

Inhaltsverzeichnis

Stand Lektion 19	2
geheim.py	2
display.py (wird momentan nicht unterstützt)	3
netzwerk.py	4
webseiten.py	6
openweathermap.py	8

Stand Lektion 19

Alle Module wurden stark geändert und an die neue Hardware angepasst.

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. Die Orts-ID fällt weg.

owm_lat = Breitengrad, owm_lon = Längengrad

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

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

display.py (wird momentan nicht unterstützt)

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

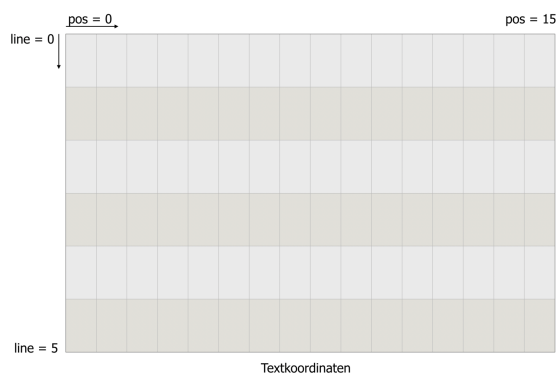
Für einfache Textausgaben wird die Funktion `text_line()` zur Verfügung gestellt.

```
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.



netzwerk.py

Es werden alle Funktionen zur Verfügung gestellt, die benötigt werden, um eine WLAN - Verbindung aufzubauen und die aktuelle Zeit über das Internet zu beziehen (Klasse **WiFi**). Ausserdem enthält das Modul Funktionen zur Darstellung von Datum und Zeit.

Es wird empfohlen das Modul mit

```
from netzwerk import *
```

zu importieren.

Ausserdem ist es notwendig, eine Instanz der Klasse WiFi zu erstellen. **Es darf nur eine einzige Instanz erzeugt werden!**

```
wifi = WiFi()
```

Klasse WiFi, Funktionen für die WLAN - Verbindung

Funktion	Funktion	Beschreibung
connect(ssid, passwort, timeout = 10000)	ssid und passswort : Zugangsdaten timeout : maximale Wartezeit in Sekunden	Verbindet mit dem WLAN und gibt True oder False zurück. Falls bereits eine Verbindung besteht, wird diese zuerst getrennt.
fast_connect(ssid, passwort, timeout = 10000)	ssid und passswort : Zugangsdaten timeout : maximale Wartezeit in Sekunden	Verbindet mit dem WLAN und gibt True oder False zurück. Falls bereits eine Verbindung besteht, wird diese verwendet.
get_wlan()		Gibt das interne wlan - Objekt aus netzwerk zurück. Das wird normalerweise nicht benötigt.
isconnected()		True, wenn Verbindung besteht.
get_ip()		Gibt die IP-Adresse zurück.
get_ssid()		Gibt die SSID zurück.
getRequest(url)	url : auf diese url wird der Get-Request gemacht	Führt einen Get-Request auf Basis des urequests - Moduls durch. Durch .json() hat man direkt Zugriff auf die JSON - Struktur.

Klasse WiFi, Funktionen zum Bezug von Zeit und Datum

Funktion	Funktion	Beschreibung
set_zeitzone(zone) Das ist eine Klassenmethode!	zone: Zeitverschiebung in Sekunden	Die Zeitverschiebung ist beim Start auf 3600 Sekunden festgelegt. Das kann hier geändert werden.
get_zeit() Das ist eine Klassenmethode!	Aktuelle Zeit in Sekunden (UTC)	Die aktuelle UTC-Zeit wird über das Internet gelesen.
get_zeit_lokal() Das ist eine Klassenmethode!	Aktuelle Zeit unter Berücksichtigung der Zeitzone.	Die aktuelle UTC-Zeit wird über das Internet gelesen.
get_zeit_lokal(zeit) Das ist eine Klassenmethode!	Umrechnung von zeit in die korrekte Zeit unter Berücksichtigung der Zeitzone.	

Hilfsfunktionen zur Textformatierung (nicht Bestandteil einer Klasse)

Funktion	Funktion	Beschreibung
datum_text(zeit)	zeit ist die Zeit in Sekunden.	Gibt tt.mm.yyyy zurück.
zeit_text(zeit)	zeit ist die Zeit in Sekunden.	Gibt hh:mm:ss zurück.
datum_zeit_text(zeit)	zeit ist die Zeit in Sekunden.	Gibt tt.mm.yyyy hh:mm:ss zurück.

webseiten.py

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
<code>BigScreen(titel, refresh, filename)</code>	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.
<code>get_version()</code>		Gibt die Versionsnummer des Moduls zurück. 100 steht dabei für Version 1.00
<code>set_filename(filename)</code>	filename (optional): Filename für Zwischenspeicherung in File	Der Filename kann auch nachträglich gesetzt oder gelöscht werden. Die bereits gespeicherten Daten bleiben erhalten.
<code>set_refresh(refresh)</code>	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.
<code>clear()</code>		Der Text der Webseite wird gelöscht. Es wird nur noch der Titel dargestellt.
<code>add(line)</code>	line: Textzeile, die hinzugefügt werden soll.	Hinzufügen einer Textzeile. <code>add()</code> 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.

openweathermap.py

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

Es stellt die Klasse **WetterService** zur Verfügung.

Klasse WetterService

Funktion	Beschreibung
WetterService(app_id, lat, lon, ort_name, land, refresh_rate=600)	Erstellt eine Instanz von WetterAktuell. app_id : ID zum Zugriff auf den Webservice lat, lon : Längen und Breitengrad ort_name : Ortsname land : Land refresh_rate : bleibt normalerweise auf 600 Sekunden, da OpenWeatherMap seine Daten nur etwa alle 10 Minuten aktualisiert.
set_ort(self, lat, lon, ort_name, land)	Die Ortsangaben können nachträglich noch verändert werden. lat, lon : Längen und Breitengrad ort_name : bleibt leer, wenn ort_id verwendet wird land : bleibt leer, wenn ort_id verwendet wird
get_ort_name()	Gibt den Ortsnamen zurück.
get_land()	Gibt das Land zurück.
request_data(excludes)	Bezieht die Daten bei OpenWeatherMap . Wenn der Bezug nicht funktioniert, wird None zurückgegeben. Andernfalls werden die Daten vom OneCallApi als JSON - Struktur zurückgegeben. excludes : gibt Elemente an, die vom Bezug ausgeschlossen werden. 'minutely' muss immer angegeben werden. Möglich sind current, hourly, daily, minutely . Das Hauptprogramm sollte diese Funktion nicht selbst verwenden.
get_alle()	Es werden die JSON - Strukturen aktuell, stündlich und täglich bezogen und zurückgegeben. Dabei wird auch die Zeitzone in WiFi auf die Zone der angeforderten Koordinaten gesetzt. Wenn der Bezug nicht funktioniert, wird None zurückgegeben.
get_aktuell()	Es werden die JSON - Strukturen aktuell und stündlich bezogen und zurückgegeben. Dabei wird auch die Zeitzone in WiFi auf die Zone der angeforderten Koordinaten gesetzt. Wenn der Bezug nicht funktioniert, wird None zurückgegeben.
get_täglich()	Es wird die JSON - Struktur aktuell bezogen und zurückgegeben. Wenn der Bezug nicht funktioniert, wird None zurückgegeben.
refresh(force = False)	Der Datenbezug erfolgt über get_aktuell() . 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.

Funktion	Beschreibung
get_messzeit_lokal()	Es wird die Messzeit der letzten aktuellen Daten (<i>aktuell</i>) zurückgegeben.
get_messzeit_text()	Die Messzeit wird als String zurückgegeben.
get_zeit_lokal(zeit)	Die Python - Zeit zeit wird korrigiert und mit der Zeitzone angepasst. Falls zeit nicht mitgegeben wird, erfolgt die Rückgabe der aktuellen Zeit.

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.