

LESSON 2: SETTING UP PYMAKR IDE

by Pycom

LESSON OBJECTIVES

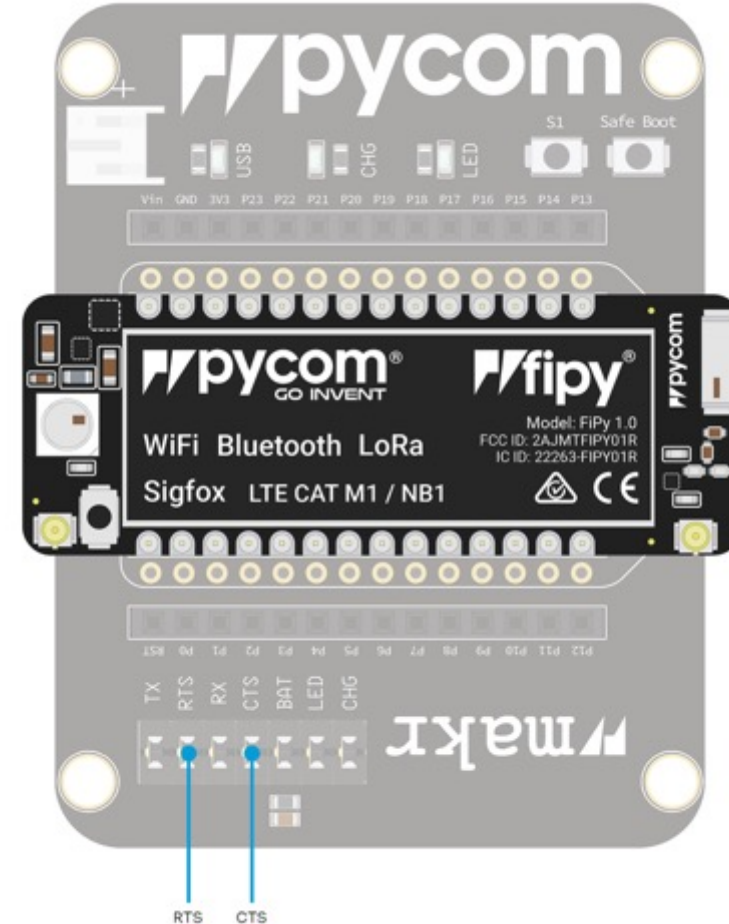
- To link the pieces of hardware together
- To download the Pymakr plug-in on Atom or VSCode
- To introduce Micropython

FIRMWARE UPDATED? THEN ON TO THE NEXT THING...

LET'S GET THAT LOPY4 ON THAT EXPANSION BOARD

BEFORE YOU START...

- Check that the CTS and RTS jumpers have been removed
- Find the reset button (it's next to the LED)
- Find the USB connector on the Expansion Board



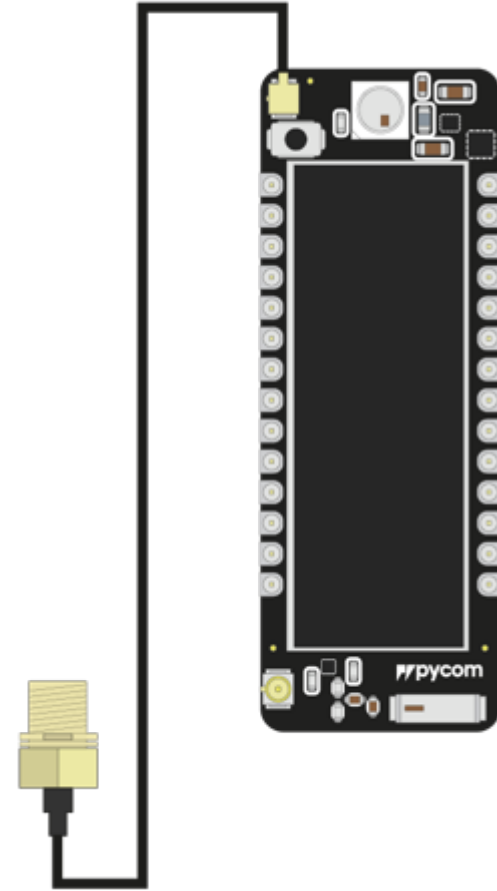
ATTACHING THE LORAWAN AND SIGFOX ANTENNA

You must connect the LoRaWAN and Sigfox Antenna if you want to use the LoRa/SigFox networks (otherwise you can damage your device)

Quick note: the FiPy only supports LoRa on the 868MHz or 915 MHz bands. It doesn't support the 433MHz band (if you want to use this band, you'll have to grab a LoPy4)

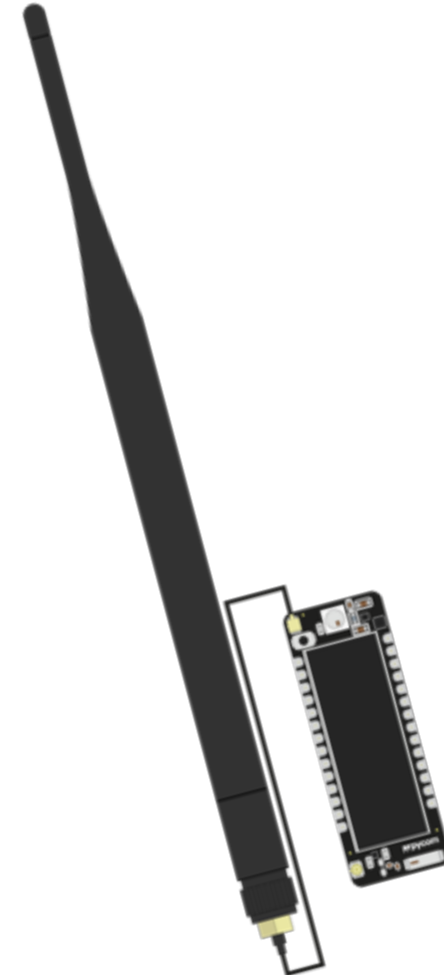
ATTACHING THE LORAWAN AND SIGFOX ANTENNA

1. You'll need to connect the U.FL to SMA pig tail to the FiPy using the U.FL connector which is on the same side of the FiPy as the LED
 - o You have to align the flat edge correctly, and screw down the connector using the nut provided
2. You will need to screw on the antenna to the SMA connector



ATTACHING THE WIFI ANTENNA

- There is a U.FL connector for an external antenna
 - This is optional – only required if better performance is needed
- Switching between the antennas is done via software



HARDWARE DONE!

SOFTWARE SET-UP – THE TOOLS YOU NEED

Drivers

If you are using Microsoft Windows, you might need to install drivers from our products so that they function correctly

SOFTWARE SET-UP – THE TOOLS YOU NEED

Pycom firmware update utility:

- This automates the process of upgrading the firmware on your Pycom device
- You should use this tool before you use your device

SOFTWARE SET-UP – THE TOOLS YOU NEED

Development environment:

- Pymakr is a plug-in for Atom and Visual Studio created by Pycom to make development for Pycom modules super easy

SOFTWARE SET-UP – THE TOOLS YOU NEED

Drivers:

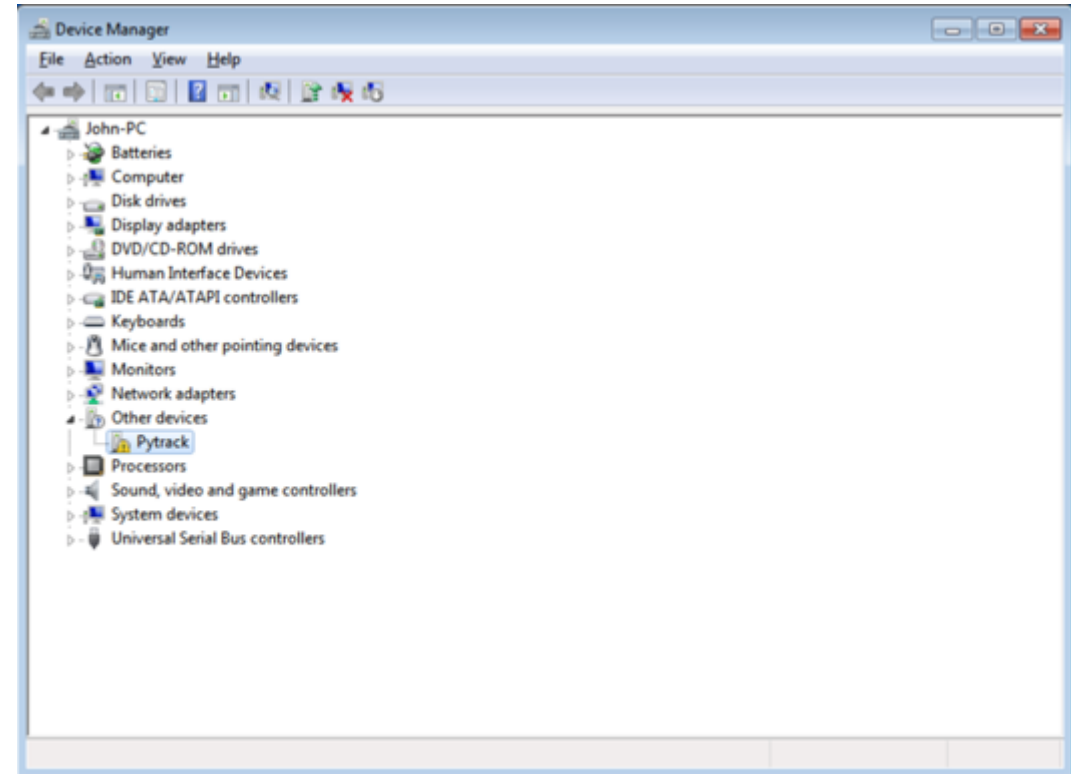
- **Windows 8/10/+**
 - All of our products will work out of the box for Windows 8/10/+
- **Windows 7**
 - drivers to support the Expansion Board will need to be installed
 - Click [here](#) (please save the file to your computer)

If you don't need the Windows 7 driver
Please [click here](#)

SOFTWARE SET-UP – WINDOWS 7

Drivers:

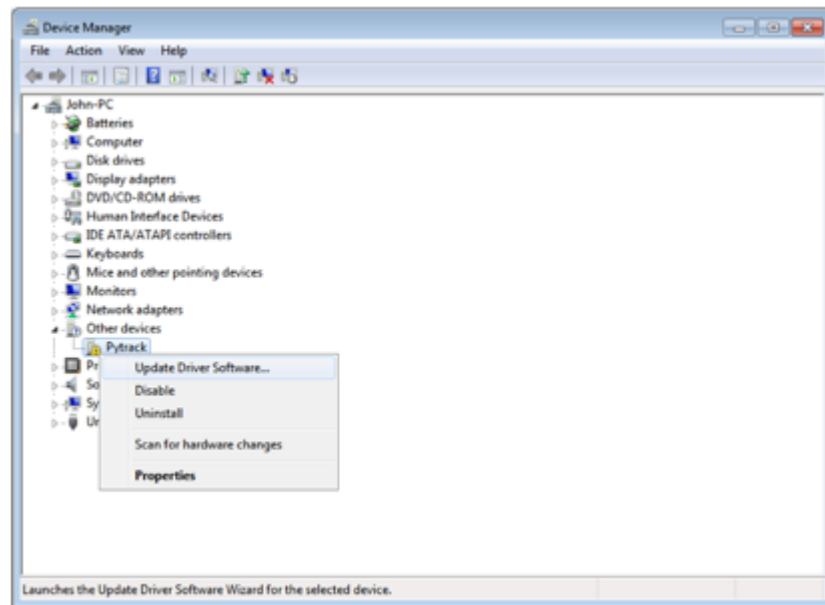
1. Navigate to the Windows start menu and search for **Device Manager**
2. Expansion Board 3.1 will be in the dropdown menu under **other devices** (a Pytrack is used in the example)



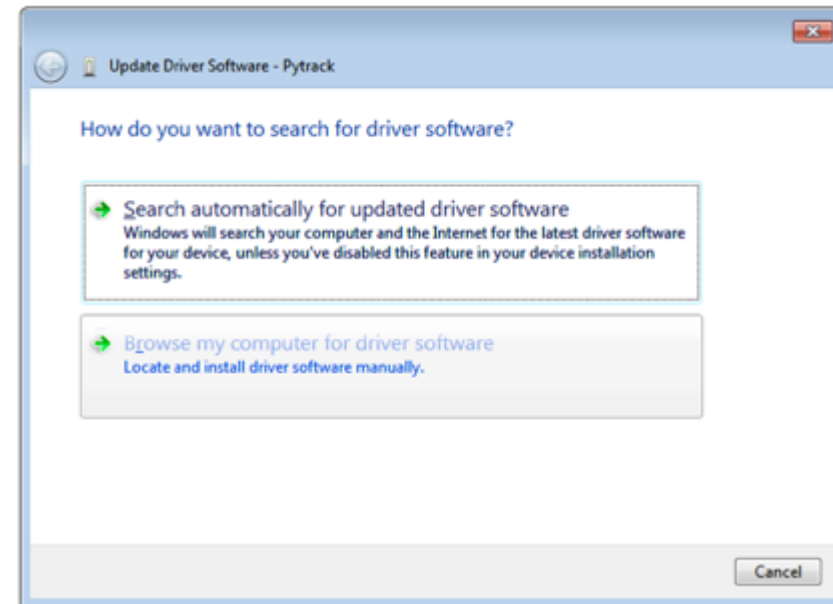
SOFTWARE SET-UP – WINDOWS 7

Drivers:

3. Right-click the device and select Update Driver Software



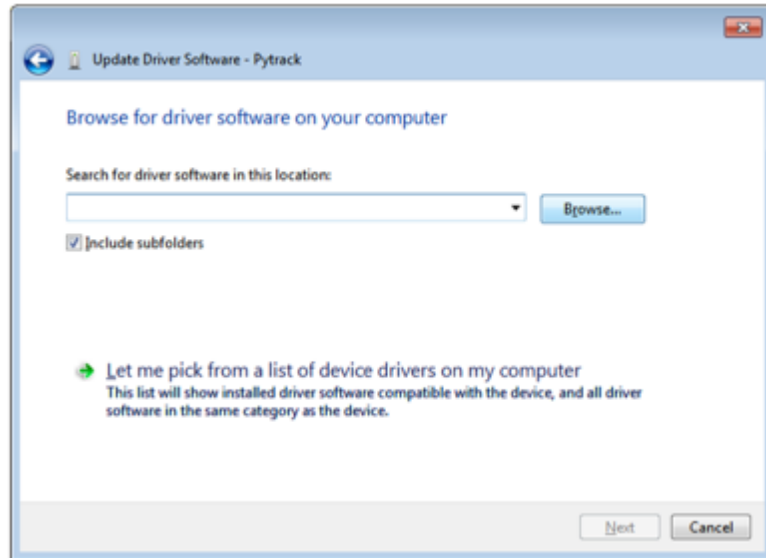
4. Right-click the device and select Update Driver Software



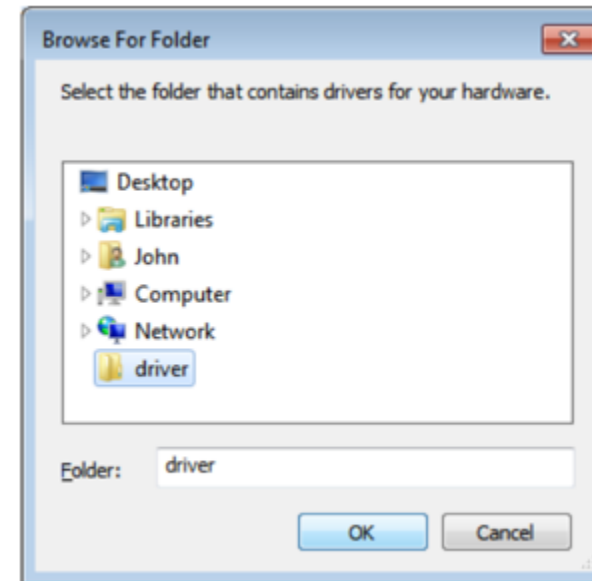
SOFTWARE SET-UP – WINDOWS 7

Drivers:

5. Need to navigate to where you downloaded the driver to (e.g. **Downloads** Folder)



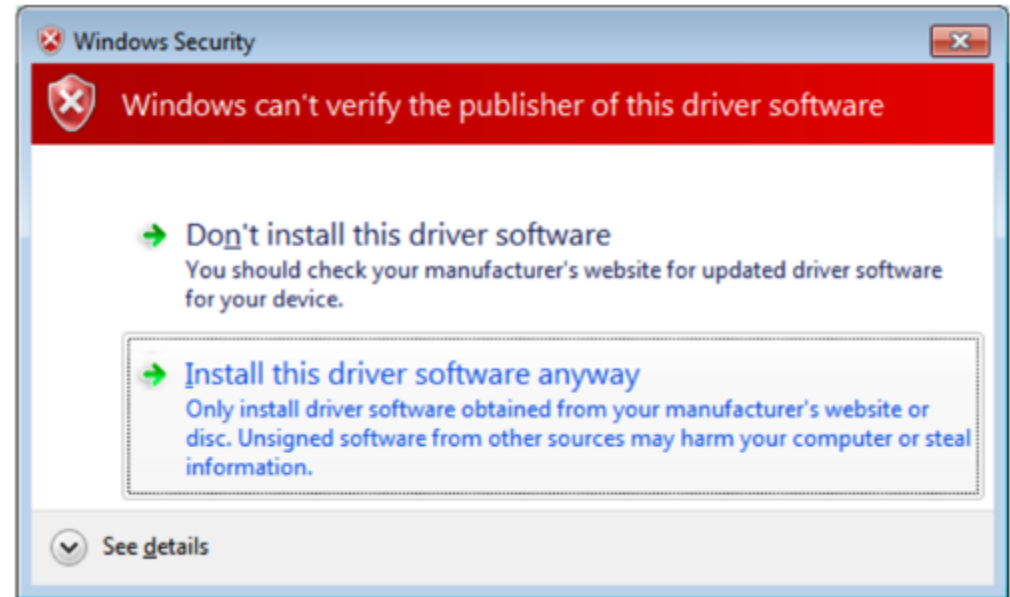
6. Specify the folder in which the drivers are contained



SOFTWARE SET-UP – WINDOWS 7

Drivers:

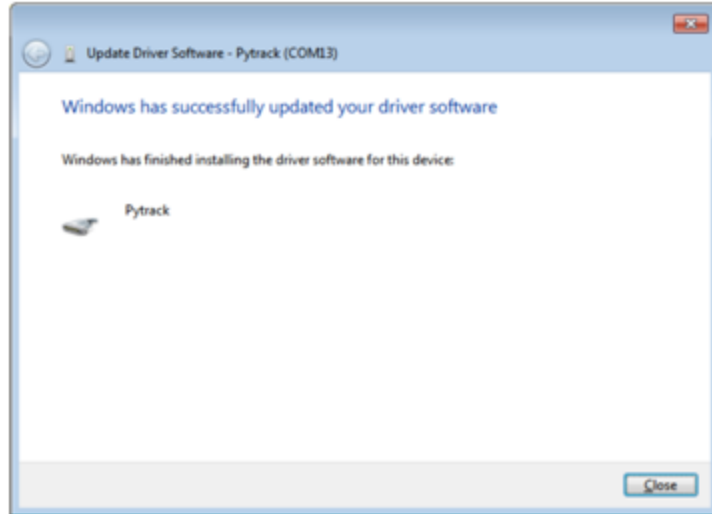
- You may receive a warning, suggesting that Windows can't verify the publisher of this driver.
- Click **Install this driver software anyway** as this link points towards our official driver



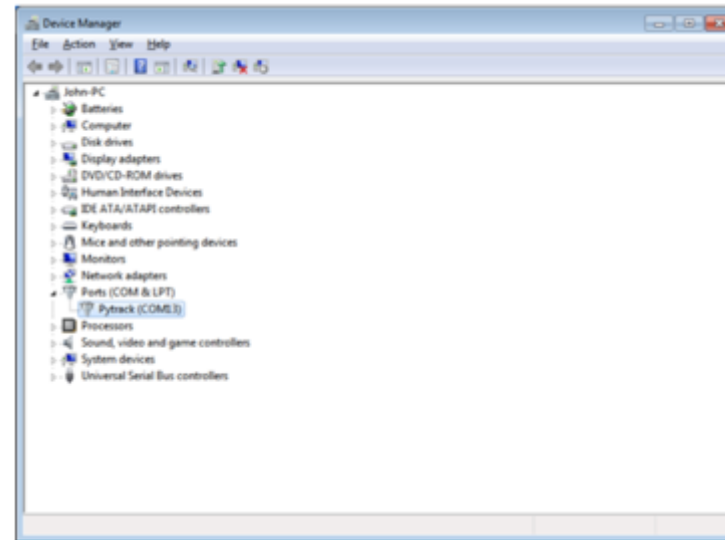
SOFTWARE SET-UP – WINDOWS 7

Drivers:

7. If the installation was successful, you should now see a window specifying that the driver was correctly installed



8. To confirm that the installation was correct, navigate back to the **Device Manager** and click the dropdown for other devices



The warning label should now be gone and (in this example) Pytrack/ Pysense should be installed

SOFTWARE SET-UP – UPDATING FIRMWARE

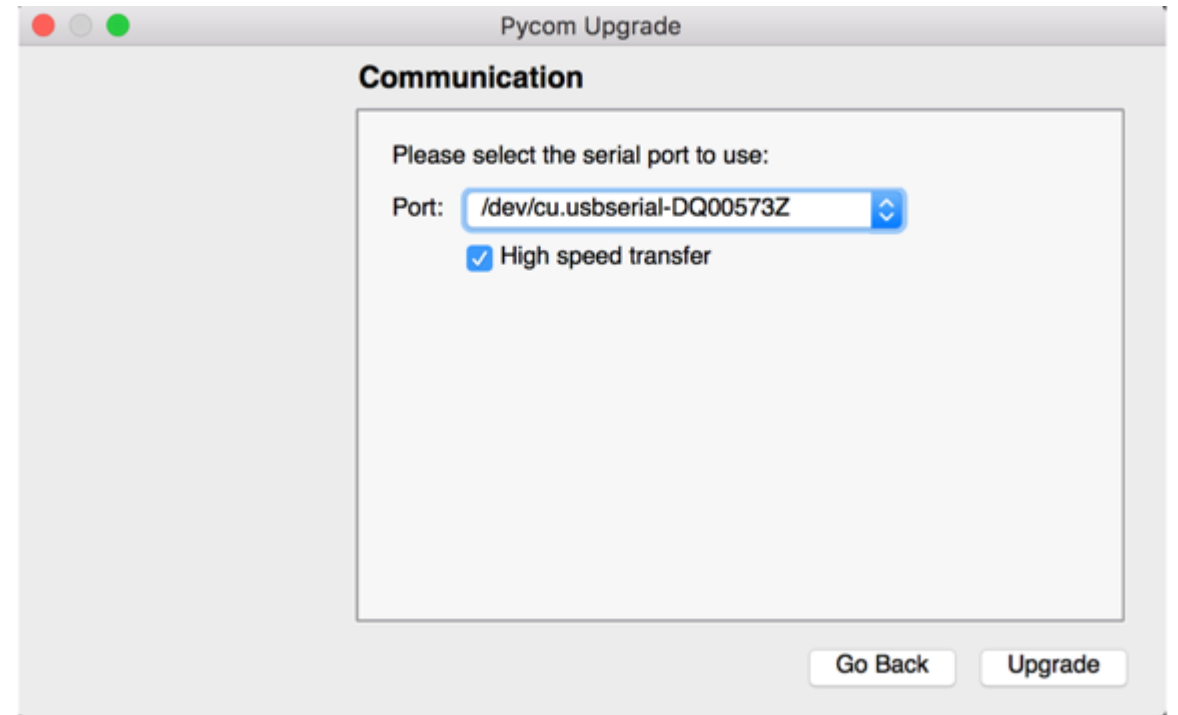
We strongly recommend you upgrade your firmware to the latest version as we are constantly making improvements and adding new features to the devices

The download links to the update tool are listed below:

- [Windows](#)
- [MacOS](#) (10.11 or higher)
- [Linux](#) (requires **dialog** and **python-serial** package)

SOFTWARE SET-UP – UPDATING FIRMWARE

1. Check that you have updated the firmware on the Expansion Board 3.1 (should have been done earlier in the lecture)
2. Disconnect the device from the computer
3. Insert module into expansion board
4. Reconnect the board via USB to your computer
5. Run the Firmware Upgrade tool
6. Disconnect the USB cable from the board and reconnect it, your device is now ready to use



AN INTRODUCTION TO PYMAKR

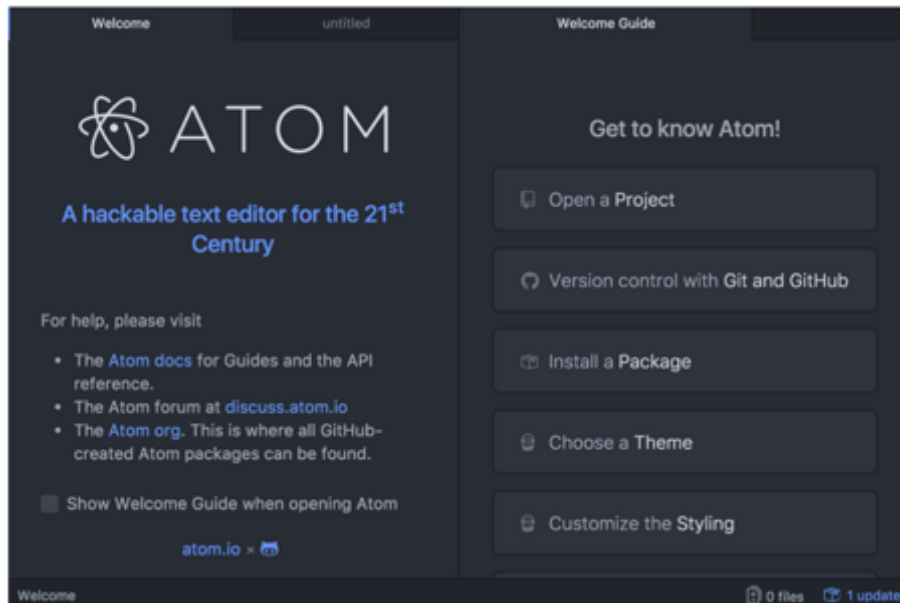


- To make it as easy as possible, Pycom has developed a plugin for two popular text editors, called Pymakr
- These plugins are built for and available in:
 - ATOM
 - Visual Studio Code

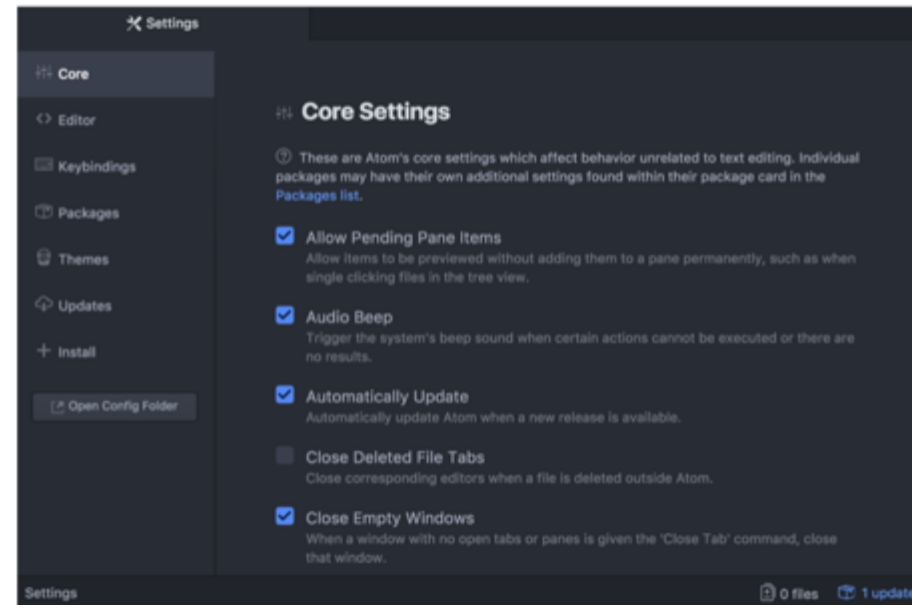
The following slides will describe how to install Pymakr for Atom, click for [Visual Studio Code](#)

PLUG-IN FOR ATOM

1. Install [Atom](#) and open it

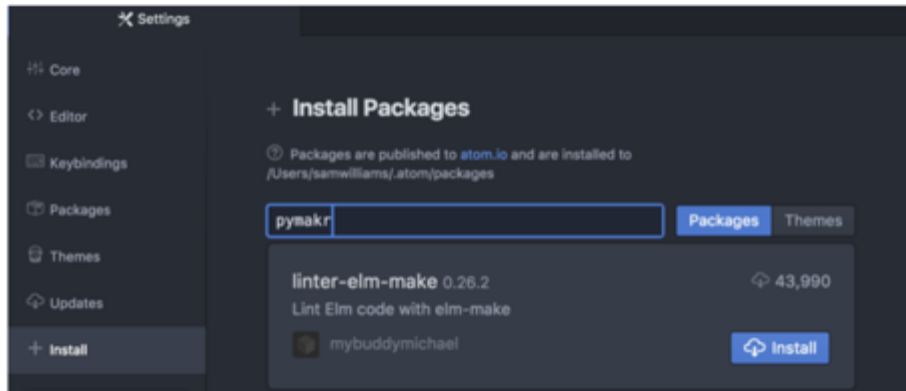


2. Navigate to the install page

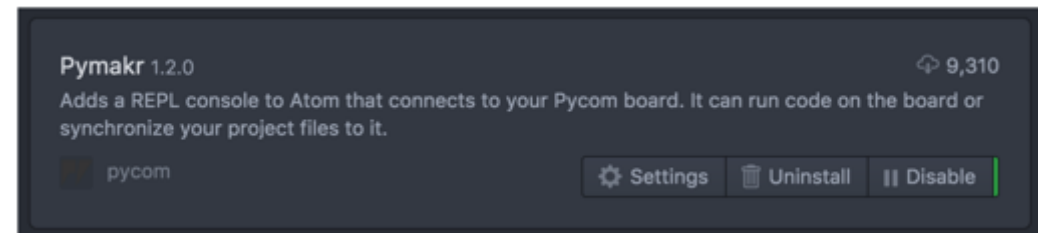
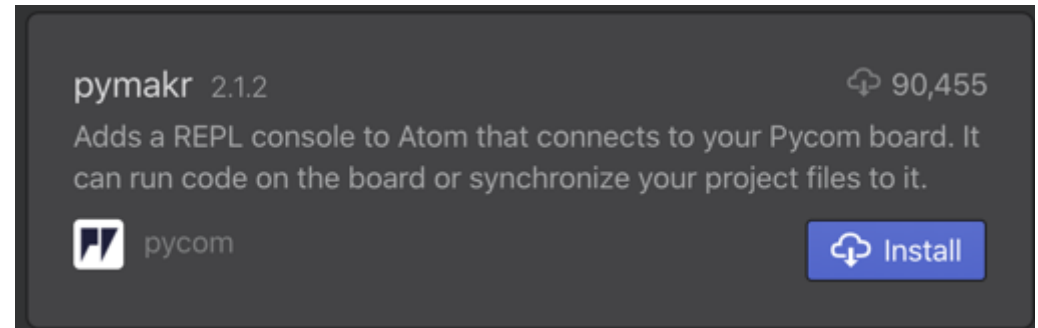


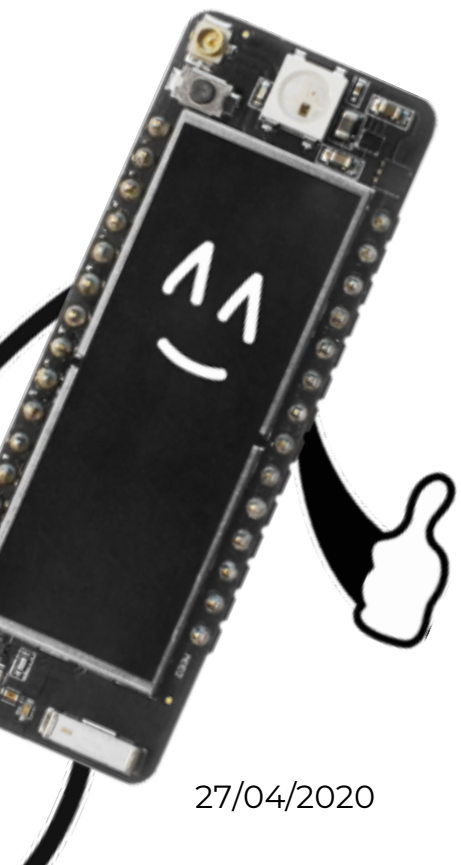
PLUG-IN FOR ATOM

3. Search for **Pymakr** and select the official Pycom Pymakr plugin



4. You should now see 'Pymakr' and click the install button

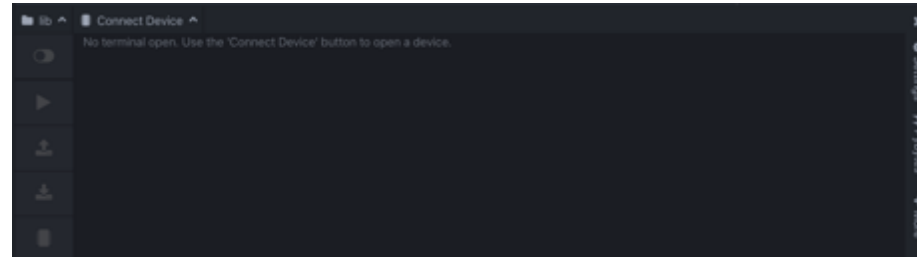




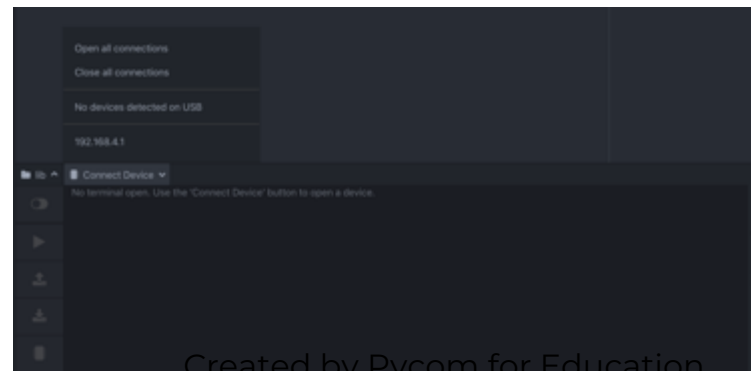
AND ATOM IS INSTALLED!

CONNECTING VIA SERIAL USB

1. Connect your device to your computer via USB.
2. Open Atom and ensure that the Pymakr Plugin has correctly installed

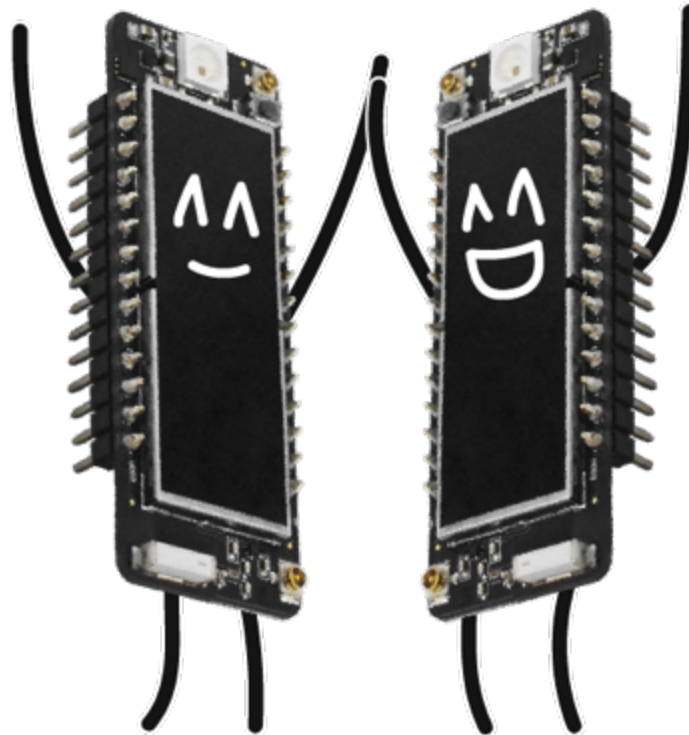


3. Open the Pymakr console by clicking on Connect Device



CONNECTING VIA SERIAL USB

4. Pymakr's autoconnect feature will have found your plugged-in device!



PLUGIN FOR VISUAL STUDIO CODE

To download Visual Studio Code, click [VS Code](#)

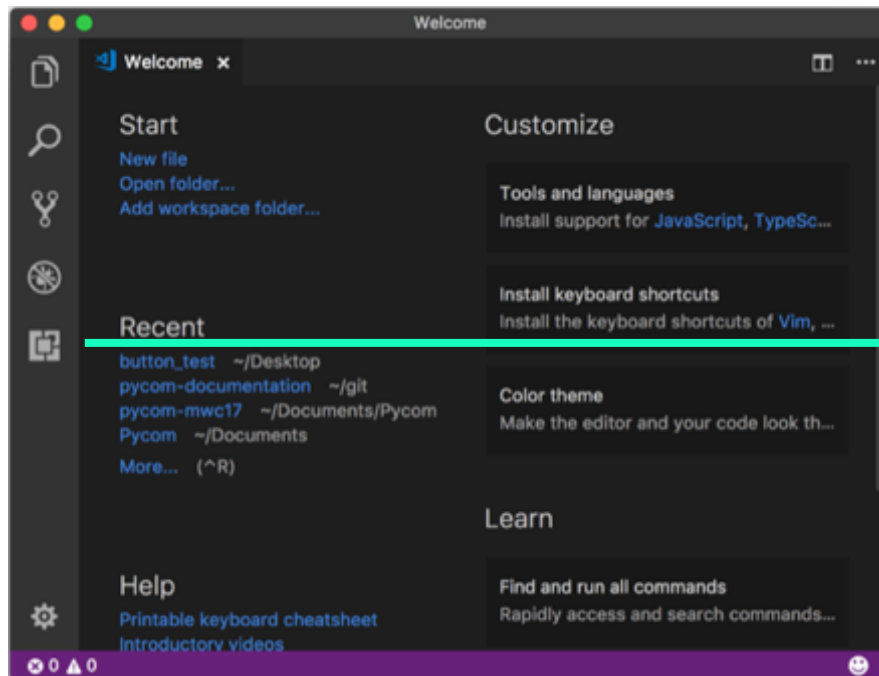
You will also need NodeJS installed on your PC

- Please download the latest LTS version available from the [NodeJS website](#)

PLUGIN FOR VISUAL STUDIO CODE

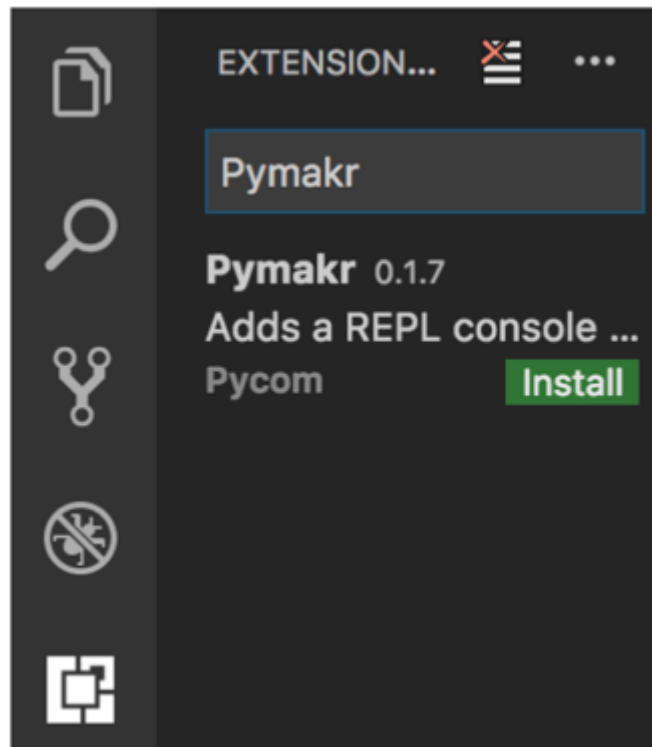
1. Open VSCode

2. Navigate to the **Extensions** page, using the 5th button in the left hand-side navigation bar

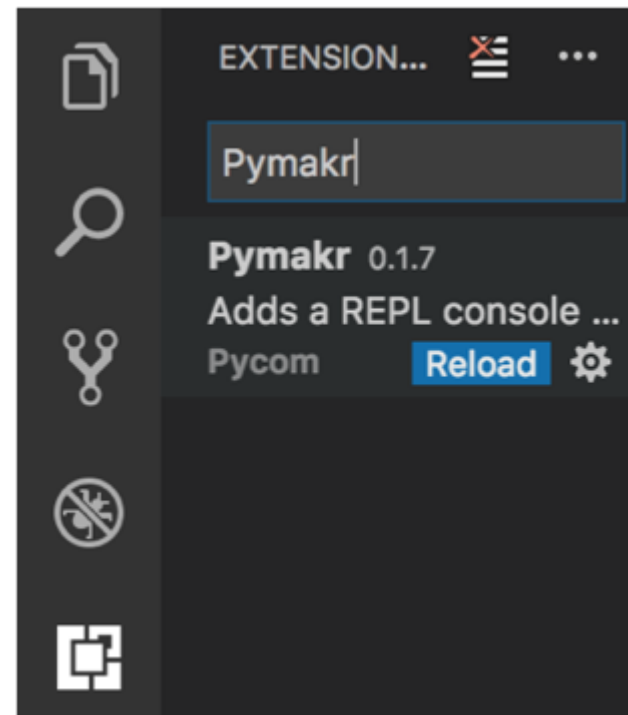


PLUGIN FOR VISUAL STUDIO CODE

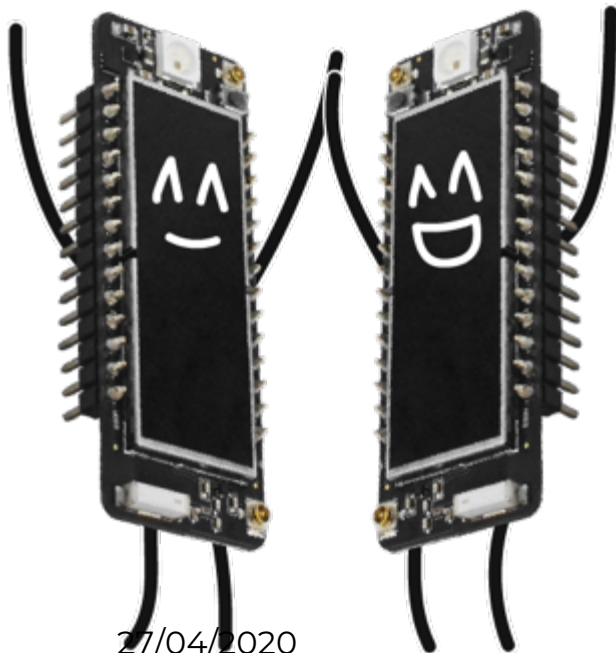
3. Search for **Pymakr** and click the install button next to it



4. Within a few minutes, a reload button should appear – press it to reload VSCode



AND VSCODE INSTALLED!



27/04/2020

A screenshot of the Visual Studio Code (VS Code) editor interface. The main window displays a Python script named 'boot.py'. The code is written in a dark theme and includes imports for 'machine', 'os', and 'time'. It sets up a UART connection, resets the machine, and configures a Wi-Fi module (WLAN) to connect to a network. The script uses a list of known networks and attempts to connect to one of them. The terminal at the bottom shows the output of the script, indicating it is connecting to a network.

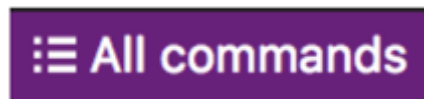
```
1 import machine
2 import os
3 import time
4 uart=machine.UART(0, 115200)
5 os.dupterm(uart)
6 if machine.reset_cause() != machine.SOFT_RESET:
7     from network import WLAN
8     known_nets=[('ssid', 'password')]
9     wlan=WLAN()
10    original_ssid=wlan.ssid()
11    original_auth=wlan.auth()
12    wlan.mode(WLAN.STA)
13    available_nets=wlan.scan()
14    nets=fromdict([n.ssid for n in available_nets])
15    known_nets_names=fromdict([n[0] for n in known_nets])
16    net_to_use=list(nets&known_nets_names)
17    try:
18        net_to_use=net_to_use[0]
19        pw=next(known_nets)[net_to_use]
20        sec=[e.sec for e in available_nets if e.ssid==net_to_use][0]
21        wlan.connect(net_to_use, [sec, pw], timeout=10000)
22        while not wlan.isconnected():
23            time.sleep(0.1)
24    except:
25        wlan.init(mode=WLAN.AP, ssid=original_ssid, auth=original_auth, channel=6, antenna=WLAN.IN)
26
```

Created by Pycom for Education

CONNECTING VIA SERIAL USB - VSCODE

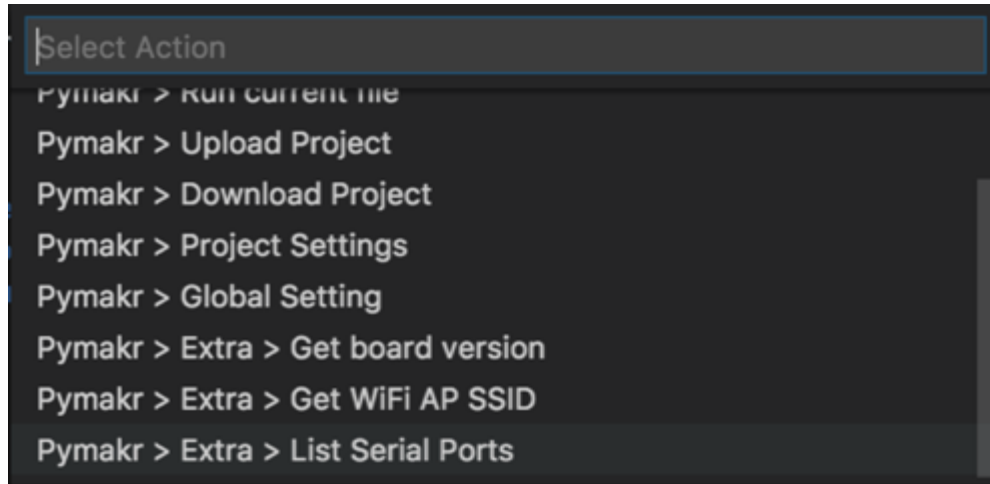
After installing the Pymakr Plugin, you'll need to configure it for first time:

1. Connect your device to your computer via USB.
2. Open VSCode and ensure that the Pymakr Plugin has correctly installed
3. Click **All Commands**



CONNECTING VIA SERIAL USB - VSCODE

4. In the list that appears, click **Pymakr>Extra>List Serial Ports**



5. All available serial ports will be listed.

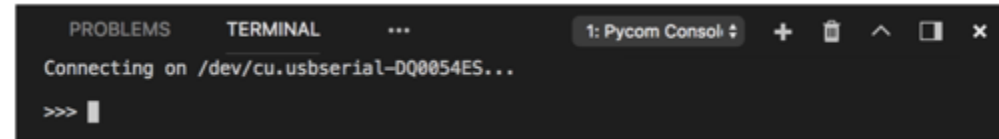


CONNECTING VIA SERIAL USB - VSCODE

6. Click **All Commands**, then **Pymakr>Global Settings**
 - o This will open a JSON file

```
1 {
2   "address": "192.168.4.1",
3   "username": "micro",
4   "password": "python",
5   "sync_folder": "Demos/FridgeSensor/",
6   "open_on_start": true,
7   "sync_file_types": "py,txt,log,json,xml",
8   "ctrl_c_on_connect": false
9 }
```

7. Close the JSON file and click **All Commands**, then **Pymakr>Connect** to connect your device



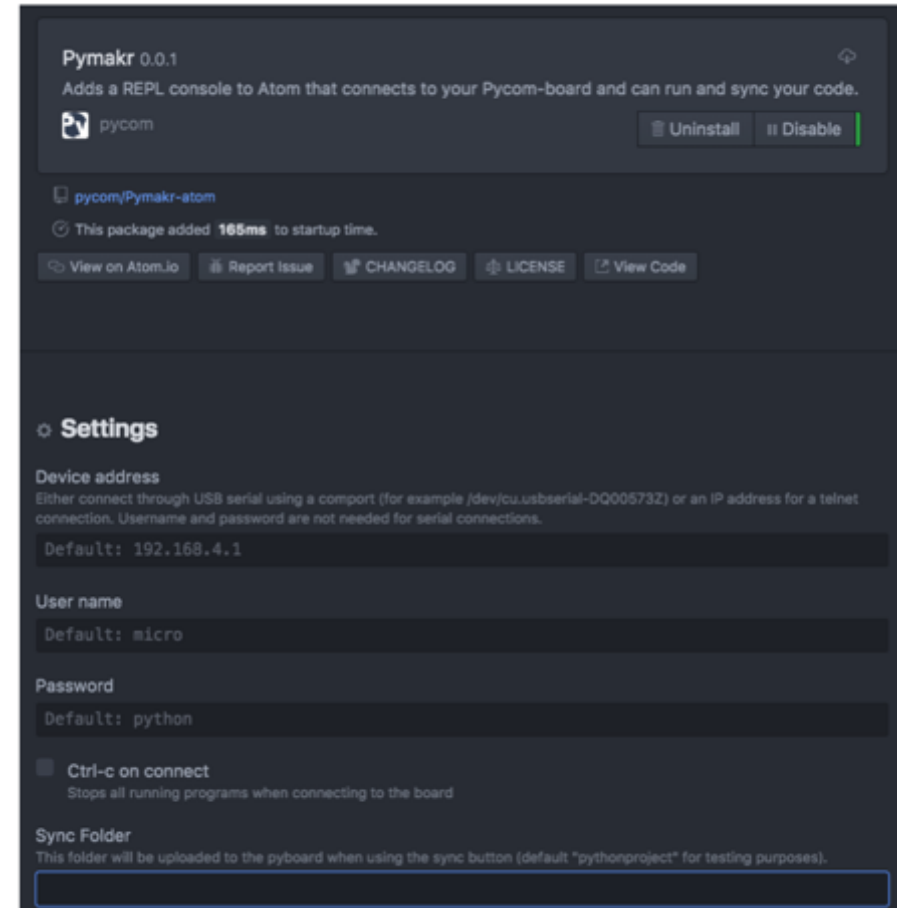
The screenshot shows the VS Code interface with the terminal panel open. The terminal title is "1: Pycom Consol". The output shows "Connecting on /dev/cu.usbserial-DQ0054ES..." followed by a prompt ">>>>".

[Click here](#) if you don't need to connect via telenet

CONNECTING VIA TELNET FOR BOTH ATOM AND VSCODE

if an IP address was provided

1. Ensure the Pycom device is turned on
2. Connect the host computer to the WiFi Access Point named after your board (the SSID will be as follows e.g. **lopy-wlan-xxx**, **wipy-wlan-xxxx** etc)
 - o The password is www.pycom.io and enter **192.168.4.1** as the address
3. The default username and password are **micro** and **python**, respectively
4. Click **connect** in the Pymakr pane and Pymakr will now connect via telnet



REMEMBER!

After writing or pasting any indented code, like a function or a while loop, you will have to press **Enter** up to three times to tell MicroPython the code is to be closed

Code written into the REPL is not saved after the device is switched off/on again

MICROPYTHON 101



MicroPython

- MicroPython is a lean version of Python 3
- Faster and a simpler development process than C
- MicroPython aims to be as compatible with normal Python as possible so that you can transfer code easily

MICROPYTHON 101



MicroPython

- It does not support the entire Python standard library (which can be found on [GitHub](#))
- It has dedicated modules to access hardware
- Pycom uses the ESP32 chipset - it has incredible raw power and a large flash size

TIPS AND TRICKS



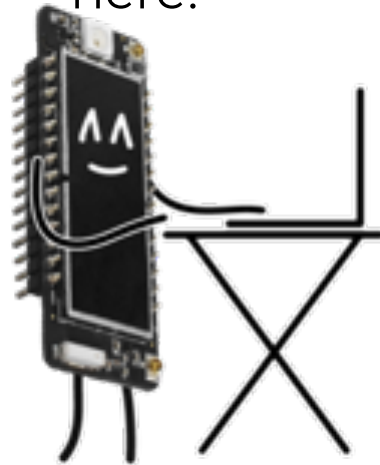
MicroPython

- MicroPython shares a large part of the syntax of Python 3.5
 - Code written for MicroPython should work in a similar manner in Python 3.5
 - A few minor variations – view them as implementation differences

Pycom docs page: <https://docs.pycom.io/gettingstarted/programming/micropython/>

DON'T FORGET YOUR LINES!

MicroPython uses line indentation to denote a block of code. No braces { } here!



The number of spaces in the indentation is variable but all statements within a block must be the same amount

Pycom docs page: <https://docs.pycom.io/gettingstarted/programming/micropython/>






STILL NOT SURE ON MICROPYTHON?

Check out this video for an overview



WHEN BOOTING INTO MICROPYTHON

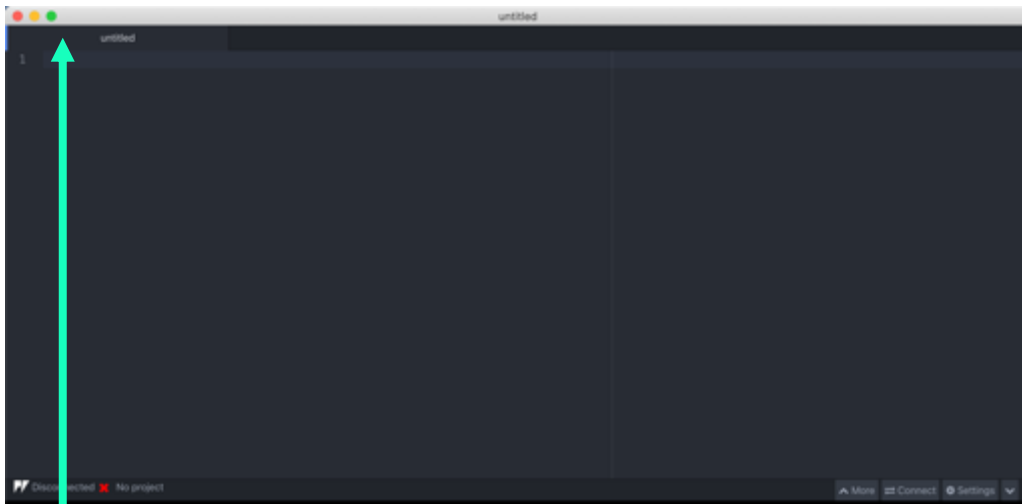
- Two files are executed automatically:
 - boot.py
 - main.py

	cert		Directory
	lib		Directory
	sys		Directory
	boot.py	1734	Python
	main.py	14	Python

Pycom docs page: <https://docs.pycom.io/gettingstarted/programming/micropython/>

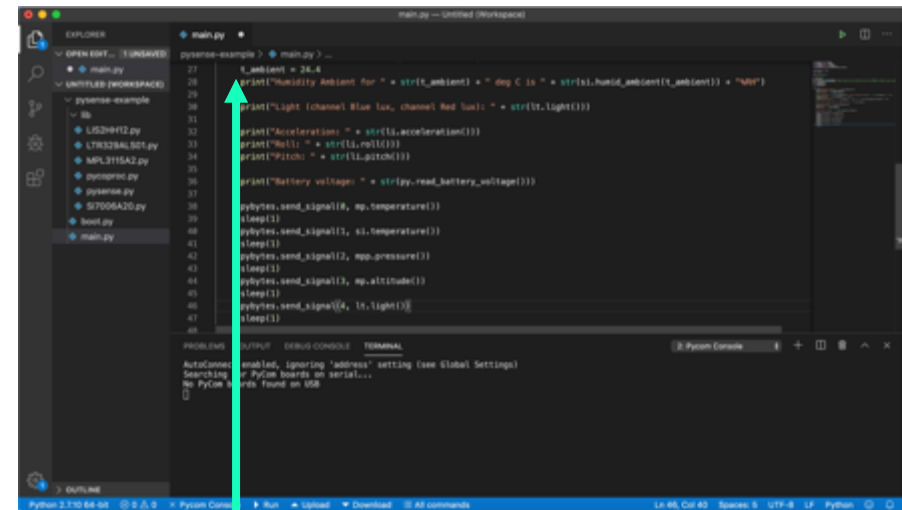
WHERE TO WRITE YOUR CODE

In Atom:



Write your code here!

In VSCode:



Write your code here!

EXAMPLES OF MICROPYTHON

Variable Assignment

Like Python 3.5, variables can be assigned to and referenced

```
variable = "Hello World"  
print(variable)
```

Setting a variable equal to a string and then printing it to the console

Conditional Statements

These allow control over which elements of code run, depending on specific cases.

```
temperature = 15  
target = 10  
if temperature > target:  
    print("Too High!")  
elif temperature < target:  
    print("Too Low!")  
else:  
    print("Just right!")
```

How a temperature sensor might be implemented in code

EXAMPLES OF MICROPYTHON

Loops

Loops allow for you to cycle and repeat your code and functions, assignments etc

- **For** – these allow you to control how many times a block of code runs within a range
- **While** – these allow you to run a loop until a specific conditional is **true/false**

```
x = 0
for y in range(0, 9):
    x += 1
print(x)
```

```
x = 0
while x < 9:
    x += 1
print(x)
```

In this example, the loop checks if **x** is less than **9** each time it passes

EXAMPLES OF MICROPYTHON

Functions

- Functions are blocks of code that are referred to by name.

Example A

```
def add(number1, number2):  
    return number1 + number2  
  
add(1, 2) # expect a result of 3
```

This function takes two numbers and adds them together, outputting the result

Example B

```
def welcome(name):  
    welcome_phrase = "Hello, " + name + "  
    print(welcome_phrase)  
  
welcome("Alex") # expect "Hello, Alex!"
```

This function takes an input name and returns a string containing a welcome phase - 'Hello Alex'

EXAMPLES OF MICROPYTHON

Lists

- A data structure that holds an ordered collection (sequence) of items

```
networks = ['lora', 'sigfox', 'wifi', 'bluetooth', 'lte-m']  
print(networks[2]) # expect 'wifi'
```

Dictionaries

- A dictionary is like an address book
- Keys (names) are associated with values (details)

```
address_book = {'Alex': '2604 Crosswind Drive', 'Joe': '1301 Hillview Drive', 'Chris': '3236 Goldleaf Lane'}  
print(address_book['Alex']) # expect '2604 Crosswind Drive'
```

EXAMPLES OF MICROPYTHON

Tuples

- Tuples are immutable - they cannot be modified once you've started

```
pycom_devices = ('wipy', 'lopy', 'sipy', 'gpy', 'fipy')
print(pycom_devices[0]) # expect 'wipy'
```

Still stuck?
More in-depth tutorials [here](#)

REPL VERSUS SCRIPTS

Please be aware that the examples that are given expect to be executed using Micropython REPL (read-evaluate-print loop)

REPL VERSUS SCRIPTS

Basic Arithmetic

```
1 + 1 # REPL will print out '2' to console  
1 + 1 # Script will not return anything the console  
print(1 + 1) # Both the REPL and a script will return '2' to the console
```


REPL VERSUS SCRIPTS

Calling Methods

```
import ubinascii

ubinascii.hexlify(b'12345') # REPL will print out "b'3132333435'" to the console
ubinascii.hexlify(b'12345') # Script will not return any the console
```

In order to use these functions, but make sure that they don't print out any functions, you'll need to either wrap them in a **print()** statement or assign them to a variable

```
# immediately print to console when using a script
print(1 + 1)
# or save variable to for later
value = 1 + 1
# do something here...
print(value)
```

YOU ARE NOW IOT-READY!

