Workshop Internet of Things

In diesem Workshop lernen Sie etwas über die Programmiersprache Python und Anwendungen im Internet der Dinge. Doch was heisst das? Das heisst soviel wie das Dinge miteinander über das Internet kommunizieren. Mit Dingen sind in der Regel Sensoren oder ganz kleine Computer gemeint. Das WiPy Board welches wir in diesem Kurs verwenden, ist genau ein solches Ding. Was nun diese Dinge genau austauschen und machen, ist Sache des Programmierers. Anhand von Programmen kann man den Dingen sagen, was Sie zu tun haben. In diesem Kurs werden wir Ihnen einen Eindruck geben, was solche Dinge in etwa können und wie ein solches Programm aussieht.

# Python

Python-Interpreter gibt es mittlerweile für diverse Plattformen, von der Waschmaschine bis hin zum NASA Satelliten. Natürlich auch auf jedem Notebook und Computer. Gleichzeitig ist die Python Programmiersprache sehr einfach zu lernen. Dadurch ist Sie sehr gut für den Einstieg aber auch für Professionelle Entwickler geeignet. Diese Eigenschaften machen Python optimal für die Entwicklung von IoT (Internet of Things) Anwendungen.

# Das WiPy Board

Das WiPy Board von der Firma pycom ist ein sogenanntes Mikrocontroller Board welches sich in der Python Programmiersprache programmieren lässt. Das heisst nichts Anderes als, dass es sich dabei um einen kleinen Computer handelt, der Python Programme versteht. Der Unterschied zu einem Handelsüblichen Computer ist nur, dass es nicht so reichhaltig mit Zubehör ausgestattet ist. Zum Beispiel hat es keinen Bildschirm, keine Festplatte mit mehreren Gigabytes an Kapazität. Dafür ist es aber sehr Energieeffizient (Laufdauer von einigen Monaten mit nur einer Batterie) und preiswert. Das erlaubt den Einsatz in IoT Anwendungen, wohingegen es für Surfen oder Spiele spielen ungeeignet ist.

Hier ein kleiner Auszug der Leistungsdaten des Boards:

Speicherplatz: 4MB  
Arbeitsspeicher: 512 KB  
Prozessorgeschwindigkeit: 160 MHz  
Drahtlos Schnittstellen: WiFi, Bluetooth Low Energy

Da das Board ein WiFi Modul besitzt lässt es sich mit dem Internet verbinden und Daten austauschen. Also ein perfektes Thing für das Internet! ☺

# Programmierung / Ausführung von Programmen

Python Programme können auf zweit Arten ausgeführt werden. Entweder im interkativen Modus oder als Skripte. Bei interkativen Modus spricht man sozusagen direkt mit Python und führt eine Anweisung nach der Anderen aus. Den interaktive Interpreter erkennen Sie an den drei grösser als Zeichen (>>>). Eine sogenannt interaktive Session eignet sich am besten um einzelne Befehle zu testen. Sobald Sie etwas grössere Programme schreiben eignet sich der Skript Modus besser.

Skripte sind ganz einfach Dateien in denen die Python anweisungen gespeichert sind. Das heisst die Abfolge von Anweisungen die Sie in den Interpreter eingeben, können Sie auch in einer Datei abspeichern und so später erneut aufrufen.

# Das erste Programm

Traditionell heisst das erste Programm, das Sie in einer neuen Sprache schreiben, »Hallo,Welt!« – weil es einfach nur die Worte »Hallo, Welt!« ausgibt. In Python sieht das folgendermaßen aus:

>>> print('Hallo, Welt!')

Das ist ein Beispiel für eine print-Funktion, die in Wahrheit natürlich nichts »druckt«. Sie zeigt den Wert einfach auf dem Bildschirm an. In diesem Fall lautet das Ergebnis Hallo, Welt! Die Apostrophe in der Programmanweisung kennzeichnen den Anfang und das Ende des anzuzeigenden Texts und erscheinen nicht im Ergebnis. Mann kann aber auch Zahlen ausgeben, z.B eine 2.

>>> print(2)

Da es sich dabei um eine Zahl, und nicht einen String handelt, benötigt es keine Apostrophe.

# Variablen

Ein Wert ist eines jener grundlegenden Dinge, mit denen ein Programm arbeitet – wie etwa ein Buchstabe oder eine Zahl. Die Werte, denen wir bisher begegnet sind, lauten 2 und 'Hallo, Welt!'. Diese Werte gehören verschiedenen Typen an: 2 ist ein Integer (eine ganze Zahl), und 'Hallo, Welt!' ist ein String, eine Folge von Zeichen. Sie (und der Interpreter) erkennen Strings daran, dass sie in Apostrophe eingefasst werden. Falls Sie sich nicht sicher sind, zu welchem Typ ein Wert gehört, kann Ihnen der Interpreter das verraten:

>>> type('Hallo, Welt!')

<class 'str'>

>>> type(17)

<class 'int'>

Es wird Sie nicht überraschen, dass Strings zum Typ str und Zahlen zum Typ int gehören. Dass Zahlen mit einem Dezimalpunkt gehören zum Typ float. Der Name dieses Typs kommt daher, dass Dezimalbrüche als Fliesskommazahlen (engl. floating-point) dargestellt werden.

>>> type(3.2)

<class 'float'>

Was ist mit Werten wie '17' und '3.2'? Sie sehen aus wie Zahlen, stehen aber in Apostrophen, genau wie Strings.

>>> type('17')

<class 'int'>

>>> type('3.2')

<class 'str'>

Es sind Strings.

# Arbeiten mit Variablen

Eine der leistungsfähigsten Funktionen einer Programmiersprache ist die Fähigkeit, mit Variablen zu arbeiten. Ein Variablenname ist dabei ein Name, der sich auf einen Wert bezieht.

Durch die Zuweisung wird eine neue Variable erstellt, und ihr wird ein Wert zugewiesen:   
>>> meldung = 'Und jetzt etwas ganz anderes'

>>> n = 17

>>> pi = 3.1415926535897932

In diesem Beispiel erfolgen drei Zuweisungen. In der ersten wird einer neuen Variablen mit dem Namen meldung ein String zugewiesen. In der zweiten wird der Integer 17 an n übergeben. Und in der dritten Zuweisung wird der (ungefähre) Wert von π der Variablen pi zugewiesen.

Üblicherweise wählen Programmierer für ihre Variablen aussagekräftige Namen (auch Bezeichner genannt) – damit zu erkennen ist, wofür die Variable verwendet wird. Variablennamen können beliebig lang sein und dürfen sowohl Buchstaben als auch Zahlen enthalten, müssen aber mit einem Buchstaben beginnen. Es ist auch zulässig, Grossbuchstaben zu verwenden, allerdings ist es besser, Variablennamen mit einem Kleinbuchstaben zu beginnen. Der Unterstrich \_ darf ebenfalls in Variablennamen vorkommen. Er wird häufig für Namen verwendet, die aus mehreren Buchstaben bestehen, zum Beispiel mein\_name oder geschwindigkeit\_einer\_unbeladenen\_schwalbe.

# Operatoren und Operanden

Operatoren sind spezielle Symbole, die Berechnungen darstellen, wie etwa Addition und Multiplikation. Die Werte, auf die der Operator angewendet wird, nennt man Operanden. Die Operatoren +, -, \*, / und \*\* stehen für Addition, Subtraktion, Multiplikation, Division und Potenzen:

20+32

stunde-1

stunde\*60+minute

minute/60

5\*\*2

(5+9)\*(15-7)

Testen Sie die verschiedenen mathematischen Operationen.

# Ausdrücke und Anweisungen

Ein Ausdruck kann eine Kombination aus Werten, Variablen und Operatoren sein. Ein einzelner Wert stellt ebenso einen Ausdruck dar, genauso wie eine Variable. Insofern sind alle folgenden Ausdrücke zulässig (unter der Voraussetzung, dass der Variablen x ein Wert zugewiesen wurde):

17

x

x + 17

Eine Anweisung ist ein Codeteil, den der Python-Interpreter ausführen kann. Wir kennen bisher zwei Arten von Anweisungen: Zuweisungen und print(). Technisch gesehen, ist ein Ausdruck ebenfalls eine Anweisung. Aber wahrscheinlich ist es einfacher, sich die beiden als zwei verschiedene Dinge vorzustellen. Der wichtigste Unterschied ist, dass ein Ausdruck einen Wert hat, eine Anweisung dagegen nicht.

# Operationen mit Text

Im Allgemeinen können Sie keine mathematischen Operationen mit Strings durchführen, selbst wenn diese wie Zahlen aussehen. Die folgenden Ausdrücke sind daher nicht zulässig:

'2'-'1'

'eier'/'leicht'

'drittel'\*'Ein Zauberspruch'

Der Operator + funktioniert mit Strings, macht aber nicht das, was Sie sich vielleicht vorstellen: Sie führen damit eine Konkatenation durch, d. h., die Strings werden aneinander angehängt. Ein Beispiel:

erster = 'donner'

zweiter = 'gurgler'

print(erster + zweiter)

Die Ausgabe dieses Programms lautet donnergurgler. Der Operator \* funktioniert ebenfalls mit Strings: Er wiederholt den angegebenen String. So ergibt

'Spam'\*3

beispielsweise 'SpamSpamSpam'. Wenn einer der Operanden ein String ist, muss der andere ein Integer sein.

# Kommentare

Wenn Programme grösser und komplizierter werden, sind sie oft auch unübersichtlich. Formale Sprachen haben eine hohe Dichte, daher ist es oft schwierig, einem Codeteil anzusehen, was er macht und warum. Aus diesem Grund ist es am besten, Ihre Programme mit Notizen zu versehen, die in einer natürlichen Sprache erklären, was das Programm macht. Solche Notizen nennt man Kommentare. Sie beginnen mit dem Symbol #:

# Berechnen, wie viel Prozent der aktuellen Stunde abgelaufen sind

prozentsatz = (minute \* 100) / 60

In diesem Fall steht der Kommentar in einer eigenen Zeile. Sie können aber auch Kommentare ans Zeilenende schreiben:

prozentsatz = (minute \* 100) / 60 # Prozentsatz der aktuellen Stunde

Alles vom # bis zum Zeilenende wird ignoriert und hat keine Auswirkung auf das Programm. Kommentare sind besonders dann nützlich, wenn Sie damit Details zum Code erläutern, die nicht offensichtlich sind. Normalerweise können Sie davon ausgehen, dass der Leser erkennt, was der Code macht. Es ist wesentlich sinnvoller, zu erklären, warum Sie das entsprechend gelöst haben. Aussagekräftige Variablennamen können den Bedarf an Kommentaren minimieren.

## Übungen

1. Angenommen, wir führen die folgenden Zuweisungsanweisungen aus:

breite = 17

hoehe = 12.0

wort = „spam“

Schreiben Sie für jeden der folgenden Ausdrücke Wert und Typ (des Werts des Ausdrucks) auf:

1. breite/2

2. breite/2.0

3. hoehe/3

4. 1 + 2 \* 5

5. wort \* 5

Überprüfen Sie Ihre Antworten mit dem Python-Interpreter.

## Übungen

Üben Sie sich im Gebrauch des Python-Interpreters als Rechner:

1. Der Flächeninhalt eines Rechteckes mit den Seitenlängen 3.5 cm und 4.5 cm ist 15.75 cm2. Wie groß ist die Fläche eines Rechtecks mit den Seitenlängen 4.2 cm und 4.7 cm?
2. Angenommen, der Verkaufspreis für ein Buch beträgt 24,90 CHF. Buchhändler erhalten einen Rabatt von 40 Prozent. Was ist der Verkaufspreis inkl. Rabatt?
3. Wenn ich um 6:52 Uhr das Haus verlasse, einen Kilometer bei langsamem Tempo laufe (5:07 pro km) und drei Kilometer etwas schneller laufe (4:28 pro km), um wie viel Uhr komme ich dann zum Frühstück nach Hause?

# Funktionen

Im Kontext eines Programms ist eine Funktion eine benannte Folge von Anweisungen, die eine Berechnung durchführen. Wenn Sie eine Funktion definieren, geben Sie einen Namen und die entsprechenden Anweisungen vor. Später können Sie dann diese Funktion über ihren Namen »aufrufen«. Wir haben bereits ein Beispiel für einen Funktionsaufruf gesehen:

>>> print(„Hallo Welt!“)

Der Name der Funktion ist print. Den Ausdruck in Klammern nennt man das Argument der Funktion. Die print Funktion hat keinen Rückgabewert. Man spricht üblicherweise davon, dass eine Funktion ein Argument »erwartet« und ein Ergebnis »zurückliefert«. Das Ergebnis bezeichnet man auch als Rückgabewert.

# Eigene Funktionen

Wir haben bislang nur jene Funktionen verwendet, die in Python enthalten sind. Es ist aber auch möglich, neue eigene Funktionen hinzuzufügen. Dazu muss man die gewünschte Funktion definieren. Eine Funktionsdefinition gibt den Namen einer neuen Funktion sowie die Reihe von Anweisungen an, die beim Aufruf der Funktion ausgeführt werden sollen. Hier ein Beispiel:

def zeige\_text(): # Funktionsname

print(„Willkommen beim INS“) # 1. Anweisung

print(„Python macht Spass!“) # 2. Anweisung

def kennzeichnet eine Funktionsdefinition und sagt Python quasi „Achtung jetzt kommt die Definition einer eigenen Funktion“. Der Name der Funktion aus dem Beispiel lautet zeige\_text. Die Regeln für Funktionsnamen sind die gleichen wie für Variablennamen: Buchstaben, Zahlen und einige Interpunktionszeichen sind zulässig, aber das erste Zeichen darf keine Zahl sein. Zudem sollten Sie vermeiden, für eine Funktion und eine Variable denselben Namen zu verwenden.

Die leeren Klammern nach dem Namen zeigen an, dass diese Funktion keine Argumente erwartet. Die erste Zeile der Funktionsdefinition bezeichnet man als Header, den Rest als Body. Der Header muss mit einem Doppelpunkt enden, und der Body muss eingerückt sein. Per Konvention muss der Body immer um vier Leerzeichen eingerückt sein. Der Body kann eine beliebige Anzahl von Anweisungen enthalten.

Wie kann man die nun definierte Funktion aufrufen? Ganz einfach:

>>> zeige\_text()

## Übungen

1. Erstellen Sie eine Funktion die Ihren Namen ausgibt
2. Erstellen Sie eine Funktion die Folgendes ausgibt:

--------------------------------------------------  
INS Institute for networked solutions  
Informatics 4 Girls  
Mein Name ist <Ihr Name>  
--------------------------------------------------

Tipps: Die Striche bestehen aus 40 Minuszeichen. Verwende den \* Operator.

# Eingaben

Das Gegenstück zur print() Funktion ist die input() Funktion. Der Rückgabewert ist die eingegebene Zeichenfolge. Als Argument nimmt die Funktion einen String welcher vor dem Einlesen der Ausgabe ausgegeben wird. Folgender Aufruf Fragt nach dem Alter und speichert die Eingabe in der Variablen alter.

>>> alter = input('Bitte geben Sie ihr alter ein: ')

Beachten Sie, dass die nach dem Eintippen ihres Alters, die Eingabe mit der Eingabe-Taste (Enter) abschliessen müssen. Anschliessen können Sie die Eingabe in ihrem Programm weiter verwenden.  
Mit

>>> print(alter)

können Sie die Eingabe ausgeben.

## Übungen

1. Erstellen Sie eine Funktion bmi(), welches ihr Gewicht in kg und ihre Körpergrösse in cm abfragt und danach den BMI (Body Mass Index) berechnet und ausgibt. Der BMI berechnet sich wie folgt: Gewicht geteilt durch Grösse im Quadrat (gewicht/groesse\*\*2).

# LED ansprechen

Funktionen sind üblicherweise in sogenannten Modulen gespeichert und nach Zweck Zusammengefasst. So gibt es Beispielsweise ein Modul math für mathematische Funktionen. Für das WiPy Board gibt es ein eigenes Modul mit dem Namen pycom, welches Funktionen für die Ansteuerung der internen LED beinhaltet. Ein Modul wird mittels import Anweisung geladen:

>>> import pycom

Sobald Sie den Befehl ausgeführt haben, können Sie die darin enthaltenen Funktionen verwenden. Wie Sie bereits bemerkt haben, blinkt die LED Standardmässig in Regelmässigen abständen in Blau. Dies soll einen Heartbeat, zu Deutsch Herzschlag, des Boards darstellen. Um den Heartbeat auszuschalten führen Sie folgenden Befehl aus:

>>> pycom.heartbeat(False)

Danach lässt sich die Farbe der LED beliebig einstellen. Die Funktion dazu heisst rgbled. Als Parameter der Funktion gibt man dazu den RGB Farbcode in hexadezimal ein. RGB steht für Rot Grün Blau. Die Grundfarben. Für jeden dieser Farbkanäle kann man die Helligkeit von 00 bis FF im Format 0xRRGGBB einstellen und so die verschiedenen Farbwerte zusammen mischen. Hier einige Werte:

|  |  |
| --- | --- |
| Farbe | Hex |
| Rot | 0xFF0000 |
| Grün | 0x00FF00 |
| Blau | 0x0000FF |
| Ausschalten | 0x000000 |
| Weiss | 0xFFFFFF (alle Farben gleichzeitig) |
| Gelb | 0xFFFF00 (Mischung aus Rot und Grün) |

Natürlich lassen sich auch Farben mischen, z.B Gelb, eine Mischung aus Rot und Grün: 0xFFFF00. Um die LED Rot leuchten zulassen rufen Sie die Funktion wie folgt auf:

>>> pycom.rgbled(0xff0000)

Wenn Sie bei Google nach „color picker“ Suchen erhalten Sie einen Farbselektor, mit dem Sie visuell den Farbcode für die gewünschte Farbe auswählen können. Achten Sie darauf, dass der Code jeweils mit vorangestellter Route angezeigt wird. Sie den Code aber mit vorangestelltem 0x in die Funktion eingeben müssen. Also wenn der Farbcode #4286f4 ist, müssen Sie die Funktion wie folgt aufrufen:

>>> pycom.rgbled(0x4286f4)

Im Modul time gibt es eine Funktion sleep welche eine gewisse Zeit (in Sekunden) wartet und erst dann mit der Nachfolgenden Instruktion weiterfährt. Dadurch lässt sich einfach ein Blinklicht programmieren.

Um die LED zuerst für 1 Sekunde Blau leuchten zu lassen und danach für 1 Sekunde Rot und am Schluss wieder auszuschalten, können Sie eine Funktion wie folgt definieren:

import time

def blau\_rot():

pycom.rgbled(0x0000FF)

time.sleep(1)

pycom.rgbled(0xFF0000)

time.sleep(1)

pycom.rgbled(0x000000)

Durch den Aufruf von

>>> blau\_rot()

Leuchten die LEDs wie gewünscht. Sie können anstelle der time.sleep Funktion auch die Funktion time.sleep\_ms verwenden um nur eine gewisse Anzahl in Millisekunden zu warten und so die LEDs schneller blinken lassen. Eine Sekunde hat 1000 Millisekunden. Eine halbe Sekunde 500 Millisekunden.

## Übungen

1. Versuchen sie die Farbwerte für Pink und Orange durch ausprobieren herauszufinden (aus welchen Grundfarben bestehen Sie?) Benennen Sie die Funktion Orange Pink um und passen Sie die Wartezeiten so an, dass die LEDs alle halbe Sekunde ihre Farbe wechseln.
2. Fügen Sie der Funktion noch eine dritte Farbe hinzu.

# Parameter und Argumente

Wie wir gesehen haben, erfordern einige eingebaute Funktionen Argumente. Wenn Sie beispielsweise time.sleep aufrufen, übergeben Sie eine Zahl als Argument, die Zeit in Sekunden wie lange gewartet werden soll. Manche Funktionen erwarten auch mehr als ein Argument. Innerhalb der Funktion werden die Argumente entsprechenden Variablen zugewiesen, den Parametern. Hier sehen Sie ein Beispiel für eine benutzerdefinierte Funktion, die ein Argument erwartet:

def print\_zweimal(name):

print(name)

print(name)

Diese Funktion weist das Argument einem Parameter mit dem Namen name zu. Wenn sie aufgerufen wird, gibt sie den Wert des Parameters zweimal aus.

## Übungen

1. Erstellen Sie eine Funktion mit dem Header def flaeche\_rechteck(breite, hoehe), welche die 2 Parameter Breite und Höhe annimmt und die Fläche des entsprechenden Rechteckes ausgibt.

# Ins Internet mit Python

Mit nachstehenden Befehlen können Sie sich mit dem WiFi der HSR Verbinden:

>>> import network

>>> import time

>>> wlan = network.WLAN(mode=network.WLAN.STA, timeout=2000)

>>> wlan.connect(“I4G”)

Anschliessen können Sie mit dem Befehl

>>> wlan.isconnected()

überprüfen, ob die Verbindung geklappt hat. Ist der Rückgabewert True (für wahr) hat die Verbindung geklappt. Ist der Wert False (für falsch) nicht. Falls es nicht geklappt hat, warten Sie einen Augenblick und überprüfen Sie den Status noch einmal oder geben Sie die Anweisungen nochmals ein.

# Chat mit Kollegen

In diesem Abschnitt erstellen wir einen Chat mit allen Teilnehmern. Als erstes benötigen wir etwas Code um uns im Chatraum „I4G“ anzumelden:

import simple, \_thread, time

name = „Dein Name“

client = simple.MQTTClient(name, "broker.hivemq.com")

def printer(t, m):  
 print(‘%s: %s’ % (t.decode(),m.decode()))

client.set\_callback(printer)

client.connect()

client.subscribe(“/I4G/#”)

def consumer():  
 while True:  
 client.check\_msg()  
 time.sleep\_ms(500)

*\_thread.start*\_new\_thread(consumer, ())

Danach können Sie mittels

>>> client.publish("/I4G/"+name, "Nachricht!")

Nachrichten ihren Kolleginnen senden. Den “Nachricht!” Teil können Sie beliebig ändern.

# Weather Beacon Applikation

Wir haben für Sie eine Applikation vorbereitet welche die Wetterprognose für den aktuellen Tag anhand der LED Farbe anzeigt. So können sie das Board beispielsweise in der Nähe ihres Regenschirmes oder der Haustüre platzieren und wissen so beim Verlassen des Hauses, was für Wetter Sie in den nächsten Stunden erwartet und können so entscheiden, ob es sich lohnt einen Schirm einzupacken.

|  |  |
| --- | --- |
| Farbe | Prognose |
| Blau | Regen |
| Weiss | Schnee |
| Orange | Sonne |

Die Anwendung bezieht die Wetterprognosen aus dem Internet über eine WLAN Verbindung. Dazu müssen Sie das Board mit ihrem WLAN zu Hause verbinden und Namen ihres WLANs und das allfällige Passwort konfigurieren. Lesen Sie dazu den Abschnitt „Konfigurationsmodus“.

## LED Status

Sobald Sie das Board mit Strom versorgen wird die Weather Beacon Applikation gestartet. Während dem Starten wird der Status des Gerätes anhand der LED angezeigt.

|  |  |
| --- | --- |
| Farbe | Status |
| Blau | Verbindung mit WLAN wird hergestellt |
| Grün | Verbindung mit WLAN wurde erfolgreich hergestellt |
| Rot | Fehler beim Verbinden mit dem WLAN oder Fehler beim Abrufen der Wetterdaten |
| Pink | Neue Wetterdaten werden aus dem Internet abgerufen |

## Konfigurationsmodus

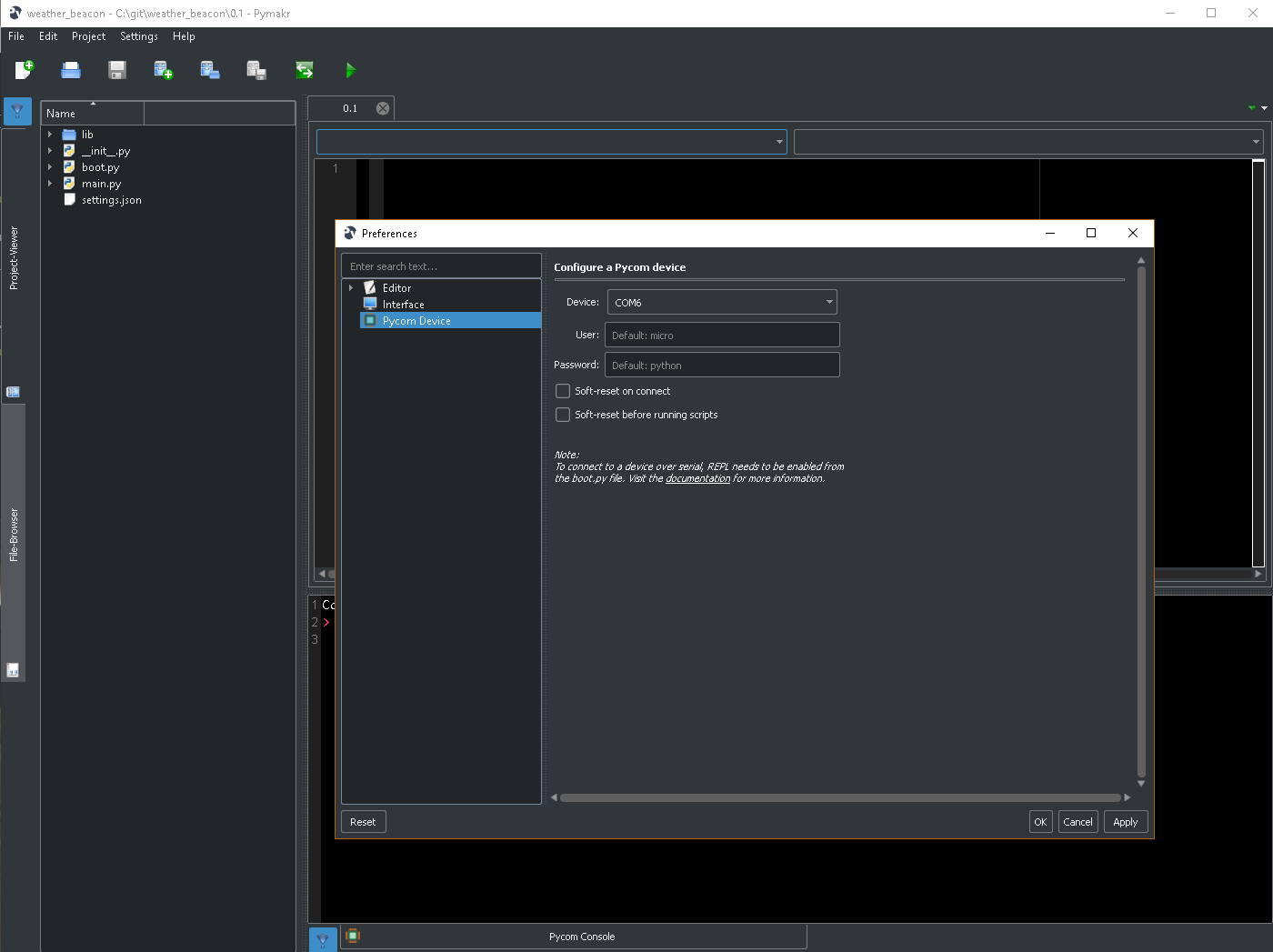


Reset

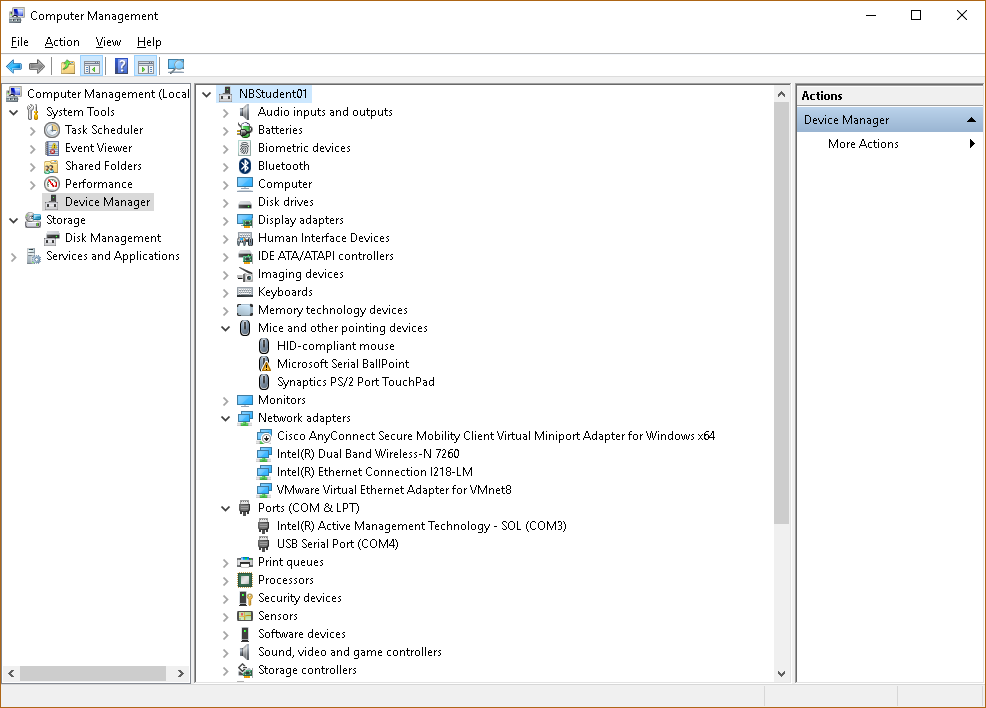
Menü

Um Konfigurationseinstellungen vorzunehmen, oder den Start der Weather Beacon Applikation zu unterbinden und eine REPL für eigene Programme zustarten, besitzt das Board einen Konfigurationsmodus. Diesen erreichen Sie wie folgt:

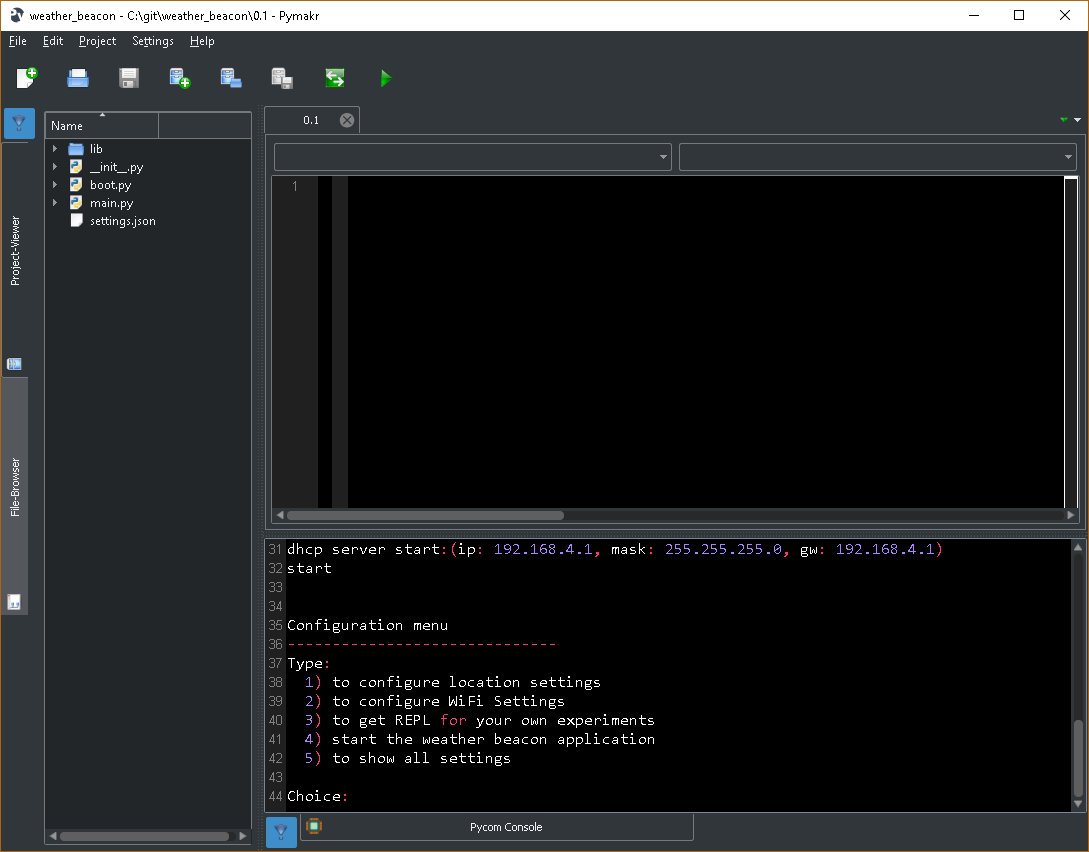
1. Installieren Sie die Pymakr IDE, welche Sie auf <https://www.pycom.io/support/supportdownloads/> herunterladen können.
2. Starten Sie die Pymakr IDE
3. Verbinden Sie das Board mit ihrem Computer
4. Wählen Sie unter Settings/Preferences/Pycom Device das richtige Gerät aus (unter Windows ist dies ein COM Port).



Falls Sie nicht wissen, welches das richtige Gerät ist, können sie im Windows Gerätemanager (Windows-Taste drücken und danach nach Gerätemanager suchen) nachsehen. Es ist das Gerät mit dem Namen „USB Serial Port“. In der Klammer sehen Sie den Namen der COM Schnittstelle.



1. Drücken Sie den Reset Button auf dem WiPy Board.
2. Halten Sie den Menü Button gedrückt, bis Sie im unteren Teil der Pymakr IDE das Auswahlmenü erhalten.



1. Wählen Sie die gwünschte Option (1-5) indem Sie die Zahl des Menüeintrages eingeben und mit Enter bestätigen.

## Bemerkungen

Damit das Weather Beacon einwandfrei Funktioniert müssen Sie das WiFi Konfigurieren und ihren Standort (Location) anpassen.

Falls das Board erfolgreich mit ihrem WLAN verbinden ist, sehen Sie das ebenfalls im Terminal im unteren Teil der IDE, oder durch dreifaches blinken der LED in blau

Falls keine Verbindung hergestellt werden konnte, blinkt die LED in rot. Starten Sie in diesem Fall das Board neu und prüfen Sie die Einstellungen.

Sie können natürlich das Board jederzeit für andere Experimente einsetzen und eigene Programme entwickeln. Wir möchten Sie sogar dazu ermutigen! Sie können jederzeit die Weather Beacon Applikation wieder auf dem Board installieren.

# Links

Dokumentation zu verfügbaren Funktionen des WiPy Boards: <http://docs.pycom.io>

Alles Material das wir in diesem Kurs verwendet haben, finden Sie in unserem Github Repository zum Download: <https://github.com/HSRNetwork/weather_beacon>

# Quellen

Teile des Inhalts wurden zum Teil direkt oder in abgeänderter Form von [www.pycom.io](http://www.pycom.io) und dem Buch „Programmieren lernen mit Python“ von Allen B. Downey übernommen.