

Inhaltsverzeichnis

Stand Lektion 18	2
geheim.py	2
display.py	3
netzwerk20.py	4
webseiten.py	6
openweathermap20.py	8

Stand Lektion 18

In dieser Lektion werden einige Module stark überarbeitet. Sie sind daher teilweise noch nicht vollständig einsatzbereit.

Die Module **bme280_i2c**, **microWebSrv** und **ssd1306** wurden aus dem Internet geladen und werden hier nicht besprochen.

Bitte nicht vergessen:

Alle hier besprochenen Module müssen auf das Board kopiert werden.

geheim.py

Das sind nur Konstanten und Einstellungsdaten. Ausführbarer Code ist hier nicht enthalten. Es sind auch die Zugangsdaten zum WLAN und dem Wetterservice gespeichert. Daher muss die Datei vertraulich behandelt werden.

Das neue OneCall API benötigt Koordinaten. Diese werden ebenfalls hier gespeichert.

owm_lat = Breitengrad, owm_lon = Längengrad

```
# WLAN
wlan_ssid      = "ssid"
wlan_passwort  = "gemeim"

# OpenWeatherMap
owm_id         = "1234567890abcd"
owm_ortID      = 2659083
owm_ort        = "Regensdorf"
owm_land       = "CH"
owm_lat        = 47.4315
owm_lon        = 8.4661
```

display.py

Dieses erstellt eine Instanz des OLEDs auf dem Heltec - Board und stellt diese zur Verfügung. Näheres zu den Funktionen findet man in **Lektion 5**.

Die wichtigsten Funktionen sind:

`oled.fill(0)` Display löschen

`oled.show()` Inhalt anzeigen

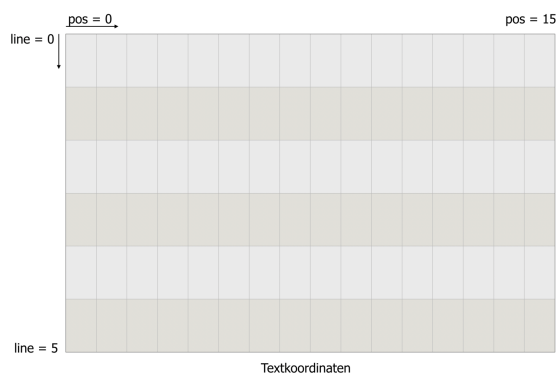
Für einfache Textausgaben wird die Funktion `text_line()` zur Verfügung gestellt.

```
text_line(text, line, pos = 0)
```

Diese Funktion schreibt den Text **text** auf Zeile **line** und Zeichenposition **pos**.

line umfasst den Bereich von 0 bis 5 und **pos** von 0 bis 15.

Wenn **pos** nicht mitgegeben wird, beginnt der Text am Zeilenanfang.



netzwerk20.py

netzwerk20.py stellt die Klasse **WiFi** und automatisch auch eine Instanz mit dem Namen **wifi** zur Verfügung. Für dich ist nur wichtig, dass du mit **from netzwerk20 import *** das Objekt **wifi** verfügbar machen kannst. Alle Funktionen werden mit **wifi.** aufgerufen. Es dürfen keine weiteren Instanzen von **WiFi** erstellt werden!

Es werden alle Funktionen zur Verfügung gestellt, die benötigt werden, um eine WLAN - Verbindung aufzubauen. Ausserdem enthält die Klassen Funktionen um die aktuelle Zeit aus dem Internet abzufragen und darzustellen.

Ab Lektion 18 steht eine zusätzliche Funktion `get_request(url)` zur Verfügung. Diese basiert auf dem Modul `urequests` und vereinfacht den Zugriff auf die JSON-Daten.

Klasse WiFi, Funktionen für WLAN

Funktion	Funktion	Beschreibung
<code>connect(ssid, passwort, timeout = 10000)</code>	ssid und passwort : Zugangsdaten timeout : maximale Wartezeit in Sekunden	Verbindet mit dem WLAN und gibt True oder False zurück.
<code>isconnected()</code>		True, wenn Verbindung besteht.
<code>get_wlan()</code>		Gibt das wlan - Objekt aus network zurück. Das wird normalerweise nicht benötigt.
<code>get_ip()</code>		Gibt die IP-Adresse zurück.
<code>get_ssid()</code>		Gibt die SSID zurück.
<code>http_get(url)</code>	url : auf diese url wird der Get-Request gemacht	Führt einen Get-Request aus und gibt das Resultat zurück. Diese Funktion wird in zukünftigen Versionen gestrichen. Stattdessen sollte <code>get_request</code> verwendet werden!
<code>get_request(url)</code>	url : auf diese url wird der Get-Request gemacht	Führt einen Get-Request auf Basis des <code>urequests</code> - Moduls durch. Durch <code>.json()</code> hat man direkt Zugriff auf die JSON - Struktur.

Klasse WiFi, Funktionen für Zeit und Datum

Funktion	Funktion	Beschreibung
set_timezone(zone)	zone: Zeitverschiebung in Sekunden	Die Zeitverschiebung ist beim Start auf 3600 Sekunden festgelegt. Das kann hier geändert werden.
get_seconds()	Aktuelle Zeit in Sekunden (UTC)	Die aktuelle UTC-Zeit wird über das Internet gelesen.
get_local_seconds()	Aktuelle Lokalzeit in Sekunden	Die aktuelle UTC-Zeit wird über das Internet gelesen und mit Hilfe der Zeitzone in lokale Zeit umgewandelt.
get_time_values(seconds)	seconds ist die Zeit in Sekunden. Diese wird in die Einzelwerte umgerechnet.	Es werden YYYY, MM, DD, HH, MM, SS, day of week (0=Montag), day of year zurückgegeben.
date_text(seconds)	seconds ist die Zeit in Sekunden.	Gibt tt.mm.yyyy zurück.
time_text(seconds)	seconds ist die Zeit in Sekunden.	Gibt hh:mm:ss zurück.
date_time_text(seconds)	seconds ist die Zeit in Sekunden.	Gibt tt.mm.yyyy hh:mm:ss zurück.

webseiten.py

Dieses Modul wurde von webseiten02.py zu webseiten.py umbenannt. Ab sofort sollen hier keine 'breaking changes' mehr erfolgen. Das bedeutet, dass neue Versionen des Moduls kompatibel zu älterem Code bleiben.

Dieses Modul stellt die Funktionen zur Darstellung einer Webseite zur Verfügung. Es wird laufend erweitert.

Mit startHTTP() wird der Webserver aktiviert.

Eine Webseite wird über die Klasse BigScreen erstellt. Momentan werden davon 3 Instanzen erzeugt.

hp: Eine einfache Homepage ("/")

log: Für die Ausgabe von Log-Informationen ("/log"). **log** schreibt standardmässig in das File **log.txt**.

wetter: Ausgabe der Wetterdaten ("/wetter")

Die Klasse BigScreen

Funktion	Funktion	Beschreibung
BigScreen(titel, refresh, filename)	titel: Titel der Seite refresh: nach soviel Sekunden wird die Webseite automatisch neu geladen filename (optional): Filename für Zwischenspeicherung in File	Erzeugen der Instanz. Dies geschieht automatisch. Wenn kein Filename angegeben ist, werden die Daten im Memory gehalten. Sonst werden sie in ein File mit dem betreffenden Namen gespeichert. Das Anlegen des Files geschieht automatisch. Beim Erzeugen des Objekts mit Filenamen wird der Inhalt einer allenfalls bestehenden Datei gelöscht.
get_version()		Gibt die Versionsnummer des Moduls zurück. 100 steht dabei für Version 1.0
set_filename(filename)	filename (optional): Filename für Zwischenspeicherung in File	Der Filename kann auch nachträglich gesetzt oder gelöscht werden. Die bereits gespeicherten Daten bleiben erhalten.
set_refresh(refresh)	refresh: nach soviel Sekunden wird die Webseite automatisch neu geladen	Es wird eine neue Refresh - Zeit gesetzt. Diese tritt erst nach dem nächsten Seitenaufruf in Kraft.
clear()		Der Text der Webseite wird gelöscht. Es wird nur noch der Titel dargestellt.
add(line)	line: Textzeile, die hinzugefügt werden soll.	Hinzufügen einer Textzeile. add() fügt eine Leerzeile hinzu.

Funktion	Funktion	Beschreibung
add_log(line)	line : Textzeile, die hinzugefügt werden soll.	Hinzufügen einer Textzeile mit vorangestellter aktueller Zeit. add() fügt eine Leerzeile hinzu.
get_content()		Gibt alle hinzugefügten Textzeilen in einem String zurück. Der Zeilenumbruch wird mit markiert.
print():		Gibt alle Textzeilen auf die Konsole aus.
getHTML():		Gibt den HTML-Code für die ganze Webseite zurück. Dazu wird die Hilfsfunktion _html() verwendet.

Wenn ein Filename angegeben ist, werden die Daten automatisch in dieser Datei verwaltet. Dazu werden keine speziellen Funktionen verwendet. Die folgenden Filefunktionen der Klasse BigScreen werden normalerweise nur intern gebraucht, müssen also von aussen nicht aufgerufen werden. Falls kein Filename definiert wurde, führen die Funktionen keine Operation aus.

Funktion	Beschreibung
add_line_to_file(line)	Der Text in line wird der Datei hinzugefügt. Falls notwendig, wird die Datei automatisch erzeugt. Ein Aufruf ohne line fügt eine Leerzeile hinzu.
read_lines_from_file()	Der ganze Text in der Datei wird gelesen. Als Ergebnis werden alle Zeilen zurückgegeben. Der Zeilenumbruch \n ist aber noch in jeder Zeile enthalten.
clear_file()	Der Inhalt des Files wird gelöscht, das File selbst bleibt aber erhalten.

openweathermap20.py

Dieses Modul ist für den Bezug der Wetterdaten bei **OpenWeatherMap.org** zuständig.

Dieses Modul ist ab Lektion 18 verfügbar. Die Klassen **WeatherWebService** und **WetterAktuell** sind weiterhin vorhanden, werden aber in einer späteren Lektion entfernt. Ersatzklassen sind noch keine vorhanden, diese werden erst in einer späteren Lektion erarbeitet.

Die neue Klassen **WetterOneCall** ist nur experimentell und nicht vollständig funktionsfähig.

Als Basis dient die Klasse **WeatherWebService**. Von dieser wird keine Instanz erstellt, die dient nur als Grundlage für die Wetterklassen. Momentan ist hier nur **WetterAktuell** definiert.

Klasse WeatherWebService (Basisklasse)

Funktion	Beschreibung
request_data()	Fragt die Daten vom Server ab.
get_json()	JSON der letzten Messung
get_messzeit()	Datum der letzten Messung
get_messzeit_text()	Messzeitpunkt als Text
get_ort_name()	Ortsname
get_land	Land
get_data(topic1,topic2="",topic3="")	Laden eines einzelnen Datensatzes. Die Erklärung erfolgt weiter unten.
get_data_idx(topic1,idx = 0,topic2="",topic3="")	Laden eines einzelnen Datensatzes aus einem Array. Die Erklärung erfolgt weiter unten.

Klasse WetterAktuell (erbt alle Eigenschaften und Funktionen von WeatherWebService)

Funktion	Beschreibung
WetterAktuell(app_id, ort_id, ort_name, land, refresh_rate=600)	Erstellt eine Instanz von WetterAktuell. app_id : ID zum Zugriff auf den Webservice ort_id : ID der Ortschaft ort_name : bleibt leer, wenn ort_id verwendet wird land : bleibt leer, wenn ort_id verwendet wird refresh_rate : bleibt normalerweise auf 600 Sekunden, da OpenWeatherMap seine Daten nur etwa alle 10 Minuten aktualisiert.
set_ort(self, ort_id, ort_name, land)	Die Ortsangaben können nachträglich noch verändert werden. ort_id : ID der Ortschaft ort_name : bleibt leer, wenn ort_id verwendet wird land : bleibt leer, wenn ort_id verwendet wird
get_request_string()	Das ist der Request-String, der den Bezug der Daten vom Server erlaubt. Diese Funktion wird normalerweise im Hauptprogramm nicht verwendet.

Funktion	Beschreibung
refresh(force = False)	Falls force True ist, dann werden die Daten sofort vom Server angefordert. Andernfalls werden die Daten nur angefordert, falls die mit refresh_rate definierte Zeit bereits verstrichen ist. Falls aber beim vorherigen regulären Datenbezug keine neuen Daten vorhanden waren, wird diese Zeit temporär auf 60 Sekunden reduziert. Wenn neue Daten vorhanden waren, werden diese gespeichert und True zurückgegeben.

Klasse WetterOneCall (erbt alle Eigenschaften und Funktionen von WeatherWebService)

Diese Klasse ist experimentell und nur teilweise funktionsfähig!

Funktion	Beschreibung
request_data()	Frägt die Daten vom Server ab.
get_json()	JSON der letzten Messung
get_messzeit()	Datum der letzten Messung
get_messzeit_text()	Messzeitpunkt als Text
get_ort_name()	Ortsname
get_land	Land
get_data(topic1,topic2="",topic3="")	Laden eines einzelnen Datensatzes. Die Erklärung erfolgt weiter unten.
get_data_idx(topic1,idx = 0,topic2="",topic3="")	Laden eines einzelnen Datensatzes aus einem Array. Die Erklärung erfolgt weiter unten.

Bezug von Daten über Topics

```
get_data(topic1,topic2="",topic3="")
```

Die Daten liegen in verschachtelten JSON - Strukturen vor.

So ist der Ortsname im ersten Level verfügbar und kann mit

```
get_data("name")
```

abgerufen werden.

Das Land steht im zweiten Level und muss mit

```
get_data("sys","country")
```

abgerufen werden.

Maximal stehen drei Levels zur Verfügung.

Falls ein Datensatz nicht geliefert wurde, erfolgt keine Fehlermeldung. Es wird einfach ein leerer String zurückgegeben.

Es gibt Daten, die als Listen zurückgegeben werden. Hier können die Einzelwerte über den Index extrahiert werden.

```
get_data_idx(topic1,idx = 0,topic2="",topic3="")
```

Das ist zum Beispiel bei der Wetterbeschreibung der Fall.

```
get_data_idx("weather",0,"description")
```

Eine Funktion zur Abfrage der Anzahl Einträge in der Liste steht momentan noch nicht zur Verfügung.