HSLU Hochschule Luzern

PH LUZERN
PÄDAGOGISCHE
HOCHSCHULE

Die Programmierumgebung - Emulator

Um das Projekt umsetzen zu können, müssen wir uns zuerst mit der Programmierumgebung vertraut machen.

Ziel:
Du lernst den Emulator, der die Funktionen des Astro Pi simuliert, zu nutzen.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 1 Pakete importieren
- 2 Farbenspie
- 3 Text ausgeben
- 4 Das Display verstehen
- 5 Bilder zeichnen
- 6 Den Emulator starten
- 7 Die Sensoren kennen lernen
- 8 Den Code ergänzen

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

1 Pakete importieren

Bevor du mit deiner eigentlichen Forschungsarbeit beginnen kannst, müssen ein paar Grundlagen geschaffen werden. Hier packst du sinnbildlich die benötigten Forschungsutensilien ein.

Aufgabe:

Starte den Emluator unter https://www.esa.int/Education/Astro... T_emulator.
Schaue dir die ersten drei Zeilen Code genau an und überlege dir, was diese drei Zeilen machen.

Tipps zum Vorgehen:

Überlege dir, hast du gewisse Code-Teile auf den bisherigen Aufgabenkarten bereits einmal gesehen und was bewirken diese.

1 Pakete importieren

Lösungsvorschlag

```
1  from sense_hat import SenseHat
2  sense = SenseHat()
3  sense.set_rotation(270)
```

1: Mit diesem Befehl wird das Modul SenseHat aus dem Paket sense_hat importiert. Damit stehen diese Module zum Programmieren zur Verfügung. Diese Module werden dazu benötigt, um die Sensoren und Licht-Matrix ansteuern zu können.

2: Der Variablen "sense" werden die Module SenseHat zugewiesen. Dadurch ist es später möglich, diese Module über einen kürzeren Befehl aufzurufen.

3: Der Anzeigebildschirm wird gedreht um 270 Grad im Uhrzeigersinn (am Bildschirm ist der Parameter 0, der Text erscheint horizontal und kann von links nach rechts gelesen werden).

2 Farbenspiel

Das Display des Astro Pi kann die verschiedensten Farben annehmen.

Aufgabe:

Schaue dir die Zeilen 5 bis 8 genau an und überlege dir, was diese Zeilen bewirken.

Tipps zum Vorgehen:

RGB-Farben setzen sich aus drei Grundfarben zusammen, auf die die lichtempfindlichen Zapfen im menschlichen Auge am empfindlichsten reagieren: Rot, Grün und Blau. Die drei Grundfarben Rot, Grün und Blau können jeweils die Stärken von 0 bis 255 einnehmen, also gibt es insgesamt 256 Abstufungen je Kanal.

2 Farbenspiel

Lösungsvorschlag

```
5   g = (0, 255, 0)
6   r = (255, 0, 0)
7   w = (155,155,155)
8   e = (0,0,0)
```

5: der Variablen "g" wird die Farbe grün zugewiesen

6: der Variablen "r" wird die Farbe rot zugewiesen

7: der Variablen "w" wird die Farbe grau zugewiesen

8: der Variablen "e" wird die Farbe schwarz zugewiesen

3 Text ausgeben

Um zu sehen, ob deine Arbeit erfolgreich ist, bist du darauf angewiesen immer wieder Rückmeldung zu erhalten.

Aufgabe:

Schaue dir die Zeile 10 genau an. Was bewirkt die Zeile?

Tipps zum Vorgehen:

Schaue dir die Zeile 2 im Code nochmals an, diese Leistet die Vorarbeit für den Befehl, den du jetzt anschaust.

3 Text ausgeben

Lösungsvorschlag

```
10  sense.show_message("Hello my name is Paxi")
```

Die Zeile bewirkt, dass der Text "Hello my name is Paxi" ausgegeben wird.

Dabei wird die Funktion show_message aufgerufen, mithilfe der eine Nachricht ausgegeben werden kann.

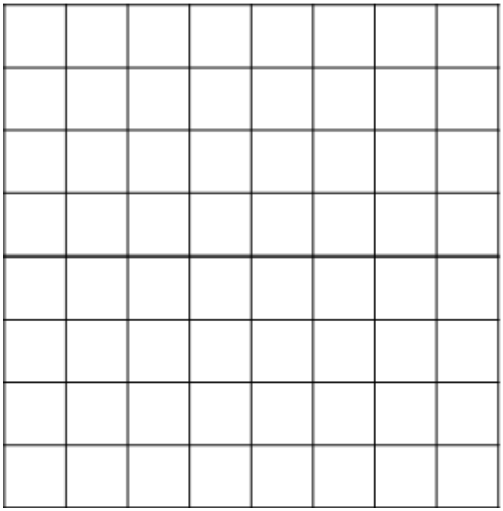
Die Funktion gehört zu einem Modul, dem Modul SenseHat(), diesem Modul haben wir in der Zeile 2 die Bezeichnung sense zugewiesen. Beim Aufrufen der Funktion muss dem Programm jeweils mitgegeben werden, aus welchem Modul der Befehl stammt, damit es diesen zum Ausführen findet. Dies ist der erste Teil des Befehls: sense. danach folgt der Aufruf der Funktion show_message, der wir als Parameter den Text "Hello my name is Paxi" mitgeben.

4 Das Display verstehen

Nicht alle Utensilien, die du zur Forschung benötigst sind auf den ersten Blick selbsterklärend.

Aufgabe:

Das Display des Astro Pi besteht aus einem Gitter von 8x8 Pixeln. In den Zeilen 12 bis 21 wird der Variabeln "paxi" ein Bild zugewiesen, dass in diesem Gitter gezeichnet wird.
Male die Pixel in untenstehendem Raster entsprechend aus.

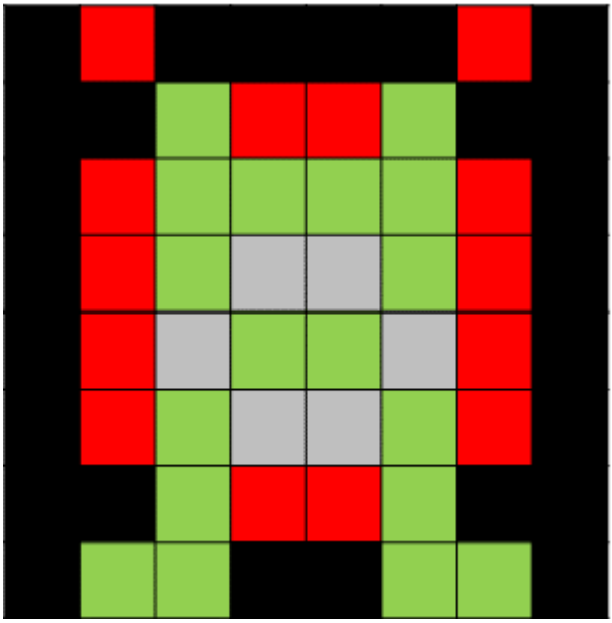


Tipps zum Vorgehen:

Welcher Pixel welche Farbe hat, hast du in den vorhergehenden Aufgaben betrachtet. Schaue dir diese nochmals an.

4 Das Display verstehen

Lösungsvorschlag



5 Bilder zeichnen

Manchmal sind es die kleinen Dinge, die entscheidend sind.

Aufgabe:

Schaue dir die Zeile 22 im Code genau an. Was bewirkt die Zeile?

Tipps zum Vorgehen:

Schaue dir die eben gelösten Aufgaben nochmals an. Dabei kannst du dir die Fragen stellen, was der Variablen "paxi" für ein Wert zugewiesen wird und wozu das set_pixels dienen kann.

5 Bilder zeichnen

Lösungsvorschlag

```
22  sense.set_pixels(paxi)
```

Das eben gezeichnete Bild wird ausgegeben. Dazu wird die Funktion set_pixels aufgerufen, der die Variable "paxi" mitgegeben wird. Der Variablen paxi wurde das Bild zugewiesen, dass du in der vorherigen Aufgabe gezeichnet hast.

Die Funktion gehört zu einem Modul, dem Modul SenseHat(), diesem Modul haben wir in der Zeile 2 die Bezeichnung sense zugewiesen. Beim Aufrufen der Funktion muss dem Programm jeweils mitgegeben werden, aus welchem Modul der Befehl stammt, damit es diesen zum Ausführen findet. Dies ist der erste Teil des Befehls: sense. danach folgt der Aufruf der Funktion set_pixels, der wir als Parameter die Variable "paxi" mitgeben.

6 Den Emulator starten

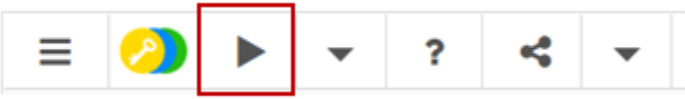
Du hast nun viel Vorarbeit geleistet und es ist an der Zeit loszulegen.

Aufgabe:

Lasse das Programm einmal laufen und schaue ob es das macht, was du vermutest dass es macht.

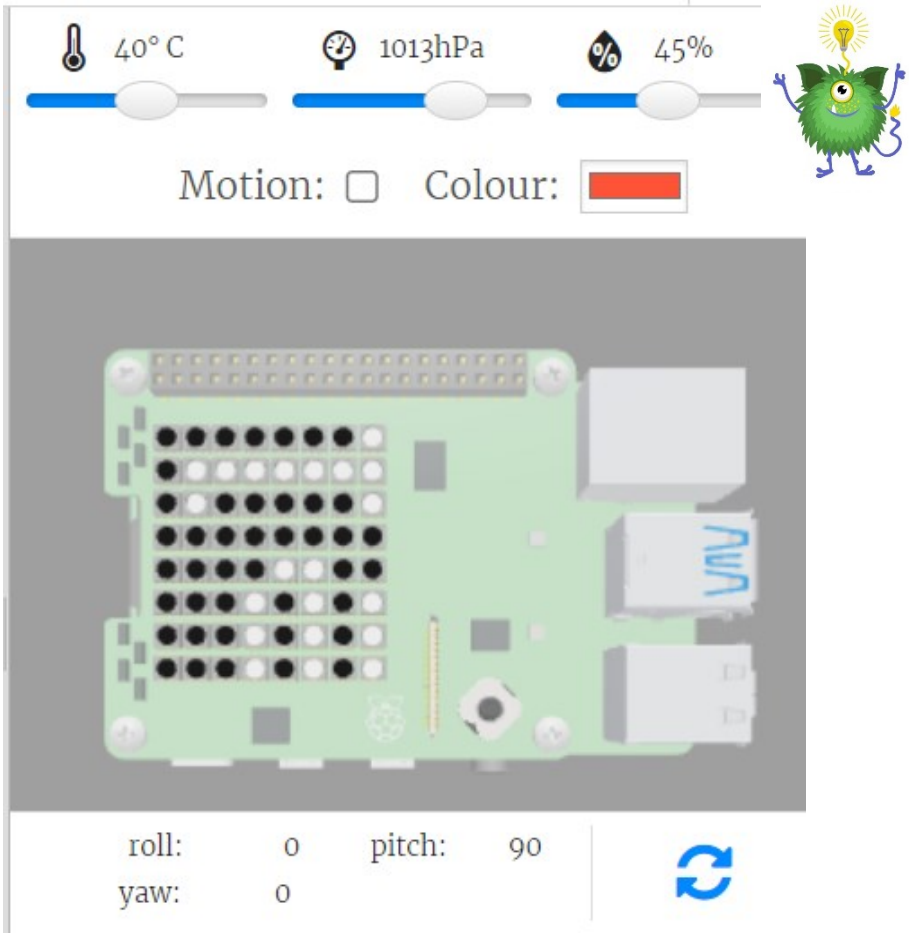
Tipps zum Vorgehen:

Du kannst das Programm mit Klick auf den Play-Pfeil starten.



6 Den Emulator starten

Lösungsvorschlag



Auf der rechten Seite wird dir gezeigt, wie das Programm auf dem Astro Pi ausgeführt wird.

7 Die Sensoren kennen lernen

Für deine Forschung musst du unterschiedliche Daten sammeln.

Aufgabe:

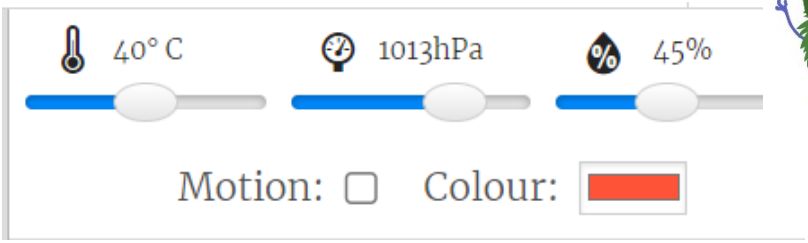
Schaue dir die Ausführung des Programmes genau an. Oberhalb des Astro Pi findest du ein paar angaben, die du mittels Reglern verstellen kannst. Was bedeuten diese Angaben?

Tipps zum Vorgehen:

Überlege dir, welche Angaben du aus deinem Alltag kennst. Kommen dir gewisse Zeichen vertraut vor?

7 Die Sensoren kennen lernen

Lösungsvorschlag



1 ° C = Temperatur

hPa = Luftdruck

% = Luftfeuchtigkeit

Motion = Bewegung des Astro Pi

Colour = Farbsensor, bestimmt die Farbe eines Zielobjektes

8 Den Code ergänzen

Du bist mit den Forschungsutensilien jetzt gut vertraut und traust dich etwas auszuprobieren.

Aufgabe:

Ergänze den bestehenden Code so, dass nach dem Text "Hello my name is Paxi", der Text "and i love the universe" ausgegeben wird.

Tipps zum Vorgehen:

Schaue dir die Zeile 10 nochmals an, wie wird hier der Text ausgegeben?

8 Den Code ergänzen

Lösungsvorschlag

```
10  sense.show_message("Hello my name is Paxi")
11  sense.show_message("and i love the universe")
```

Unterhalb der bestehenden Ausgabe in Zeile 10, kannst du eine weitere Zeile hinzufügen in welcher du den Text "and i love the universe" ausgibst.

Dafür musst du erst das Modul SenseHat() mittels sense. aufrufen und danach die Funktion show_Message. Dieser gibst du als Parameter den gewünschten Text mit.

Aufgabenkarte 1: Die Temperatur messen

Die Astronauten sind auf ein angenehmes Raumklima angewiesen, um gut arbeiten zu können. Mittels Temperaturmessung kannst du prüfen, ob die Astronauten ein angenehmes Raumklima haben.

Ziel:
Du lernst den Temperatursensor des Astro Pi kennen und wie du Temperaturdaten auslesen kannst.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 1.1 Temperatur messen
- 1.2 Temperatur in Variable schreiben
- 1.3 Temperatur runden

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

1.1 Temperatur messen

Die Werkzeuge sind nun einsatzfähig. Du probierst gleich das erste Werkzeug aus.

Aufgabe:

Miss die aktuelle Temperatur und gib diese via print-Befehl aus.

Tipps zum Vorgehen:

Über die Funktion get_temperature() kann die aktuelle Temperatur ausgegeben werden. Vergiss dabei nicht, den Namen des Modules SenseHat voranzustellen und diesen Mittels Punkt abzutrennen:

nameModul.get_temperatur()

Wenn du dir nicht mehr sicher bist, wie der print-Befehl funktioniert, schaue dir die blauen Aufgabenkarten nochmals an.

1.1 Temperatur messen

Lösungsvorschlag

```
1  from sense_hat import SenseHat
2  import time
3  sense = SenseHat()
4  print(sense.get_temperature())
```

Die ersten drei Zeilen sind die notwendige Vorarbeit, damit der Code in Zeile 4 funktioniert. Darin werden die benötigten Module importiert sowie der Variablen sense die Module SenseHat() zugewiesen (vgl. auch vorherige Aufgaben).

Die Zeile 4 ist das eigentliche Lösen der Aufgabe: Was in der Klammer des print-Befehles steht, wird ausgegeben.

Über sense. Wird dem Programm mitgeteilt, in welchem Modul benötigte Funktion vorhanden ist. Nach dem Punkt wird die entsprechende Funktion get_temperature() aufgerufen. Es benötigt nach dem Funktionsaufruf get_temperature eine öffnende und eine schliessende Klammer ().

1.2 Temperatur in Variable schreiben

Damit du die Temperatur nicht nur einmalig anzeigen, sondern auch damit arbeiten kannst, speicherst du den Wert in einer Variablen ab.

Aufgabe:

Schreibe die Temperatur in eine Variable temp. Danach gibst du die Variable temp mittels print-Befehl aus.

Tipps zum Vorgehen:

Der Ablauf des gesamten Programmes ist wie folgt:

1. Importieren der benötigten Pakete und Module
2. Zuweisen des Modules SenseHat an die Variable sense
3. Temperatur in der Variablen temp speichern
4. Variable temp mittels print-Befehl ausgeben

1.2 Temperatur in Variable schreiben

Lösungsvorschlag

```
1  from sense_hat import SenseHat
2  import time
3  sense = SenseHat()
4  temp = sense.get_temperature()
5  print(temp)
```

Die ersten drei Zeilen sind die notwendige Vorarbeit, damit der Code in Zeile 4 funktioniert. Darin werden die benötigten Module importiert sowie der Variablen sense die Module SenseHat() zugewiesen (vgl. auch vorherige Aufgaben).

Zeile 4 und 5 sind das eigentliche Lösen der Aufgabe. Die Temperatur wird in die Variable temp geschrieben, über sense. Wird dem Programm mitgeteilt, in welchem Modul benötigte Funktion vorhanden ist. Nach dem Punkt wird die entsprechende Funktion get_temperature() aufgerufen.
Ist die Temperatur in die Variable temp geschrieben, wird der print-Befehl ausgeführt, welcher die Variable temp ausgibt.

1.3 Temperatur runden

Die Temperatur wird auf mehrere Nachkommastellen genau ausgegeben. So genau benötigst du die Temperaturangabe nicht, deshalb rundest du diese.

Aufgabe:

Runde die Temperatur auf zwei Nachkommastellen, bevor du diese mittels print-Befehl ausgibst.

Tipps zum Vorgehen:

Gerundet werden kann mittels dem Befehl round. Diesem müssen zwei Parameter mitgegeben werden, als erstes die Zahl, die gerundet werden soll (in unserem Fall temp) sowie die Anzahl der Nachkommastellen.

round(zahlDieGerundetWerdenSoll, anzahlNachkommastellen)

1.3 Temperatur runden

Lösungsvorschlag

```
1  from sense_hat import SenseHat
2  import time
3  sense = SenseHat()
4  temp = sense.get_temperature()
5  temp = round(temp,2)
6  print(temp)
```

Auf Zeile 5 ist das Runden der Temperatur auf zwei Nachkommastellen hinzugefügt worden. Die gerundete Temperatur wird wiederum in die Variable temp geschrieben.

Aufgabenkarte 2: Die Temperatur speichern

Die Angabe zu der Temperatur, nützt dir für deine Forschung noch nicht viel. Du musst immer wieder auf die Daten zugreifen können, um diese analysieren zu können.

Ziel:
Du lernst, wie du die ausgelesenen Temperaturdaten in einer Datei speichern kannst.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 2.1 File erstellen
- 2.2 Ins File schreiben
- 2.3 Neue Zeile einfügen
- 2.4 Temperatur in Variable schreiben
- 2.5 Temperatur schreiben

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

2.1 File erstellen

Die gesammelten Daten sollen in einer Datei abgespeichert werden. Dazu erstellst du eine Datei.

Aufgabe:

Erstelle eine leere csv-Datei, die mittels der Variablen file angesprochen werden kann. Die Datei selbst soll unter dem Namen Datafile gespeichert werden.

Tipps zum Vorgehen:

Mittels der Funktion open kann ein neues File angelegt werden. Dabei kann als Parameter der Filename mitgegeben werden, dann wird das File im aktuellen Verzeichnis erstellt. Alternativ kann hier der ganze Pfad spezifiziert werden, in welchem Das Dokument gespeichert werden soll. Welche Dokumentenart du erstellst, spezifizierst du anhand der Dateiendung. In unserem Fall .csv.
Als zweites wird ein Buchstabe mitgegeben, welcher den Modus spezifiziert, für dich reicht zu wissen, dass du hier ein a als String, also in Anführungszeichen mitgeben musst.

```
open("Dateiname.csv", "a")
```

2.1 File erstellen

Lösungsvorschlag

```
file = open("Datafile.csv", "a")
```

Da wir die csv-Datei später mittels file ansprechen wollen, weisen wir der Variablen file die Datei mittels dem = zu.
Der Funktion open geben wir als Parameter zuerst den Filenamen Datafile.csv mit. Mit dem .csv spezifizieren wir auch gleich, dass eine csv-Datei erstellt werden soll. Anstelle des Dokumentnamens könnte hier auch der ganze Pfad inkl. Dokumentname mitgegeben werden, z.B. "C:\Program Files\Results\Datafile.csv".
Der zweite Parameter ist der Modus, ein "a".

2.2 Ins File schreiben

Die von dir erstellte Datei soll natürlich nicht leer bleiben, deshalb beginnst du damit, in die Datei zu schreiben.

Aufgabe:

Schreibe "Hallo Welt!" in das in der vorherigen Aufgabe angelegte csv-File.
Öffne anschliessend das File und überprüfe, ob das reinschreiben geklappt hat.

Tipps zum Vorgehen:

Mittels der Funktion write kannst du in das Dokument schreiben. Als Parameter muss der Funktion der Text mitgegeben werden.
Vergiss nicht, dass du der Funktion mitteilen musst, wohin sie schreiben soll. Unter welchem Namen soll das Dokument angesprochen werden können?

2.2 Ins File schreiben

Lösungsvorschlag

```
file = open("Datafile.csv", "a")
file.write("Hallo Welt!")
```

Im Befehl auf der ersten Zeile erstellen wir das csv-File (vgl. vorherige Aufgabe).

Mit dem Befehl write und der Übergabe des Textes als Parameter, schreiben wir nun in das File. Achte darauf, dass der Text den du als Parameter übergibst in Anführungszeichen geschrieben ist.

2.3 Neue Zeile einfügen

Um die Datei übersichtlich zu gestalten, schreibst du nicht alles auf eine Zeile, sondern fügst gezielt neue Zeilen ein.

Aufgabe:

Schreibe "Hallo Welt!" in das in der Aufgabe 2.1 angelegte File. Wechsle dann auf eine neue Linie uns schriebe dort "Ich sehe dich vom Weltall aus!"
Öffne anschliessend das File und überprüfe, ob das reinschreiben geklappt hat.

Tipps zum Vorgehen:

Eine neue Zeile wird mittels \n erstellt. Dies wird als String, also in Anführungszeichen geschrieben, mitgegeben.

2.3 Neue Zeile einfügen

Lösungsvorschlag

```
file = open("Datafile.csv", "a")
file.write("Hallo Welt! \n")
file.write("Ich sehe dich vom Weltall aus!")
```

Der Befehl \n kann gleich anschliessend an den Text Hallo Welt! platziert werden.

2.4 Temperatur in Variable schreiben

Damit du die Temperatur nicht nur einmalig anzeigen, sondern auch damit arbeiten kannst, speicherst du den Wert in einer Variablen ab.

Aufgabe:

Schreibe die aktuelle Temperatur in die Variable temp und runde diese auf zwei Nachkommastellen.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie das Funktioniert, schaue dir die Aufgabenkarten 1: Die Temperatur messen nochmals gut an.

2.4 Temperatur in Variable schreiben

Lösungsvorschlag

```
temp = sense.get_temperature()
temp = round(temp,2)
```

Die Temperatur wird in die Variable temp geschrieben, über sense. Wird dem Programm mitgeteilt, in welchem Modul benötigte Funktion vorhanden ist. Nach dem Punkt wird die entsprechende Funktion get_temperature() aufgerufen. Die gerundete Temperatur wird wiederum in die Variable temp geschrieben.

2.5 Temperatur schreiben

Damit dir die gemessenen Daten auch nach dem ausführen des Programms zur Verfügung stehen, speicherst du diese in deiner Datei ab.

Aufgabe:

Schreibe "Temperatur" in das in der Aufgabe 2.1 angelegte File. Wechsle dann auf eine neue Linie uns schriebe dort, mithilfe der Variablen temp die aktuelle Temperatur auf, die du auf zwei Nachkommastellen gerundet hast.
Öffne anschliessend das File und überprüfe, ob das Reinschreiben geklappt hat.

Tipps zum Vorgehen:

Denk daran, dass du zuerst die Temperatur auslesen und runden musst, bevor du diese in die csv-Datei schreiben kannst.

2.5 Temperatur schreiben

Lösungsvorschlag

```
temp = sense.get_temperature()
temp = round(temp,2)
file = open("Datafile.csv", "a")
file.write("Temperatur \n")
file.write(temp)
```

Aufgabenkarte 3: Zeit und Temperatur speichern

Um Daten gezielt auswerten zu können, müssen diese mit anderen Angaben verknüpft werden. Dabei hilft es dir zu wissen, wann genau du deine Messdaten erhoben hast.

Ziel:
Du lernst, wie du neben den Temperaturdaten auch die aktuelle Zeit in ein Dokument speichern kannst.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 3.1 Zeit auslesen
- 3.2 Datum auslesen
- 3.3 Zeit und Datum in Datei schreiben
- 3.4 Zeit und Temperatur schreiben
- 3.5 Warten

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

3.1 Zeit auslesen

Um deine Messdaten mit anderen Daten verknüpfen zu können, möchtest du die aktuelle Zeit auslesen können.

Aufgabe:

Lies die aktuelle Zeit aus und lasse dir diese mittels print-Befehl ausgeben.

Tipps zum Vorgehen:

Die aktuelle Zeit erhältst du mittels des Befehles

```
time.strftime("%X")
```

Achtung: Das X muss grossgeschrieben sein.

3.1 Zeit auslesen

Lösungsvorschlag

```
print(time.strftime("%X"))
```

Was in der Klammer des print-Befehles steht, wird ausgegeben.
Darin wird der eben kennengelernte Befehl ausgeführt, um die aktuelle Zeit auszugeben.
Damit der Befehl ausgeführt werden kann, muss vorgängig das Paket time importiert werden, vgl. dazu Aufgabenkarte 1.1.

3.2 Datum auslesen

Um deine Messdaten mit anderen Daten verknüpfen zu können, möchtest du das aktuelle Datum auslesen können.

Aufgabe:

Lies das aktuelle Datum aus und lasse dir diese mittels print-Befehl ausgeben

Tipps zum Vorgehen:

Das aktuelle Datum erhältst du mittels des Befehles
time.strftime("%x")

Achtung: Das x muss kleingeschrieben sein.
Hinweis: Das Datum wird im Format Monat / Tag / Jahr ausgegeben.

3.2 Datum auslesen

Lösungsvorschlag

```
print(time.strftime("%x"))
```

Was in der Klammer des print-Befehles steht, wird ausgegeben.

Darin wird der eben kennengelernte Befehl ausgeführt, um die aktuelle Zeit auszugeben.

Damit der Befehl ausgeführt werden kann, muss vorgängig das Paket time importiert werden, vgl. dazu Aufgabenkarte 1.1.

3.3 Zeit und Datum in Datei schreiben

Die angaben zu Zeit und Datum hältst du in deiner Datei fest, damit dir diese beim Auswerten deiner Daten noch zur Verfügung stehen.

Aufgabe:

Schreibe das aktuelle Datum und die aktuelle Zeit in eine csv-Datei. Die csv-Daei soll "Datafile" heissen. Das Datum soll auf einer Zeile stehen, auf der neuen Zeile dann die Zeit.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du eine Datei erstellen und in diese schreiben kannst, schaue dir die Aufgabenkarten 2 nochmals an.

3.3 Zeit und Datum in Datei schreiben

Lösungsvorschlag

```
file = open("Datafile.csv", "a")
file.write(time.strftime("%x"))
file.write("\n")
file.write(time.strftime("%X"))
```

Was in der Klammer des file.write-Befehles steht, wird in die angelegte csv-Datei geschrieben.

Das \n bedeutet, dass auf einer neuen Zeile weitergeschrieben werden soll.

3.4 Zeit und Temperatur schreiben

Um erfolgreich Forschen zu können, ist es nicht nur wichtig Daten zu haben, sondern diese auch zuordnen zu können. Deshalb speicherst du nicht nur die gemessene Temperatur ab, sondern auch den Messzeitpunkt.

Aufgabe:

Schreibe die aktuelle Zeit und die Temperatur die du auf zwei Nachkommastellen gerundet hast, in eine csv-Datei, die Datafile heisst.
Öffne anschliessend die Datei und überprüfe, ob das Reinschreiben funktioniert hat.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du die Temperatur runden und in das Dokument schreiben kannst, schaue dir die Aufgabenkarten 2 nochmals an.

3.4 Zeit und Temperatur schreiben

Lösungsvorschlag

```
file = open("Datafile.csv", "a")
temp = sense.get_temperature()
temp = round(temp,2)
file.write(time.strftime("%X"))
file.write(" ")
file.write(temp)
```

In der ersten Zeile wird die csv-Datei Datafile angelegt. In Zeile 2 und 3 wird die Temperatur ausgelesen und auf zwei Nachkommastellen gerundet. In Zeile 4 bis 6 wird die aktuelle Zeit, ein Leerschlag und die gerundete Temperatur in die csv-Datei geschrieben.

Damit die Befehle ausgeführt werden können, müssen vorgängig die benötigten Module und Pakete importiert werden, vgl. dazu Aufgabenkarte 1.1.

3.5 Warten

Du benötigst nicht unendlich viele Daten, dass macht nur unnötig viel Arbeit bei der Auswertung. Deshalb pausierst du zwischen einzelnen Messungen.

Aufgabe:

Schreibe die aktuelle Zeit in die csv-Datei Datafile. Warte drei Sekunden und schreibe erneut die aktuelle Zeit in die csv-Datei Datafile. Die zweite Zeit, soll auf einer neuen Zeile geschrieben werden.
Öffne anschliessend die Datei und überprüfe, ob das Reinschreiben funktioniert hat.

Tipps zum Vorgehen:

Mit der Funktion sleep() kann eine bestimmte Zeit gewartet werden. Die Wartezeit wird der Funktion als Parameter mitgegeben, sie wird in Sekunden angegeben. Soll eine Sekunde gewartet werden, kann folgender Befehl genutzt werden:

```
time.sleep(1)
```

3.5 Warten

Lösungsvorschlag

```
file = open("Datafile.csv", "a")
file.write(time.strftime("%X"))
file.write("\n")
time.sleep(3)
file.write(time.strftime("%X"))
```

In Zeile 1 wird die csv-Datei Datafile angelegt. In Zeile 2 wird die aktuelle Zeit reingeschrieben, auf Zeile 3 wird ein Abstand eingefügt.
In Zeile 4 wird die sleep-Funktion genutzt, um 3 Sekunden zu warten. In Zeile 5 wird erneut die aktuelle Zeit in die csv-Datei geschrieben.

Aufgabenkarte 4: Mehrere Temperaturen speichern

Um Muster in Daten feststellen zu können, benötigst du viele Messdaten.

Ziel:
Du lernst, wie du die Temperatur in regelmässigen Zeitabständen abspeichern kannst.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 4.1 Mehrere Temperaturen speichern
- 4.2 Die Schleife verstehen
- 4.3 Zeit und Temperatur speichern
- 4.4 Einträge Analysieren
- 4.5 Warten zwischen Einträgen
- 4.6 Datei schliessen

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

4.1 Mehrere Temperaturen speichern

Um erfolgreich Forschen zu können, benötigst du viele Messdaten. Denn erst aus einer genügend grossen Datenmenge, kannst du interessante Muster identifizieren.

Aufgabe:

Speichere mithilfe der while-Schleife 20-mal die Temperatur in die csv-Datei Datafile. Die Temperatur soll dabei immer auf eine neue Zeile geschrieben werden.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie die while-Schlaufe funktioniert, schaue dir die blauen Aufgabenkarten nochmals an.

4.1 Mehrere Temperaturen speichern

Lösungsvorschlag

```
file = open("Datafile.csv", "a")

anzahl = 1
while anzahl <= 20:
  temp = sense.get_temperature()
  temp = round(temp,2)

  file.write(temp)
  file.write("\n")
  anzahl = anzahl + 1
```

Die while-Schlaufe wird 20-mal ausgeführt. Gestartet wird mit der Zahl 1, welche sich innerhalb der Schleife hochzählt. Ist die Zahl grösser als 20, wird die Schleife abgebrochen.

4.2 Die Schleife verstehen

Du sollst nicht nur viele Daten sammeln können, sondern auch verstehen, wie diese Daten gesammelt werden. Denn dies kann einen Einfluss auf deine Auswertung haben.

Aufgabe:

Schaue dir die folgende while-Schlaufe genau an. Weshalb wird die Temperatur des Temperatur-Sensors innerhalb der Schlaufe ausgelesen und gerundet und nicht davor?

```
file = open("Datafile.csv", "a")

anzahl = 1
while anzahl <= 20:
  temp = sense.get_temperature()
  temp = round(temp,2)

  file.write(temp)
  file.write("\n")
  anzahl = anzahl + 1
```

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie die while-Schlaufe funktioniert, schaue dir die blauen Aufgabenkarten nochmals an.

Achte dich genau darauf, welche Bereiche eingerückt sind und welche nicht.

4.2 Die Schleife verstehen

Lösungsvorschlag

Die Temperatur wird innerhalb der Schleife aus dem Temperatursensor ausgelesen, damit bei jedem abspeichern die aktuelle Temperatur gespeichert wird.

Würde die Temperatur vor dem Beginn der Schleife ausgelesen, so würde immer wieder die Temperatur abgespeichert werden, welche beim Starten der Schleife gemessen worden ist.

4.3 Zeit und Temperatur speichern

Neben den effektiven Messwerten, speicherst du weitere Werte ab, welche dir bei der Auswertung nützlich sein können.

Aufgabe:

Speichere mithilfe der while-Schleife 20x die aktuelle Zeit und Temperatur in die csv-Datei Datafile. Die Zeit soll mit einem Leerschlag von der Temperatur getrennt eingetragen werden. Jeder Eintrag, bestehend aus Zeit und Temperatur, soll dabei immer auf eine neue Zeile geschrieben werden.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du die Zeit auslesen kannst, schaue dir die Aufgabenkarten 3 nochmals an.

4.3 Zeit und Temperatur speichern

Lösungsvorschlag

```
anzahl = 1
while anzahl <= 20:
  temp = sense.get_temperature()
  temp = round(temp,2)

  file.write(time.strftime('%X'))
  file.write(" ")
  file.write(temp)
  file.write("\n")
  anzahl = anzahl + 1
```

Die while-Schlaufe wird 20-mal ausgeführt. Gestartet wird mit der Zahl 1, welche sich innerhalb der Schleife hochzählt. Ist die Zahl grösser als 20, wird die Schleife abgebrochen.

4.4 Einträge Analysieren

Um eine gute Forscherin oder ein guter Forscher zu werden, musst du nicht nur Daten sammeln, sondern diese auch interpretieren können.

Aufgabe:

Öffne die csv-Datei Datafile, welches du in der vorherigen Aufgabe 4.3 erstellt hast. Schaue dir die Zeit- und Temperatureinträge genau an. Was fällt dir dabei auf?

Tipps zum Vorgehen:

13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17

13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17
13:26:07 22.17

4.4 Einträge Analysieren

Lösungsvorschlag

Es fällt auf, dass die Temperaturmessungen sehr dicht aufeinander erfolgen. Die 20 Einträge aus dem Beispiel-Ergebnis, wurden alle innerhalb derselben Sekunde erfasst.

4.5 Warten zwischen Einträgen

Du beschliesst, dass mehrmals pro Sekunde die Temperatur zu messen übertrieben ist.

Aufgabe:

Speichere mithilfe der while-Schleife 20x die aktuelle Zeit und Temperatur in die csv-Datei Datafile. Die Zeit soll mit einem Leerschlag von der Temperatur getrennt eingetragen werden. Jeder Eintrag, bestehend aus Zeit und Temperatur, soll dabei immer auf eine neue Zeile geschrieben werden. Nach jedem Eintrag soll eine Sekunde gewartet werden.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du Pausieren kannst, schaue dir die Aufgabenkarten 3 nochmals an.

4.5 Warten zwischen Einträgen

Lösungsvorschlag

```
anzahl = 1
while anzahl <= 20:
  temp = sense.get_temperature()
  temp = round(temp,2)

  file.write(time.strftime('%X'))
  file.write(" ")
  file.write(temp)
  file.write("\n")
  anzahl = anzahl + 1
  time.sleep(1)
```

Die while-Schlaufe aus der Aufgabe 4.3 wurde um die sleep-Funktion ergänzt.

4.6 Datei schliessen

Nachdem du alle benötigten Daten für deine Forschung in die Datei geschrieben hast, schliesst du die Datei.

Aufgabe:

Nachdem du 20 Einträge in die csv-Datei geschrieben hast, schliesse die Datei.

Tipps zum Vorgehen:

Mit der close-Funktion, kannst du die Datei schliessen. Da wir die Datei jeweils mit file ansprechen, heisst der benötigte Code:

```
file.close()
```

4.6 Datei schliessen

Lösungsvorschlag

```
anzahl = 1
while anzahl <= 20:
  temp = sense.get_temperature()
  temp = round(temp,2)

  file.write(time.strftime('%X'))
  file.write(" ")
  file.write(temp)
  file.write("\n")
  anzahl = anzahl + 1
  time.sleep(1)

file.close()
```

Das Schliessen des Files mittels der close-Funktion erfolgt ausserhalb der while-Schleife. Denn das File soll erst geschlossen werden, nachdem alle Einträge reingeschrieben wurden.

Die Programmierumgebung – Astro Pi

Um das Projekt umsetzen zu können, müssen wir uns zuerst mit der Programmierumgebung vertraut machen.

Ziel:
Du lernst die Programmierumgebung des Astro Pi zu nutzen.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 1 Eine Verbindung herstellen
- 2 Die Umgebung starten
- 3 Das erste kleine Programm

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

1 Eine Verbindung herstellen

Als erstes musst du eine Verbindung zwischen dem Astro Pi und einem Monitor herstellen.

Aufgabe:

Schliesse den Astro Pi an einen Monitor an.

Tipps zum Vorgehen:

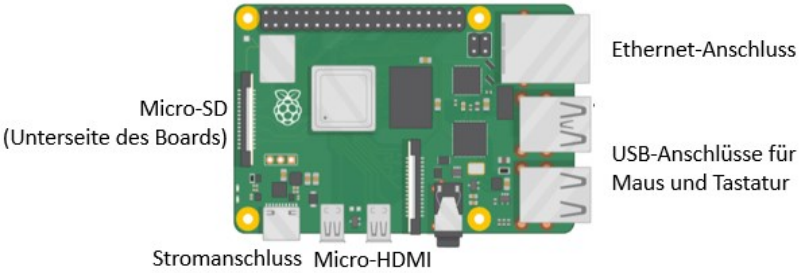
Um den Astro Pi an einen Monitor anzuschliessen, benötigst du:

- Astro Pi
- Netzgerät
- Micro-HDMI-Kabel oder HDMI-Kabel
- Tastatur
- Maus
- SD-Karte

1 Eine Verbindung herstellen

Lösungsvorschlag

Schliesse das Stromkabel als letztes an.



2 Die Umgebung starten

Ist der Astro Pi an den Monitor angeschlossen, kannst du die Programmierumgebung starten.

Aufgabe:

Starte die Astro Pi Programmierumgebung.

Tipps zum Vorgehen:

Monitor:
Sobald das Stromkabel angeschlossen ist, startet der Astro Pi automatisch

Auf das Himbeer-Symbol klicken, dann *Thonny Phyton IDE* starten

2 Die Umgebung starten

Lösungsvorschlag

3 Das erste kleine Programm

Nun bist du bereit, dein erstes Programm für den Astro Pi zu schreiben.

Aufgabe:

Schreibe ein Programm, das einen kurzen Text auf der Konsole ausgibt.

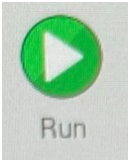
Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du einen Text ausgeben kannst, schaue dir die blauen Aufgabenkarten nochmals an.

3 Das erste kleine Programm

Lösungsvorschlag

```
print("Hallo ich bin ein Text der angezeigt wird!")
```



Mit dem Button "Run" kannst du das Programm ausführen. Bevor das Programm ausgeführt wird, musst du es abspeichern. Wähle als Speicherort den Desktop aus und nenne das Programm "Hello".

Aufgabenkarte 5: Gemessene Temperatur verifizieren

Damit deine Forschung zu verwertbaren Ergebnissen führt, musst du immer wieder kontrollieren, ob du richtig misst.

Ziel:
Du lernst, deine Messergebnisse kritisch zu Hinterfragen und gegebenenfalls Korrekturen vorzunehmen.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 5.1 Korrektheit der Messung
- 5.2 Abweichungen der Messungen
- 5.3 Die Messung korrigieren
- 5.4 Schleife mit korrigierter Messung

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

5.1 Korrektheit der Messung

Um sicherzustellen, dass korrekt gemessen wird, müssen die Messdaten des Astro Pi mit anderen Daten verglichen werden.

Aufgabe:

Miss mit dem Astro Pi die Temperatur und lasse dir diese mittels print-Funktion ausgeben. Miss zudem mit einem gewöhnlichen Thermometer die Temperatur.
Wiederhole das Messen mindestens 3-mal. Trage deine Messergebnisse in die Tabelle ein.

Messung Nr.	Astro Pi	Thermometer
1		
2		
3		
4		
5		
Durchschnittliche Temperatur		
Unterschied Astro Pi - Thermometer		

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du die Temperatur messen kannst, schaue dir die Aufgabenkarte 1 nochmals an.

5.1 Korrektheit der Messung

Lösungsvorschlag

Wahrscheinlich ist die Temperatur, welche der Astro Pi misst, höher als die Temperatur, welche du mit dem Thermometer misst.
Dies kann daran liegen, dass der Astro Pi selbst wärme abgibt.

5.2 Abweichungen der Messungen

Damit du weisst, um wie viel du die Messung korrigieren musst, musst du die unterschiedlichen Messdaten genau anschauen.

Aufgabe:

Fülle in der Tabelle aus der vorherigen Aufgabe 5.1 die durchschnittlich gemessene Temperatur für den Astro Pi und den Thermometer auf.
Wie gross ist der Messunterschied im Durchschnitt?

Tipps zum Vorgehen:

Den Durchschnitt errechnest du, indem du alle Messergebnisse zusammenzählst und durch die Anzahl der Messergebnisse teils.
Bei den drei Messergebnissen 20.00 Grad, 22.50 Grad und 21.50 Grad ergäbe die folgende Rechnung:
(20.00 + 22.50 + 21.50) / 3 = 21.35

5.2 Abweichungen der Messungen

Lösungsvorschlag

Individuelle Lösungen: Frag deine Lehrperson ob du richtig gerechnet hast.

5.3 Die Messung korrigieren

Es erleichtert die Auswertung, wenn du die Originalwerte (sprich die gemessene Temperatur) und die korrigierten Werte abspeicherst. So kannst du je nach Bedarf mit den einen oder anderen Daten arbeiten.

Aufgabe:

Schreibe die aktuelle Zeit die gemessene Temperatur, die du auf zwei Nachkommastellen gerundet hast und die Temperatur, die du um die durchschnittliche Abweichung korrigiert hast, in eine csv-Datei, die Datafile heisst. Die Zeit soll mit einem Leerschlag von der Temperatur und mit einem Minus - getrennt von der korrigierten Temperatur eingetragen werden.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du die Zeit und Temperatur in eine csv-Datei speichern kannst, schaue dir die Aufgabenkarten 3 nochmals an.

5.3 Die Messung korrigieren

Lösungsvorschlag

```
file = open("Datafile.csv", "a")
temp = sense.get_temperature()
temp = round(temp,2)
tempkorr = round(temp - 2.30,2)
file.write(time.strftime('%X'))
file.write(" ")
file.write(temp)
file.write("-")
file.write(tempkorr)
```

In Zeile 4 wird mit Hilfe der Variable tempkorr die Temperatur um 2.3 Grad korrigiert / reduziert. Mit round wird auch die korrigierte Temperatur gleich wieder auf zwei Nachkommastellen gerundet.

5.4 Schleife mit korrigierter Messung

Du möchtest nicht nur einmal die Temperatur messen, sondern mehrmals.

Aufgabe:

Speichere mithilfe der while-Schleife 20x die aktuelle Zeit, die gemessene Temperatur und die um die durchschnittliche Abweichung korrigierte Messung in die csv-Datei Datafile. Die Zeit soll mit einem Leerschlag von der Temperatur und mit einem Minus - getrennt von der korrigierten Temperatur eingetragen werden. Jeder Eintrag, bestehend aus Zeit gemessener und korrigierte Temperatur, soll dabei immer auf eine neue Zeile geschrieben werden. Nach jedem Eintrag soll eine Sekunde gewartet werden.

Tipps zum Vorgehen:

Wenn du dir nicht mehr sicher bist, wie du mehrmals die Zeit und Temperatur in eine csv-Datei speichern kannst, schaue dir die Aufgabenkarten 4 nochmals an.

5.4 Schleife mit korrigierter Messung

Lösungsvorschlag

```
anzahl = 1
while anzahl <= 20:
  temp = sense.get_temperature()
  temp = round(temp,2)
  tempkorr = round(temp - 2.30,2)

  file.write(time.strftime('%X'))
  file.write(" ")
  file.write(temp)
  file.write("-")
  file.write(tempkorr)
  file.write("\n")
  anzahl = anzahl + 1
  time.sleep(1)
```

Die while-Schleife aus Ausgabe 4.5 wurde um die Ausgabe der korrigierten Temperatur mit Hilfe der Variable tempkorr erweitert.