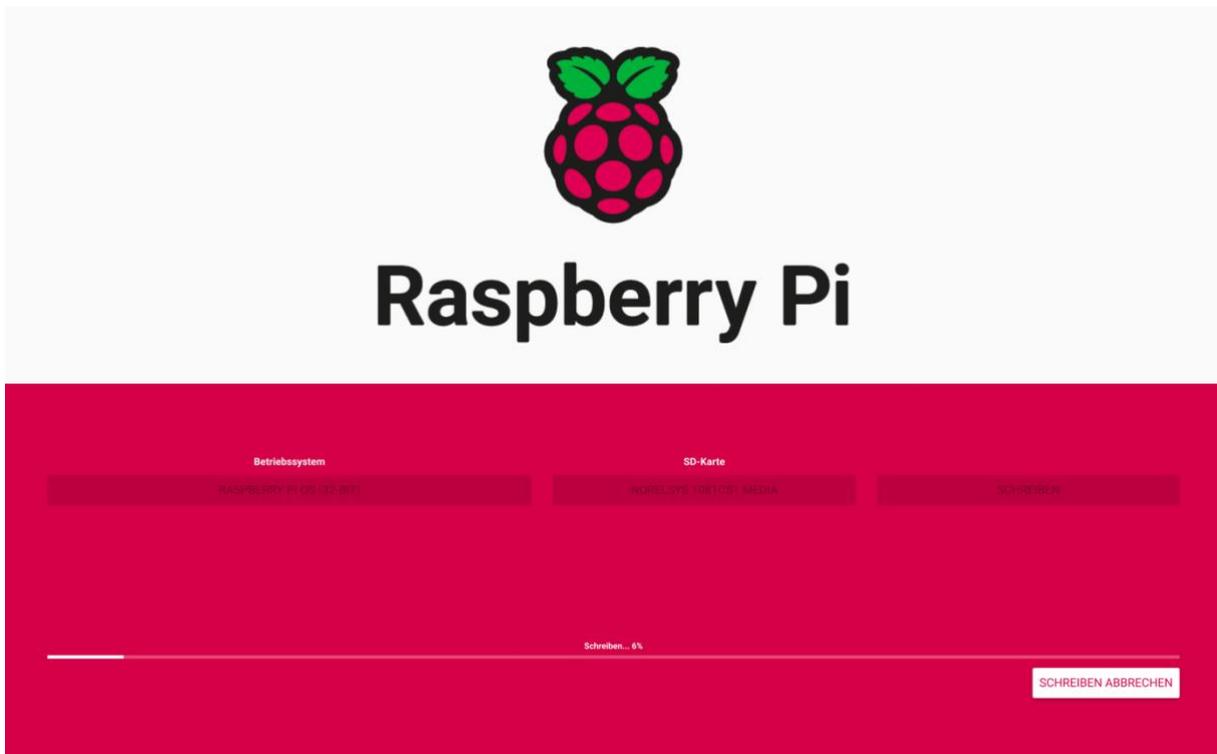
(Abb. 5)



(Abb. 6)



(Abb. 7)



(Abb. 8)

Nach dem Schreibvorgang kann die SD-Karte entfernt werden und somit ist der erste Schritt beendet.

Schritt 2

Medien / Material:

- Monitor mit Netzteil
- Maus
- Tastatur
- HDMI-Kabel
- Raspberry Pi
- Breadboard
- MCP 3008
- Helligkeitssensor
- Jumper Kabel
- Netzteil Raspberry Pi (Raspberry Pi)
- 10k Ohm Widerstand
- Wenn nicht genug Steckdosen vorhanden sind Steckdosenleisten

Im folgenden Schritt wird nun erklärt, wie das Betriebssystem bereitgestellt wird und anschließend der Helligkeitssensor ausgelesen werden kann. Zuerst wird die microSD Karte in das microSD Kartenfach des Raspberry Pi. Anschließend werden Maus und Tastatur per USB verbunden und der Monitor per HDMI-Kabel. Als Letztes werden die Netzteile des Monitors sowie das vom Raspberry Pi an die Geräte als auch an die Steckdosen geschlossen. (Abb. 9&10&11)

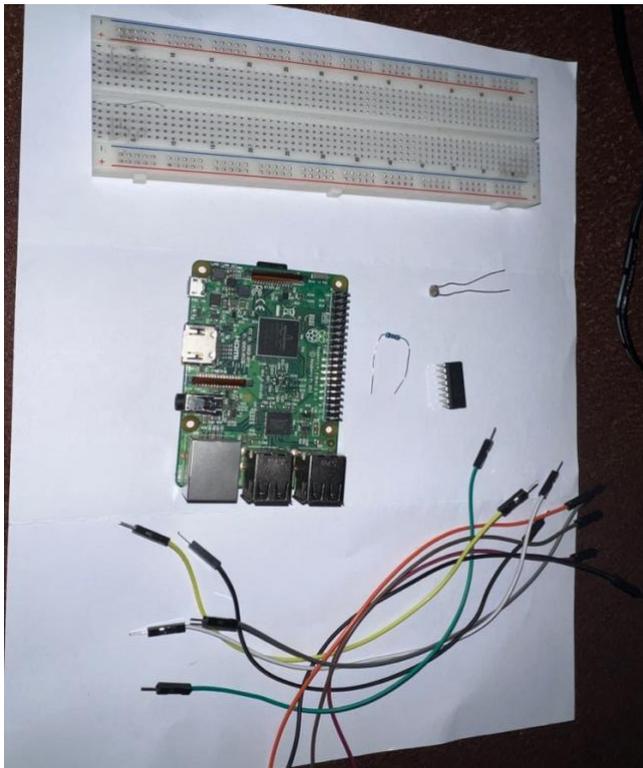


(Abb. 9&10&11)

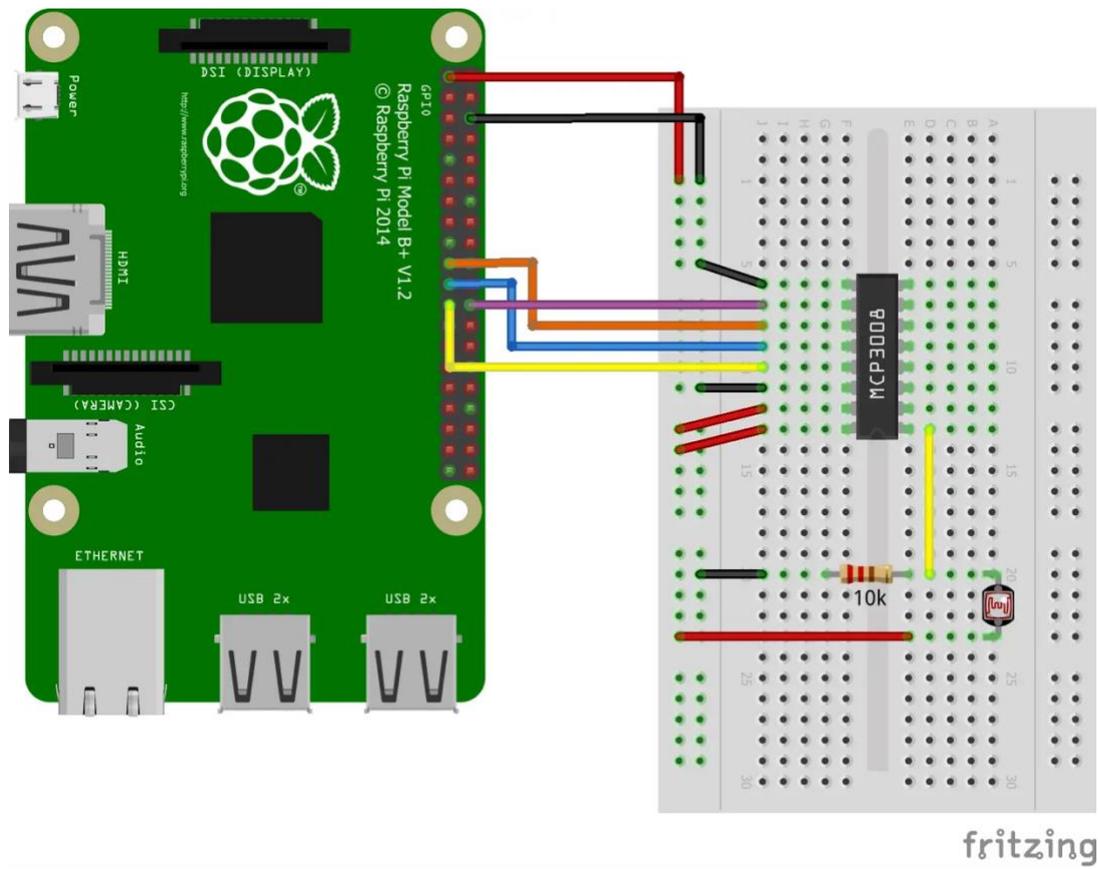
Es wird folgendes auf dem Bildschirm angezeigt, hier wird auf Next geklickt. Daraufhin wird das Land sowie die Sprache eingestellt. Als Nächstes wird ein Benutzerkonto angelegt, wobei es empfehlenswert ist den Benutzernamen und das Passwort abzufotografieren. Beim Klicken

auf Next erscheint nun eine Frage, ob der Bildschirm richtig angezeigt wird, ist dies der Fall wird wieder auf Next geklickt oder auf das Feld neben „Reduce the size of the desktop monitor“. Im nächsten Schritt wird die Internetverbindung bereitgestellt. Da der Raspberry Pi über einen Lan Anschluss verfügt, kann dies über ein Lan-Kabel erfolgen oder es wird eine W-LAN Verbindung hergestellt. Dafür wird im Fenster das entsprechende W-LAN Netzwerk ausgewählt und das Passwort eingegeben. Anschließend wird im nächsten Fenster nach einem „Software-Update“ gefragt, hier wird ebenfalls auf Next geklickt. Das Software-Update kann einige Zeit in Anspruch nehmen. Wenn das Update erfolgreich installiert wurde, wird das System neu gestartet, in dem auf Restart geklickt wird. Für den nächsten Schritt wird der Raspberry Pi ausgeschaltet und von den Anschlüssen entfernt, außer der microSD Karte.

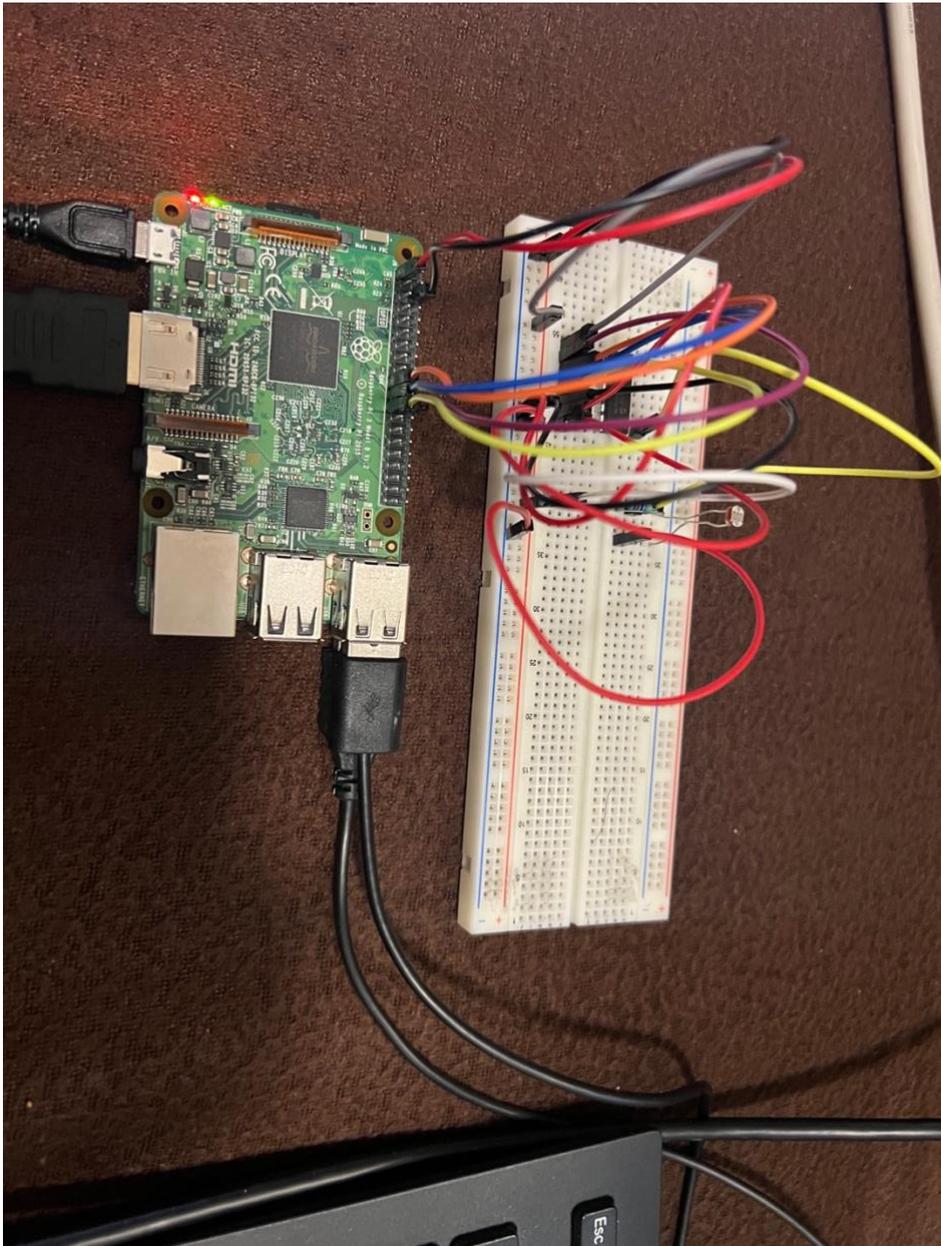
Im folgenden Schritt werden nun die Hardware-Komponenten mit dem Raspberry Pi verbunden. Dafür werden wie auf Abb. 12, das Breadboard, der Helligkeitssensor (Fotowiderstand), die Jumper-Kabel, ein 10k Ohm Widerstand, sowie der MCP-3008 benötigt. Die Verkabelung erfolgt, wie auf Abb. 13. Nachdem alles verkabelt ist, wird das Raspberry-Pi wieder an den Monitor, Maus etc. angeschlossen und gestartet. Wenn das System erfolgreich startet und der Homescreen angezeigt wird, wurde alles richtig angeschlossen.



(Abb. 12)



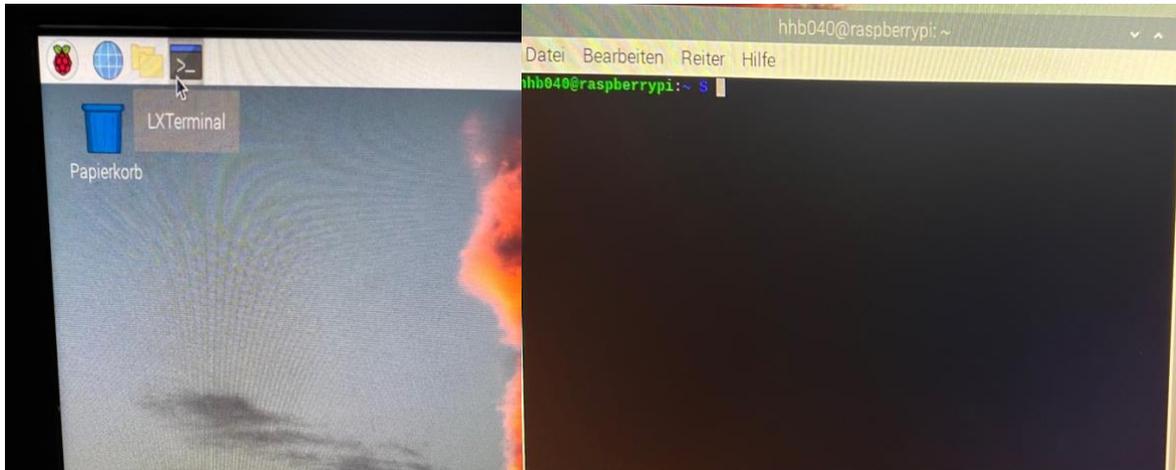
(Abb. 13) (Vgl. <https://tutorials-raspberrypi.de/wp-content/uploads/2017/01/Raspberry-Pi-Helligkeitssensor-Fototransistor-Steckplatine.png>)



(Abb. 14)

Schritt 3

Im folgenden Schritt erfolgt die Programmierung, dafür wird oben Links das LXTerminal geöffnet. In diesem Terminal können nun Skripte geschrieben werden. Es wird dabei die Programmiersprache „Python“ verwendet.



(Abb. 15&16)

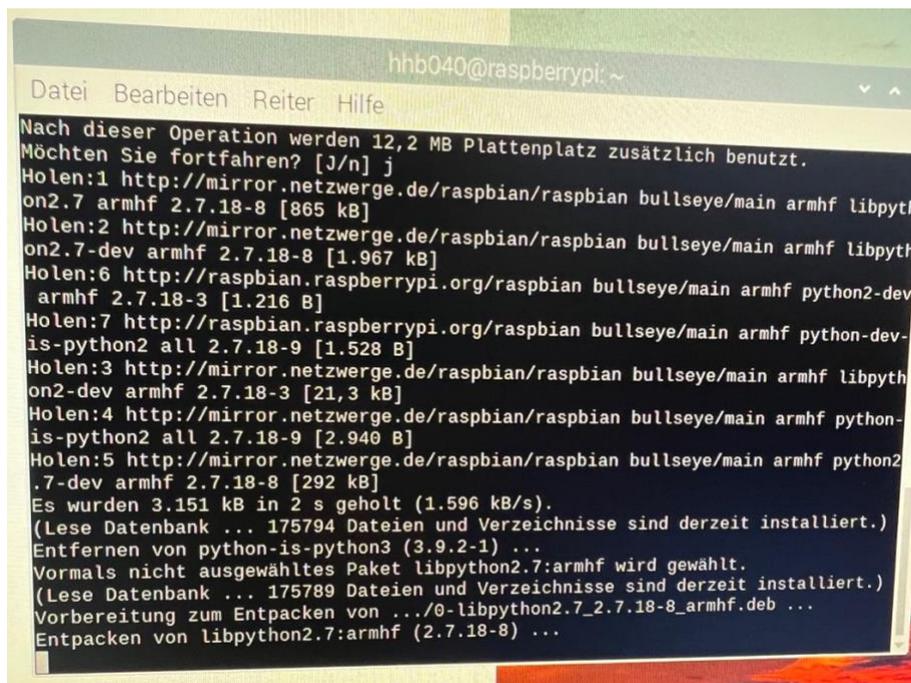
Es werden folgende Befehle nacheinander in das Terminal eingefügt:

```
sudo apt-get update  
sudo apt-get upgrade  
sudo apt-get install python-dev
```

Anschließend wird folgender Befehl eingegeben:

```
wget https://github.com/doceme/py-spidev/archive/master.zip  
unzip master.zip  
cd py-spidev-master  
sudo python setup.py install
```

Das Fenster wird ungefähr, wie auf Abb. 17 aussehen.

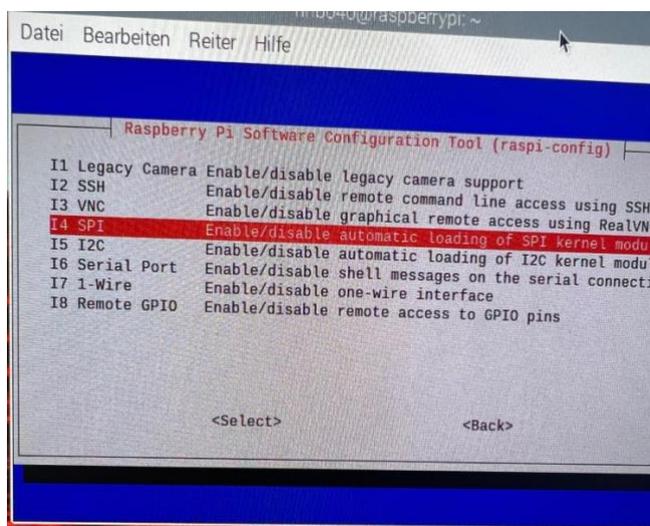


(Abb. 17)

Als Nächstes wird der SPI-Bus aktiviert, das erfolgt, indem folgender Befehl eingegeben wird:

```
sudo raspi-config
```

Beim aufkommenden Fenster wird mit den Pfeiltasten zum 3. Punkt navigiert und mit Enter bestätigt. Anschließend wird mit der Pfeiltaste zum 4. Punkt navigiert und ebenfalls bestätigt, wie auf Abb. 18. Zum Abschluss wird dann auf Ja navigiert und mit Finish wird der Vorgang beendet.



(Abb.18)

Nun wird der Raspberry Pi neu gestartet, indem folgender Befehl in das Terminal eingegeben wird:

Sudo reboot

Wenn der Raspberry neu gestartet ist, kann nun mit dem Skript für das Auslesen des Helligkeitswertes begonnen werden.

Sudo nano

Anschließend kann dieses Skript in das kommende Fenster kopiert werden:

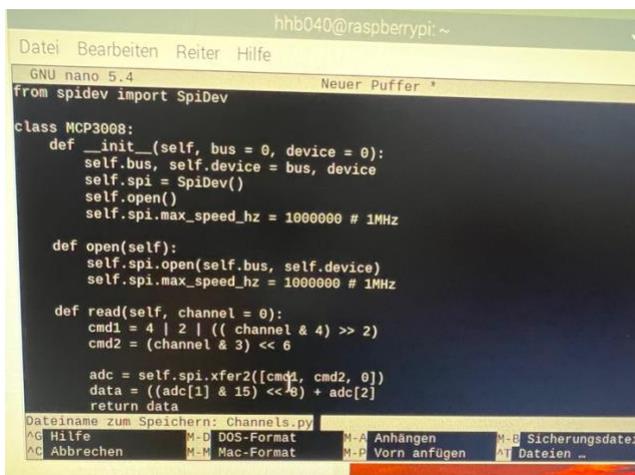
```
from spidev import SpiDev

class MCP3008:
    def __init__(self, bus = 0, device = 0):
        self.bus, self.device = bus, device
        self.spi = SpiDev()
        self.open()

    def open(self):
        self.spi.open(self.bus, self.device)

    def read(self, channel = 0):
        adc = self.spi.xfer2([1, (8 + channel) << 4, 0])
        data = ((adc[1] & 3) << 8) + adc[2]
        return data

    def close(self):
        self.spi.close()
```



```
hhb040@raspberrypi: ~
Datei Bearbeiten Reiter Hilfe
GNU nano 5.4 Neuer Puffer *
from spidev import SpiDev

class MCP3008:
    def __init__(self, bus = 0, device = 0):
        self.bus, self.device = bus, device
        self.spi = SpiDev()
        self.open()
        self.spi.max_speed_hz = 1000000 # 1MHz

    def open(self):
        self.spi.open(self.bus, self.device)
        self.spi.max_speed_hz = 1000000 # 1MHz

    def read(self, channel = 0):
        cmd1 = 4 | 2 | ((channel & 4) >> 2)
        cmd2 = (channel & 3) << 6

        adc = self.spi.xfer2([cmd1, cmd2, 0])
        data = ((adc[1] & 15) << 8) + adc[2]
        return data

    def close(self):
        self.spi.close()

Dateiname zum Speichern: Channels.py
^G Hilfe M-D DOS-Format M-A Anhängen M-B Sicherungsdatei
^G Abbrechen M-M Mac-Format M-D Vorn anfügen M-I Dateien ..
```

```

hhb040@raspberrypi: ~
Datei Bearbeiten Reiter Hilfe
GNU nano 5.4 Channels.py
from spidev import SpiDev
class MCP3008:
    def __init__(self, bus = 0, device = 0):
        self.bus, self.device = bus, device
        self.spi = SpiDev()
        self.open()
        self.spi.max_speed_hz = 1000000 # 1MHz
    def open(self):
        self.spi.open(self.bus, self.device)
        self.spi.max_speed_hz = 1000000 # 1MHz
    def read(self, channel = 0):
        cmd1 = 4 | 2 | (( channel & 4) >> 2)
        cmd2 = (channel & 3) << 6
        adc = self.spi.xfer2([cmd1, cmd2, 0])
        data = ((adc[1] & 15) << 6) + adc[2]
        return data
23 Zeilen geschrieben
Hilfe Speichern Wo ist Ausschneiden Ausführen Position
Beenden Datei öffnen Ersetzen Einfügen Ausrichten Zu Zeile

```

(Abb. 19&20)

Anschließend wird das Skript gespeichert. Das erfolgt, indem Strg+O gedrückt wird und dann kann ein Dateiname eingegeben werden. (Der Dateiname kann beliebig gewählt werden, hier heißt er Channels.py dann kann das Fenster mit Strg+X geschlossen werden. Um das Skript nun auszulesen, wird Folgendes eingegeben:

Python Channels.py

Nun sieht man auf dem Abb. 20 den Helligkeitswert. Es empfiehlt sich mehrere Testläufe wie auf der Abbildung durchzuführen. Die oberen beiden Werte ergaben sich bei normalen Lichtverhältnissen, beim dritten Wert wurde ein Finger auf den Helligkeitssensor gelegt.

```

hhb040@raspberrypi: ~/py-spidev-master $ python channels.py
Anliegende Spannung: 12.08
hhb040@raspberrypi: ~/py-spidev-master $ python channels.py
Anliegende Spannung: 10.38
hhb040@raspberrypi: ~/py-spidev-master $ nano channels.py
hhb040@raspberrypi: ~/py-spidev-master $ nano channels.py
hhb040@raspberrypi: ~/py-spidev-master $ python channels.py
Anliegende Spannung: 6.66

```

(Abb.

20)

Schritt 4:

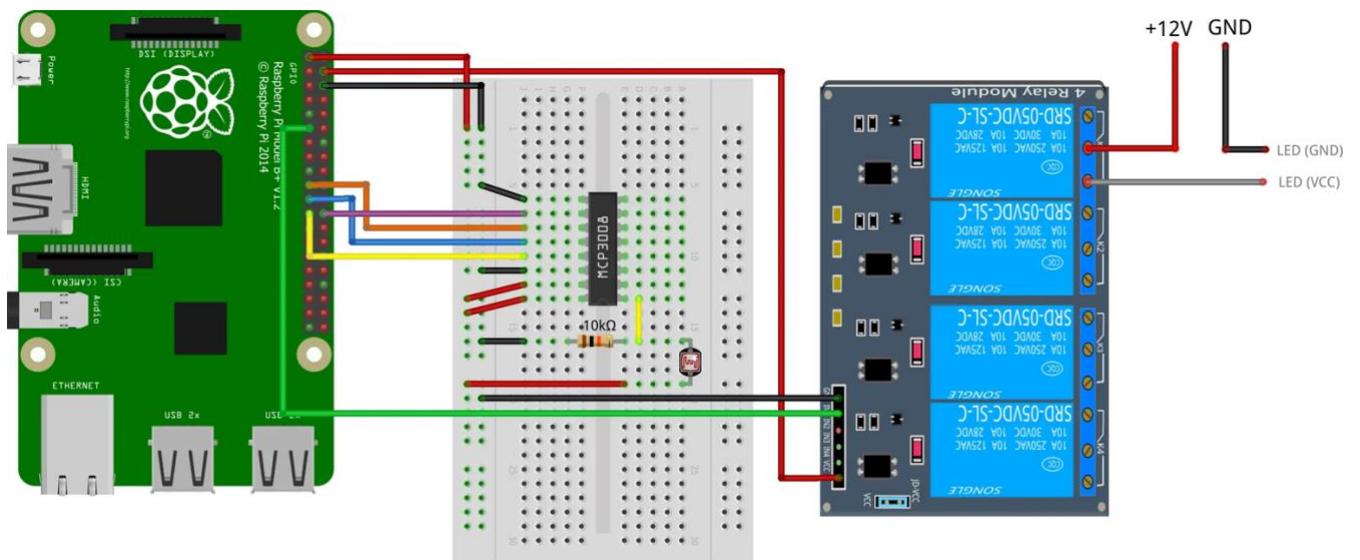
Materialien / Medien:

- Raspberry Pi
- Monitor
- Maus
- Tastatur
- Spannungswandler
- Helligkeitssensor
- 10k Ohm Widerstand
- Steckdosenleiste
- Relaismodul mit Optokoppler
- MCP3008
- Breadboard
- Jumper Kabel / benötigt werden verschiedene benötigt wie Stecker zu Stecker und Stift zu Stecker
- Spannungsmesser

In diesem Schritt wird erklärt, wie die LED-Leiste programmiert wird. Dafür wird mit dem Aufbau der Hardware begonnen. Der Aufbau erfolgt, wie auf Abb. 22&23 da ein Spannungswandler verwendet wird, ist es vor Beginn die Ausgabe des Spannungswandlers wie zu messen mit einem Spannungsmesser. Ist die Ausgabe zu hoch, kann es vorkommen, dass andere Hardwarekomponenten wie der MCP3008 oder sogar der Raspberry Pi durchbrennen. Ist die Spannung zu hoch, wird die Schraube wie auf Abb. 21 zu drehen bis 12V erreicht sind. Anschließend kann mit dem Aufbau begonnen werden. Da aufgrund des Spannungswandlers auf eine weitere Steckdose verzichtet wird, erfolgt der Anschluss der LED-Leiste wie auf Abb. 23 Die roten Kabel werden an das Relaismodul geschlossen, und die schwarzen werden miteinander verbunden und anschließend mit Panzertape befestigt. Anschließend kann der Monitor, Maus etc. wieder verbunden werden. Es soll, wie auf Abb. 24 aussehen.

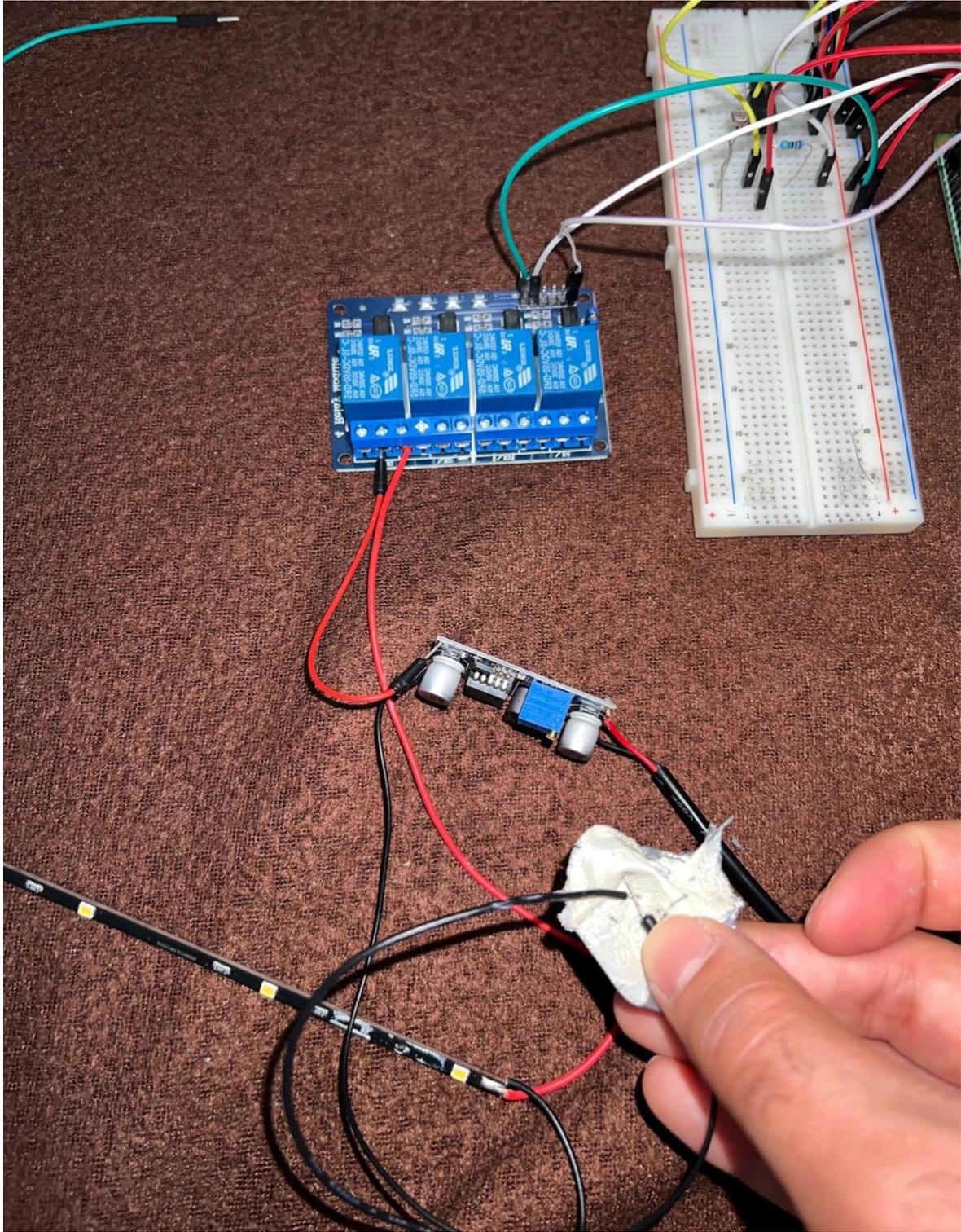


(Abb. 21)



fritzing

(Abb. 22) (<https://tutorials-raspberrypi.de/wp-content/uploads/2017/04/Raspberry-Pi-Lightsensor-MCP3008-Schema.png>)



(Abb. 23)

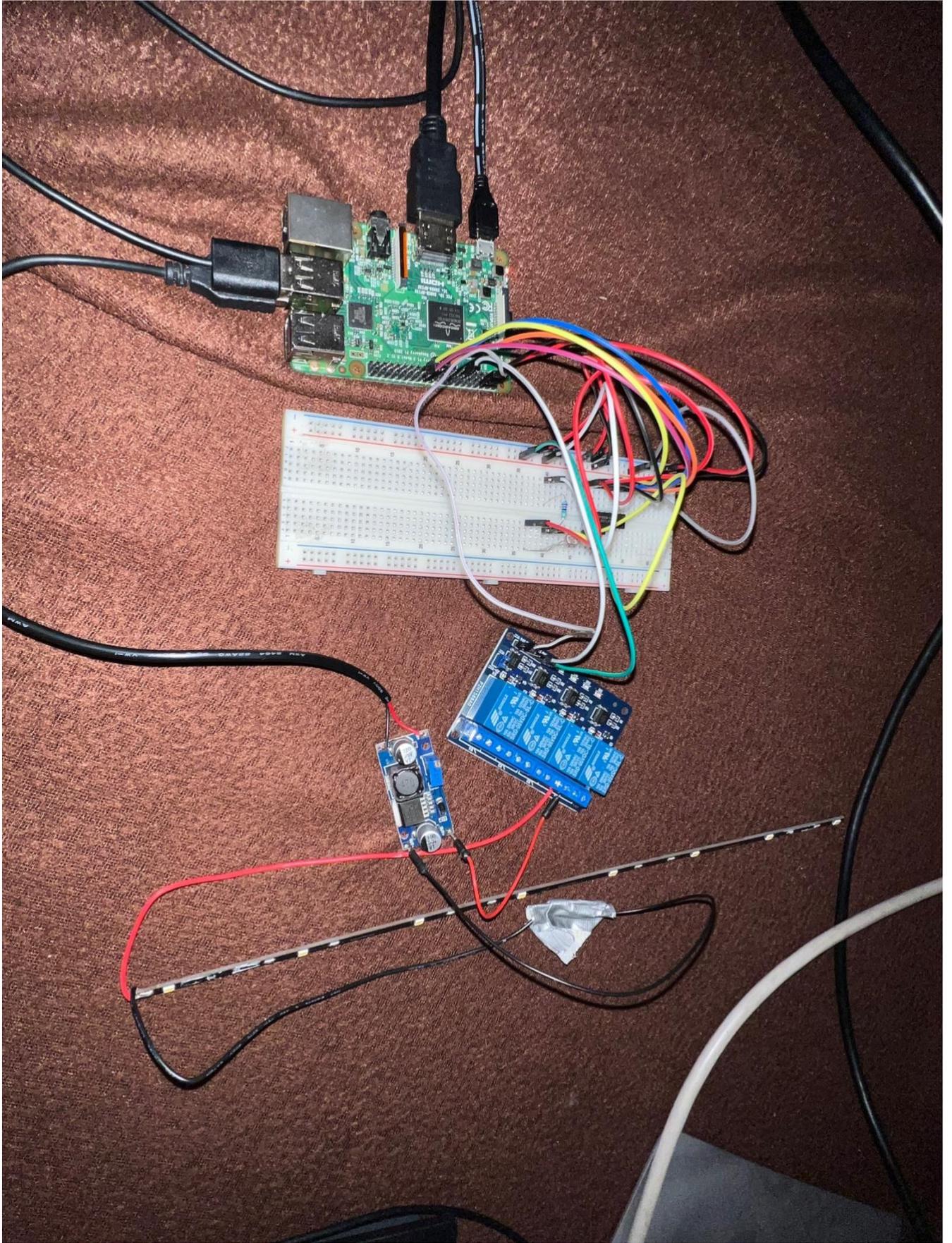


Abb. 24

Schritt 4.1

In diesem Schritt wird nun die Programmierung des Raspberry Pi erklärt. Dazu wird das Terminal geöffnet und folgende Befehle eingegeben:

```
sudo apt-get update --yes && sudo apt-get upgrade --yes  
sudo apt-get install build-essential python-dev python-pip python-smbus python-opensl git --  
yes
```

Nach dem alle Pakete geladen sind, wird der SPI-Bus aktiviert wie bei Schritt 3, dazu wird folgender Befehl eingegeben.

Sudo raspi-config

Anschließend wird zu „Interfacing Options“ navigiert und der SPI-Bus aktiviert. Danach wird der Raspberry Pi neugestartet durch folgenden Befehl:

Sudo Reboot

Nach dem Neustart wird dieser Befehl ins Terminal eingegeben:

```
git clone https://github.com/doceme/py-spidev  
cd py-spidev  
sudo python setup.py install  
cd ..
```

Daran anschließend dieser Befehl:

```
sudo pip install adafruit_python_dht
```

Nun wird das Skript für den MCP3008 geschrieben dazu wird zuerst folgender Befehl eingegeben:

Sudo Nano

Anschließend wird dieses Skript eingefügt in das aufkommende Fenster:

```

from spidev import SpiDev

class MCP3008:
    def __init__(self, bus = 0, device = 0):
        self.bus, self.device = bus, device
        self.spi = SpiDev()
        self.open()

    def open(self):
        self.spi.open(self.bus, self.device)

    def read(self, channel = 0):
        adc = self.spi.xfer2([1, (8 + channel) << 4, 0])
        data = ((adc[1] & 3) << 8) + adc[2]
        return data

    def close(self):
        self.spi.close()

def checkLight():
    timestamp = readTime()

    if SETTINGS["LIGHT_FROM"] <= timestamp.hour <= SETTINGS["LIGHT_UNTIL"]:
        # check light sensors
        adc = MCP3008()
        # read 10 times to avoid measuring errors
        value = 0
        for i in range(10):
            value += adc.read( channel = SETTINGS["LIGHT_CHANNEL"] )
        value /= 10.0

        if value <= SETTINGS["LIGHT_THRESHOLD"]:
            # turn light on
            GPIO.setup(SETTINGS["LIGHT_GPIO"], GPIO.OUT, initial=GPIO.LOW) # Relay
LOW = ON
        else:
            # turn light off
            GPIO.setup(SETTINGS["LIGHT_GPIO"], GPIO.OUT, initial=GPIO.HIGH)
        else:
            # turn light off
            GPIO.setup(SETTINGS["LIGHT_GPIO"], GPIO.OUT, initial=GPIO.HIGH)

```

Abschließend wird das Skript unter dem Namen MCP3008.py (Abb.25) gespeichert. Die Speicherung erfolgt durch Betätigen von Strg + O. Es wird gefragt, ob die alte Klasse ersetzt werden soll, dies wird mit Eingabe von J bestätigt. Das Skript ist länger als bei Schritt 3, dies hat den Hintergrund, dass hierbei das Skript erweitert, wurde um die Lampe bei einem Helligkeitswert niedriger als 10.0 eingeschaltet wird. Der Prozess wird durch Strg+X beendet.

```

hhb040@raspberrypi:~
Datei Bearbeiten Reiter Hilfe
GNU nano 5.4 MCP3008.py
From spidev import SpiDev

class MCP3008:
    def __init__(self, bus = 0, device = 0):
        self.bus, self.device = bus, device
        self.spi = SpiDev()
        self.open()

    def open(self):
        self.spi.open(self.bus, self.device)

    def read(self, channel = 0):
        adc = self.spi.xfer2([1, (8 + channel) << 4, 0])
        data = ((adc[1] & 3) << 8) + adc[2]
        return data

    def close(self):
        self.spi.close()

def checkLight():

```

41 Zeilen geschrieben

^G Hilfe ^O Speichern ^W Wo ist ^K Ausschneiden ^T Ausführen ^C Position
 ^X Beenden ^R Datei öffnen ^E Ersetzen ^U Einfügen ^J Ausrichten ^_ Zu Zeile

(Abb. 25)

Schritt 4.2

Damit die Belichtungsanlage funktioniert, wird ein weiteres Skript erstellt, dazu wird in das Terminal eingegeben:

Sudo Nano

Anschließend wird folgendes Skript hineinkopiert:

```

Import RPi.GPIO as GPIO
import Adafruit_DHT
from MCP3008 import MCP3008
import time

```

```

SETTINGS = {
    "LIGHT_GPIO": 17, # GPIO Number (BCM) for the Relay
    "LIGHT_FROM": 10, # from which time the light can be turned on (hour)
    "LIGHT_UNTIL": 20, # until which time (hour)
    "LIGHT_CHANNEL": 0, # of MCP3008
    "LIGHT_THRESHOLD": 500, # if the analog Threshold is below any of those,
the light will turn on
    "DHT_GPIO": 27, # GPIO Number (BCM) of the DHT Sensor
    "DHT_SENSOR": Adafruit_DHT.DHT22, # DHT11 or DHT22
    "TEMP_THRESHOLD": 23.0, # in Celcius. Above this value, the window will be
opened by the servo
    "SERVO_GPIO": 22, # GPIO Number (BCM), which opens the window
    "SERVO_OPEN_ANGLE": 90.0, # degree, how much the servo will open the
window
    "PLANTS": [
        {

```

```

    "NAME":          "Tomaten",
    "MOISTURE_CHANNELS": [1, 2], # of MCP3008
    "MOISTURE_THRESHOLD": 450, # if the average analog value of all sensors is
above of this threshold, the Pump will turn on
    "WATER_PUMP_GPIO": 23, # GPIO Number (BCM) for the Relais
    "WATERING_TIME": 10, # Seconds, how long the pump should be turned on
    },
    {
    "NAME":          "Salat",
    "MOISTURE_CHANNELS": [3, 4],
    "MOISTURE_THRESHOLD": 450,
    "WATER_PUMP_GPIO": 24,
    "WATERING_TIME": 12,
    },
    ]
}

```

```

def readTime():
    try:
        ds1307 = SDL_DS1307.SDL_DS1307(1, 0x68)
        return ds1307.read_datetime()
    except:
        # alternative: return the system-time:
        return datetime.datetime.utcnow()

```

```

def checkLight():
    timestamp = readTime()

    if SETTINGS["LIGHT_FROM"] <= timestamp.hour <= SETTINGS["LIGHT_UNTIL"]:
        # check light sensors
        adc = MCP3008()
        # read 10 times to avoid measuring errors
        value = 0
        for i in range(10):
            value += adc.read( channel = SETTINGS["LIGHT_CHANNEL"] )
        value /= 10.0

        if value <= SETTINGS["LIGHT_THRESHOLD"]:
            # turn light on
            GPIO.setup(SETTINGS["LIGHT_GPIO"], GPIO.OUT, initial=GPIO.LOW) # Relay
LOW = ON
        else:
            # turn light off
            GPIO.setup(SETTINGS["LIGHT_GPIO"], GPIO.OUT, initial=GPIO.HIGH)
        else:
            # turn light off
            GPIO.setup(SETTINGS["LIGHT_GPIO"], GPIO.OUT, initial=GPIO.HIGH)

```

```

if __name__ == '__main__':
    try:
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)

```

```
# execute functions
checkLight()
except:
    GPIO.cleanup()
```

Das Skript wird unter dem Namen Greenhouse.py gespeichert durch Strg+O und dann mit Strg+X kehren wir ins Terminal zurück. Jetzt wird getestet, ob das Programm funktioniert, dazu wird folgender Befehl ins Terminal eingegeben. Es sollten dunkle Lichtverhältnisse vorhanden sein oder es wird ein Finger auf den Helligkeitssensor gelegt.

```
Sudo Greenhouse.py
```

Leuchtet die LED-Leiste wie bei Abbildung 26 wurde alles richtig gemacht.



(Abb. 26)

Schritt 4.3

Die Skripte sind so weit fertig, damit der Prozess aber automatisiert und in Betrieb genommen werden kann, wird Folgendes gemacht. Es wird das Skript vom Greenhouse.py geöffnet und durch folgende Erweiterung ergänzt. Dazu wird zuerst folgender Befehl eingegeben:

```
Sudo nano greenhouse.py
```

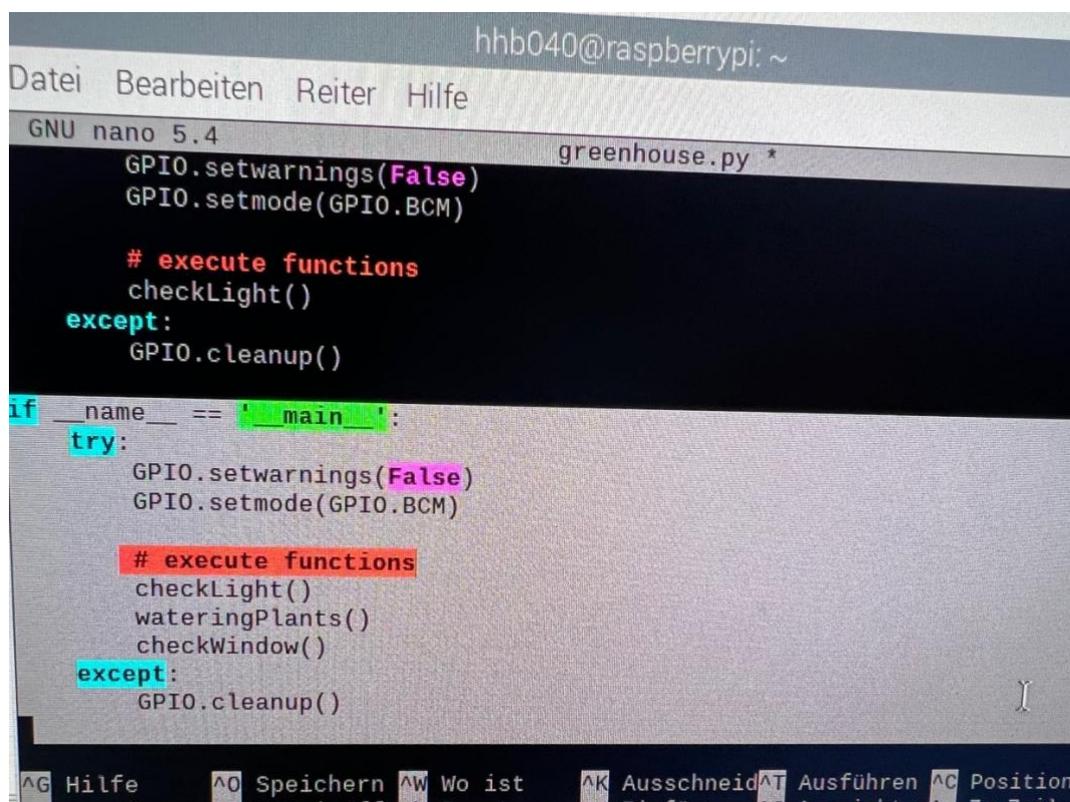
Jetzt ist das Skriptfenster geöffnet, es wird zum Ende gescrollt und folgendes Skript wird kopiert und hinzugefügt:

```

if __name__ == '__main__':
    try:
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)

        # execute functions
        checkLight()
        wateringPlants()
        checkWindow()
    except:
        GPIO.cleanup()

```



(Abb. 27)

Nun wird das Skript gespeichert, es sollte ein Leerzeichen wie auf Abb.27 zwischen dem Ende des ursprünglichen Skriptes vorhanden sein. Gespeichert wird mit Strg+O und mit Strg+X kehrt ihr ins Terminal zurück.

Damit das Skript alle 10 abgerufen wird folgendes ins Terminal eingegeben:

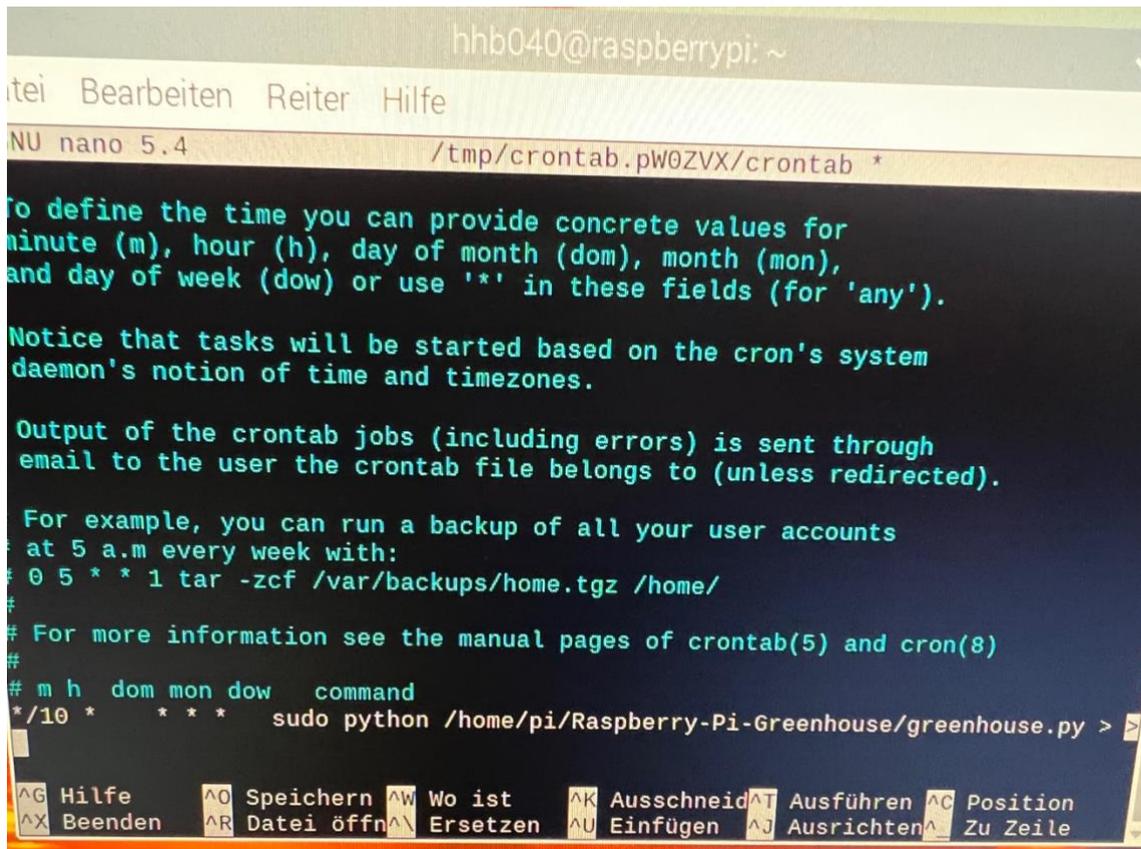
Crontab -e

Anschließend wird 1 eingegeben und Crontab wird installiert, danach wird wieder folgender Befehl eingegeben:

Crontab-e

Das sich nun öffnende Skript wird am Ende um folgendes Skript erweitert wie auf Abb 28:

```
*/10 * * * * sudo python /home/pi/Raspberry-Pi-Greenhouse/greenhouse.py > /dev/null 2>&1
```



```
hhb040@raspberrypi: ~
Datei Bearbeiten Reiter Hilfe
GNU nano 5.4 /tmp/crontab.pw0ZVX/crontab *
To define the time you can provide concrete values for
minute (m), hour (h), day of month (dom), month (mon),
and day of week (dow) or use '*' in these fields (for 'any').
Notice that tasks will be started based on the cron's system
daemon's notion of time and timezones.
Output of the crontab jobs (including errors) is sent through
email to the user the crontab file belongs to (unless redirected).
For example, you can run a backup of all your user accounts
at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
*/10 * * * * sudo python /home/pi/Raspberry-Pi-Greenhouse/greenhouse.py >
^G Hilfe      ^O Speichern  ^W Wo ist     ^K Ausschneid^T Ausführen  ^C Position
^X Beenden    ^R Datei öffn^N Ersetzen   ^U Einfügen  ^J Ausrichten^_ Zu Zeile
```

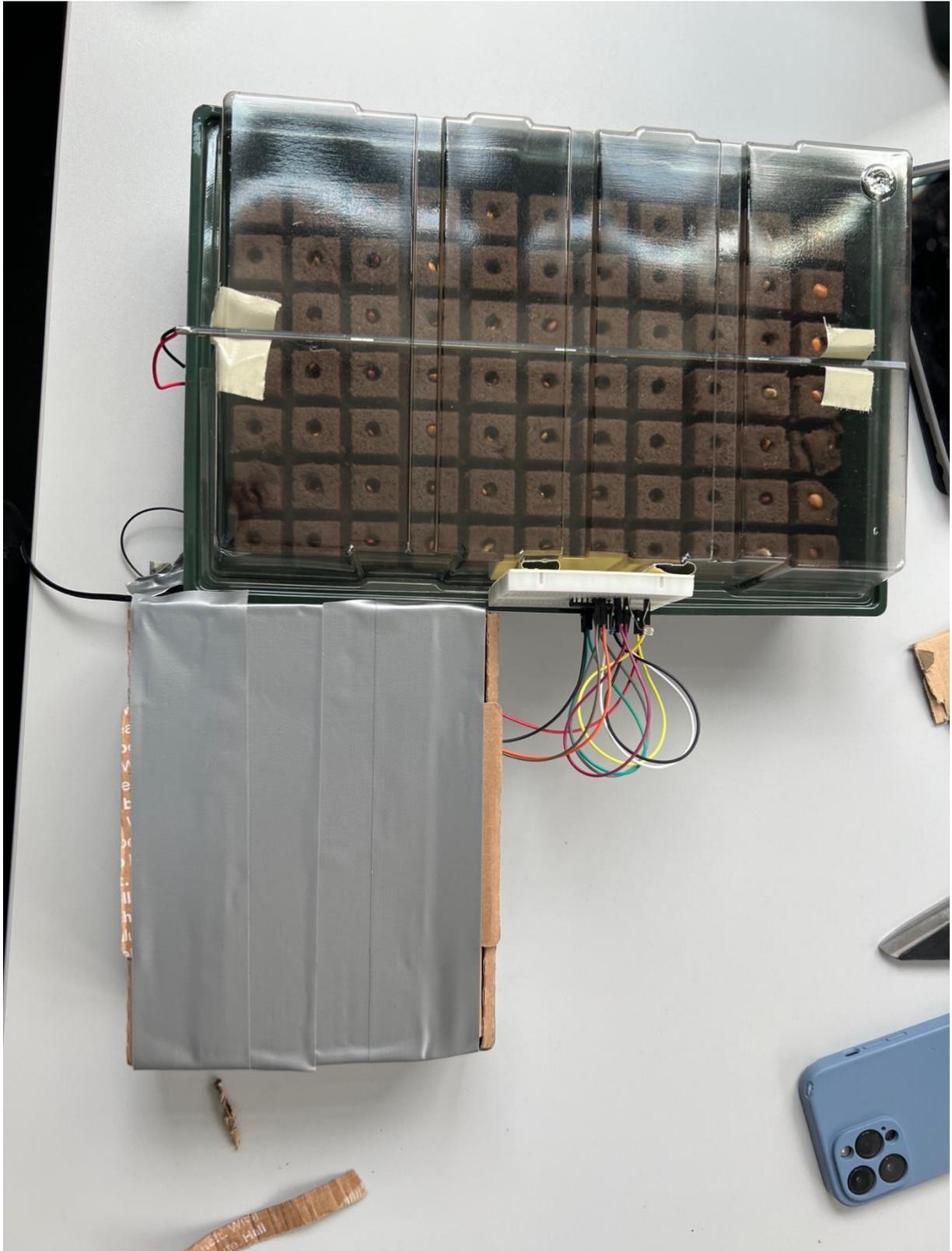
(Abb.28)

Damit wurde die Belichtungsanlage erfolgreich programmiert. Wichtig ist das, dass Terminal offenbleibt, damit das Skript alle 10 Minuten abgerufen wird.

Schritt 5

- Raspberry Pi mit allen Hardwarekomponenten
- Mini-Gewächshaus
- Pappkarton oder Kasten 17cm x 7cm x 20 cm
- Panzertape / Isolierband
- Pflanzensamen (Balkontomaten, Radieschen, Pflücksalat)
- Düngermittel

Nun folgt der letzte Schritt, hierbei kann zuerst das W-LAN und Bluetooth am Raspberry Pi ausgeschaltet werden. Die Maus, Tastatur und Monitor werden vom Raspberry Pi getrennt. Wichtig ist, dass der Raspberry und die anderen Verbindungen wie die Jumper, Kabel, Spannungsregler etc. bestehen bleiben. Jetzt kann mit dem Aufbau des Mini-Gewächshauses begonnen werden. Du kannst damit beginnen, den Pappkarton zu isolieren, mit dem Panzertape. Siehe dabei auf Abbildung 29. Anschließend werden die weiteren Komponenten in den Karton gelegt. Damit die LED-Leiste angebracht werden kann, muss ein Stück von der Seite des Deckels abgeschnitten werden. (Abb. 31) Anschließend wird die Leiste an die Decke des Minigewächshauses mittig angebracht, mit Panzertape. Das Breadboard wird außen angebracht, damit der Helligkeitssensor nicht bedeckt wird, wie auf Abb. 30. Zum Schluss werden die Samen in die Löcher eingegeben, Pflanzendünger dazugegeben sowie Wasser gegossen. Als Position eignet sich am besten eine Fensterbank, da dort genug Licht am Tag auf die Samen scheint.



(Abb. 29)



Abb 30

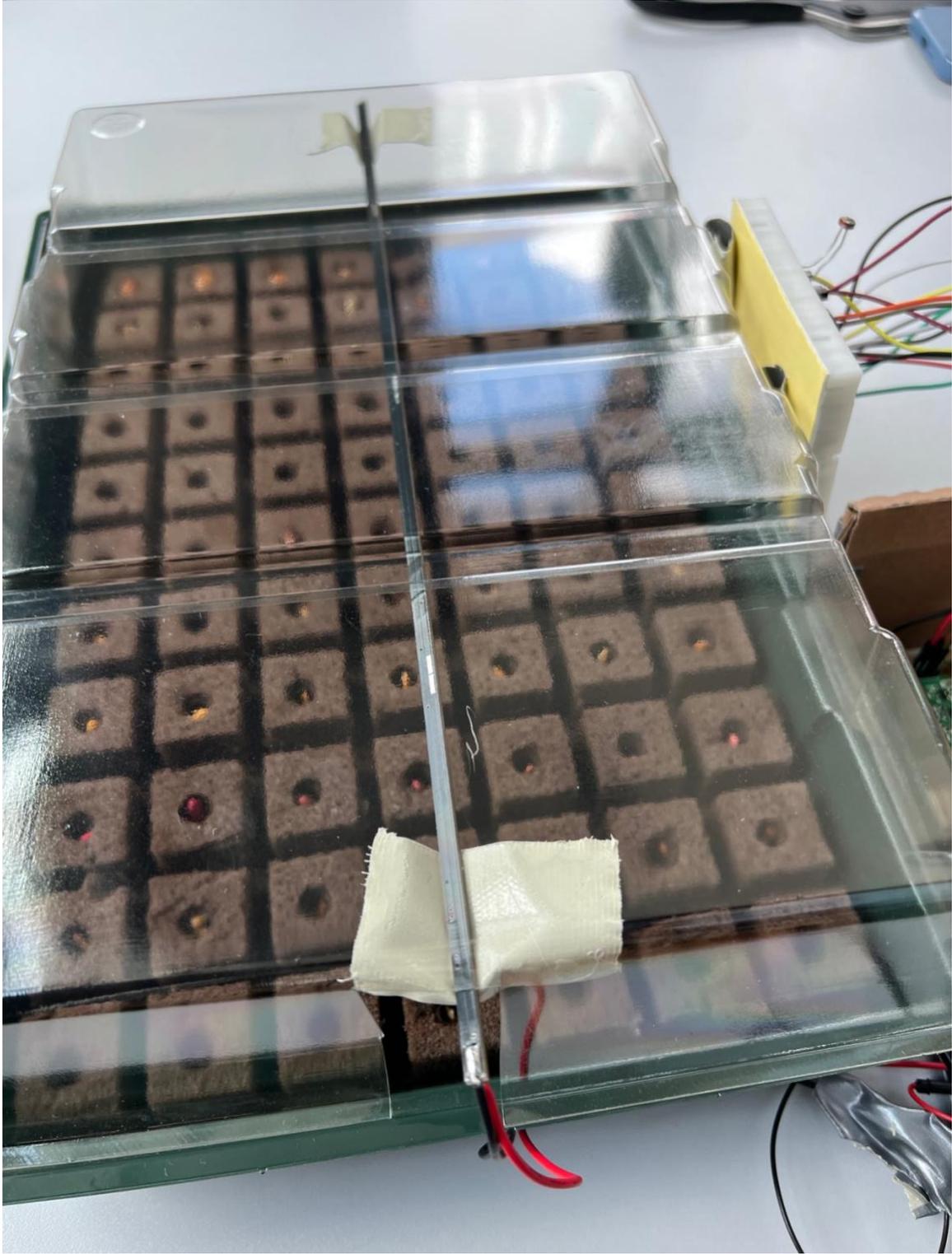


Abb. 31

Literaturverzeichnis und Abbildungsverzeichnis

- <https://www.raspberrypi.com/software/>
- Abb. 13 (<https://tutorials-raspberrypi.de/wp-content/uploads/2017/01/Raspberry-Pi-Helligkeitssensor-Fototransistor-Steckplatine.png>)
- Abb. 22 (<https://tutorials-raspberrypi.de/wp-content/uploads/2017/04/Raspberry-Pi-Lightsensor-MCP3008-Schema.png>)
- Abb. 21 https://www.amazon.de/Batterietester-Spannungsprüfer-Durchgangsprüfer-LCD-Anzeige-Hintergrundlicht/dp/B09DKFTR7M/ref=sr_1_6?__mk_de_DE=ÅMÅŽŦÑ&crid=33AYSGLSJCU0U&keywords=spannungsmesser&qid=1693492966&prefix=spannungsme sse%2Caps%2C199&sr=8-6