

Lösungsblatt

Diese Fassung enthält die Lösungen.

Wie die Aufgaben aufgebaut sind

Standard ist das Ziel, fang immer dort an. **Einstieg** führt hin, falls du nicht reinkommst.

Experte geht weiter, wenn du fertig bist. Nicht jede Aufgabe hat alle drei Stufen.

Zum Nachschlagen liegen **Kärtchen** in der Tischmitte.

Heute wird erwartet

Zwischen zwei Nachrichten müssen mindestens **10 Sekunden** liegen. Schickst du früher die nächste, nimmt das Funkmodul sie nicht an.

Und die Bestätigung kommt nicht sofort. Erst abwarten, dann nachsehen, dann etwas ändern. In dieser Reihenfolge.

Aufgabe 1 – Die erste Nachricht

Standard

a) Die Bibliothek aufs Board bringen

Zuerst braucht das Board die Bibliothek, die mit dem Funkmodul spricht. Sie heißt `loramint` und ist ein ganzer Ordner, nicht nur eine Datei.

1. Öffne `loramint.de/workshop` und lade den Ordner `loramint` herunter.
2. Kopier ihn in Thonny auf das Board, so wie an Tag 3 die Datei `bme280.py`.
3. Prüf in der Kommandozeile, ob er angekommen ist:

```
>>> from loramint import LoRaMINT
```

Was passiert, wenn du dich beim Namen vertippst?

Lösung

Läuft die Zeile ohne Meldung durch, liegt der Ordner richtig.

Bei einem Tippfehler kommt ein `ImportError`, genau wie an Tag 3 bei `bme281`. Die Meldung heißt: Diese Bibliothek gibt es auf dem Board nicht.

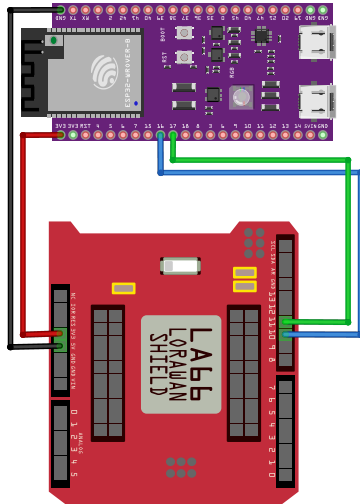
Anders als gestern ist es diesmal ein **Ordner** mit mehreren Dateien darin. Wer nur eine davon kopiert, bekommt dieselbe Meldung.

b) Den Sender anschließen

Der Aufbau von gestern bleibt stehen. Dazu kommen das **LA66-Shield** und die Antenne.

Erst die Antenne, dann der Strom

Ein Funkmodul, das ohne Antenne sendet, nimmt Schaden. Antenne anschrauben, danach das USB-Kabel einstecken. Nie umgekehrt.



Vier Drähte, mehr braucht es nicht

von	nach	Kabel
3V3	LA66 3V3	rot
GND	LA66 GND	schwarz
GPIO 17	LA66 11	grün
GPIO 16	LA66 10	blau

Zwei Drähte für den Strom, zwei für die Daten. Ein Breadboard braucht es nicht, die vier gehen direkt von Board zu Board.

Der Sensor von gestern

kann steckenbleiben. Gebraucht wird er erst in Aufgabe 2.

Über die beiden mittleren Drähte reden Board und Funkmodul miteinander. Sie sind **über Kreuz** gesteckt: Was das eine sendet, hört das andere.

Die Probe. Steck das USB-Kabel ein und tipp in die Kommandozeile:

```
>>> from loramint import LoRaMINT
>>> lora = LoRaMINT()
>>> lora.check_connection()
```

Was kommt zurück, und was bedeutet es?

Wenn False zurückkommt

Dann antwortet das Funkmodul nicht. Prüf in dieser Reihenfolge:

1. Sitzt jeder Draht fest, an beiden Enden?
2. Sind grün und blau vertauscht? Der häufigste Fehler, weil die beiden über Kreuz gehören.
3. Ist das LA66 mit 3V3 und GND verbunden?

Zwei Minuten suchen, dann Hand heben.

Lösung

Sitzt alles, meldet sich das Funkmodul mit seiner Version, und `check_connection` gibt `True` zurück. Das ist dieselbe Probe wie `i2c.scan()` an Tag 3: erst prüfen, ob das Bauteil überhaupt antwortet, und dann erst programmieren.

c) Das Programm verstehen und starten

Öffne `erste-nachricht.py`. Es besteht aus fünf Teilen. Geh sie einzeln durch und schreibe sie Stück für Stück ab.

```
from loramint import LoRaMINT
```

Holt die Bibliothek, wie gestern der `import`. Neu ist das `from`: Es holt nicht die ganze Bibliothek, sondern einen einzelnen Namen daraus. Deshalb steht später `LoRaMINT()` und nicht `loramint.LoRaMINT()`.

```
lora = LoRaMINT()
```

Öffnet die Leitung zum Funkmodul und legt es als `lora` ab. Dasselbe Muster wie gestern beim Sensor: erst anlegen, dann benutzen. Anschlüsse brauchst du keine anzugeben, die Bibliothek kennt sie.

```
if not lora.check_connection():  
    raise SystemExit("Abbruch: keine Verbindung zum LA66.")
```

`check_connection` fragt das Funkmodul, ob es da ist, und gibt `True` oder `False` zurück. `not` dreht das um: Bei `False` wird der eingerückte Teil ausgeführt. `raise SystemExit` beendet das Programm mit dieser Meldung, statt mit einem toten Funkmodul weiterzumachen.

```
if not lora.join():  
    raise SystemExit("Die Anmeldung hat nicht geklappt.")
```

`join` meldet die Node im Netz an. Das ist der langsame Teil, es kann mehrere Anläufe brauchen. Zurück kommt wieder `True` oder `False`. War die Node schon angemeldet, kommt `join` sofort zurück.

```
if lora.sendLog("Hallo aus Raum 101"):  
    print("Nachricht abgeschickt.")
```

`sendLog` verschickt einen Text, höchstens 140 Zeichen. Auch hier ein Rückgabewert: `True`, wenn das Funkmodul die Nachricht angenommen hat. Diesmal ohne `not`, denn gemeldet werden soll der Erfolg.

Jetzt starten. Das dauert. Schreib auf, was nacheinander in der Kommandozeile erscheint, und sieh danach im **LoRaMINT-Status-Fenster** nach, ob die Nachricht angekommen ist.

Lösung

Auf dem Bildschirm erscheint der Reihe nach: die Meldung von `check_connection`, dann "Melde mich im Netz an ...", nach einigen Sekunden "Angemeldet." und zuletzt "Nachricht abgeschickt."

Die Anmeldung ist der langsame Teil. Wer das Programm dabei neu startet, fängt von vorn an und verlängert es.

Im Status-Fenster steht der Text danach als Log-Eintrag. Bis er dort auftaucht, vergehen Sekunden. Das ist kein Fehler, das ist Funk.

Alle drei Funktionen geben `True` oder `False` zurück. Deshalb steht jede in einem `if`: Das Programm prüft nach jedem Schritt, ob es weitergehen kann. Bei `print` wäre das sinnlos, es gibt nichts zurück; das war der Unterschied aus Aufgabe 1 von Tag 3.

Experte

Schreib einen eigenen Text hinein, etwa "Hallo LoRaMINT" oder den Raum, in dem ihr sitzt. Schick ihn ab und sieh nach, ob er ankommt.

Keine Namen

Was ihr abschickt, landet in der Datenbank und bleibt dort. Namen und alles andere, was zu einer Person gehört, haben darin nichts zu suchen.

Zwei Grenzen

Höchstens **140 Zeichen**, und zwischen zwei Nachrichten **10 Sekunden** Pause.

Lösung

```
lora.sendLog("Hallo LoRaMINT")
```

Wer zwei Nachrichten hintereinander schickt, bekommt beim zweiten Mal `False` zurück: Das Funkmodul ist noch mit dem Empfangsfenster der ersten beschäftigt. Mit `time.sleep(10)` dazwischen geht es.

Warum keine Namen: Ein Log-Eintrag ist kein Chat. Er wird gespeichert, ist für alle mit Zugang sichtbar und lässt sich später nicht einzeln herausnehmen. Ein Raum ist eine Ortsangabe, ein Name eine Person.

Und noch eine Stufe weiter: Lass das Programm warten, bis jemand den **BOOT-Knopf** auf dem Board drückt, und erst dann senden.

```
from machine import Pin

knopf = Pin(0, Pin.IN, Pin.PULL_UP)
```

Gedrückt bedeutet `knopf.value() == 0`.

Lösung

```
print("Warte auf den Knopf ...")
while knopf.value() == 1:
    pass

lorawan.sendLog("Jetzt aber")
```

Die Schleife hat einen leeren Rumpf. `pass` heißt genau das: tu nichts, warte nur. Solange der Knopf nicht gedrückt ist, liefert `value()` eine Eins, und die Bedingung bleibt wahr.

Das ist dieselbe `while`-Schleife wie an Tag 2, nur dass die Bedingung diesmal nicht von einer Zählvariablen abhängt, sondern von der Außenwelt.

S044

S046

Aufgabe 2 – Ein eigener Messwert geht auf die Reise

Einstieg

Ein Log-Eintrag ist Text. Ein Messwert ist mehr: Zu der Zahl gehört, was sie bedeutet. Genau die Angaben, die gestern im Dictionary standen.

```
# Ein einzelner Messwert geht auf die Reise. Fertiges Beispiel ohne Sensor.

from loramint import LoRaMINT, MintValue

lora = LoRaMINT()

if not lora.check_connection():
    raise SystemExit("Abbruch: keine Verbindung zum LA66.")

if not lora.join():
    raise SystemExit("Die Anmeldung hat nicht geklappt.")

# Wert, Einheit, Ort, Messgroesse, Sensor
messwert = MintValue(21.5, "*C", "Raum 101", "Temperatur", "BME280")

if lora.sendValue(messwert):
    print("Messwert abgeschickt.")
```

Start das Programm und trage ein, wofür die fünf Angaben in den Klammern stehen.

Angabe	Wofür
--------	-------

21.5

"*C"

"Raum 101"

"Temperatur"

"BME280"

Lösung

Angabe	Wofür
--------	-------

21.5 der Messwert selbst

"*C" die Einheit

"Raum 101" der Ort, an dem gemessen wurde

"Temperatur" die Messgröße

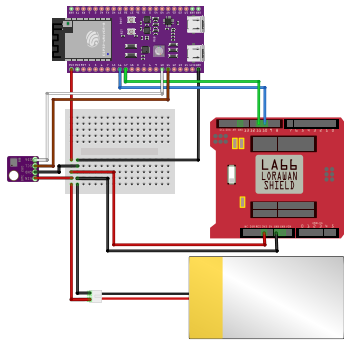
"BME280" der Sensor, von dem der Wert stammt

Die Reihenfolge ist festgelegt und lässt sich nicht vertauschen. Wer Ort und Messgröße verwechselt, bekommt keine Fehlermeldung, sondern einen Messwert mit falscher Beschriftung.

Das ist das Dictionary von gestern, nur in anderer Form. Dort stand der Name neben dem Wert, hier stehen fünf Angaben in einer festen Reihenfolge. Der Zweck ist derselbe: Eine Zahl allein sagt nichts.

Standard

Jetzt kommt der Sensor dazu. Er hängt wie gestern am Board, das Shield bleibt daneben.



Der vollständige Aufbau

am BME280	ans Board	Kabel
VIN	3V3	rot
GND	GND	schwarz
SDA	GPIO 10	weiß
SCL	GPIO 11	braun

Das ist derselbe Anschluss wie an Tag 3. Neu ist nur, dass Strom und Masse jetzt über das Breadboard verteilt werden, weil sich Sensor und Shield beides teilen.

Nimm dein Sensorprogramm von gestern und schick die gemessene Temperatur los, Schritt für Schritt:

1. Sensor anlegen wie gestern, mit `I2C(0, sda=Pin(10), scl=Pin(11))`.
2. Temperatur auslesen mit `read_compensated_data()`.
3. Auf eine Nachkommastelle runden.
4. Einen `MintValue` daraus bauen, mit deinem Ort.
5. Mit `sendValue` abschicken.

Lösung

eigener-messwert.py

```
from machine import Pin, I2C
import bme280
from loramint import LoRaMINT, MintValue

i2c = I2C(0, sda=Pin(10), scl=Pin(11))
sensor_bme = bme280.BME280(i2c=i2c)

lora = LoRaMINT()
lora.join()

temperatur, druck, feuchte = sensor_bme.read_compensated_data()

messwert = MintValue(round(temperatur, 1), "*C", "Raum 101",
                      "Temperatur", "BME280")

if lora.sendValue(messwert):
    print("Messwert abgeschickt.")
```

Gerundet wird **vor** dem Senden. Was einmal unterwegs ist, lässt sich nicht mehr ändern.

Der Ort ist deine Angabe und sollte stimmen: An Tag 5 ist er das Einzige, woran sich erkennen lässt, welcher Wert von wo kommt.

Experte

In der Projektbox liegt der **BH1750** von gestern, der Helligkeitssensor. Schick auch seinen Messwert los.

Was du dafür brauchst

Anschließen und auslesen wie an Tag 3, an dieselben zwei Datenleitungen. Die Bibliothek `bh1750` liegt schon auf dem Board. Zum Verschicken die fünf Schritte von oben.

Überleg vorher, was in die fünf Angaben des `MintValue` gehört. Die Einheit der Helligkeit ist **Lux**, abgekürzt `lx`.

Lösung

helligkeit-senden.py

```
from machine import Pin, I2C
import bh1750
from loramint import LoRaMINT, MintValue

i2c = I2C(0, sda=Pin(10), scl=Pin(11))
sensor_lux = bh1750.BH1750(i2c)

lora = LoRaMINT()
if not lora.join():
    raise SystemExit("Die Anmeldung hat nicht geklappt.")

lux = sensor_lux.luminance(bh1750.BH1750.ONCE_HIRES_1)

messwert = MintValue(round(lux, 1), "lx", "Raum 101",
                      "Helligkeit", "BH1750")
lora.sendValue(messwert)
```

Das Gerüst ist dasselbe wie in der Standardstufe. Ausgetauscht sind nur die Bibliothek, die Zeile zum Auslesen und die drei Angaben, die den Messwert beschreiben: Einheit "`lx`", Messgröße "`Helligkeit`", Sensor "`BH1750`".

Wer beide Werte schicken will, muss zwischen die zwei Aufrufe eine Pause von mindestens zehn Sekunden legen. Sonst nimmt das Funkmodul den zweiten nicht an, und `sendValue` liefert `False`.

Beide Sensoren hängen an denselben zwei Leitungen und teilen sich ein einziges `i2c`. Das war die Erkenntnis von gestern.

S041

S042

S043

Aufgabe 3 – Immer weiter messen

Standard

Bisher schickt dein Programm einen Wert und ist fertig. Für die Langzeitmessung muss es von allein weitermachen: messen, senden, warten, wieder von vorn.

Die Schleife kennst du von Tag 2. Neu ist nur das Warten:

```
import machine

machine.lightsleep(60000)
```

Die Zahl sind **Millisekunden**, hier also eine Minute. Bau damit eine Schleife um dein Programm aus Aufgabe 2, sodass es jede Minute einen Messwert schickt.

Was in die Schleife gehört und was nicht

Anmelden ist einmalig. Was vor der Schleife steht, passiert einmal; was darin steht, jede Minute.

Lösung

dauermessung.py

```
from machine import Pin, I2C
import machine
import bme280
from loramint import LoRaMINT, MintValue

i2c = I2C(0, sda=Pin(10), scl=Pin(11))
sensor_bme = bme280.BME280(i2c=i2c)

lora = LoRaMINT()
if not lora.join():
    raise SystemExit("Die Anmeldung hat nicht geklappt.")

while True:
    temperatur, druck, feuchte = sensor_bme.read_compensated_data()

    messwert = MintValue(round(temperatur, 1), "*C", "Raum 101",
                          "Temperatur", "BME280")
    lora.sendValue(messwert)

    machine.lightsleep(60000)
```

import, i2c, sensor_bme, lora und join stehen **vor** der Schleife. Wer die Anmeldung hineinzieht, meldet sich jede Minute neu an, und das dauert jedes Mal.

Häufiger Fehler: das Einrücken. Was sich wiederholen soll, steht eingerückt unter while, das Warten am Ende gehört dazu.

Experte

Warum eigentlich `lightsleep` und nicht `time.sleep(60)`? Probier beides aus und schreib auf, was du bemerkst.

Lösung

	Strom	Was danach passiert
<code>time.sleep(60)</code>	einige zehn mA	Das Programm läuft weiter.
<code>machine.lightsleep(60000)</code>	etwa 1 mA	Das Programm läuft weiter.
<code>machine.deepsleep(60000)</code>	10 bis 20 μ A	Das Board startet neu .

`time.sleep` hält den Chip die ganze Minute wach, er tut nur nichts. `lightsleep` schaltet ab, was gerade nicht gebraucht wird, kehrt aber zurück: Der Arbeitsspeicher bleibt, die Schleife läuft weiter.

`deepsleep` kehrt **nicht** zurück. Das Board startet neu und läuft die Datei von oben, der Neustart ist der nächste Durchlauf. Eine Schleife braucht es dann gar nicht mehr.

Und deshalb steht es hier nicht: Beim Neustart verschwindet der USB-Anschluss, und Thonny verliert die Verbindung. Für eine Node, die allein im Treppenhaus steht, ist `deepsleep` richtig; solange ihr zuschaut, ist es `lightsleep`.

S045

Anhang • Senden mit LoRaWAN

Alles von den Kärtchen noch einmal, zum Mitnehmen und Nachschlagen.

Verkabelung

LA66-Shield			BME280		
3V3	3V3	rot	VIN	3V3	rot
GND	GND	schwarz	GND	GND	schwarz
GPIO 17	11	grün	SDA	GPIO 10	weiß
GPIO 16	10	blau	SCL	GPIO 11	braun

Erst die Antenne, dann der Strom. Grün und blau gehen **über Kreuz**: Was das eine Board sendet, hört das andere.

Anmelden und senden

```
from loramint import LoRaMINT

lora = LoRaMINT()
lora.check_connection()
lora.join()
lora.sendLog("Hallo LoRaMINT")
```

Zeile	Was sie tut
LoRaMINT()	öffnet die Leitung zum Funkmodul
check_connection()	prüft, ob das Modul antwortet
join()	meldet die Node im Netz an
sendLog()	schickt einen Text, höchstens 140 Zeichen

Alle vier geben True oder False zurück. Die Anmeldung gilt weiter, auch nach einem Neustart des Programms. Zwischen zwei Nachrichten liegen **10 Sekunden**.

Einen Messwert verschicken

```
messwert = MintValue(21.5, "*C", "Raum 101", "Temperatur", "BME280")

lora.sendValue(messwert)
```

Die fünf Angaben in dieser Reihenfolge:

	Bedeutung	höchstens
21.5	der Messwert	
"*C"	die Einheit	10 Zeichen
"Raum 101"	der Ort	30 Zeichen
"Temperatur"	die Messgröße	15 Zeichen
"BME280"	der Sensor	10 Zeichen

Ist eine Angabe zu lang, steht später "too long" an ihrer Stelle. Vertauscht gibt es keine Fehlermeldung, nur eine falsche Beschriftung.

Den Sensor auslesen

```
from machine import Pin, I2C
import bme280

i2c = I2C(0, sda=Pin(10), scl=Pin(11))
sensor_bme = bme280.BME280(i2c=i2c)

temperatur, druck, feuchte = sensor_bme.read_compensated_data()
```

Reihenfolge: Temperatur, Druck, Feuchte. Der Druck kommt in Pascal, `druck / 100` ergibt hPa.

Der zweite Sensor hängt am selben i2c:

```
import bh1750

sensor_lux = bh1750.BH1750(i2c)
lux = sensor_lux.luminance(bh1750.BH1750.ONCE_HIRES_1)
```

Immer weiter messen

```
import machine

while True:
    <messen und senden>
    machine.lightsleep(60000)
```

Die Zahl sind **Millisekunden**. Alles, was einmal passieren soll, steht **vor** der Schleife: die `import`-Zeilen, `i2c`, der Sensor, `lora` und `join`.

	Strom	Was danach passiert
<code>time.sleep(60)</code>	einige zehn mA	Das Programm läuft weiter.
<code>machine.lightsleep(60000)</code>	etwa 1 mA	Das Programm läuft weiter.
<code>machine.deepsleep(60000)</code>	10 bis 20 µA	Das Board startet neu.

Wenn nichts ankommt

Was du siehst	Was zu tun ist
<code>check_connection</code> liefert <code>False</code> <code>join</code> liefert <code>False</code>	Drähte prüfen. Sind grün und blau vertauscht? Sie gehören über Kreuz. Sitzt die Antenne? Stimmen die Zahlen aus der Registrierung? Noch einmal versuchen, nicht neu starten.
<code>sendLog</code> oder <code>sendValue</code> liefert <code>False</code> <code>ImportError</code>	Zu schnell hintereinander gesendet. Zehn Sekunden warten. Die Bibliothek liegt nicht auf dem Board. Bei <code>loramint</code> ist es ein ganzer Ordner, nicht eine Datei.
Alles läuft, im Status-Fenster steht nichts	Erst abwarten. Bis eine Nachricht dort auftaucht, vergehen Sekunden bis Minuten.

Zwei Minuten suchen, dann Hand heben.