

# Communication in IoT systems based on the Bluetooth Low Energy standard



**Grażyna Gilewska, PhD**

# IoT - Internet of Things

system of electronic devices with sensors that can collect, process and exchange data through wired or wireless communication:

- smart home devices (washing machines, cleaning robots, fridges, televisions, air purifiers, thermostats, light bulbs),
- industry,
- energy sector,
- medical devices etc.



# IoT - Internet of Things

The condition for the creation of IoT is the compatibility and capabilities of devices in terms of having appropriate sensors and modules responsible for mutual communication.

Due to multiplicity of devices, it is necessary to unambiguously identify a specific device:

- IP addresses
- MAC addresses
- product serial codes

# IoT - Internet of Things

The main requirements of the IoT application to the networks used:

- low energy consumption (large number of sensors deployed in remote locations);
- low fees for using network;
- equipment used should be simple and cheap;
- long-distance transmission resistant to disturbances;
- large network capacity to support a large number of devices;
- adequate level of security of the communication network.

# Bluetooth

Bluetooth is a standard for short-range wireless communication between various electronic devices such as:

- keyboard,
- computer, laptop,
- smartphone,
- wireless headphones
- and many others.

## Solution Areas



AUDIO STREAMING



DATA TRANSFER



## Bluetooth

- Open standard described in the IEEE 802.15.1 specification but no longer supported by the IEEE
- Currently managed by the Bluetooth Special Interest Group (SIG)
- SIG has more than 35,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics.



<https://www.bluetooth.com>



# Bluetooth

- Bluetooth SIG oversees development of the specification, manages the qualification program, and protects the trademarks
- Each manufacturer must comply with the Bluetooth SIG standards in order to market their product as a Bluetooth device
- Technology applies to a network of patents that are licensed for each qualifying device (in 2021 the amount of Bluetooth chips produced was nearly 5 billion units – 5 000 000 000).



## Why the name Bluetooth

- The name "Bluetooth" was proposed in 1997 by Jim Kardach of Intel, one of the founders of the SIG
- Kardach was fascinated by the first Danish king Harald, who contributed to the unification of rival clans (just like the technology under development)
- King Harald was called Bluetooth
- Kardach proposed Bluetooth as the codename for the short-range wireless technology
- He chose the first character from the runic notation of King Harald's nickname (ᚢ) as the logo of the standard.





## Bluetooth - development

In 1999, the Bluetooth SIG published a 1500-page specification for the first version of Bluetooth technology and is constantly working to improve it.

The current version of the specification has more than 3,000 pages and covers the complete system, from the physical layer to the application layer.

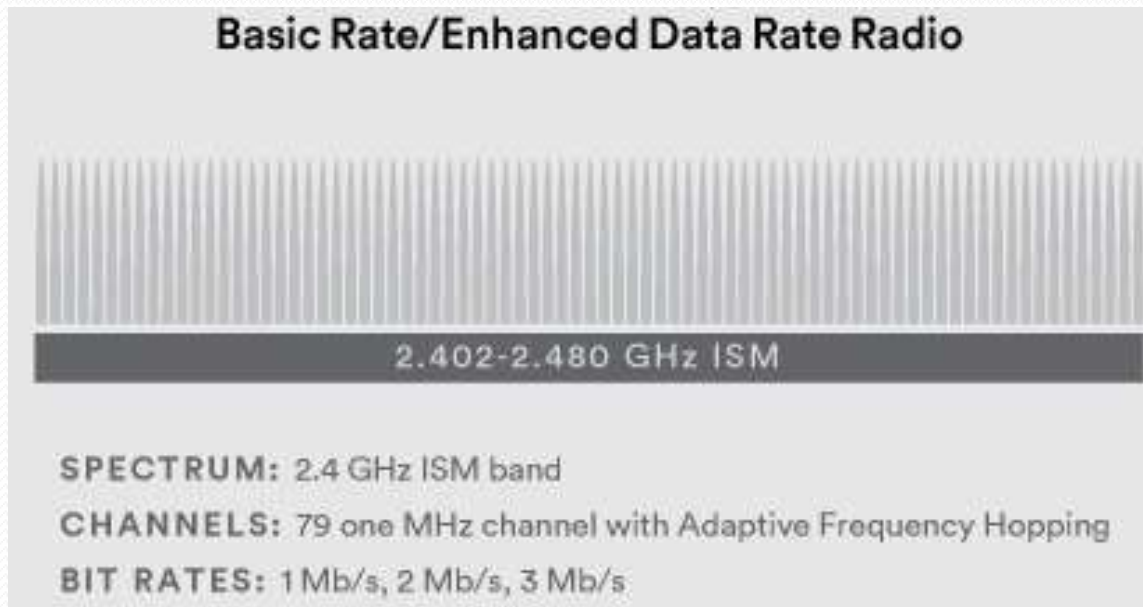


<https://www.bluetooth.com>

## Bluetooth - Classic

The Bluetooth Classic is a low power radio that streams data in the 2.4GHz (2.402 - 2.480 GHz) unlicensed frequency band supporting point-to-point device communication.

The spectrum of this band is divided into 79 channels, each of them has a 1MHz band.

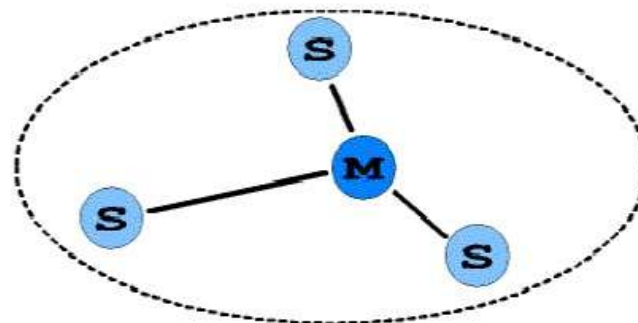


## Bluetooth - Classic

- Bluetooth uses a radio technology called frequency-hopping spread spectrum. Bluetooth divides transmitted data into packets (parts), and transmits each packet on one of 79 Bluetooth channels. It usually performs 1600 hops per second, with adaptive frequency-hopping enabled.
- Each packet is sent on a specific channel, after which the air interface selects a new channel on which the next packet will be sent. Thanks to this process, the message is transmitted over the entire available frequency spectrum.
- For this reason, it is required that the transmitter and receiver are properly tuned so that the receiver knows the hop pattern and can receive the packets and then assemble them into a complete message.

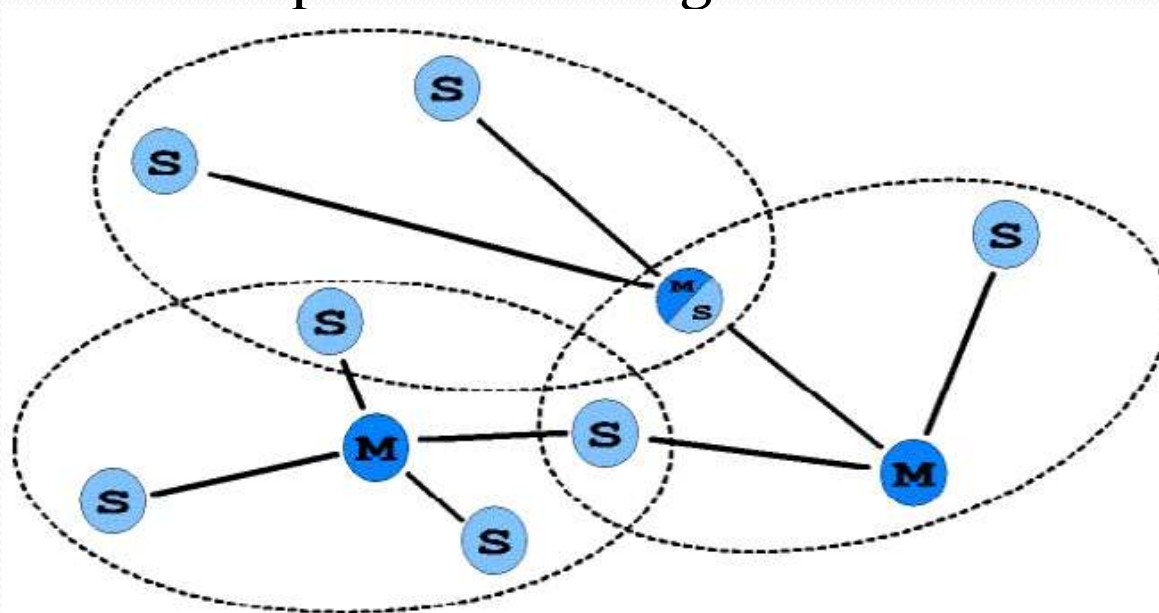
## Bluetooth - Classic

- The Bluetooth network model is a proximity network-based communication model. Which means devices can automatically, spontaneously connect whenever they are within range.
- They are based on the master-slave model, when devices are establishing a connection with each other, one is the master and the other is the slave.
- All slave devices that communicate with the master device create a piconet, where the number of active slaves cannot exceed seven.



## Bluetooth - Classic

- The Bluetooth Core Specification provides for the connection of two or more piconets to form a scatternet, in which certain devices simultaneously play the master /slave role in one piconet and the slave role in another.
- All devices within a given piconet use the clock provided by the master as the base for packet exchange.

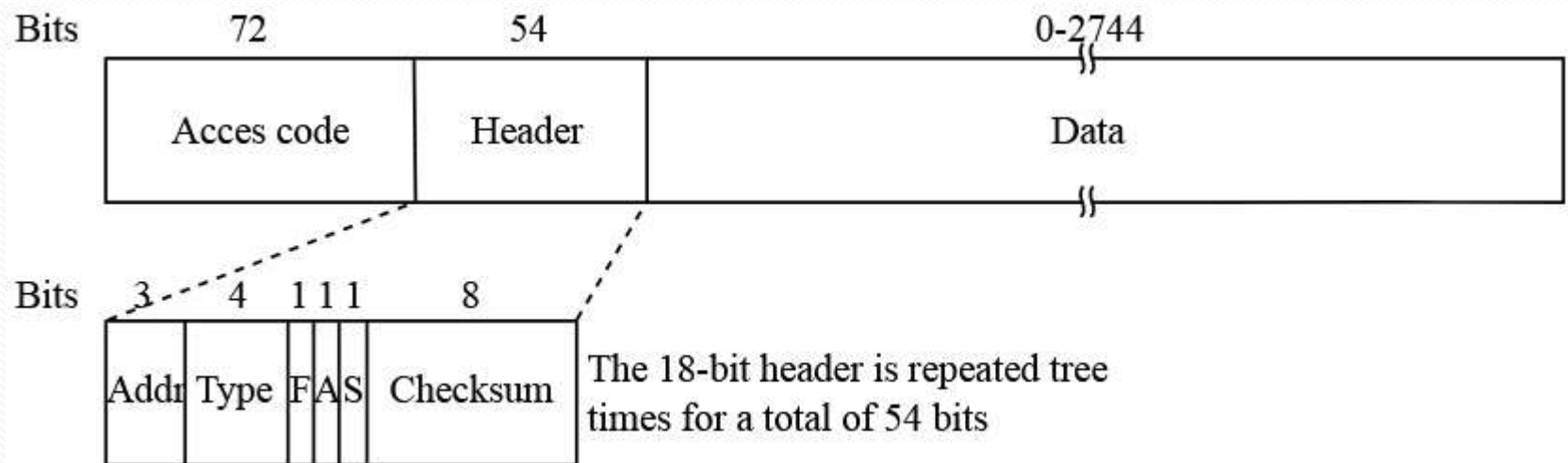


## Bluetooth - Classic

- Each Bluetooth device has a unique address provided by the manufacturer. Which excludes the situation that two different Bluetooth devices, e.g. a mouse and a keyboard, will stop connecting to the computer and start with each other.
- Receiver sensitivity is the measure of the minimum signal strength a receiver can interpret.
- Bluetooth technology specifies that devices must be able to achieve a minimum receiver sensitivity of -70 dBm to -82 dBm, depending on the physical layer used.

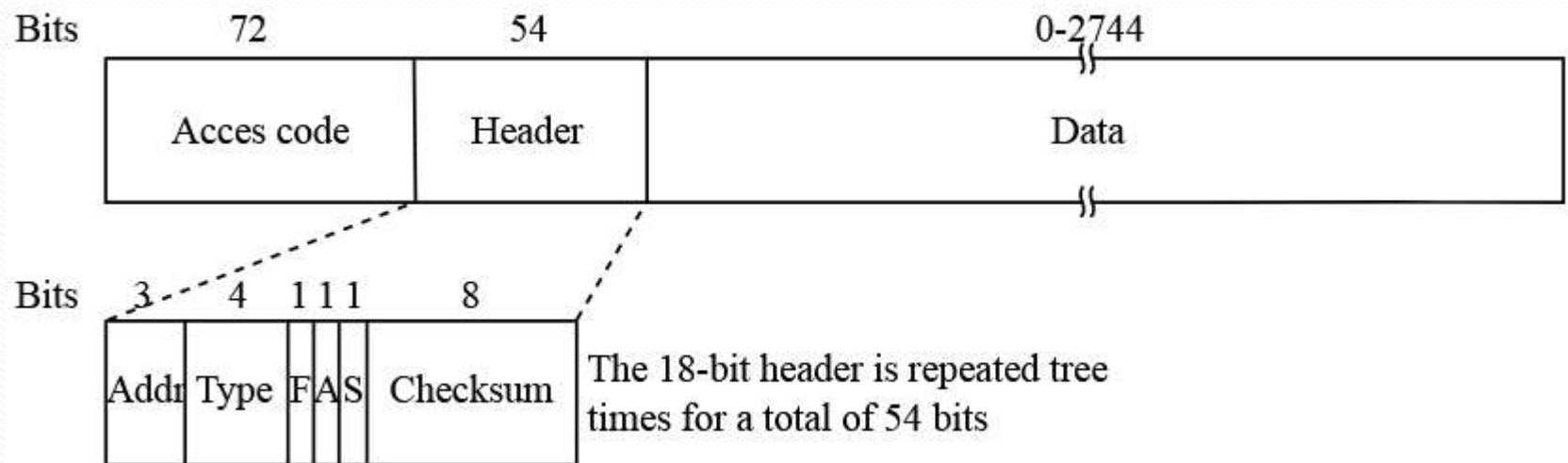


## Bluetooth - frame structure



- Access code - identifies the master for the slave device within the range of two master devices to which the transmission is to take place.
- Header - 54 bits - 18 bits of header repeated three times, on the receiving side, all three copies of each bit are checked.

## Bluetooth - frame structure



- Data - up to 2744 bits containing data (for a 5-slot frame). For a one-shot transmission, the frame contains 240 bits of the data field..

# Bluetooth

Ranges of Bluetooth devices by power-class:

Power class	Maximum output power	Typical range
1	100 mW	100 m
1.5	10 mW	20 m
2	2.5 mW	10 m
3	1 mW	1 m

The effective range varies depending on propagation conditions, antenna configurations and battery conditions.

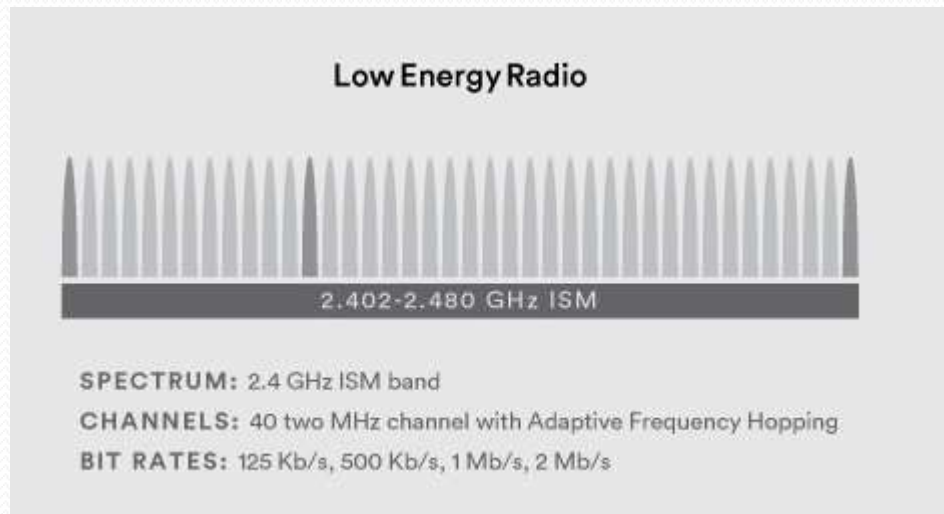
## Comparison of the basic parameters of the Bluetooth version:

Version	Publication date	Maximum bandwidth
Bluetooth 1.0a, 1.0B	1999	21 kb/s
Bluetooth 1.1	2001	124 kb/s
Bluetooth 1.2	2003	721 kb/s
Bluetooth 2.0	2005	2,1 Mb/s
Bluetooth 2.1	2007	2,1 Mb/s
Bluetooth 3.0	2009	24 Mb/s
Bluetooth 4.0 + LE	2009	24 Mb/s lub 1 Mb/s *
Bluetooth 4.1	2013	25 Mb/s lub 1 Mb/s *
Bluetooth 4.2	2014	25 Mb/s lub 1 Mb/s *
Bluetooth 5.0	2016	50 Mb/s lub 2 Mb/s *
Bluetooth 5.1	2019	50 Mb/s lub 2 Mb/s *

\* for wearables devices.

# BLE - Bluetooth Low Energy (Bluetooth LE)

The Bluetooth Low Energy (BLE) radio is designed for very low power operation, uses the same 2.4GHz frequency band, but divides it into 40 channels with a width of 2MHz each.



# BLE - Bluetooth Low Energy (Bluetooth LE)

The Bluetooth Low Energy (BLE) is designed to operate with very low energy consumption.

BLE supports multiple communication topologies:

- point-to-point
- broadcast
- mesh

BLE now includes features that enable one device to determine the presence, distance, and direction of another device.

<https://www.bluetooth.com>



## BLE - Bluetooth Low Energy (Bluetooth LE)

BLE greatly reduces power consumption by turning off the data transmission module when nothing is transmitted.

Contrary to the classic bluetooth module, the transmission is carried out on the basis of intervals and not a permanent connection.

<https://www.bluetooth.com>

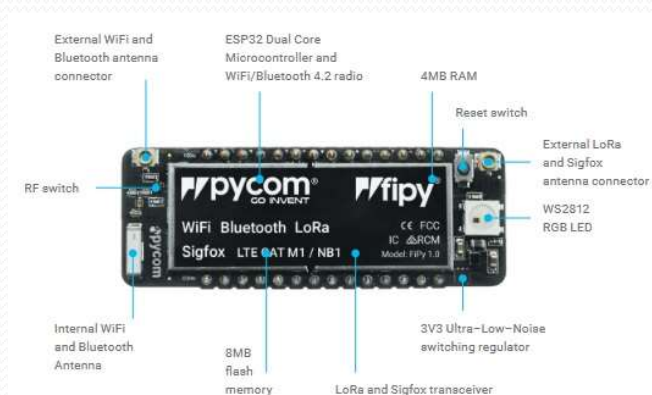
## Applied systems and devices:

IoT systems will be used, consisting of FiPy ESP32 Pycom modules that allow for communication with the Bluetooth network, and a Pysense expansion board with additional sensors of environmental parameters.

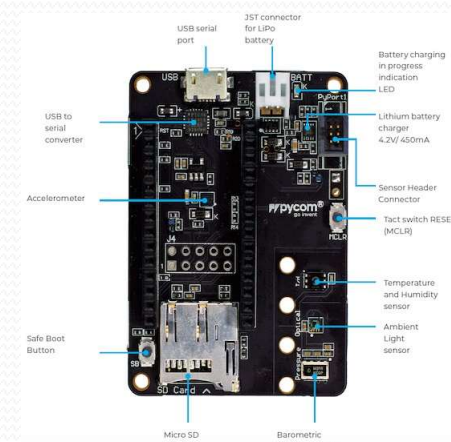
- FiPy with ESP32 Pycom
- Pysense expansion board
- iNode Care Sensor PHT



<https://inode.pl/>



<https://pycom.io>

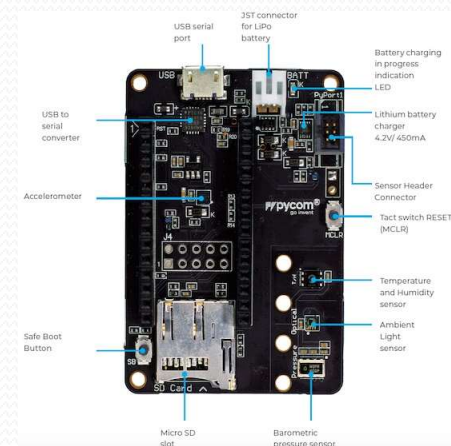


## FiPy with ESP32 Pycom:

enables communication using Bluetooth LE network and Pysense expansion board with additional sensors of environmental parameters.



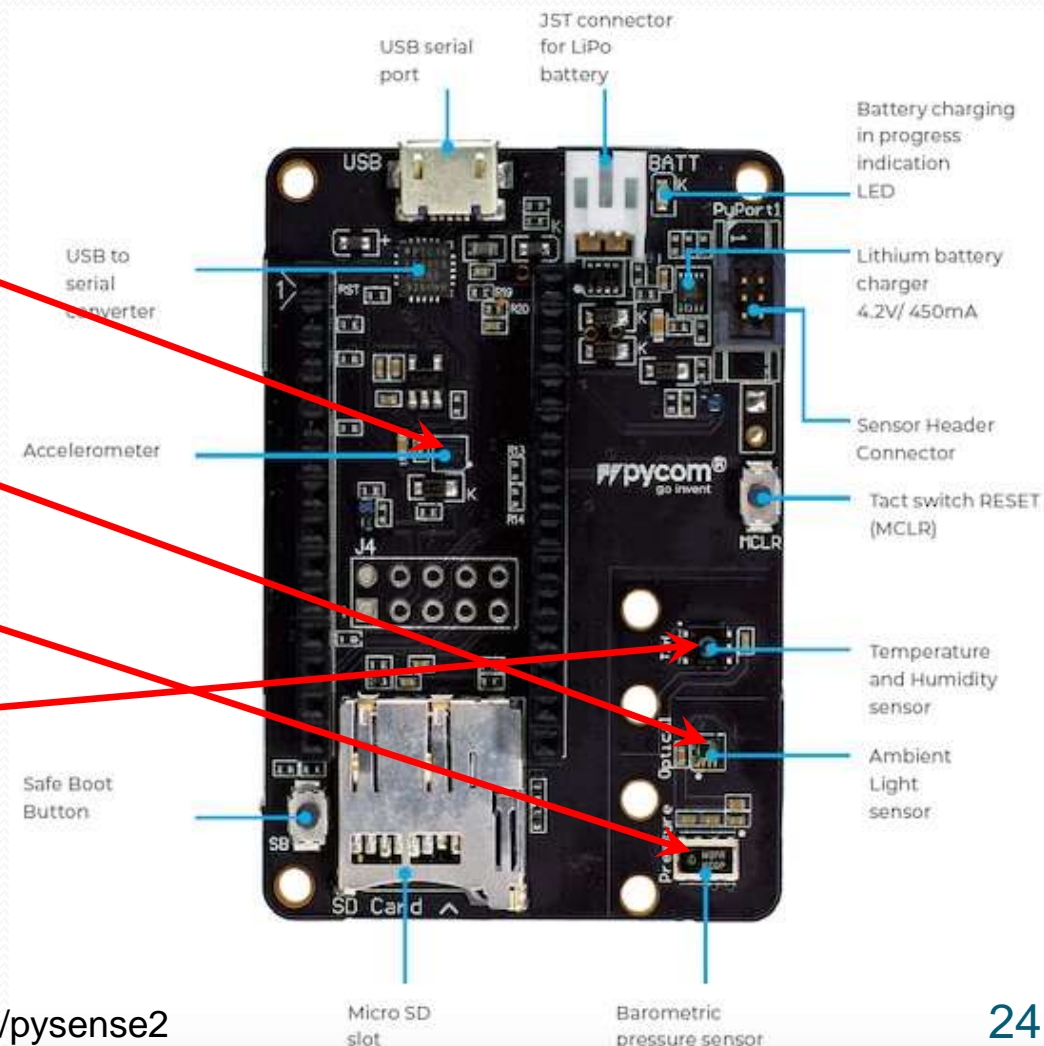
<https://pycom.io>



## Pysense shield:

enables the measurement of environmental parameters using 5 sensors:

- Accelerometer (LIS2HH12)
- Light sensor (LTR329ALS01)
- Pressure sensor (MPL3115A2)
- Temperature / Humidity sensor (SI7006A20)



# Preparation of environment for device programming

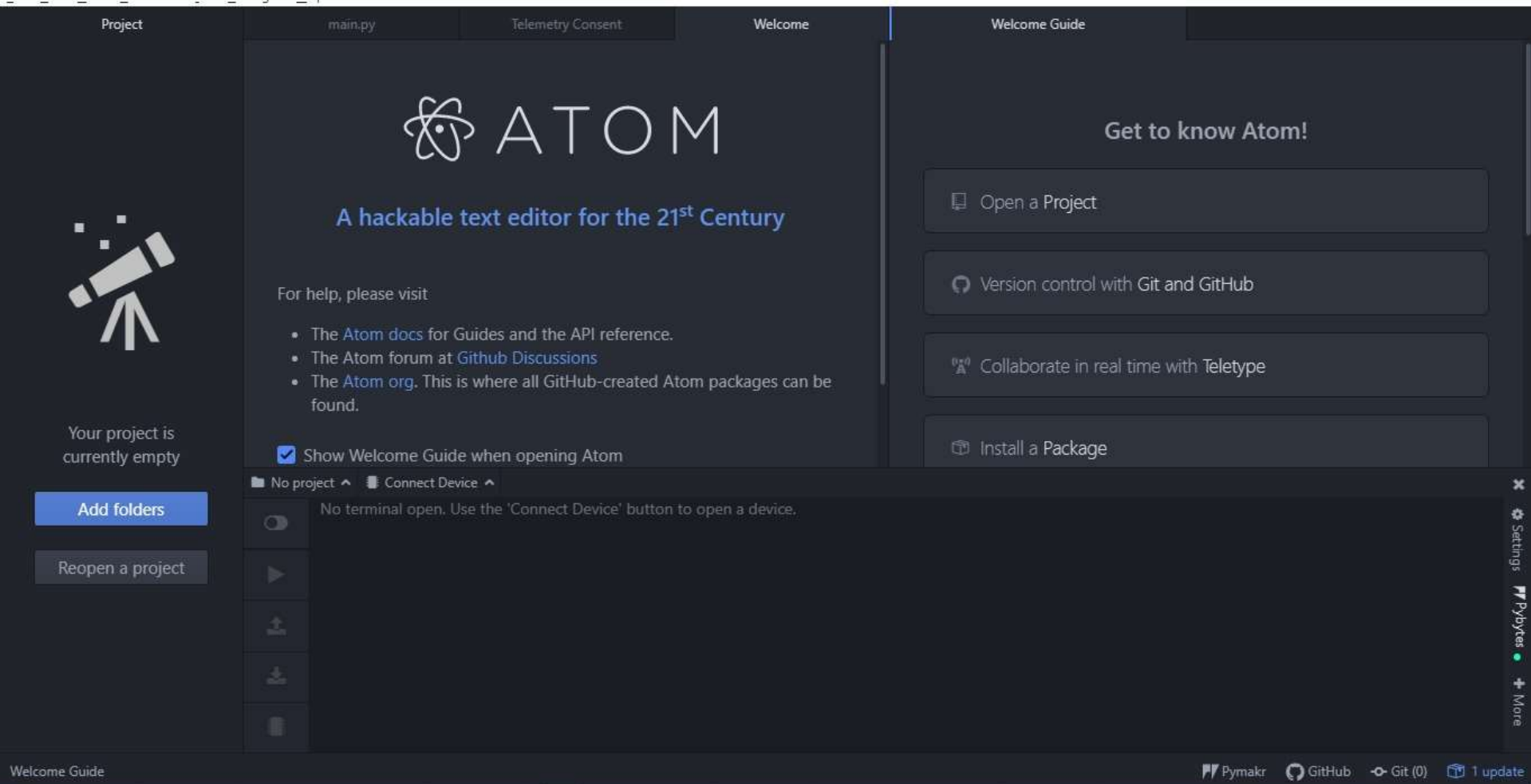
Use Atom Text Editor & Pymakr Plugin environment:

- download and install Atom (<https://atom.io>)
- install the official Pycom Pymakr Plugin via Atom
- connect FiPy module to computer via USB
- test some basic MicroPython commands



# Atom and Pycom Pymakr Plugin:

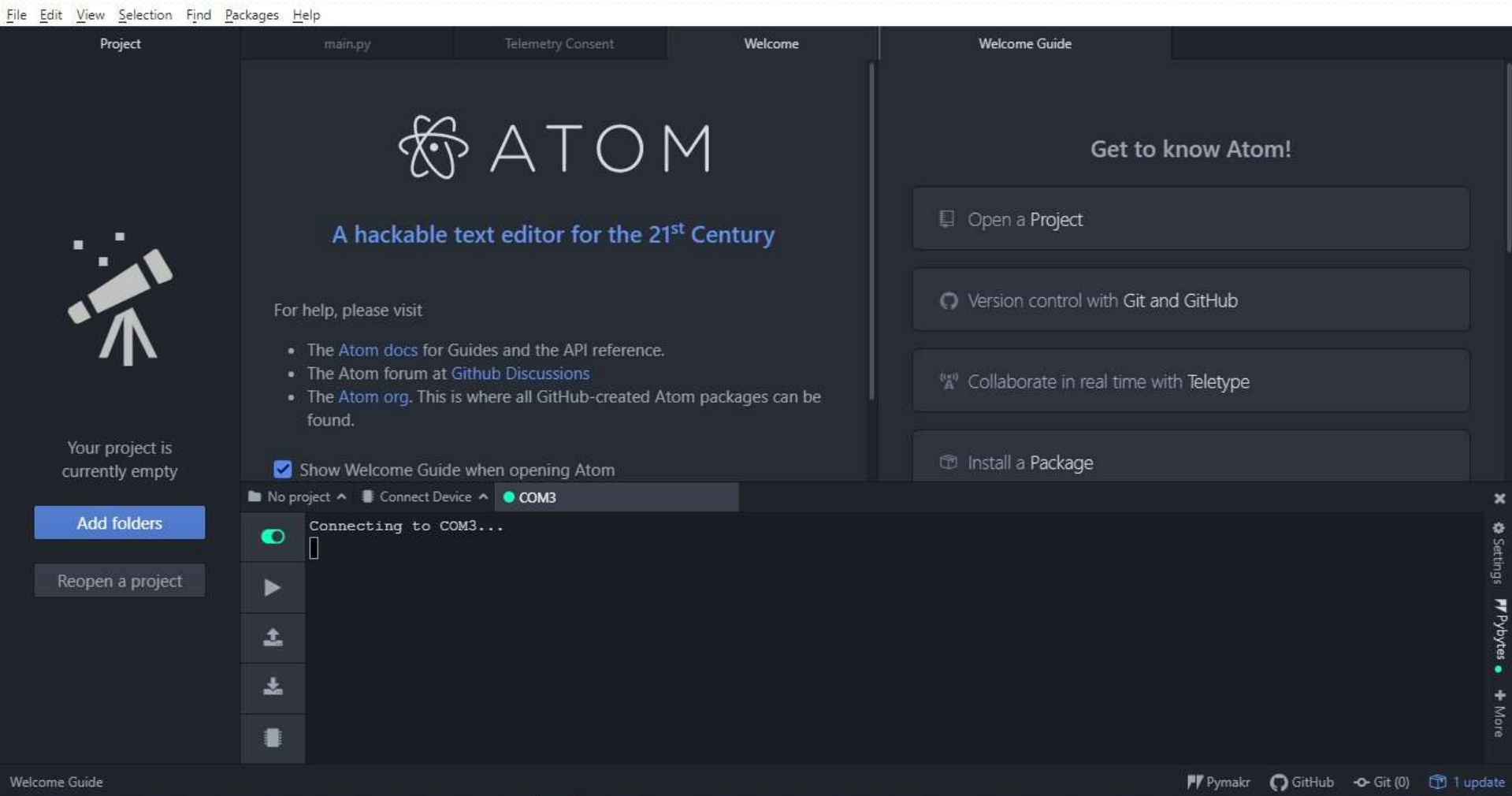
File Edit View Selection Find Packages Help



The screenshot shows the Atom text editor interface. The main window displays the Atom logo and the text "A hackable text editor for the 21<sup>st</sup> Century". Below this, there is a section for help links and a checkbox for "Show Welcome Guide when opening Atom". The left sidebar contains a "Project" panel with a telescope icon and buttons for "Add folders" and "Reopen a project". The right sidebar shows a "Welcome Guide" with buttons for "Open a Project", "Version control with Git and GitHub", "Collaborate in real time with Teletype", and "Install a Package". The bottom status bar shows the "Pycom Pymakr" plugin and other system information.



# Connected FiPy module to computer via USB (COM3)



The screenshot shows the Atom text editor interface. The main window displays the Atom logo and the text "A hackable text editor for the 21<sup>st</sup> Century". Below this, there is a section for help links: "For help, please visit" followed by a list of links: "The Atom docs for Guides and the API reference.", "The Atom forum at Github Discussions", and "The Atom.org. This is where all Github-created Atom packages can be found." A checkbox labeled "Show Welcome Guide when opening Atom" is checked.

On the right side, there is a "Welcome Guide" panel with the title "Get to know Atom!". It contains four buttons: "Open a Project", "Version control with Git and GitHub", "Collaborate in real time with Teletype", and "Install a Package".

At the bottom, there is a terminal window. The terminal shows the text "Connecting to COM3..." and a green light icon. The terminal is titled "COM3" and has a "Connect Device" button next to it.

The bottom status bar shows the following information: "Welcome Guide", "Pymakr", "GitHub", "Git (0)", and "1 update".

## Devices used for tasks:



# Test some basic MicroPython commands:

File Edit View Selection Find Packages Help

Project

Telemetry Consent

Welcome

Welcome Guide



A hackable text editor for the 21<sup>st</sup> Century

For help, please visit

- The [Atom docs](#) for Guides and the API reference.
- The Atom forum at [Github Discussions](#)
- The [Atom org](#). This is where all GitHub-created Atom packages can be found.

☒ Show Welcome Guide when opening Atom

No project ^ Connect Device ^ COM3

```
>>> 7/3
2.333333
>>> 7//3
2
>>> x=7
>>> y=2
>>> x+y
9
>>> 
```

Get to know Atom!

Open a Project

Version control with Git and GitHub

Collaborate in real time with Teletype

Install a Package

Your project is currently empty

Add folders

Reopen a project

Settings  
Python  
+ More

Welcome Guide

Pymakr

GitHub

Git (0)

1 update

# Test some basic MicroPython commands:

File Edit View Selection Find Packages Help

Project

Telemetry Consent

Welcome

Welcome Guide



A hackable text editor for the 21<sup>st</sup> Century

For help, please visit

- The [Atom docs](#) for Guides and the API reference.
- The Atom forum at [Github Discussions](#)
- The [Atom org](#). This is where all GitHub-created Atom packages can be found.

☒ Show Welcome Guide when opening Atom

No project ^ Connect Device ^ COM3

```
>>>
>>> word='Python'
>>> print(word[0:2])
Py
>>> print(word[2:])
thon
>>> print(word[:-2])
Pyth
>>>
```

Get to know Atom!

Open a Project

Version control with Git and GitHub

Collaborate in real time with Teletype

Install a Package

Your project is currently empty

Add folders

Reopen a project

Welcome Guide

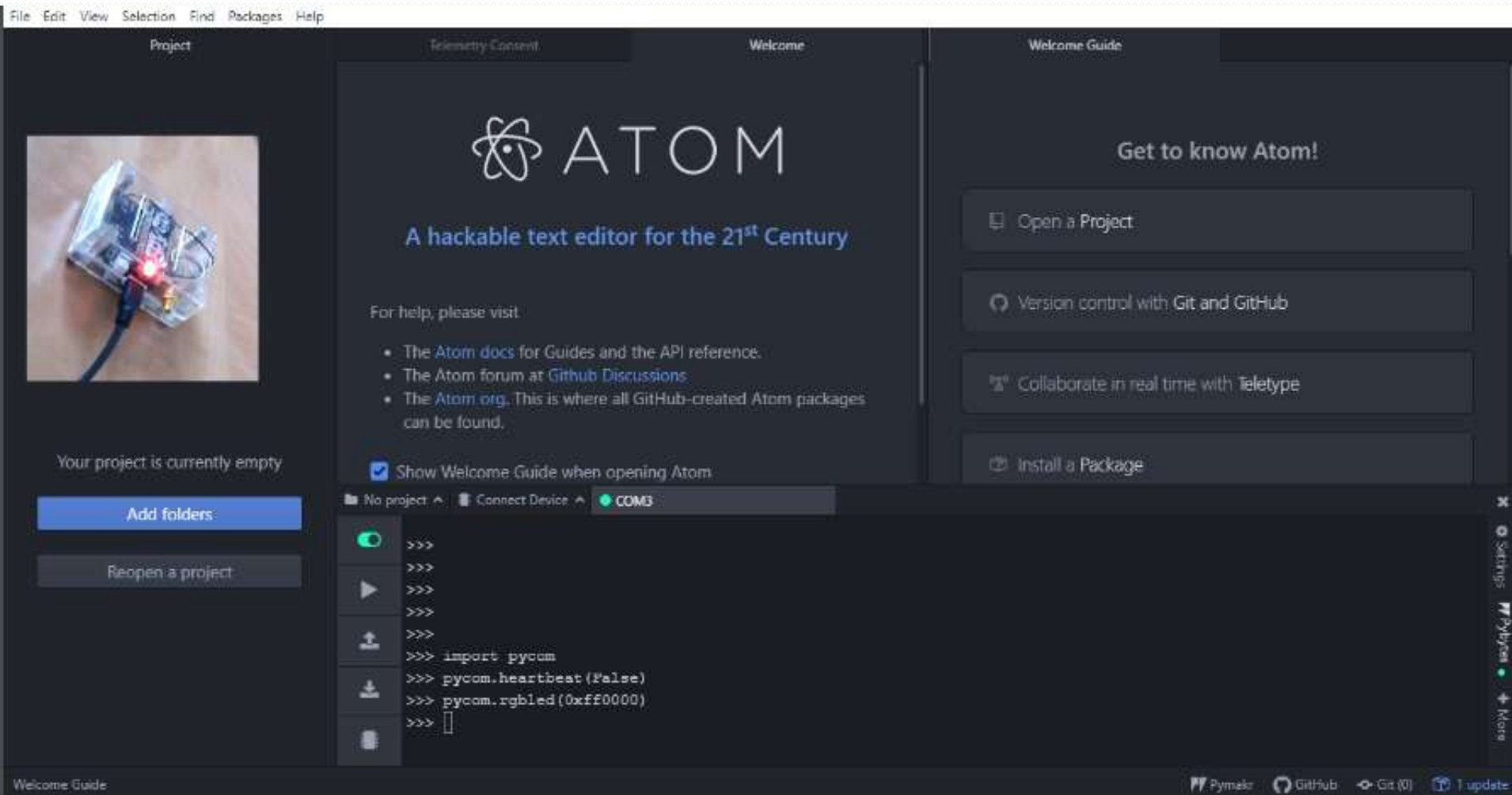
Pymakr

GitHub

Git (0)

1 update

# Programming of the FiPy module (set red color fo LED)



The screenshot shows the Atom IDE interface. On the left, there's a sidebar with a 'Project' tab showing a photo of a microcontroller board with a red LED. Below it, buttons for 'Add folders' and 'Reopen a project' are visible. The main area displays the 'Welcome' screen with the 'ATOM' logo and the text 'A hackable text editor for the 21<sup>st</sup> Century'. It provides links for help, including 'Atom docs', 'Atom forum', and 'Atom.org'. A 'Show Welcome Guide when opening Atom' checkbox is checked. On the right, a 'Welcome Guide' sidebar lists actions like 'Open a Project', 'Version control with Git and GitHub', 'Collaborate in real time with Teletype', and 'Install a Package'. At the bottom, a terminal window shows the following Python code:

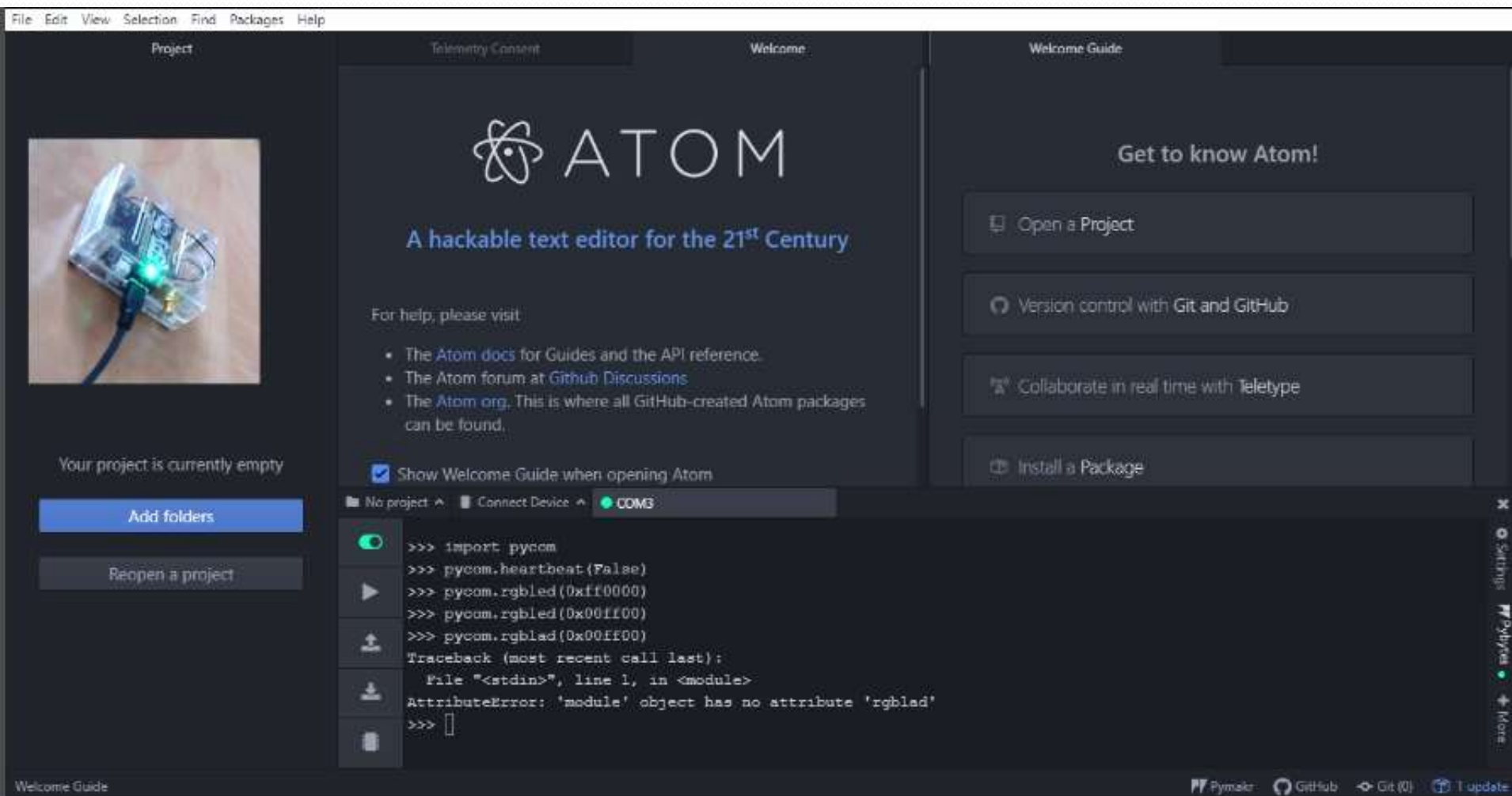
```
>>>
>>>
>>>
>>>
>>> import pycom
>>> pycom.heartbeat(False)
>>> pycom.rgbled(0xff0000)
>>> 
```

The terminal window also shows a status bar at the bottom with 'Pymakr', 'GitHub', 'Git (0)', and '1 update'.



# Programming of the FiPy module

## (set green color fo LED and response to an incorrect command)



The screenshot shows the Atom IDE interface. On the left, there's a 'Project' panel with a photo of a microcontroller board (likely an Arduino) and a green LED. Below it, buttons for 'Add folders' and 'Reopen a project' are visible. The main editor area displays the 'ATOM' logo and the text 'A hackable text editor for the 21<sup>st</sup> Century'. Below this, there's a list of links for help and documentation. A checkbox 'Show Welcome Guide when opening Atom' is checked. At the bottom, a status bar shows 'No project', 'Connect Device', and 'COM3'. The terminal window at the bottom right shows the following code and error:

```
>>> import pycom
>>> pycom.heartbeat(False)
>>> pycom.rgbled(0xff0000)
>>> pycom.rgbled(0x00ff00)
>>> pycom.rgbled(0x00ff00)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute 'rgbled'
>>>
```

On the right side, there's a 'Welcome Guide' panel with buttons for 'Open a Project', 'Version control with Git and GitHub', 'Collaborate in real time with Teletype', and 'Install a Package'.



# Create project directory:

File Edit View Selection Find Packages Help

Project

Telemetry Consent

Welcome

Welcome Guide

> Bluetooth



A hackable text editor for the 21<sup>st</sup> Century

For help, please visit

- The [Atom docs](#) for Guides and the API reference.
- The Atom forum at [Github Discussions](#)
- The [Atom org](#). This is where all GitHub-created Atom packages can be found.

☒ Show Welcome Guide when opening Atom

Bluetooth Connect Device COM3

```
>>> pycom.heartbeat(False)
>>> pycom.rgbled(0xFF0000)
>>> pycom.rgbled(0x00FF00)
>>> pycom.rgbled(0x00FF00)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute 'rgbled'
>>>
>>> |
```

Get to know Atom!

Open a Project

Version control with Git and GitHub

Collaborate in real time with Teletype

Install a Package

Welcome Guide

Pymakr

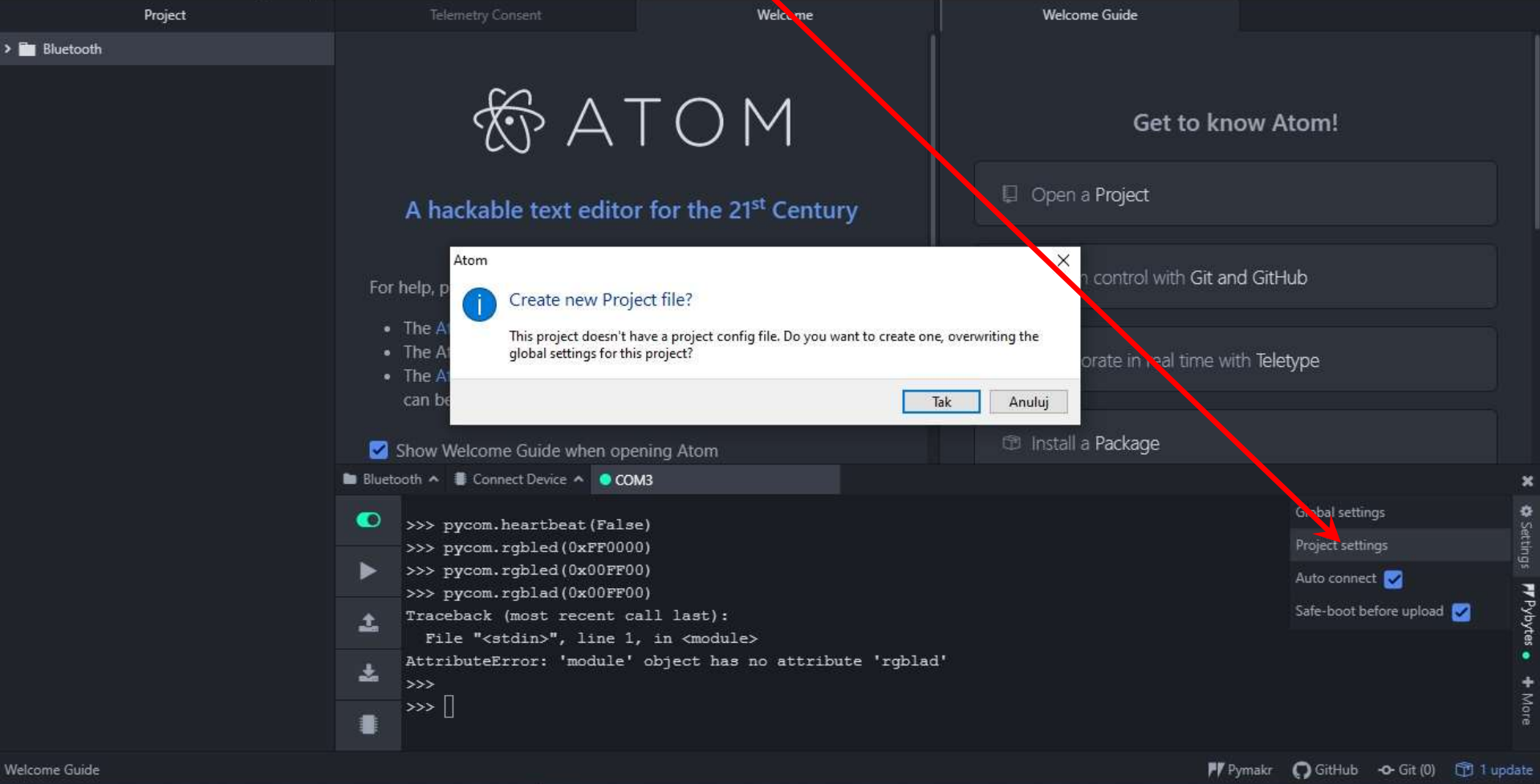
GitHub

Git (0)

1 update

# Configure project settings

File Edit View Selection Find Packages Help



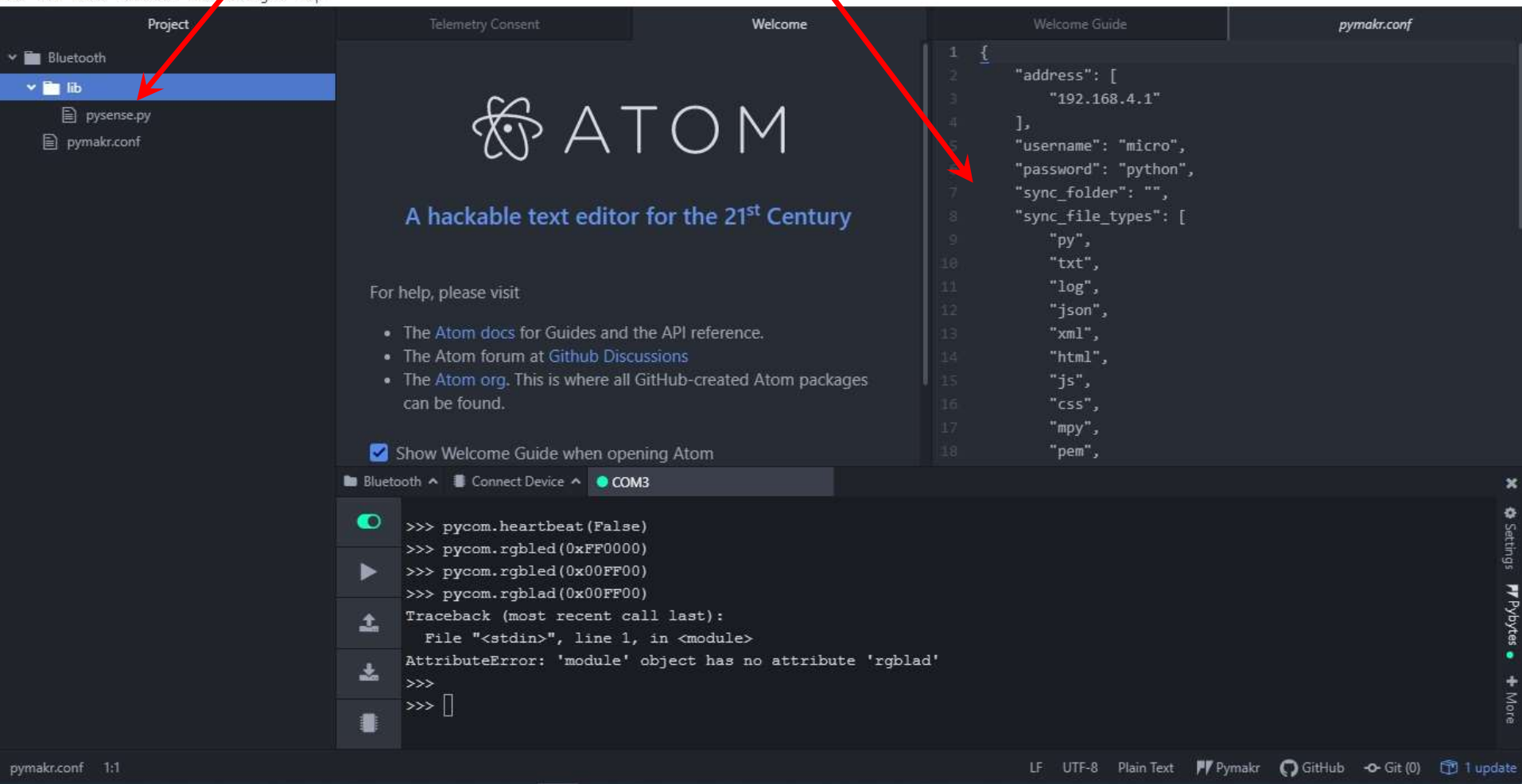
The screenshot shows the Atom text editor interface. A dialog box titled 'Atom' is open, asking 'Create new Project file?' with the message: 'This project doesn't have a project config file. Do you want to create one, overwriting the global settings for this project?'. The dialog has 'Tak' (Yes) and 'Anuluj' (Cancel) buttons. A red arrow points from the title 'Configure project settings' to the 'Project settings' option in the 'Settings' menu on the right. The background shows the Atom welcome screen with the text 'ATOM A hackable text editor for the 21st Century' and a terminal window at the bottom displaying a traceback error: 'AttributeError: 'module' object has no attribute 'rgblad''.

Welcome Guide

Pymakr GitHub Git (0) 1 update

# The contents of the config file and additional library

File Edit View Selection Find Packages Help



The screenshot shows the Atom text editor interface. On the left, the 'Project' sidebar shows a file explorer with a 'lib' folder containing 'pysense.py' and 'pymakr.conf'. A red arrow points from the title 'The contents of the config file and additional library' to the 'lib' folder. The main editor area displays the 'Welcome' screen for Atom, which includes the Atom logo and the text 'A hackable text editor for the 21<sup>st</sup> Century'. A red arrow points from the title to the 'pymakr.conf' file in the file explorer. The right sidebar shows the 'pymakr.conf' file with the following content:

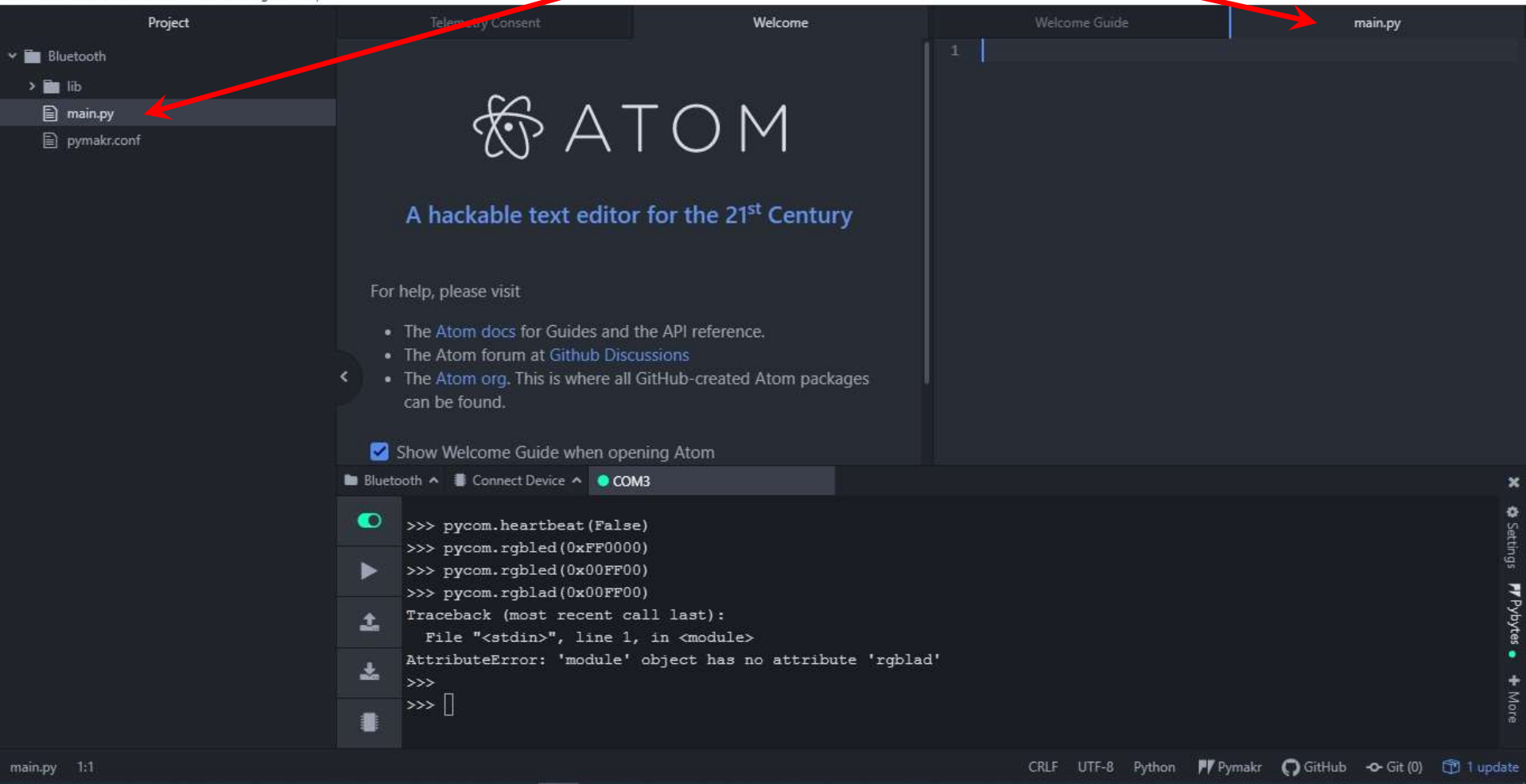
```
1 {
2   "address": [
3     "192.168.4.1"
4   ],
5   "username": "micro",
6   "password": "python",
7   "sync_folder": "",
8   "sync_file_types": [
9     "py",
10    "txt",
11    "log",
12    "json",
13    "xml",
14    "html",
15    "js",
16    "css",
17    "mpy",
18    "pem",
19  ]
20 }
```

At the bottom, the 'Console' panel shows a Python REPL session with the following output:

```
>>> pycom.heartbeat(False)
>>> pycom.rgbled(0xFF0000)
>>> pycom.rgbled(0x00FF00)
>>> pycom.rgbled(0x00FF00)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute 'rgblad'
>>>
>>> []
```

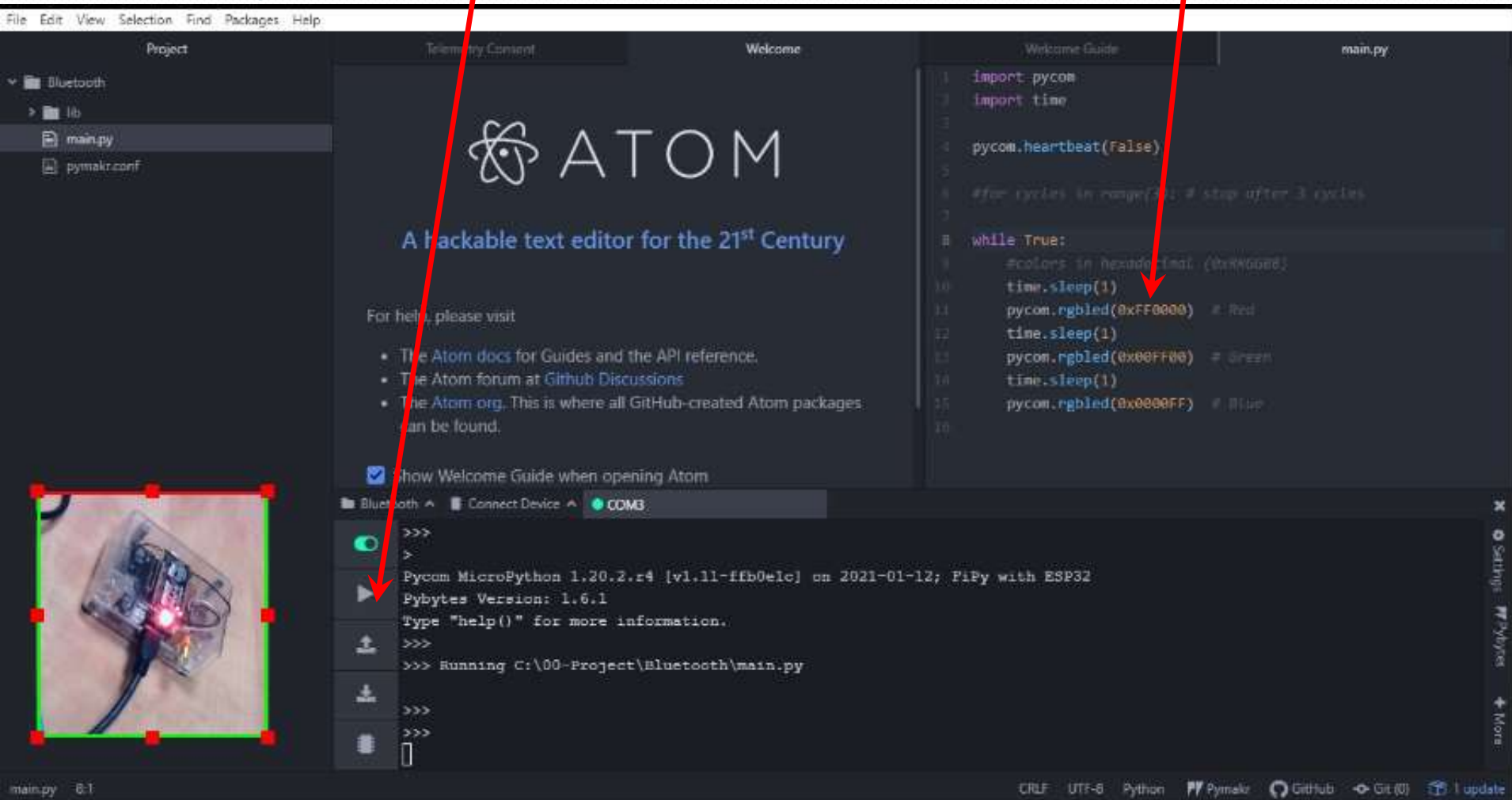
# Create the main project file

File Edit View Selection Find Packages Help



The screenshot shows the Atom text editor interface. The left sidebar displays the project structure with a folder named 'Bluetooth' containing a subfolder 'lib' and two files: 'main.py' and 'pymakr.conf'. A red arrow points from the title 'Create the main project file' to the 'main.py' file. The main editor area shows the 'Welcome' screen with the ATOM logo and the text 'A hackable text editor for the 21<sup>st</sup> Century'. Below this, there are links for help and a checkbox for 'Show Welcome Guide when opening Atom'. The bottom status bar shows the current file is 'main.py' at line 1, column 1. The bottom right corner shows various icons for settings, Pymakr, GitHub, and Git.

# Create and run project that changes LED color every 1 second



The screenshot shows the Atom text editor interface. The left sidebar displays the project structure with files: `main.py` and `pymkr.conf`. The main editor area shows the `main.py` file with the following code:

```
1 import pycom
2 import time
3
4 pycom.heartbeat(False)
5
6 #for cycles in range(10): # stop after 3 cycles
7
8 while True:
9     #colors in hexadecimal (0xXXXX)
10    time.sleep(1)
11    pycom.rgbled(0xFF0000) # Red
12    time.sleep(1)
13    pycom.rgbled(0x00FF00) # Green
14    time.sleep(1)
15    pycom.rgbled(0x0000FF) # Blue
16
```

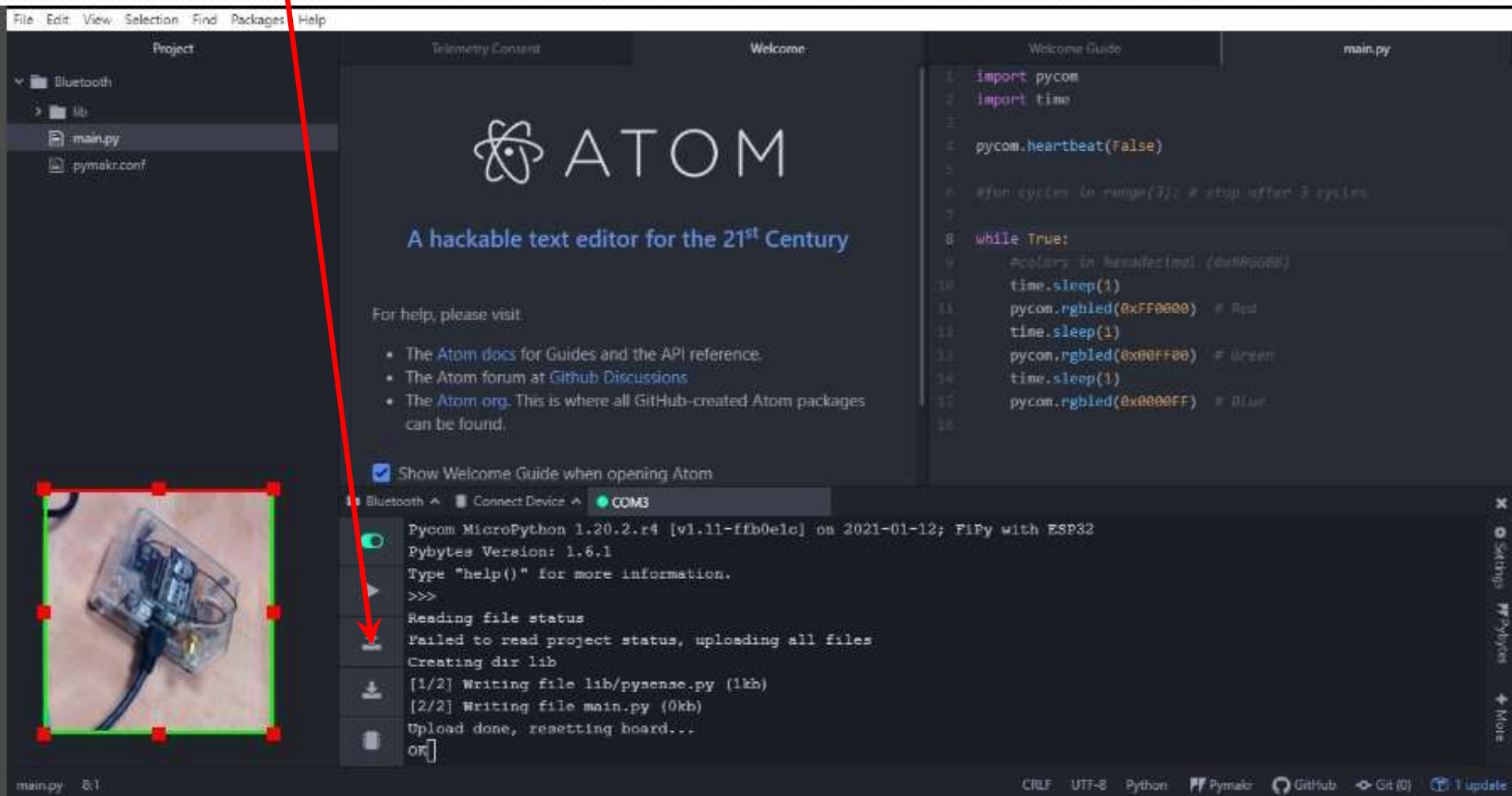
At the bottom left, a red box highlights the `Run` button (a green play icon). A red arrow points from the `time.sleep(1)` line in the code to the `Run` button. Below the `Run` button, the terminal output shows the execution status:

```
>>>
>
Pycom MicroPython 1.20.2.r4 [v1.11-fffb0e1c] on 2021-01-12; PiPy with ESP32
Pybytes Version: 1.6.1
Type "help()" for more information.
>>>
>>> Running C:\00-Project\Bluetooth\main.py
>>>
>>>
```

In the bottom left corner, there is a small inset image of a physical LED module with a red light, enclosed in a red box.



# Upload project to device



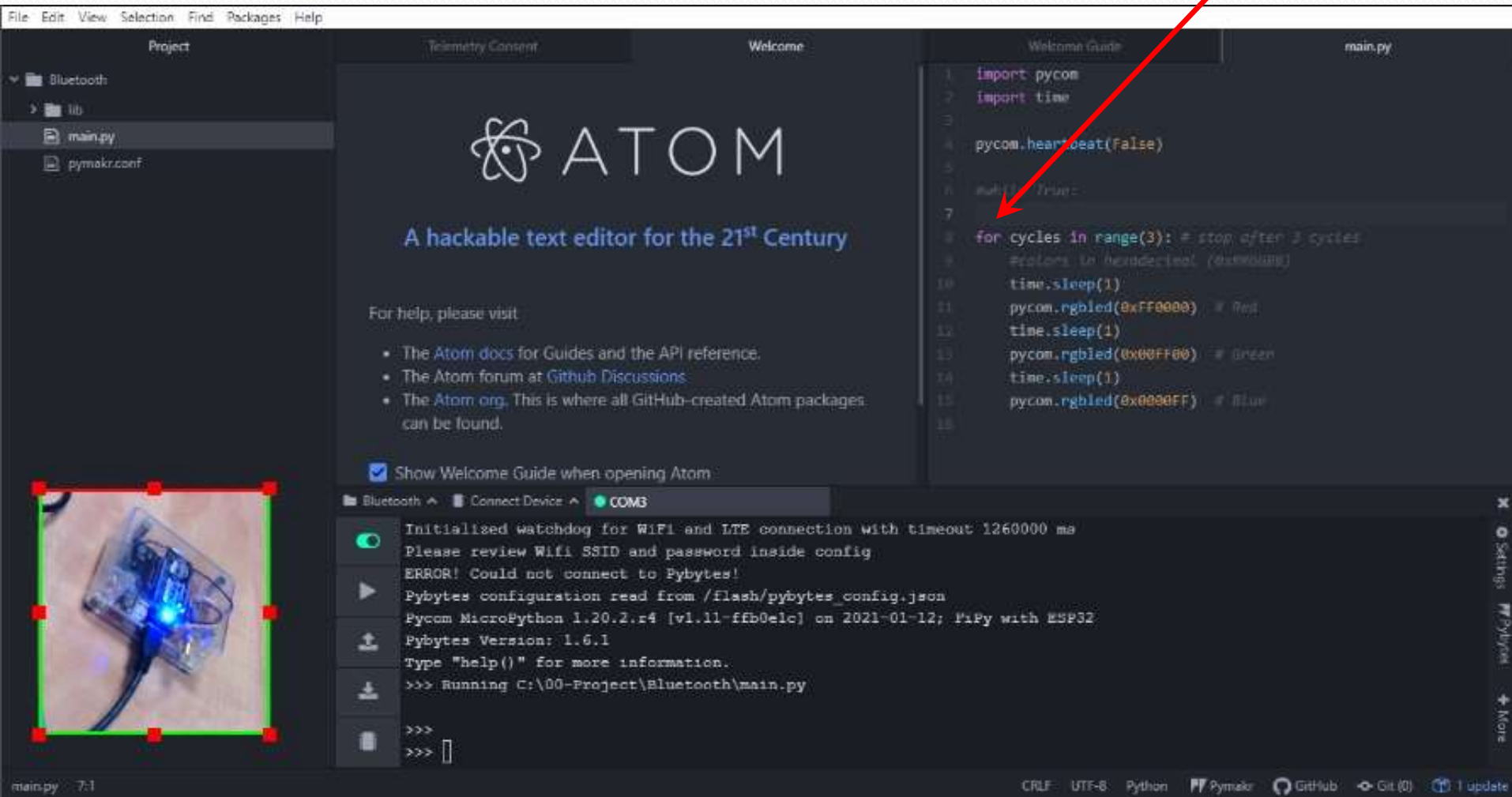
The screenshot shows the Atom IDE interface. The left sidebar displays the project structure with files like `main.py` and `pymakr.conf`. The main editor area shows the Atom logo and a welcome message. The right sidebar displays the code for `main.py`, which includes imports for `pycom` and `time`, and a loop that sets RGB LED colors (Red, Green, Blue) with a 1-second delay between each color change.

A red arrow points from the title "Upload project to device" to the "Connect Device" button in the bottom left corner of the IDE. The "Connect Device" button is located in the bottom left corner of the IDE, next to the "Bluetooth" and "COM3" labels. The "Connect Device" button is highlighted with a red border.

The bottom status bar shows the upload progress: "Pycom MicroPython 1.20.2.r4 [v1.11-ffb0e1c] on 2021-01-12; PiPy with ESP32", "Pybytes Version: 1.6.1", "Type 'help()' for more information.", "Reading file status", "Failed to read project status, uploading all files", "Creating dir lib", "[1/2] Writing file lib/pysense.py (1kb)", "[2/2] Writing file main.py (0kb)", "Upload done, resetting board...", and "OK".



# Changed project: infinite While-loop to For-loop



The screenshot shows the Atom text editor with a project named 'Bluetooth'. The file 'main.py' is open, showing the following Python code:

```
1 import pycom
2 import time
3
4 pycom.heartbeat(False)
5
6 while True:
7
8     for cycles in range(3): # stop after 3 cycles
9         # colors in hexadecimal (0x000000)
10            time.sleep(1)
11            pycom.rgbled(0xFF0000) # Red
12            time.sleep(1)
13            pycom.rgbled(0x00FF00) # Green
14            time.sleep(1)
15            pycom.rgbled(0x0000FF) # Blue
16
```

A red arrow points from the title 'Changed project: infinite While-loop to For-loop' to the code change. The console output shows the following messages:

```
Initialized watchdog for WiFi and LTE connection with timeout 1260000 ms
Please review Wifi SSID and password inside config
ERROR! Could not connect to Pybytes!
Pybytes configuration read from /flash/pybytes_config.json
Pycom MicroPython 1.20.2.r4 [v1.11-ff0e1c] on 2021-01-12; PiPy with ESP32
Pybytes Version: 1.6.1
Type "help()" for more information.
>>> Running C:\00-Project\Bluetooth\main.py
>>>
>>>
```

# Task topic: Connect to a BLE Device and read data

File Edit View Selection Find Packages Help

Project

main.py

Welcome

Bluetooth

lib

main.py

pymkr.conf

ATOM

A hackable text editor for the 21<sup>st</sup> Century

For help, please visit

- The Atom docs for Guides and the API reference.
- The Atom forum at [Github discussions](#)
- The Atom org. This is where all GitHub-created Atom packages can be found.

☒ Show Welcome Guide when opening Atom

Bluetooth Connect Device COM3



```
Retrieved data:
Mac address of device: b'd0f01843e444'
Name of device: iNode-43E444
Pressure: 1008.188 hPa
Temperature: 24.98671 °C
Humidity: 25.35681 %

Pycam MicroPython 1.20.2.r4 [v1.11-ffb0e1c] on 2021-01-12; PiPy with ESP32
Pybytes Version: 1.6.1
Type "help()" for more information.
>>> 
```

Welcome

Pymkr

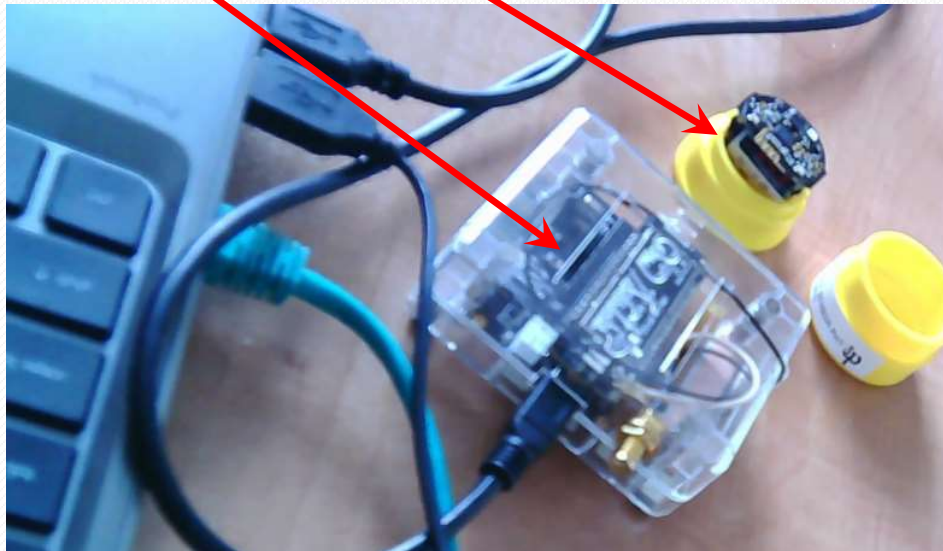
GitHub

Git (0)

Update

## Devices used for task:

- FiPy with ESP32 Pycom and Pysense expansion board
- iNode Care Sensor PHT



## Use property constructors:

- `class network.Bluetooth`

## methods:

- `bluetooth.start_scan`
- `bluetooth.get_adv()`
- `bluetooth.resolve_adv_data`

## and commands:

- `ubinascii.hexlify`

## Realise the task in 3 parts:

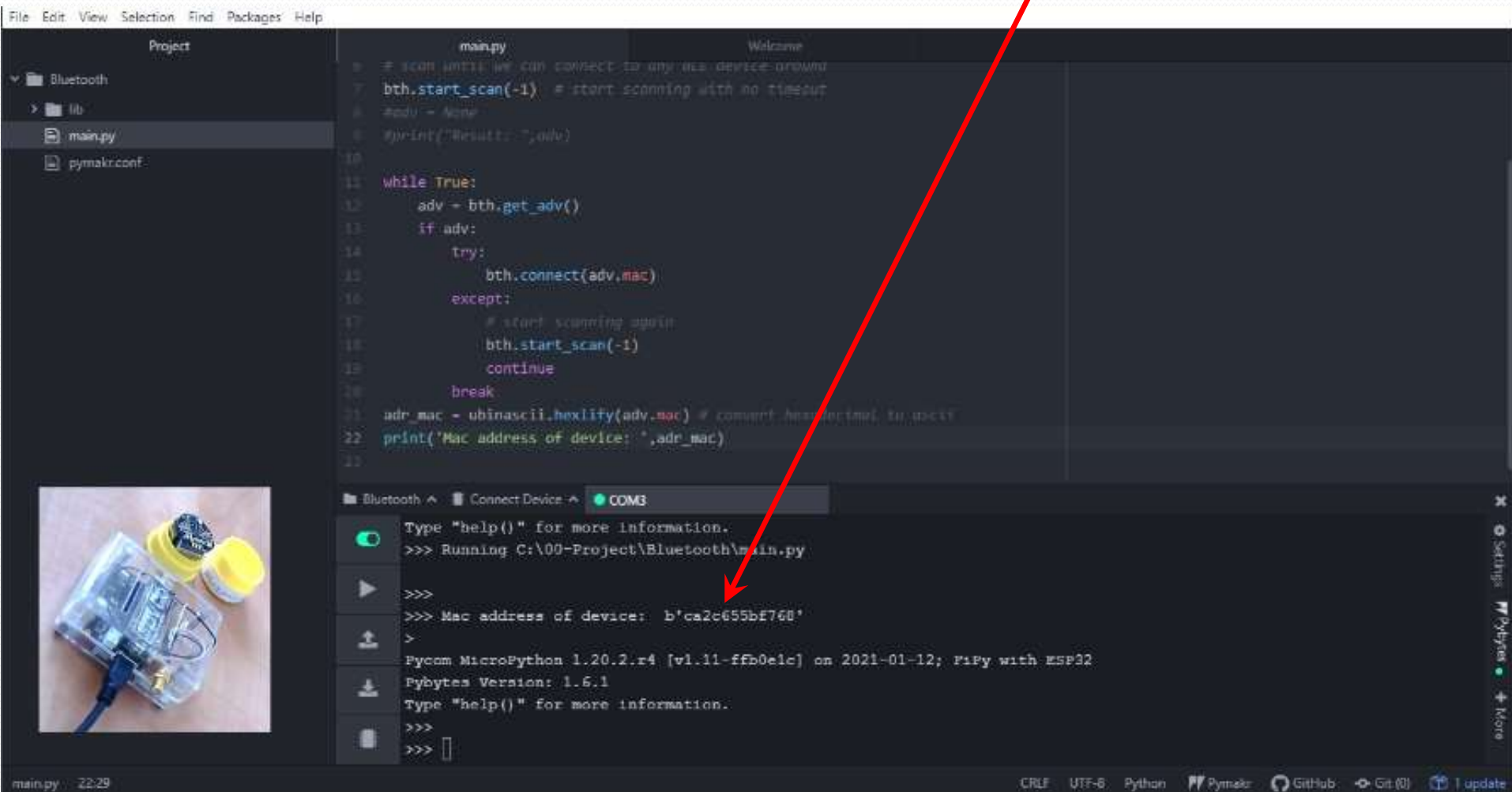
- Connecting to a device that is sending advertisements and receiving advertisement data
- Connecting to the device sending advertisements and receiving device data
- Connecting to the device sending advertisements and receiving measurement data

## Part 1 Scan until we can connect to any BLE device around:

```
from network import Bluetooth
import ubinascii
bth = Bluetooth() # create a Bluetooth object
bth.start_scan(-1) # start scanning with no timeout
while True:
    adv = bth.get_adv()
    if adv:
        try:
            bth.connect(adv.mac)
        except:
            # start scanning again
            bth.start_scan(-1)
            continue
        break
adr_mac = ubinascii.hexlify(adv.mac) # convert hexadecimal to ascii
print('Mac address of device: ',adr_mac)
```



# Part 1 Scan until we can connect to any BLE device around:



The screenshot shows a code editor with a Python script named `main.py` in a project named `Bluetooth`. The script is designed to scan for BLE devices until it can connect to one. The code is as follows:

```

1 # scan until we can connect to any BLE device around
2
3 bth.start_scan(-1) # start scanning with no timeout
4
5 adv = None
6
7 #print("Results: ",adv)
8
9
10
11 while True:
12     adv = bth.get_adv()
13     if adv:
14         try:
15             bth.connect(adv.mac)
16         except:
17             # start scanning again
18             bth.start_scan(-1)
19             continue
20         break
21
22 adr_mac = ubinascii.hexlify(adv.mac) # convert hex decimal to ascii
23 print('Mac address of device: ',adr_mac)
24
25

```

The terminal output shows the execution of the script. A red arrow points from the title to the output line: `>>> Mac address of device: b'ca2c655bf760'`. The terminal also shows the version information for Pycam MicroPython and Pybytes.

```

Type "help()" for more information.
>>> Running C:\00-Project\Bluetooth\main.py
>>>
>>> Mac address of device: b'ca2c655bf760'
>
Pycam MicroPython 1.20.2.r4 [v1.11-ffb0e1c] on 2021-01-12; PiPy with ESP32
Pybytes Version: 1.6.1
Type "help()" for more information.
>>>
>>>

```



## Part 1 Scan until we can connect to any BLE device around:

We used the method:

```
bluetooth.get_adv()
```

gets tuple which has the following structure:

```
(mac, addr_type, adv_type, rssi, data)
```

mac - mac address of the device that sent the advertisement

addr\_type - address type

adv\_type - advertisement type received

rssi - signed integer with the signal strength of the advertisement

data - contains the complete 31 bytes of the advertisement message

## Part 1 Write a script for the task:

Using the method of:

```
bluetooth.get_adv()
```

read the following parameters of the BLE device we connected to:

```
mac , addr_type, adv_type, rssi
```

### Constants:

**Advertisement type:** `Bluetooth.CONN_ADV`,  
`Bluetooth.CONN_DIR_ADV`, `Bluetooth.DISC_ADV`,  
`Bluetooth.NON_CONN_ADV`, `Bluetooth.SCAN_RSP`

**Address type:** `Bluetooth.PUBLIC_ADDR`, `Bluetooth.RANDOM_ADDR`,  
`Bluetooth.PUBLIC_RPA_ADDR`, `Bluetooth.RANDOM_RPA_ADDR`

<https://docs.pycom.io/firmwareapi/pycom/network/bluetooth/>

## Part 2 Connect to BLE device and get requested data type:

```
from network import Bluetooth
import time
import pycom
import ubinascii

bth = Bluetooth() # create a Bluetooth object
bth.start_scan(-1) # start scanning with no timeout
while True:
    adv = bth.get_adv()
    if adv:
        try:
            bth.connect(adv.mac)
        except:
            # start scanning again
            bth.start_scan(-1)
            continue
        break
    adr_manuf = bth.resolve_adv_data(adv.data, bth.ADV_NAME_CMPL)
    print('\nnName of device: ' + str(adr_manuf) + '\n')
    pycom.heartbeat(False)
    time.sleep(1)
    pycom.rgbled(0xFF0000) # Red
```

## Part 2 Connect to BLE device and get requested data type:

File Edit View Selection Find Packages Help

Project

Bluetooth

lib

main.py

pytnakr.conf

main.py

Welcome

```
6  bth = Bluetooth() # create a Bluetooth object
7  bth.start_scan(-1) # start scanning with no timeout
8
9  while True:
10     adv = bth.get_adv()
11     if adv:
12         try:
13             bth.connect(adv.mac)
14         except:
15             # start scanning again
16             bth.start_scan(-1)
17             continue
18         break
19     adr_manuf = bth.resolve_adv_data(adv.data, bth.ADV_NAME_CPL)
20     print('\nName of device: ' + str(adr_manuf) + '\n')
21     pycom.heartbeat(False)
22     time.sleep(1)
23     pycom.rgbled(0xFF0000) # off
```

Bluetooth Connect Device COM3

```
>>>
>>>
Name of device: iNode-43E444
>
Pycom MicroPython 1.20.2.r4 [v1.11-ffbf0e1c] on 2021-01-12; PiPy with ESP32
Pybytes Version: 1.6.1
Type "help()" for more information.
>>>
>>>
```



main.py 20:24

CRLF UTF-8 Python Pytnakr GitHub Git (0) 1 update

## Part 2 Connect to BLE device and get requested data type:

We used the method:

```
bluetooth.resolve_adv_data(data, data_type)
```

returns the requested `data_type` if present

`data` - bytes object with the complete advertisement data

`data_type` - data type to resolve from from the advertisement data.



## Part 2 Write a script for the task:

Using the method of:

```
bluetooth.resolve_adv_data(data, data_type)
```

read the following parameters of the BLE device we connected to:

flag, short name, manufacturer data

### Constants:

```
Advertisement data type: Bluetooth.ADV_FLAG,  
Bluetooth.ADV_16SRV_PART, Bluetooth.ADV_16SRV_CMPL,  
Bluetooth.ADV_32SRV_PART, Bluetooth.ADV_32SRV_CMPL,  
Bluetooth.ADV_128SRV_PART, Bluetooth.ADV_128SRV_CMPL,  
Bluetooth.ADV_NAME_SHORT, Bluetooth.ADV_NAME_CMPL,  
Bluetooth.ADV_TX_PWR, Bluetooth.ADV_DEV_CLASS,  
Bluetooth.ADV_SERVICE_DATA, Bluetooth.ADV_APPEARANCE,  
Bluetooth.ADV_ADV_INT, Bluetooth.ADV_32SERVICE_DATA,  
Bluetooth.ADV_128SERVICE_DATA,  
Bluetooth.ADV_MANUFACTURER_DATA
```

<https://docs.pycom.io/firmwareapi/pycom/network/bluetooth/>

## Part 3 Connect to BLE device with known MAC number and receive measurement data:

```
from network import Bluetooth
import ubinascii
import struct
import pycom
import time

bt = Bluetooth()
bt.start_scan(10)      # starts scanning and stop after 10 seconds
pycom.heartbeat(False)
pycom.rgbled(0xFF0000)  # Red

while bt.isscanning():
    adv = bt.get_adv()
    if adv and ubinascii.hexlify(adv.mac) == b'd0f01843e444':
        time.sleep(1)
        pycom.rgbled(0x0000FF)  # Blue
        print('\nRetrieved data:')
        adv_manuf = bt.resolve_adv_data(adv.data, bt.ADV_NAME_CMPL)
        print('\nName of device: ' + str(adv_manuf))
        data_manuf = ubinascii.hexlify(bt.resolve_adv_data(adv.data, bt.ADV_MANUFACTURER_DATA))
        press=data_manuf[12:16]
        press2 = ubinascii.unhexlify(press)
        press3=(struct.unpack("<H",press2)[0])/16
        print('\nPressure: ' + str(press3) + ' hPa\n')
pycom.rgbled(0x00FF00)  # Green
```

## Part 3 Connect to BLE device with known MAC number and receive measurement data:

File Edit View Selection Find Packages Help

Project

Bluetooth

lib

main.py

pymakr.conf

main.py

```

1 bt.start_scan(10) # starts scanning and stop after 10 seconds
2 pycom.heartbeat(False)
3 pycom.rgbled(0xFF0000) # Red
4
5
6
7
8
9
10
11 while bt.is_scanning():
12     adv = bt.get_adv()
13     if adv and ubinascii.hexlify(adv.mac) == b'd0f01843e444':
14         time.sleep(1)
15         pycom.rgbled(0x0000FF) # Blue
16         print('\nRetrieved data:')
17         adv_manuf = bt.resolve_adv_data(adv.data, bt.ADV_NAME_CMPL)
18         print('\nName of device: ' + str(adv_manuf))
19         data_manuf = ubinascii.hexlify(bt.resolve_adv_data(adv.data, bt.ADV_MANUFACTURER_DATA))
20         press = data_manuf[12:16]
21         press2 = ubinascii.unhexlify(press)
22         press3 = (struct.unpack("<f", press2)[0])/16
23         print('\nPressure: ' + str(press3) + ' hPa\n')
24
25 pycom.rgbled(0x00FF00) # Green

```

Bluetooth Connect Device COM3

```

Retrieved data:
Name of device: iNode_43E444
Pressure: 1009.938 hPa
Pycom MicroPython 1.20.2.r4 [v1.11-fffb0e1c] on 2021-01-12; PiPy with ESP32
Pybytes Version: 1.6.1
Type "help()" for more information.
>>>

```



main.py 2:17

CRLF UTF-8 Python Pymakr GitHub Git (0) 1 update

## Part 3 Connect to BLE device with known MAC number and receive measurement data:

We used iNode Manufacturer Specific Data:

iNode Care Sensor PHT (0x9D)

12 9D 01 C0 00 00 4F 3E 3F 19 95 12 03 00 3C C0 91 99 BB A2 CC 23 AC 82

12	bit 2: rtto bit 3: lowBattery
9D	iNode Care Sensor PHT
01 c0	groupsAndBattery (uint16le);
00 00	Alarms (uint16le);
4f 3e	rawPressure (uint16le);
3f 19	rawTemperature (uint16le);
95 12	rawHumidity (uint16le);
03 00	rawTime1 (uint16le);
3c c0	rawTime2 (uint16le)
91 99 bb a2 cc 23 ac 82	AES128 digital signature for the above data

<https://inode.pl>

## Part 3 Write a script for the task:

Using iNode Manufacturer Specific Data iNode Care Sensor PHT

converting hexadecimal to ascii by `ubinascii`  
decoding Little endian by `struct`

Calculation of humidity ( $H[\%]$ ):

$$H = (125 * \text{rawHumidity} * 4 / 65536) - 6$$

Calculation of temperature ( $T[^\circ\text{C}]$ ):

$$T = (175.72 * \text{rawTemperature} * 4 / 65536) - 46.85$$

<https://inode.pl>

Connect to BLE device with known name: `iNode-43E444`  
and read the following parameters of the BLE device we connected to:

Temperature, Humidity