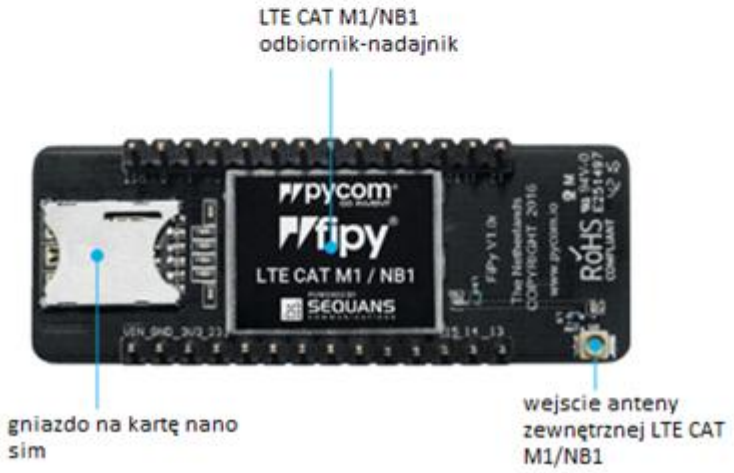
Przygotowanie środowiska do programowania urządzenia;

Zaprogramowanie mikrokontrolera do transmisji danych w sieci Internetu Rzeczy. Główne cele mikrokontrolera: pobranie pomiaru przy pomocy odpowiedniej biblioteki, utworzenie gniazda, wysłanie danych do serwera oraz zapisanie ich w bazie danych. Serwer ma wysyłać dane dotyczące parametrów pomiarów: ilość pomiarów oraz czas pomiędzy nimi.

Realizacja:

1. Połączenie urządzenia przez kabel usb z komputerem i ustanowienie komunikacji z oprogramowanie Atom.

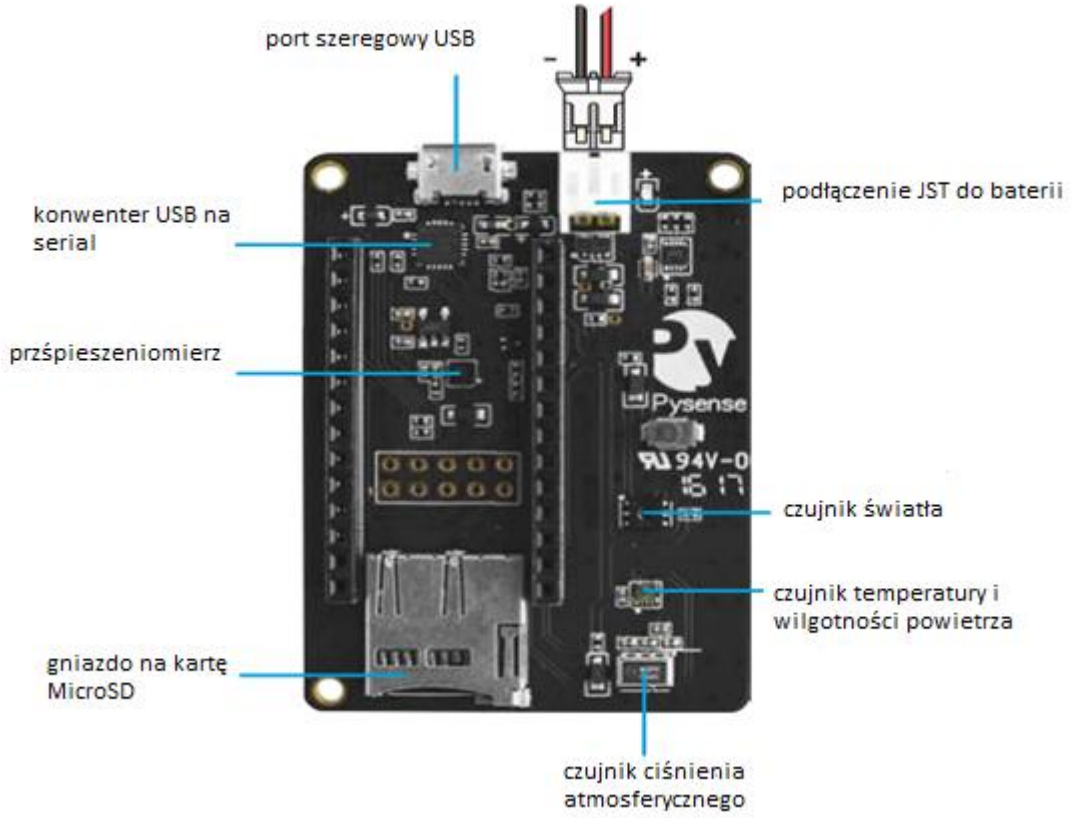
Moduł **FiPy ESP32** oraz płytka rozszerzeń Pysense opisane w instrukcji - dodatkowy opis:



Opis modułu **FiPy ESP32**

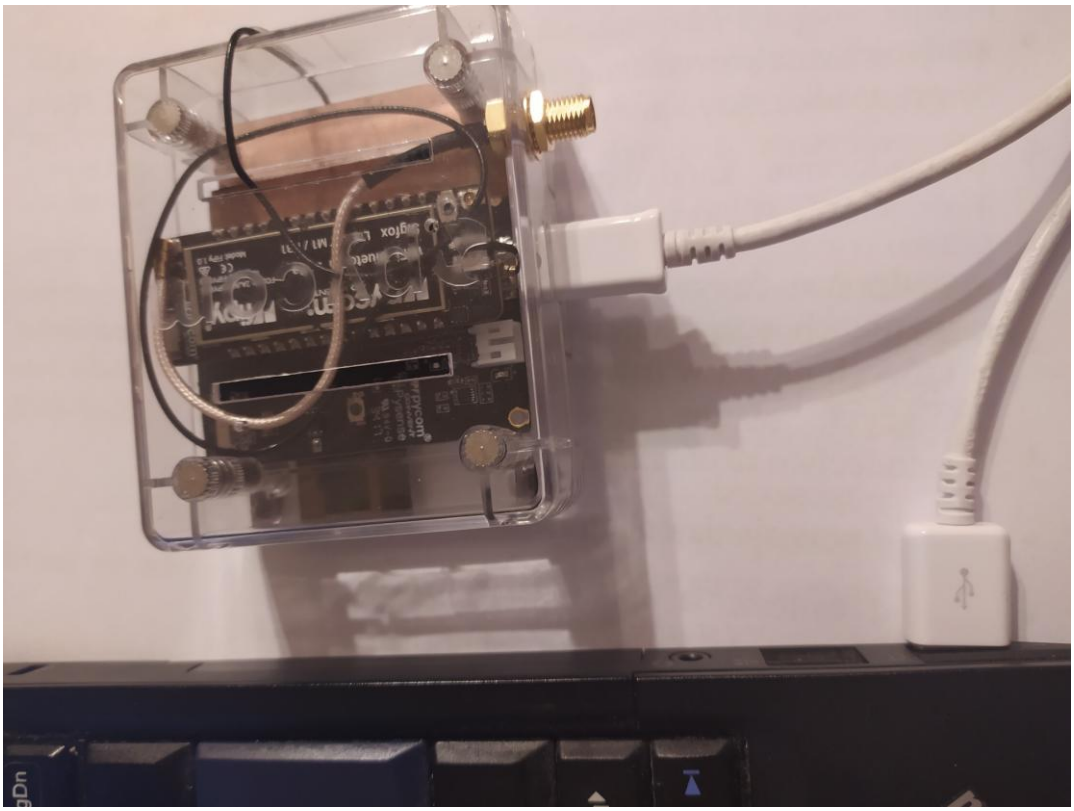
https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_FiPy_v2.pdf

PYSENSE – płytka rozszerzeń dla FIPY z dodatkowymi czujnikami takimi jak: czujnik światła, czujnik temperatury, barometr oraz akcelerometr.



Opis płytki rozszerzeń Pysense
https://docs.pycom.io/gitbook/assets/pysense-specsheet.pdf

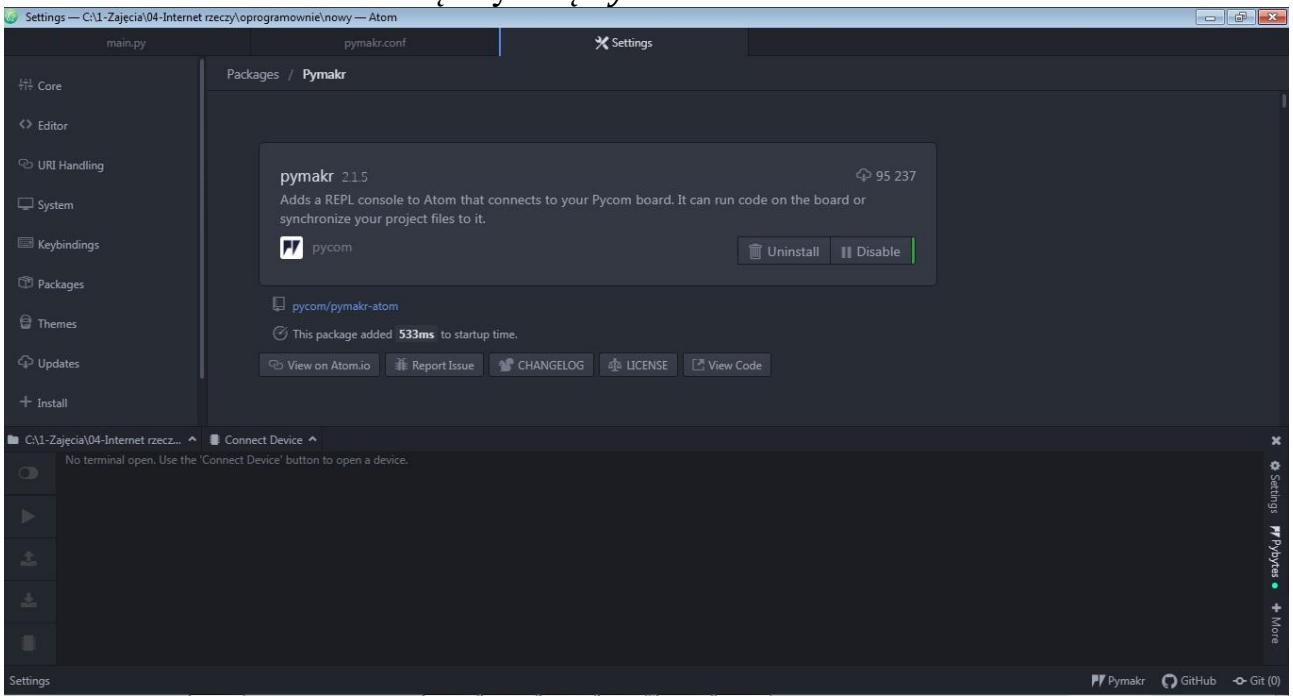
Mamy dostępne moduły z płytkami rozszerzeń z anteną LTE w obudowie:



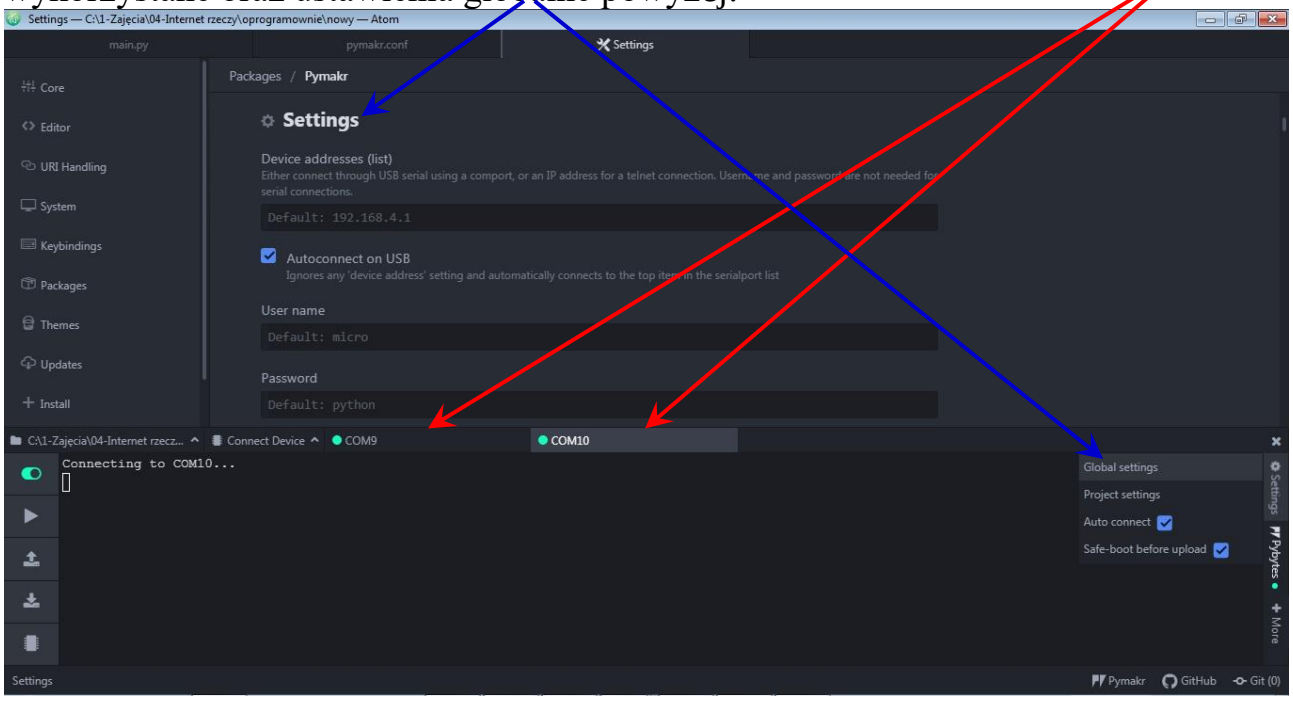
Dla Windows starszego niż 8 należy pobrać sterownik:
https://docs.pycom.io/gitbook/assets/pycom.inf
W Menedżerze urządzeń zaktualizować sterownik z lokalizacji gdzie go zapisaliśmy.

Do programowania urządzenia w języku Python przydatna jest wtyczka ***Pymakr*** do edytora ***Atom*** (https://atom.io)
opis na stronie: https://docs.pycom.io/pymakr/installation/atom

Widok ***Atom*** z zainstalowaną wtyczką ***Pymakr***:



po podłączeniu modułów do komputera przez USB widzimy które porty są wykorzystane oraz ustawienia globalne powyżej:



2. Projekt Pymakr w Atom:

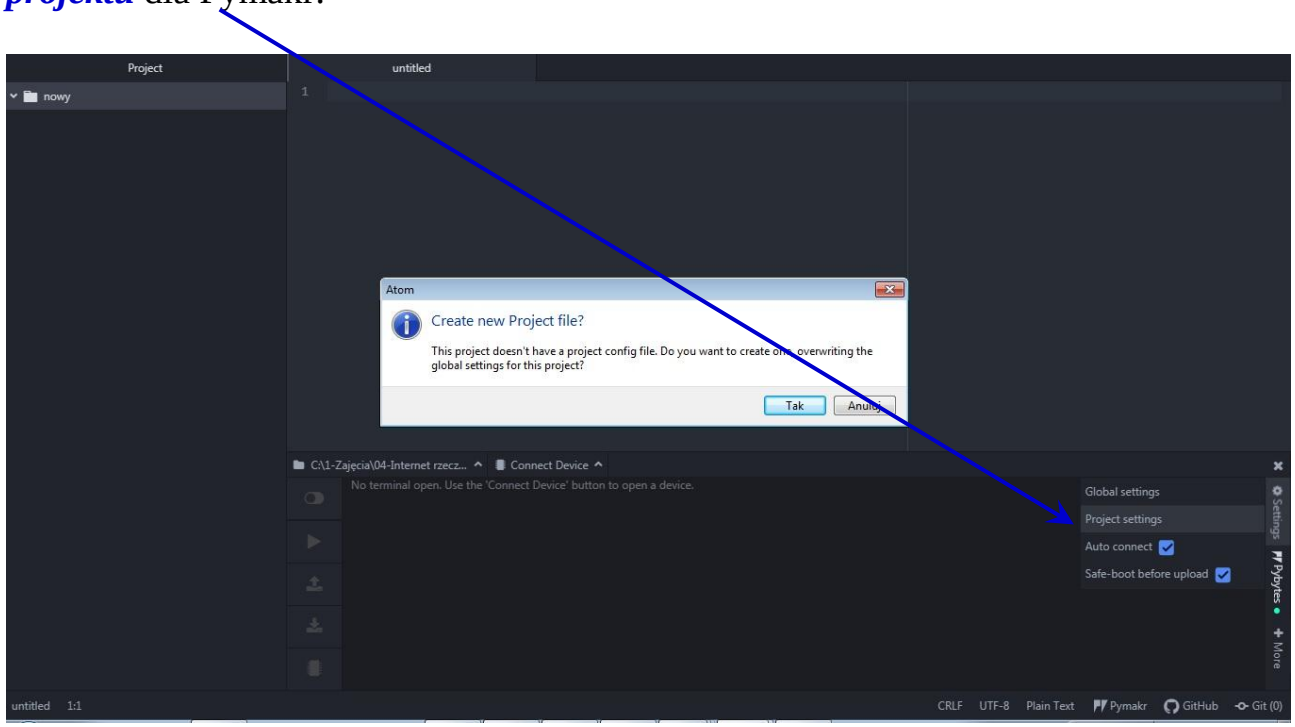
- utwórz nowy, pusty katalog na komputerze (pod nazwą np. `nowy`),
- otwórz Atom, otwórz utworzony katalog (`nowy`),

Standardowy projekt MicroPython ma następującą strukturę:

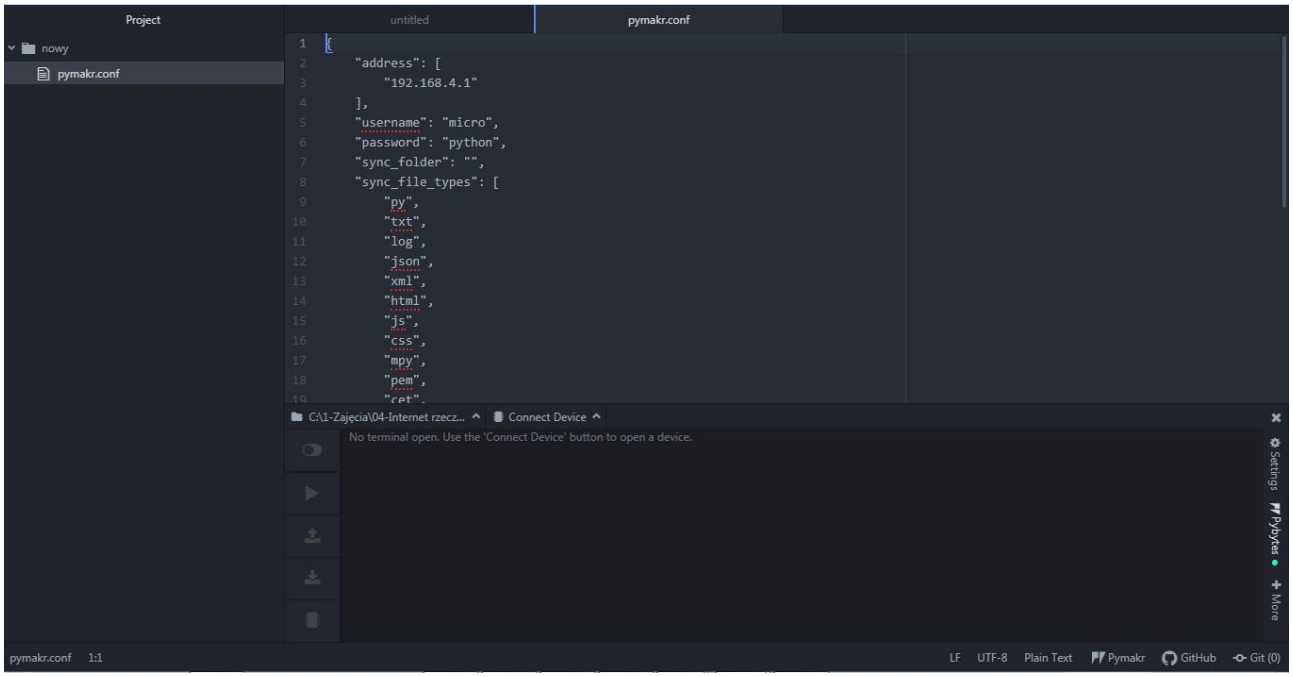
```
/nowy
[lib]
some_library.py
boot.py
main.py
```

- `boot.py` - to pierwszy skrypt uruchamiany w module po włączeniu.
  Często używane do łączenia modułu z siecią Wi-Fi, dzięki czemu można używać Telnet i FTP bez łączenia się z punktem dostępowym WiFi utworzonym przez moduł i bez zaśmiecania pliku `main.py`.
- `main.py` - ten skrypt działa bezpośrednio po `boot.py` i powinien zawierać główny kod, który chcemy uruchomić na urządzeniu.
- często dobrze jest podzielić kody wielokrotnego użytku na biblioteki.
  Jeśli chcemy tworzyć lub używać bibliotek utworzonych przez innych, musimy utworzyć katalog `lib` i umieścić w nim pliki bibliotek. Umieszczamy pliki `.py` bezpośrednio w `lib` zamiast tworzyć drzewo katalogów. Domyślnie MicroPython nie wykrywa żadnych bibliotek w podkatalogach.

Po skonfigurowaniu struktury projektu można skonfigurować specyficzne ***ustawienia projektu*** dla Pymakr:

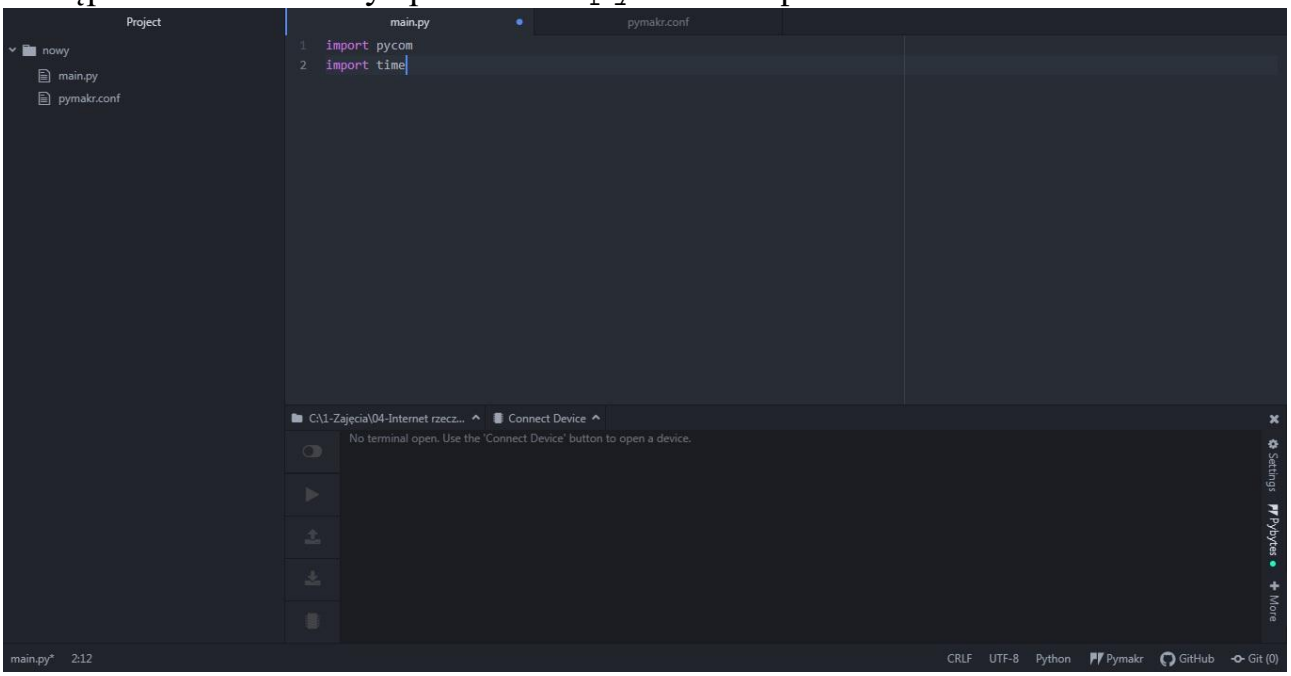


Na początku utworzy nam się plik o nazwie `pymakr.conf` w projekcie i zostanie zapełniony ustawieniami domyślnymi skopiowanymi z ustawień globalnych:



Pierwszą rzeczą, którą musimy zrobić, to zaimportować niektóre biblioteki, aby współpracować np. z czujnikami na płytce rozszerzeń. Oprogramowanie układowe Pycom zawiera dużą liczbę bibliotek dla wbudowanych standardowych funkcji (dokumentacja API).

Następnie trzeba utworzyć plik `main.py` i dodać np.:



Spowoduje to zaimportowanie dwóch bibliotek, Pycom, które są odpowiedzialne za funkcje specyficzne dla Pycom, takich jak wbudowana dioda LED i czas, który jest standardową biblioteką wykorzystywaną do taktowania i opóźniania.

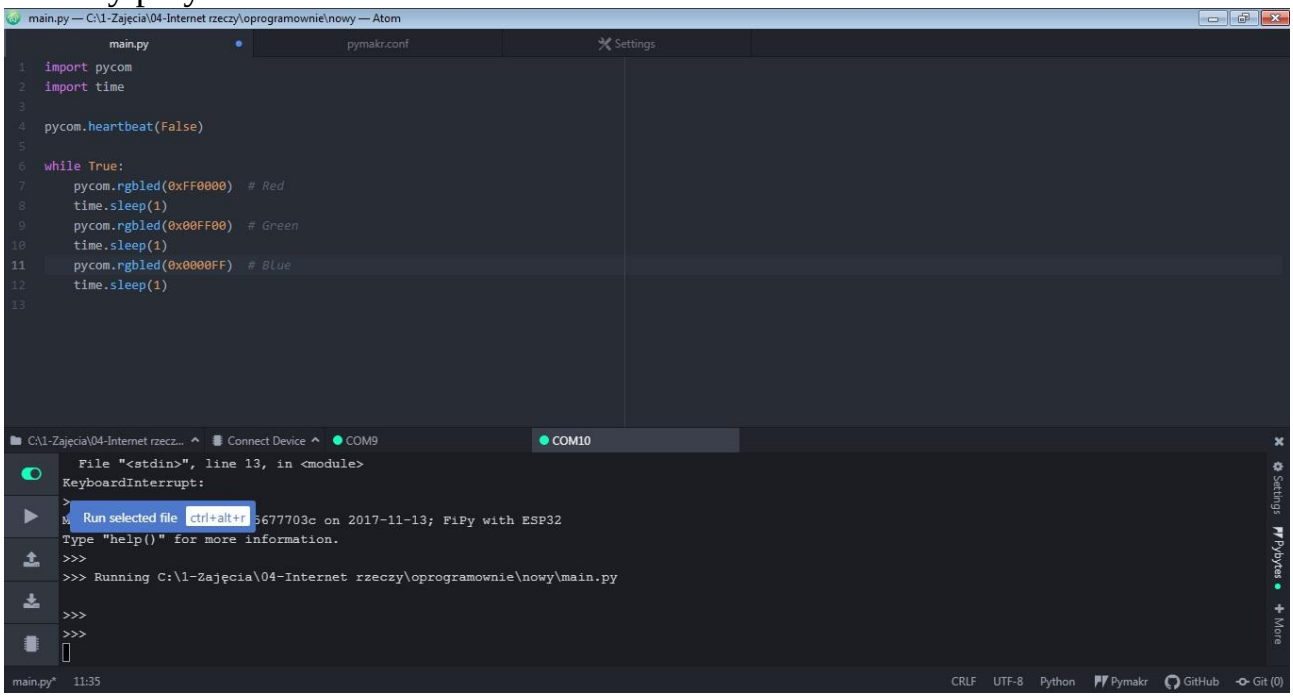
Standardowo po włączeniu modułu Pycom dioda LED w układzie miga regularnie na niebiesko, w celu sprawdzenia, czy moduł został włączony i uruchomił się poprawnie. Wyłączmy miganie jak poniżej:

```
pycom.heartbeat(False)
```

Kod do zmian koloru diody:

```
while True:
    pycom.rgbled(0xFF0000)  # Red
    time.sleep(1)
    pycom.rgbled(0x00FF00)  # Green
    time.sleep(1)
    pycom.rgbled(0x0000FF)  # Blue
    time.sleep(1)
```

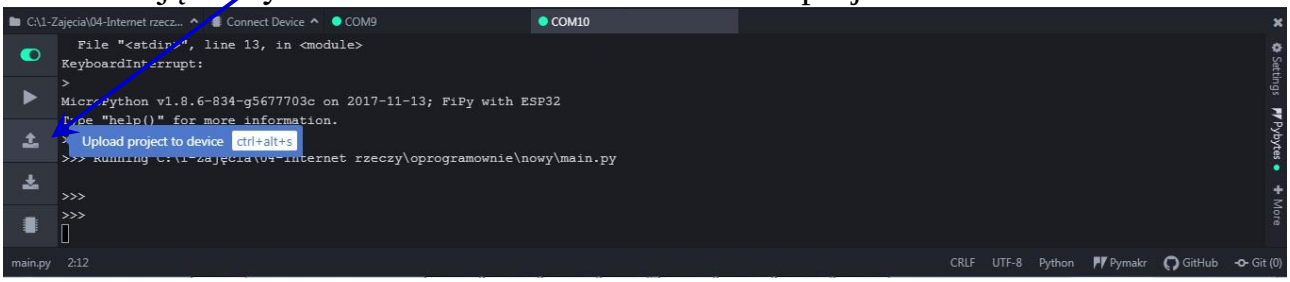
klikamy przycisk Run:



Kod w aktualnie otwartym pliku zostanie wykonany na urządzeniu, ale nie zostanie skopiowany na urządzenie. Jest to przydatne do szybkiego testowania, ale zadziała tylko wtedy, gdy korzystasz z bibliotek wbudowanych w oprogramowaniu wewnętrznym.

- Aby zatrzymać skrypt w terminalu `Pymakr` wybieramy `ctrl+c` na klawiaturze.
- Jeśli potrzebne są dodatkowe biblioteki, musimy je najpierw skopiować na urządzenie.

Jeśli wybierzemy przycisk `Upload`, Pymakr prześle wszystkie pliki w projekcie (o ile ich typ znajduje się w ustawieniu `sync_file_types` dla projektu w pliku `pymakr.conf`).
Są one następnie przechowywane na urządzeniu, nawet między restartami i umożliwiają korzystanie z bibliotek z folderu `lib` w projekcie.



Aby usunąć pliki z urządzenia można:
- połączyć się przez FTP i zarządzać swoimi plikami,
- sformatować (zresetować) wewnętrzną pamięć flash urządzenia:

```
>>> import os
>>> os.fsformat('/flash')
```

3. Resetowanie

- miękki reset usuwa stan maszyny wirtualnej MicroPython, ale nie wpływa na sprzętowe urządzenia peryferyjne, wykonujemy przez `Ctrl+D` na REPL (terminalu `Pymakr`) lub ze skryptu:

```
>>> import sys
>>> sys.exit()
```

- twardy reset naciskając przycisk resetowania na płytce lub uruchamiając skrypt:

```
>>> import machine
>>> machine.reset()
```

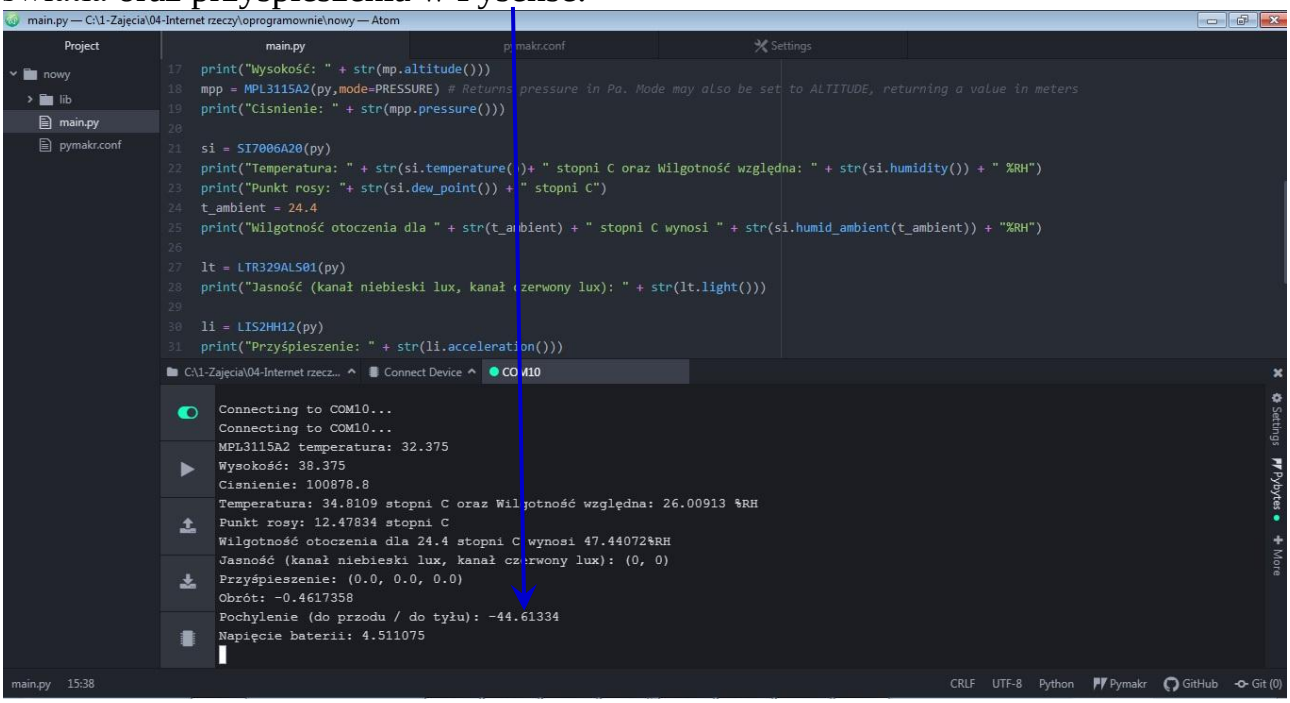
4. Biblioteki

Do wykorzystania czujników na Pysense potrzebne są biblioteki, dzięki którym odczyt do oraz z różnych czujników. Te biblioteki znajdują się w repozytorium Pycom GitHub, a najnowsze wersje można znaleźć na stronie z wydaniami.

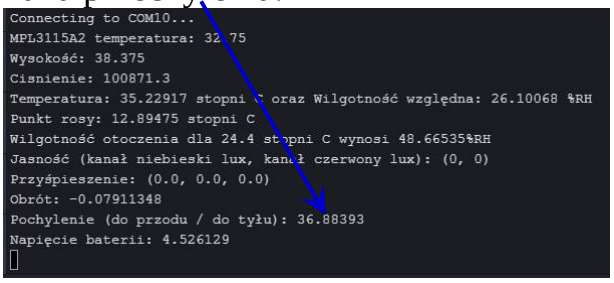
Zawierają informacje na temat różnych ***metod i klas*** dostępnych dla każdego z czujników Pysense: https://github.com/pycom/pycom-libraries

W pobieranym pliku **… .zip** wybrać odpowiednie urządzenie (Pysense), rozpakować, odpowiednie pliki `.py` umieścić w folderze `/lib` w projekcie.
Po przesłaniu bibliotek do urządzenia można je wykorzystać czy zaimportować tak jak typową bibliotekę MicroPython.

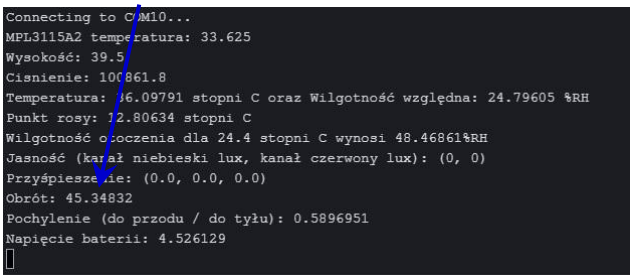
Na przykład importowanie i używanie czujników wysokości, ciśnienia, temperatury, światła oraz przyspieszenia w Pysense:



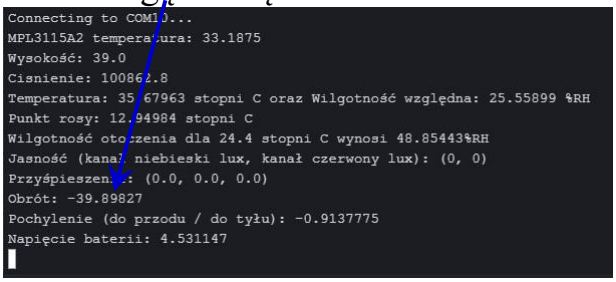
zmiana przechylenia:



zmiana obrotu:



obrót w drugą stronę:



Akcelerometr 3-osiowy (LIS2HH12)

Akcelerometr służy do pomiaru przyśpieszenia kątowego lub liniowego.

Konstruktor:

klasa (LIS2HH12) (`pysense = None, sda = 'P22', scl = 'P21'`)

Metody:

`LIS2HH12.acceleration ()`

Zwraca krotkę z 3 wartościami przyspieszenia (G)

`LIS2HH12.roll ()`

Zwraca liczbę zmiennoprzecinkową w stopniach w zakresie od -180 do 180

`LIS2HH12.pitch ()`

Zwraca liczbę zmiennoprzecinkową w stopniach w zakresie od -90 do 90. Gdy układ przechyli się poza ten zakres, wartości się powtórzą. Wynika to z braku pomiaru odchylenia, co uniemożliwia dokładne określenie położenia układu.

Przykład:

```
li = LIS2HH12(py)
print("Przyśpieszenie: " + str(li.acceleration()))
print("Obrót (w prawo / w lewo): " + str(li.roll()))
print("Pochylenie (do przodu / do tyłu): " + str(li.pitch()))
```

Cyfrowy czujnik światła otoczenia (LTR-329ALS-01)

Pysense ma podwójny czujnik światła, który zapewnia pomiar natężenia zewnętrznego światła w luksach.

Konstruktor:

klasa LTR329ALS01 (`pysense = None, sda = 'P22', scl = 'P21', gain = ALS_GAIN_1X, integration = ALS_INT_100, rate = ALS_RATE_500`)

Metody:

`LTR329ALS01.light ()`

Zwraca krotkę z dwiema wartościami poziomów światła w luksach.

Przykład:

```
lt = LTR329ALS01(py)
print("Jasność (kanał niebieski lux, kanał czerwony lux): " +
str(lt.light()))
```

Czujnik wilgotności i temperatury (SI7006A20)

Pysense ma czujniki wilgotności i temperatury, który podaje wartości wilgotności względnej i temperatury zewnętrznej.

Konstruktor

klasa SI7006A20 `(pysense = None, sda = „P22”, scl = „P21”)`

Konstruktor tworzy obiekt, który odczytuje i zwraca wartości temperatury(^{o}C) i wilgotności (%).

Metody:

`SI7006A20.humidity()`

zwracana jest wilgotność względna jako liczba zmiennoprzecinkowa wyrażona w procentach.

`SI7006A20.temperature()`

zwracana jest temperatura zewnętrzna jako liczba zmiennoprzecinkowa wyrażona w stopniach Celsjusza.

Przykład:

```
si = SI7006A20(py)
print("Temperatura: " + str(si.temperature())+ " stopni C oraz
Wilgotność względna: " + str(si.humidity()) + " %")
```

<u>Barometr z wysokościomierzem</u> (MPL3115A2)

Pysense ma czujnik barometryczney, który zapewnia odczyty ciśnienia, wysokości, a także dodatkowy czujnik temperatury.

Konstruktor

class MPL3115A2(`pysense = None, sda = 'P22', scl = 'P21', mode = PRESSURE`). Konstruktor tworzy obiekt, który zwraca wartości ciśnienia (Pa), wysokości (m) i temperatury (^{o}C).

Metody:

`MPL3115A2.pressure()`

zwracane jest ciśnienie atmosferyczne jako liczba zmiennoprzecinkowa wyrażona w Pa

`MPL3115A2.altitude()`

zwracana jest wysokość jako liczba zmiennoprzecinkowa wyrażona w metrach

`MPL3115A2.temperature()`

zwracana jest temperatura jako liczba zmiennoprzecinkowa wyrażona w stopniach Celsjusza.

Do konstruktora można przekazać argumenty:

`mode: PRESSURE, ALTITUDE`

Przykład:

```
mp = MPL3115A2(py,mode=ALTITUDE)
print("Wysokość: " + str(mp.altitude()))
mpp = MPL3115A2(py,mode=PRESSURE)
print("Cisnienie: " + str(mpp.pressure()))
```

Metody uśpienia i budzenia, które są osadzone w bibliotece Pysense:

Przykład:

```
from pysense import Pysense
from LIS2HH12 import LIS2HH12
import time
py = Pysense()

py.setup_sleep(3) # Ustawia interwał snu określony w sekundach.
py.go_to_sleep() # przełącza układ w tryb uśpienia
```

```
import pycom
import time
from pysense import Pysense
import machine

from LIS2HH12 import LIS2HH12
from SI7006A20 import SI7006A20
from LTR329ALS01 import LTR329ALS01
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE

pycom.heartbeat(False)

for cycles in range(2):
   time.sleep(1)
   pycom.rgbled(0xFF0000) # red
   py = Pysense()

   mp = MPL3115A2(py,mode=ALTITUDE) # Returns height in meters.
   Mode may also be set to PRESSURE, returning a value in Pascals
# print("MPL3115A2 temperatura: " + str(mp.temperature()))
   print("Wysokość: " + str(mp.altitude()))
   mpp = MPL3115A2(py,mode=PRESSURE) # Returns pressure in Pa.
   Mode may also be set to ALTITUDE, returning a value in meters
   print("Cisnienie: " + str(mpp.pressure()))

   si = SI7006A20(py)
   print("Temperatura: " + str(si.temperature())+ " stopni C oraz
   Wilgotność względna: " + str(si.humidity()) + " %")
#    print("Punkt rosy: "+ str(si.dew_point()) + " stopni C")
# t_ambient = 24.4
# print("Wilgotność otoczenia dla " + str(t_ambient) + " stopni C
wynosi " + str(si.humid_ambient(t_ambient)) + "%RH")

   lt = LTR329ALS01(py)
   print("Jasność (kanał niebieski lux, kanał czerwony lux): " +
   str(lt.light()))

   li = LIS2HH12(py)
   print("Przyśpieszenie: " + str(li.acceleration()))
   print("Obrót (w prawo / w lewo): " + str(li.roll()))
   print("Pochylenie (do przodu / do tyłu): " + str(li.pitch()))
#    print("Napięcie baterii: " + str(py.read_battery_voltage()))
   pycom.rgbled(0x00FF00)  # Green
```