ESP32 und MicroPython

Bearbeiten Sie für den Einstieg in MicroPython und ESP32 folgende Posten.

Posten 1 - Hello World!

- Wenn nicht schon geschehen, installieren Sie auf ihrem Notebook Thonny.
- Verbinden Sie den ESP32 mit USB-C-Kabel mit ihrem Notebook. Die weisse LED auf dem Mikrocontroller sollte nun dauerhaft leuchten.
- Öffnen Sie Thonny und wählen Sie unter `Extras → Optionen… → Interpreter` den Interpreter `MicroPython (ESP32)` aus.
- Klicken Sie auf `Stop`, nun sollte ihm Shell-Fenster von Thonny `MicroPython v1.21.0…` stehen und >>> erscheinen. Diese Zeichen bedeuten, dass MicroPython-Befehle eingegeben werden können, welche anschliessend an den ESP32 gesendet und auf diesem ausgeführt werden.
- Geben Sie zum Testen die Rechnung 34**4 ein, der ESP32 berechnet nun 34^4. (Befehle müssen mit Enter bestätigt/ausgeführt werden.)
- Probieren Sie noch andere Rechnungen aus.
- Was passiert, wenn Sie 12345^{98765} ausrechnen möchten?
- Sie können auch Text vom ESP32 ausgeben lassen: z.B. mit `print('Hello World!')`.
- Schreiben Sie eine `for`-Schleife, die alle Dreieckszahlen bis n = 15 zusammenzählt.

Posten 2 - LED

- Bauen Sie folgenden Schaltkreis nach.

- Um die LED zu verwenden müssen Sie zuerst die nötigen Module mit dem Befehl `from machine import Pin` laden.
- Die LED ist am Pin 3 angeschlossen. Ordnen Sie die LED diesem Pin in MicroPython zu: `led = Pin(3, Pin.OUT)`
- Die LED können Sie nun mit dem Befehl `led(1)` ein- bzw. mit `led(0)` ausschalten.
- Mit einer for-Schleife kann die LED zum Blinken gebracht werden. Dazu wird noch das time Modul benötigt:

```
import time
led = Pin(2, Pin.OUT)
for i in range(10):
    led(1)
    time.sleep(0.5)
    led(0)
    time.sleep(0.5)
```

Posten 3 - Druckknopf

Es kann auch ein Druckknopf verwendet werden, um die LED zu steuern. Ergänzen Sie dazu den Schaltkreis der LED um den Druckknopf. Stecken Sie dazu die LED an Pin 3 (und nicht mehr Pin 2). Stecken Sie den Druckknopf wie folgt auf das Breadboard und verbinden Sie das untere/rote Kabel mit GND und das obere/grüne Kabel mit Pin 2.

```
from machine import Pin
led = Pin(3, Pin.OUT)
button = Pin(2, Pin.IN, Pin.PULL_UP)
while True:
    if not button():
        led(not led())
        while not button():
            pass
```

Hinweis: Um eine while-Schleife (oder auch andere Befehle) zu unterbrechen, drücken Sie die Tastenkombination CTRL+C.

Posten 4 - Ampel

Bauen Sie folgende Ampelschaltung nach.

(schwarzes Kabel → GND, grünes Kabel → Pin 6, gelbes Kabel → Pin 4, rotes Kabel → Pin 3, violettes Kabel → Pin 2)

Programmieren Sie die Schaltung so, dass die Ampel von rot über orange auf grün schaltet, sobald der Knopf gedrückt wird. Die Ampel soll dann für 2 Sekunden grün sein und danach automatisch wieder auf Rot schalten.

Code-Hilfe:

```
from machine import Pin
import time
led1 = Pin(3, Pin.OUT)
led2 = Pin(4, Pin.OUT)
led3 = Pin(6, Pin.OUT)
button = Pin(2, Pin.IN, Pin.PULL_UP)
led3(1)
while True:
    if not button.value():
        led3(1)
        time.sleep(0.1)
        ...
```

Posten 5 - NeoPixel

Auf dem Mikrocontroller ist auch eine dreifarbige LED (NeoPixel) verbaut. Diese kann über den Pin 7 angesteuert werden (keine Verkabelung nötig).

Laden Sie die NeoPixel-LED mit

```
import machine, neopixel
np = neopixel.NeoPixel(machine.Pin(7),1)
```

Die Farbe kann mit RGB-Farbcode gesetzt werden. Bevor die NeoPixel leuchtet muss der Farbcode gesetzt werden: z.B. rot mit `np[0]=(255,0,0)`. Danach kann mit `np.write()` die LED eingeschaltet werden.

Schreiben Sie einen Code für einen Farbverlauf der NeoPixel-LED. Die RGB-Farbcodes finden Sie z.B. hier.

Posten * - Pulsweitenmodulation (PWM)

Mit dem ESP32 lässt sich auch ein Dimmungseffekt hinzufügen, die LED wird also heller und wieder dunkler. Ermöglicht wird dies durch die Funktion Pulse Width Modulation (PWM, auf Deutsch: Pulsweitenmodulation).

- Recherchieren Sie im Internet, wie die Pulsweitenmodulation funktioniert und schauen Sie folgendes Youtube-Video dazu: PWM.
- Mit dem Befehl `led(1)` bzw. `led.on()` sendet der ESP32 ein auerhaftes Stromsignal an den Pin 2 und somit an die LED. Die Spannung hat einen Wert von ca. 3.3 V. Im U -t-Diagramm sieht dies wie folgt aus:

- Skizzieren Sie ein Signal, welches mit per Pulsweitenmodulation erstellt wurde. Die Periodendauer sei 3 s und der Tastgrad betrage 33%. Die Spannung erreicht dabei einen maximalen Wert von $U_{max} = 3$ V.
- In MicroPython lässt sich die Pulsweitenmodulation mit der Funktion PWM vom `machine`-Modul bewerkstelligen. Definieren Sie dazu die LED mit folgendem Befehl neu: `led = machine.PWM(machine.Pin(2), freq=1000)`. Dabei gibt der Parameter `freq` die Frequenz für die Pulse an.

Mit `led.duty(x)` kann nun der Tastgrad eingestellt werden. Der Wert x kann dabei von 0 bis 1023 variiert werden. Bei $x = 1023$ ist der Tastgrad bei 100%, bei $x = 0$ entsprechend bei 0%. Testen Sie einige Werte für x. Was passiert mit der LED?

- Verwenden Sie nun folgenden Code, um die LED während dem Blinken zu dimmen.

```
while True:
    for tast in range(0,1024):
        led.duty(tast)
```

```
        time.sleep(0.005)
```

Hinweis: Um die while-Schleife zu unterbrechen, drücken Sie Ctrl+C.

Geben Sie folgenden Code ein:

```
import math
for i in range(50):
    led.duty(int(math.sin(i / 10 * math.pi) * 500 + 500))
    time.sleep_ms(50)
```

- Wie ändert sich das Verhalten gegenüber dem Code, welchen Sie im letzten Schritt verwendet haben?
- Was passiert wenn Sie die Parameter von `range` und `time.sleep_ms` verändern?
- Definieren Sie eine Funktion `pulse(x,t)` bei welcher x die gewünschte LED ist und t die Periodendauer des Blinkens.

Posten 6 - Lichtsensor

Ein IoT-Gerät besteht nicht nur aus dem Mikrocontroller sondern besitzt auch mindestens einen Sensor. An diesem Posten möchten wir ein Beispiel für einen solchen Sensor betrachten: einen Lichtsensor. Als Lichtsensor verwenden wir einen Fotowiderstand, welchen wir mit dem ESP32 verbinden und ansteuern.

Verbinden Sie die elektronischen Komponenten nach folgendem Bauplan mit dem ESP32:

(rotes Kabel → 3V3-Pin, grünes Kabel → Pin 0, schwarzes Kabel → GND-Pin)

Der Fotowiderstand funktioniert als Lichtsensor, denn seine Grösse ändert sich in Abhängigkeit der Lichtstärke als Messgrösse. Fällt viel Licht auf den Fotowiderstand, so ist seine Grösse gering. Sind die Lichtverhältnisse hingegen schwach, so ist der Fotowiderstand gross. Im Falle des verwendeten Fotowiderstandes ist die Abhängigkeit nicht linear. Der Mikrocontroller kann nicht direkt die Grösse des Fotowiderstandes messen. Er besitzt jedoch sogenannte Pins, die als Analog-Digital-Wandler (ADC) verwendet werden können. Der Pin mit der Beschriftung 'Pin 0' ist ein solcher ADC-Pin. Er konvertiert ein analoges Signal in eine Digitalzahl. Im Falle des Fotowiderstands kann mit dem Pin 0 eine Spannung gemessen werden, diese steht laut dem Ohmschen Gesetz im Zusammenhang mit dem Widerstand. Somit kann über die Spannung indirekt die Lichtstärke gemessen werden.

Um die Spannung über dem Fotowiderstand zu messen, wird das Prinzip des Spannungsteilers verwendet. Schauen Sie sich die Funktionsweise und den Aufbau eines Spannungsteilers (z.B. auf https://www.leifiphysik.de/elektrizitaetslehre/komplexere- schaltkreise/versuche/unbelasteter-spannungsteiler) an.

Wir verwenden den Pin mit der Nummer 0 als ADC-Pin. An diesem Pin können Spannungen zwischen 0 und 3.3 V gemessen werden. Dabei wir der Spannungswert linear in eine Zahl zwischen 0 und 4095 übersetzt. Die Auflösung des ADC-Pins ist also 12-bit (da $2^{12}=4096$). Wir benötigen die Funktionen ADC und Pin des machine-Moduls. Laden Sie diese Funktionen mit `from machine import ADC, Pin`.

- Initialisieren Sie anschliessend den Fotowiderstand (wird oft mit LDR für light dependent resistor

abgekürzt) mit folgendem Befehl:

```
ldr = ADC(Pin(0))
```

- Nun können Sie mit `ldr.read()` die Zahl auslesen, welche der ADC-Pin für die gerade angelegte Spannung ausgibt.
- Verwenden Sie nun folgende while-Schleife, um den ADC-Pin alle 200 ms auszulesen:

```
import time
while True:
    a=ldr.read()
    print(a)
    time.sleep(0.2)
```

- Was stellen Sie fest, wenn Sie den Fotowiderstand abdunkeln?
- Der ADC-Pin gibt für grössere Lichtstärke auch eine grössere Zahl aus als für wenig Lichteinfall. Der Fotowiderstand sollte jedoch genau ein gegenteiliges Verhalten zeigen. Weshalb ist die Ausgabe also anders

als erwartet? Beim verwendeten Spannungsteiler wird mit dem ADC-Pin nicht die Spannung über dem Fotowiderstand sondern über dem 10 kΩ-Widerstand gemessen. Trifft viel Licht auf den Fotowiderstand, so ist sein Widerstandswert gering, das bedeutet auch, dass wenig Spannung über ihm abfällt. Somit ist aber auch der Spannungsabfall über dem 10 kΩ-Widerstand grösser, denn die Spannung über beiden Widerständen ist konstant bei 3.3 V. Der ADC-Pin zeigt also eine grosse Zahl (nahe 4095) an. Bei schwachen Lichtverhältnissen ist der Spannungsabfall über dem Fotowiderstand gross und dementsprechend misst der ADC-Pin eine kleine Spannung über dem 10 kΩ-Widerstand. Der ADC-Pin gibt dann eine kleine Zahl (nahe 0) aus.

- Recherchieren Sie, welches Material für Fotowiderstände verwendet wird. Welche speziellen Eigenschaften besitzt dieses?
- Verwenden Sie folgenden Code:

```
led = Pin(2,Pin.OUT)
while True:
    a=ldr.read()
    if a<4000:
        led(1)
    else:
        led(0)
```

- Was bewirkt die while-Schleife?

Posten 7 - OLED-Display

Versuchen Sie mithilfe von ChatGPT den OLED-Display mit dem ESP32 zum Laufen zu bringen. Verwenden Sie dazu folgende Verkabelung (OLED-Display → ESP32):

- GND → GND
- VDD → 3V3
- SCK → Pin 1

- SDA → Pin 2

Sie werden einen Treiber für den Display benötigen, diesen können Sie hier herunterladen: SSD1306-Treiber Den Treiber müssen Sie via Thonny auf den ESP32 laden.

Beispiel ChatGPT-Prompts:

- Ich habe einen Treiber für den SSD1306 als ssd1306.py heruntergeladen. Ich möchte diesen Treiber nun mit Thonny auf den ESP32 mit MicroPython installieren. Beschreibe mir genau und Schritt für Schritt, was ich machen muss.
- Schreibe mir einen Beispiel-Code für MicroPython, der mir 'Hallo' auf dem OLED-Display SSD1306 anzeigt.
- Bei mir funktioniert es nicht mit machine.I2C was muss ich machen?

Posten 8 - BME280

Versuchen Sie den Temperatur- und Druck-Sensor mit dem ESP32 zu verbinden und Messwerte auszulesen. Recherchieren Sie dazu im Internet oder verwenden Sie ChatGPT.