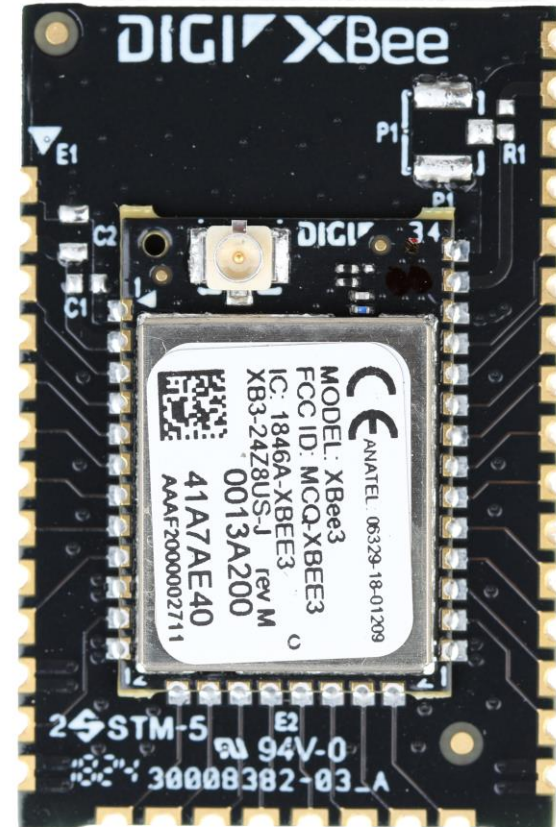


# XBee Radio Modules



## Coding XBee Module Applications Using MicroPython

January 28, 2020

Fred Eady

Presented by:

**DesignNews**

Information Classification: General

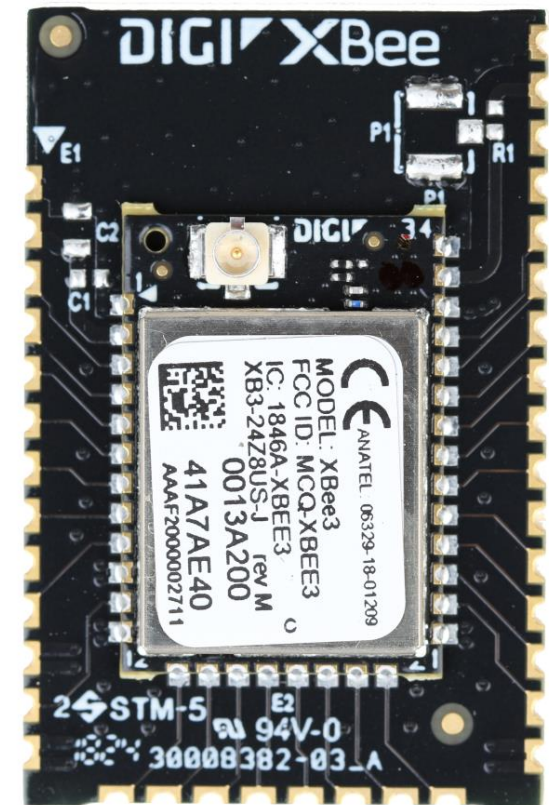
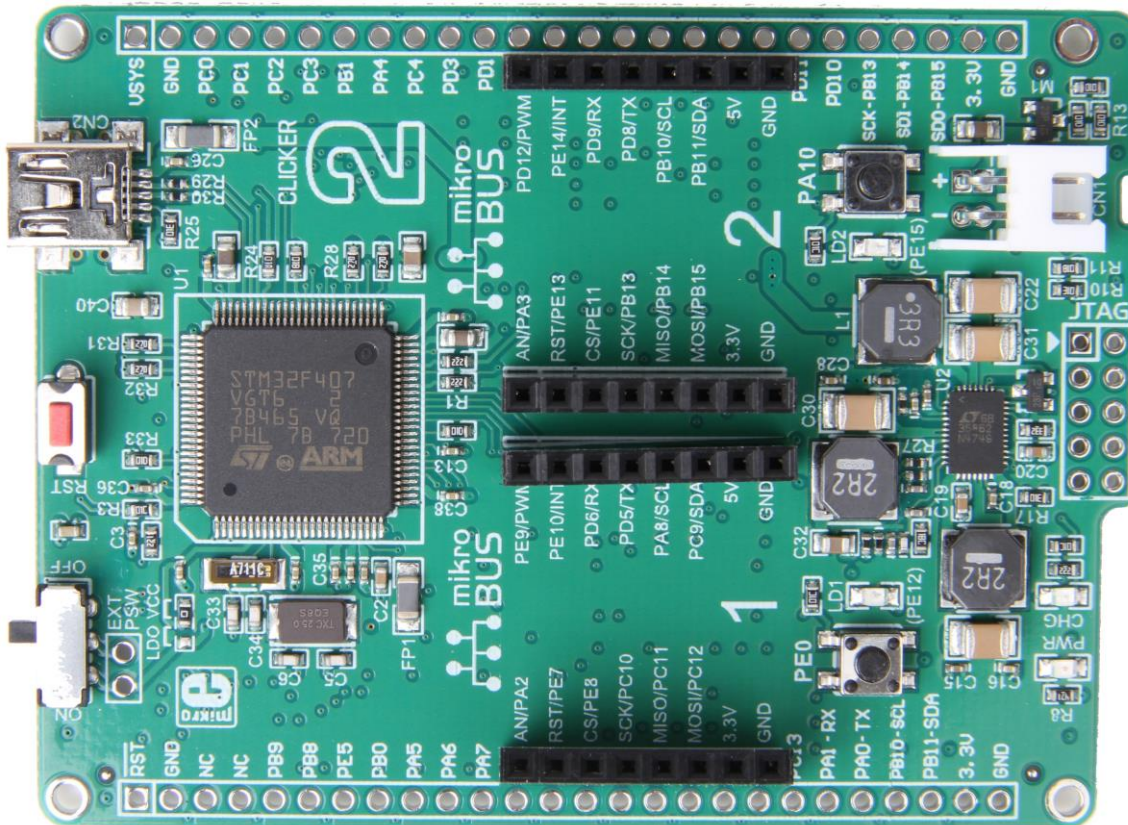
**CEC** CONTINUING  
EDUCATION  
CENTER



# XBee Radio Modules

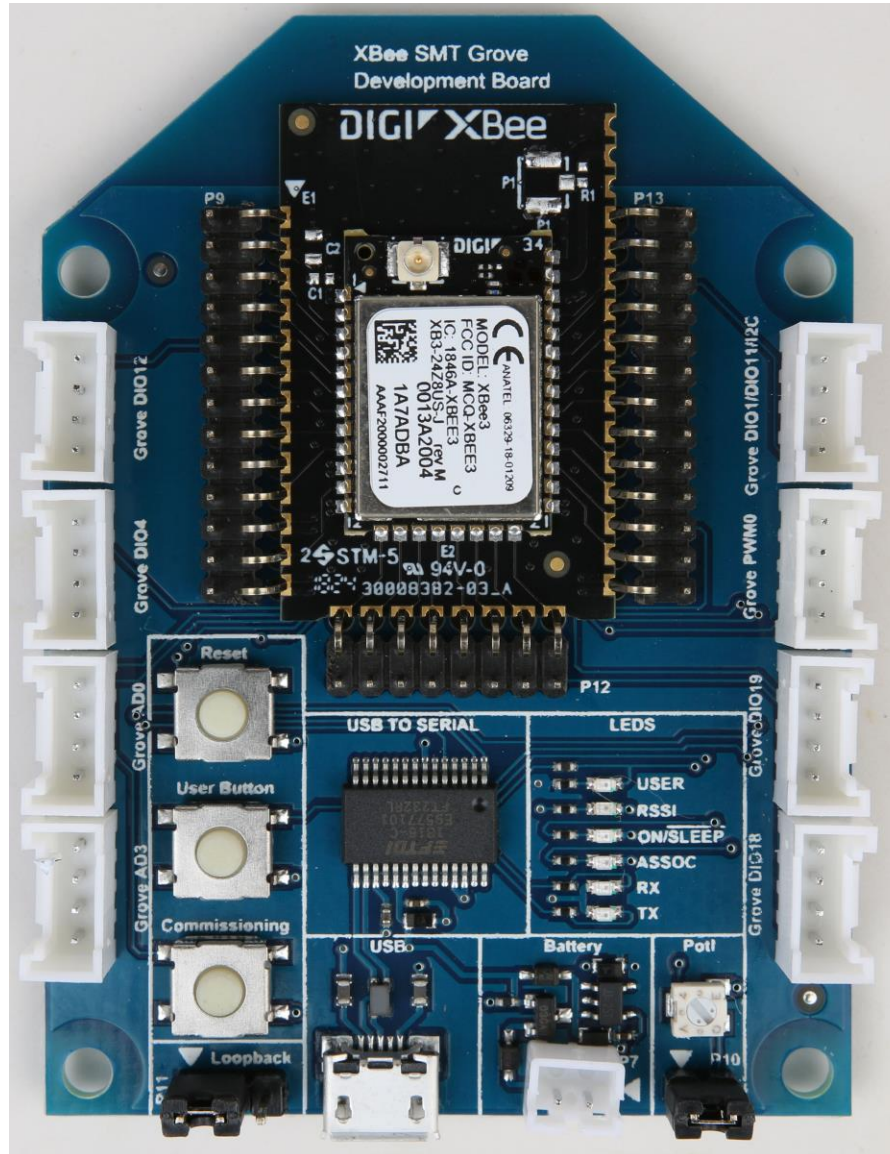
# AGENDA

- PyCharm XBee 3 Application
- PyCharm XBee/ARM/MicroPython Application
- Day 2 Summary



# XBee Radio Modules

## PyCharm XBee 3 Application



# XBee Radio Modules

## PyCharm XBee 3 Application



# XBee Radio Modules

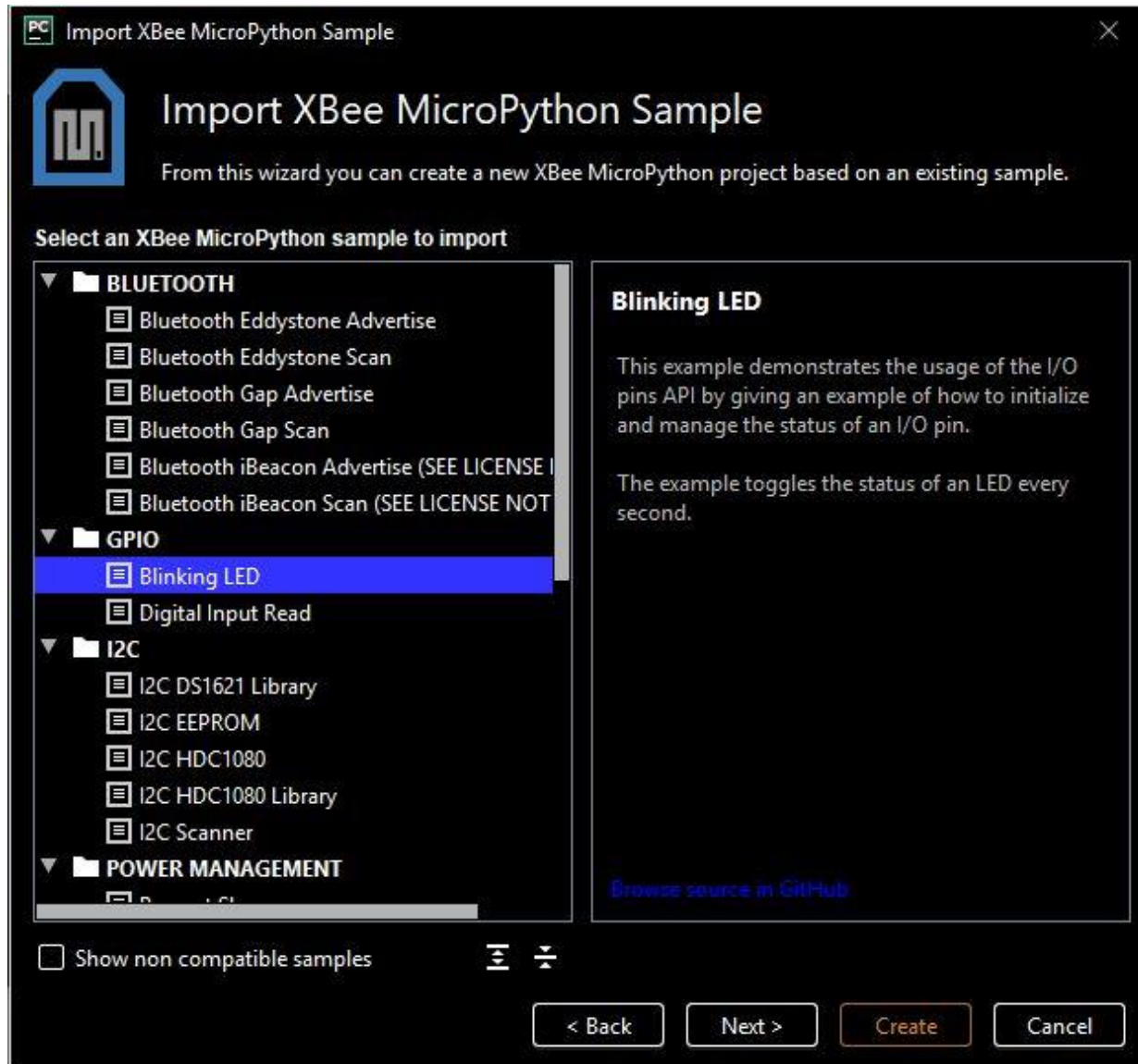
## PyCharm XBee 3 Application



Presented by:

# XBee Radio Modules

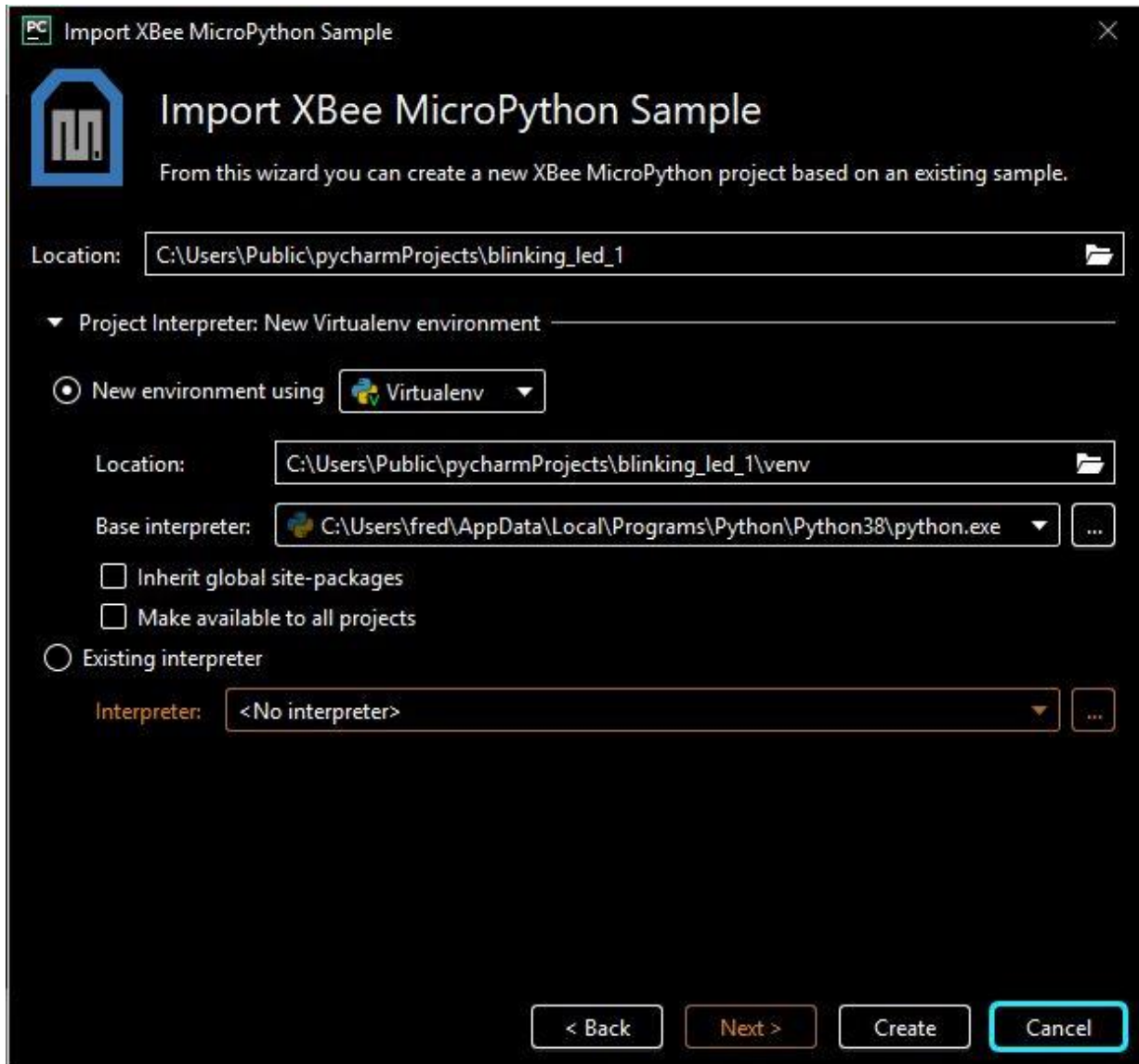
## PyCharm XBee 3 Application



Presented by:

# XBee Radio Modules

## PyCharm XBee 3 Application



Presented by:

# XBee Radio Modules

## PyCharm XBee 3 Application

The screenshot displays the PyCharm IDE interface. The main editor window shows a Python script named `main.py` for a blinking LED application. The code includes a copyright notice, imports for `Pin` and `time`, and a `while` loop that toggles an LED pin every second. A dialog box titled "XBee Device Selector" is overlaid on the right side of the IDE. The dialog contains the following information:

- Node ID: [blank]
- Protocol: 802.15.4
- MAC address: 0013A20041A7ADBA
- Port: COM3 - 9600/8/N/1/N

The dialog also features a "Select an XBee device" dropdown at the top, a "Refresh" button, and "OK" and "Cancel" buttons at the bottom.

Presented by:



# XBee Radio Modules

## PyCharm XBee 3 Application

The screenshot displays the PyCharm IDE interface. The main editor shows a Python script named `main.py` with the following content:

```
9 #
10 # The above copyright notice and this permission notice shall be included in
11 # all copies or substantial portions of the Software.
12 #
13 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
14 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
15 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
16 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
17 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
18 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
19 # SOFTWARE.
20
21 from machine import Pin
22 import time
23
24 # Pin D9 (ON/SLEEP/DIO9)
25 LED_PIN_ID = "D9"
26
27
28 print(" +-----+")
29 print(" | XBee MicroPython Blinking LED Sample |")
30 print(" +-----+\n")
31
32 # Set up the LED pin object to manage the LED status. Configure the pin
33 # as output and set its initial value to off (0).
34 led_pin = Pin(LED_PIN_ID, Pin.OUT, value=0)
35
```

The Run console at the bottom shows the deployment process to an XBee device (COM3 - 9600/8/N/1/N...):

```
Run: blinking_led_1 x
Deploying application in selected XBee device (COM3 - 9600/8/N/1/N)...
Cleaning stored code... [OK]
Flashing file /build/main.mpy... [OK]
Running application...
```

The status bar at the bottom indicates the file encoding is UTF-8 with 4 spaces, and the Python version is 3.8 (blinking\_led\_1).

Presented by:



# XBee Radio Modules

## PyCharm XBee 3 Application

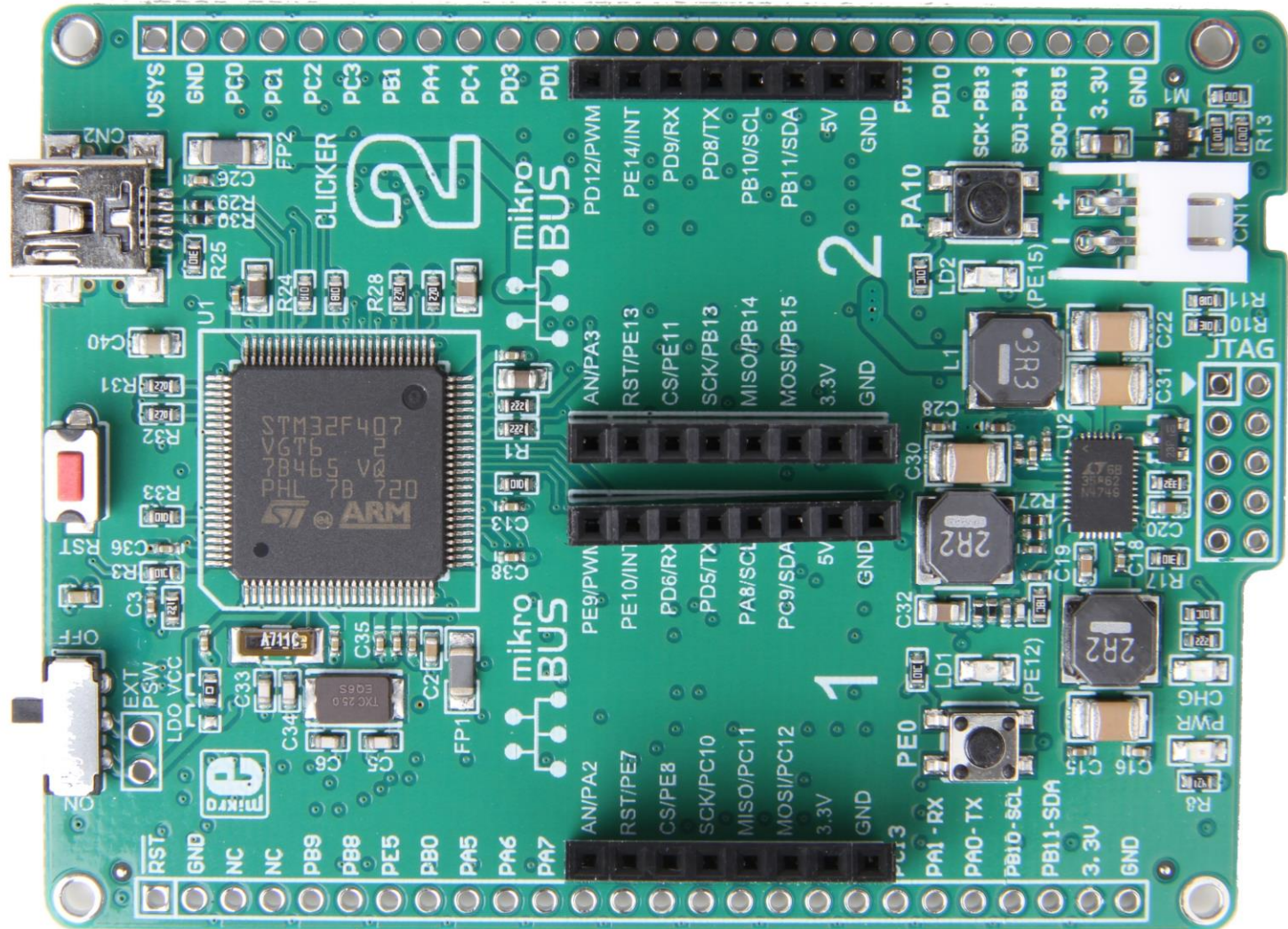
```
9 #
10 # The above copyright notice and this permission
11 # all copies or substantial portions of the Softw
12 #
13 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRA
14 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRA
15 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRI
16 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
17 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TO
18 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR TH
19 # SOFTWARE.
20
21 from machine import Pin
22 import time
23
24 # Pin D9 (ON/SLEEP/DIO9)
25 LED_PIN_ID = "D9"
26
27
28 print(" +-----+
29 print(" | XBee MicroPython Blinking LED Sample |
30 print(" +-----+
31
32 # Set up the LED pin object to manage the LED sta
33 # as output and set its initial value to off (0).
34 led_pin = Pin(LED_PIN_ID, Pin.OUT, value=0)
35
```

XBee REPL Console

```
Selected XBee device: COM3 - 9600/8/N/1/N | 0013A20041A7ADBA
- LED OFF
- LED ON
- LED OFF
- LED ON
- LED OFF
- LED ON
- LED OFF
- LED ON
- LED OFF
- LED ON
- LED OFF
- LED ON
- LED OFF
- LED ON
- LED OFF
```

# XBee Radio Modules

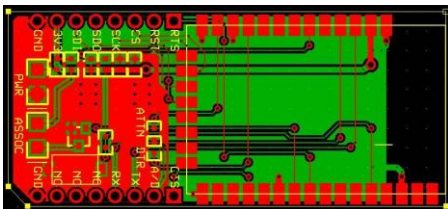
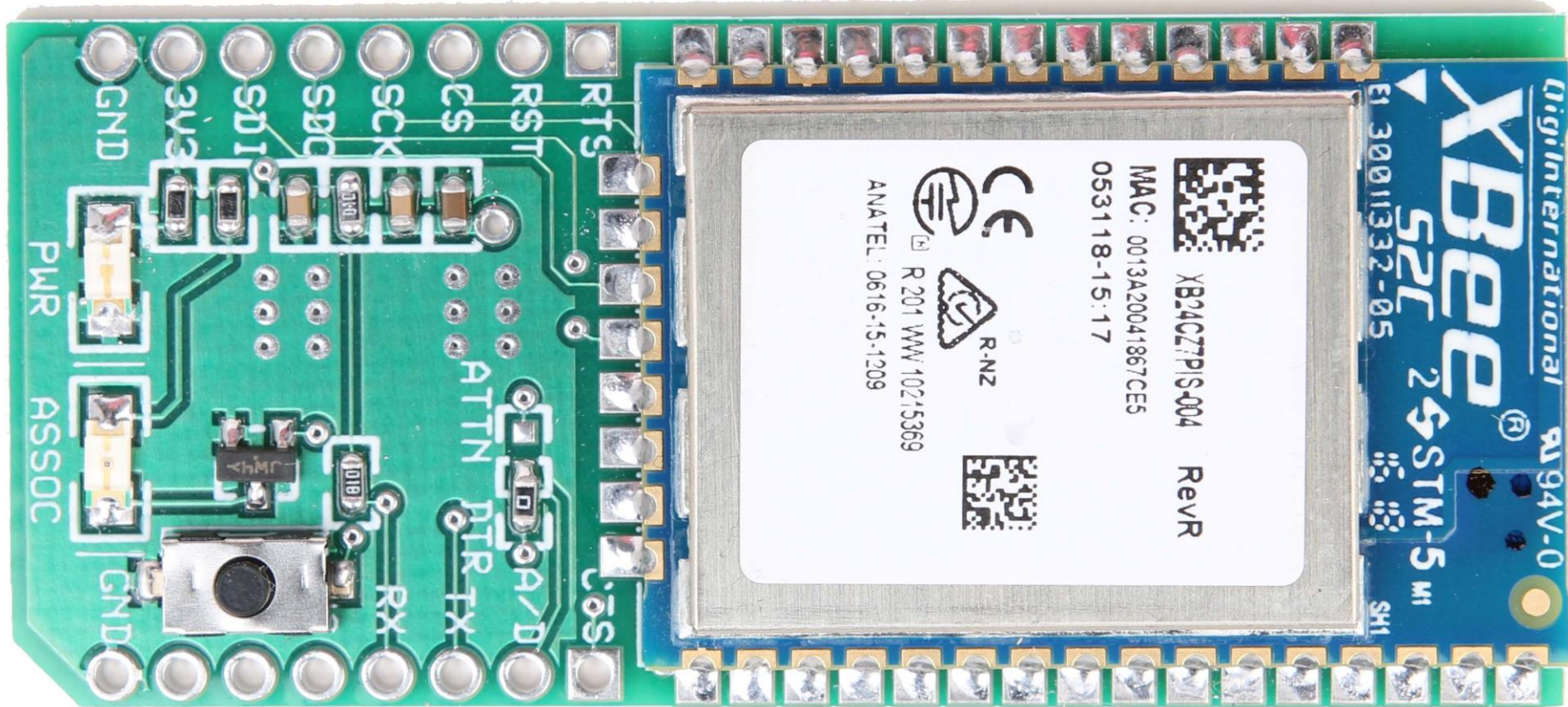
## PyCharm XBee/ARM/MicroPython Application



Presented by:

# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application

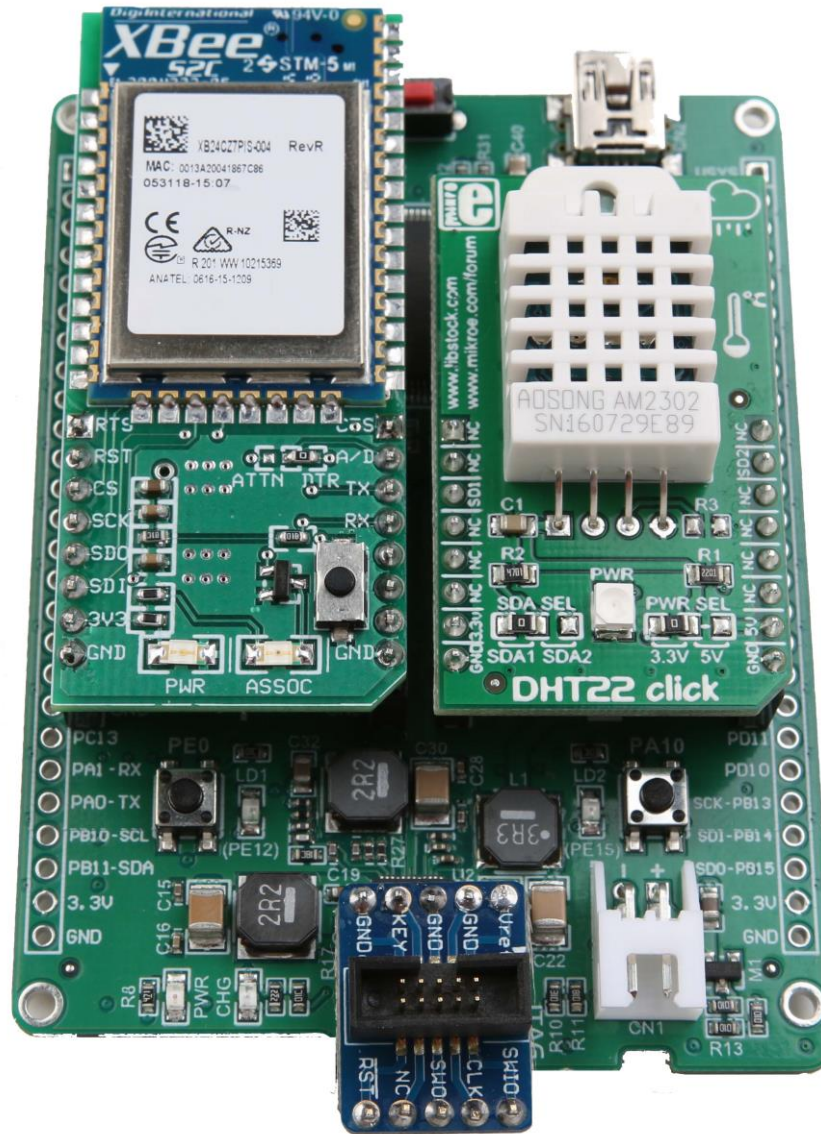


Presented by:



# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application



# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application

```
mpconfigboard.h
~/micropython/ports/stm32/boards/STM32F407

#define MICROPY_HW_BOARD_NAME "CLICK2"
#define MICROPY_HW_MCU_NAME "STM32F407"

#define MICROPY_HW_HAS_SWITCH (0)
#define MICROPY_HW_HAS_FLASH (1)
#define MICROPY_HW_ENABLE_RNG (1)
#define MICROPY_HW_ENABLE_RTC (1)
#define MICROPY_HW_ENABLE_DAC (1)
#define MICROPY_HW_ENABLE_USB (1)

// HSE is 25MHz
#define MICROPY_HW_CLK_PLLM (25)
#define MICROPY_HW_CLK_PLLN (336)
#define MICROPY_HW_CLK_PLLP (RCC_PLLP_DIV2)
#define MICROPY_HW_CLK_PLLQ (7)

// UART config
#if 0
// A9 is used for USB VBUS detect, and A10 is used for USB_FS_ID.
// UART1 is also on PB6/7 but PB6 is tied to the Audio SCL line.
// Without board modifications, this makes UART1 unusable on this board.
#define MICROPY_HW_UART1_TX (pin_A9)
#define MICROPY_HW_UART1_RX (pin_A10)
#endif
#define MICROPY_HW_UART2_TX (pin_D5)
#define MICROPY_HW_UART2_RX (pin_D6)
// #define MICROPY_HW_UART2_RTS (pin_A1)
// #define MICROPY_HW_UART2_CTS (pin_A0)
#define MICROPY_HW_UART3_TX (pin_D8)
#define MICROPY_HW_UART3_RX (pin_D9)
// #define MICROPY_HW_UART3_RTS (pin_D12)
// #define MICROPY_HW_UART3_CTS (pin_D11)
#if MICROPY_HW_HAS_SWITCH == 0
// NOTE: A0 also connects to the user switch. To use UART4 you should
// set MICROPY_HW_HAS_SWITCH to 0, and also remove SB20 (on the back
// of the board near the USER switch).
#define MICROPY_HW_UART4_TX (pin_A0)
```

C/ObjC Header Tab Width: 8 Ln 61, Col 1 INS

Presented by:

# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application

```
stm32f4xx_hal_conf.h
~/micropython/ports/stm32/boards/STM32F407

/* ##### HSE/HSI Values adaptation ##### */
/**
 * @brief Adjust the value of External High Speed oscillator (HSE) used in your application.
 * This value is used by the RCC HAL module to compute the system frequency
 * (when HSE is used as system clock source, directly or through the PLL).
 */
#if !defined (HSE_VALUE)
#define HSE_VALUE ((uint32_t)25000000) /*!< Value of the External oscillator in Hz */
#endif /* HSE_VALUE */

#if !defined (HSE_STARTUP_TIMEOUT)
#define HSE_STARTUP_TIMEOUT ((uint32_t)100U) /*!< Time out for HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */

/**
 * @brief Internal High Speed oscillator (HSI) value.
 * This value is used by the RCC HAL module to compute the system frequency
 * (when HSI is used as system clock source, directly or through the PLL).
 */
#if !defined (HSI_VALUE)
#define HSI_VALUE ((uint32_t)16000000) /*!< Value of the Internal oscillator in Hz*/
#endif /* HSI_VALUE */

/**
 * @brief Internal Low Speed oscillator (LSI) value.
 */
#if !defined (LSI_VALUE)
#define LSI_VALUE ((uint32_t)40000) /*!< Value of the Internal Low Speed oscillator in
Hz
The real value may vary depending on the variations
in voltage and temperature. */
#endif /* LSI_VALUE */

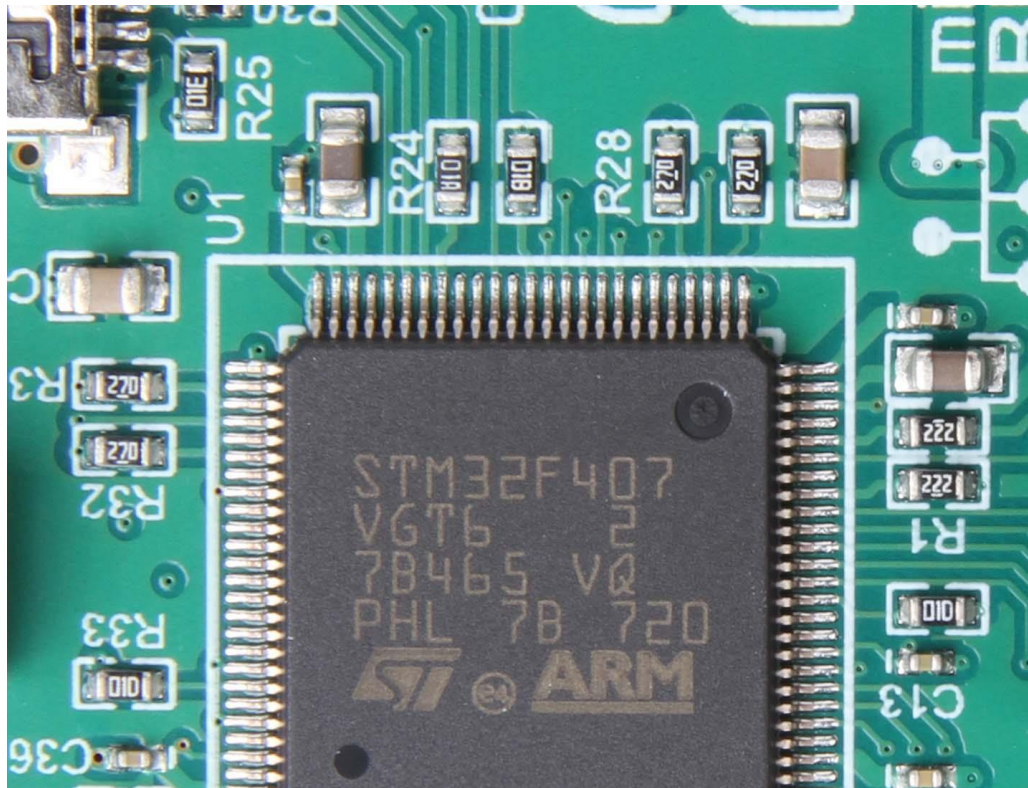
/**
 * @brief External Low Speed oscillator (LSE) value.
 */
#if !defined (LSE_VALUE)
```

C/ObjC Header Tab Width: 8 Ln 100, Col 37 INS

Presented by:

# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application



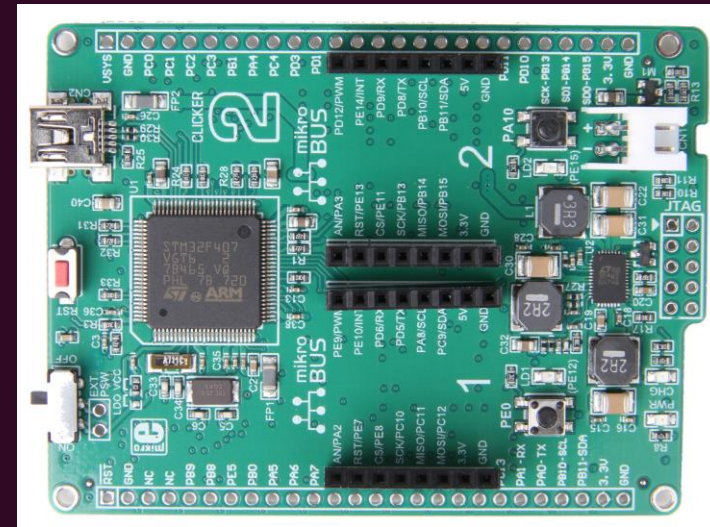
```
userfred@userfred-Inspiron-5767:~/micropython/ports/stm32$ sudo make BOARD=STM32F407
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
GEN build-STM32F407/genhdr/pins.h
GEN build-STM32F407/genhdr/qstrdefs.collected.h
QSTR not updated
GEN build-STM32F407/genhdr/qstrdefs.generated.h
CC ../../py/mpstate.c
CC ../../py/nlr.c
CC ../../py/nlrx86.c
```



# XBee Radio Modules

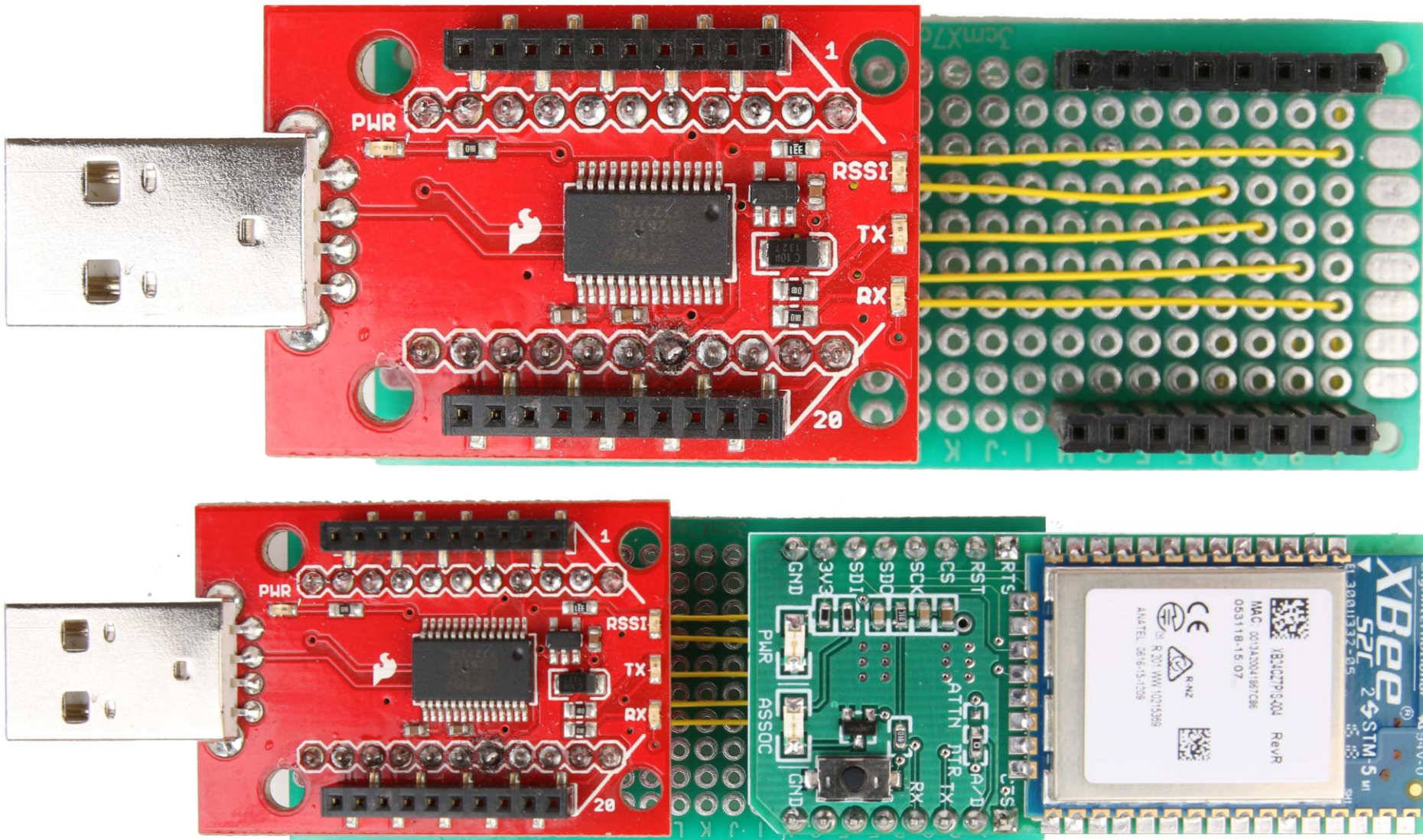
## PyCharm XBee/ARM/MicroPython Application

```
userfred@userfred-Inspiron-5767:~/micropython/ports/stm32$ cd build-STM32F407
userfred@userfred-Inspiron-5767:~/micropython/ports/stm32/build-STM32F407$ st-flash --format ihex write firmware.hex
st-flash 1.5.0
2020-01-05T17:14:18 INFO common.c: Loading device parameters....
2020-01-05T17:14:18 INFO common.c: Device connected is: F4 device, id 0x10076413
2020-01-05T17:14:18 INFO common.c: SRAM size: 0x30000 bytes (192 KiB), Flash: 0x100000 bytes (1024 KiB) in pages of 16384 bytes
2020-01-05T17:14:18 INFO common.c: Attempting to write 431628 (0x6960c) bytes to stm32 address: 134217728 (0x8000000)
Flash page at addr: 0x08060000 erasedEraseFlash - Sector:0x7 Size:0x20000
2020-01-05T17:14:28 INFO common.c: Finished erasing 8 pages of 131072 (0x20000) bytes
2020-01-05T17:14:28 INFO common.c: Starting Flash write for F2/F4/L4
2020-01-05T17:14:28 INFO flash_loader.c: Successfully loaded flash loader in sram
enabling 32-bit flash writes
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 32768
size: 5644
2020-01-05T17:14:34 INFO common.c: Starting verification of write complete
2020-01-05T17:14:38 INFO common.c: Flash written and verified! jolly good!
userfred@userfred-Inspiron-5767:~/micropython/ports/stm32/build-STM32F407$
```



# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application



Presented by:

# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application

The screenshot shows the PyCharm Settings window with the Plugins Marketplace tab selected. The search bar contains 'mi'. The search results list several plugins, with 'MicroPython' highlighted. The details for the 'MicroPython' plugin are shown on the right, including its version (1.0.13), release date (Oct 02, 2019), and a list of features.

**MicroPython** (1.0.13, Oct 02, 2019)

- Code completion and docs for some ESP8266, Pyboard, and MicroPython modules
- Flash a Python file or directory to a device
- Run REPL on a device

Currently the plugin supports ESP8266, Pyboard, and Micro:bit devices. Your feedback and contributions are welcome! See the [project page](#) on GitHub.

Presented by:

# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application

The screenshot shows the PyCharm Settings dialog for MicroPython. The 'Languages & Frameworks' section is expanded to 'MicroPython'. The 'Enable MicroPython support' checkbox is checked. The 'Device type' is set to 'Pyboard' and the 'Device path' is set to 'COM6'. A 'Detect' button is visible next to the device path field. The 'Tools' section is also visible in the left sidebar.

Settings

PC

Search

Appearance & Behavior

Keymap

Editor

Plugins

Version Control

Project: DHT22\_XBEE

Build, Execution, Deployment

Languages & Frameworks

Schemas and DTDs

Markdown

MicroPython

ReStructured Text

Tools

Languages & Frameworks > MicroPython For current project Reset

Enable MicroPython support

Device type: Pyboard

Device path: COM6 Detect

Learn more about setting up Pyboard devices

The photograph shows a green Pyboard microcontroller board. A white XBee radio module is plugged into the top header. A DHT22 digital temperature and humidity sensor is plugged into the bottom header. The board is populated with various components including resistors, capacitors, and integrated circuits. The text 'Pyboard' and 'www.mikroe.com/forum' are visible on the board.

OK Cancel Apply

Presented by:



# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application

The screenshot shows the PyCharm IDE interface. The main editor displays a Python script named `main.py` with the following code:

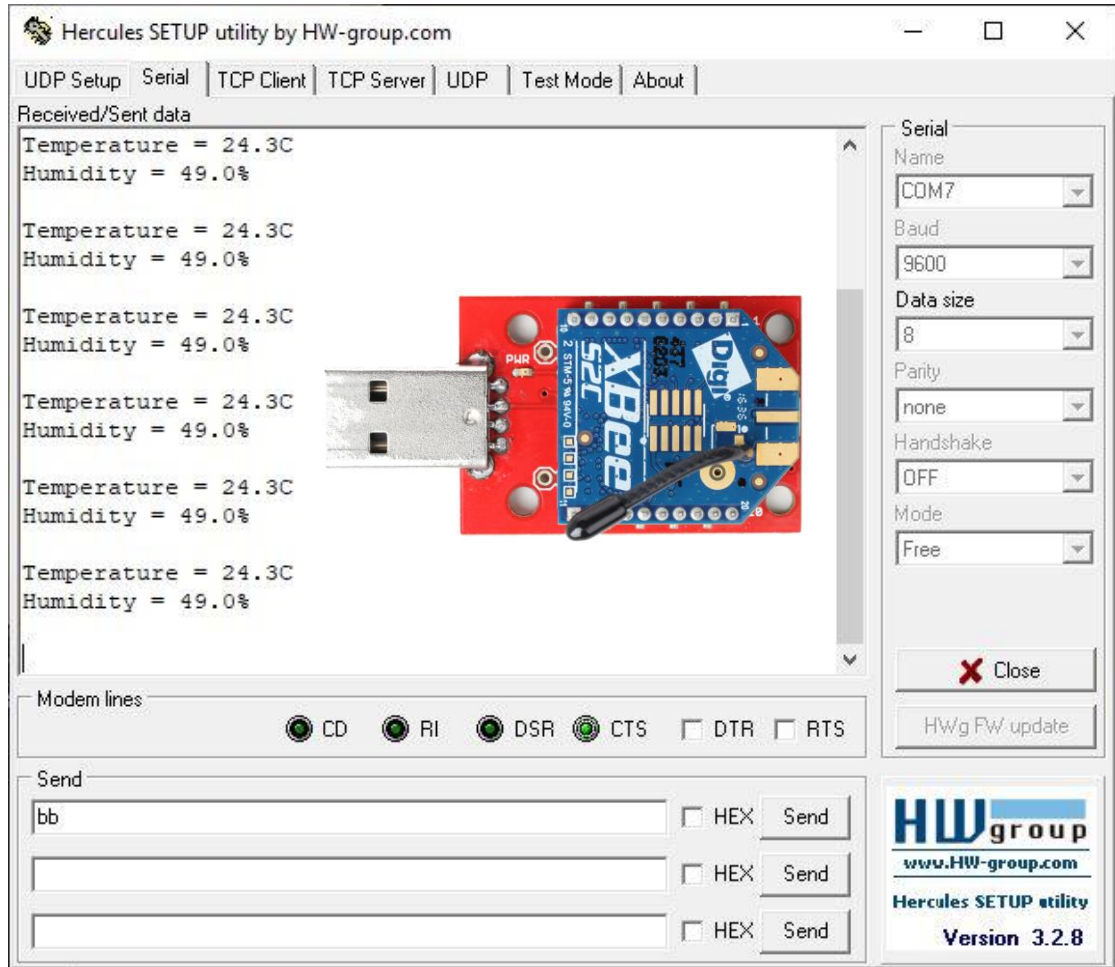
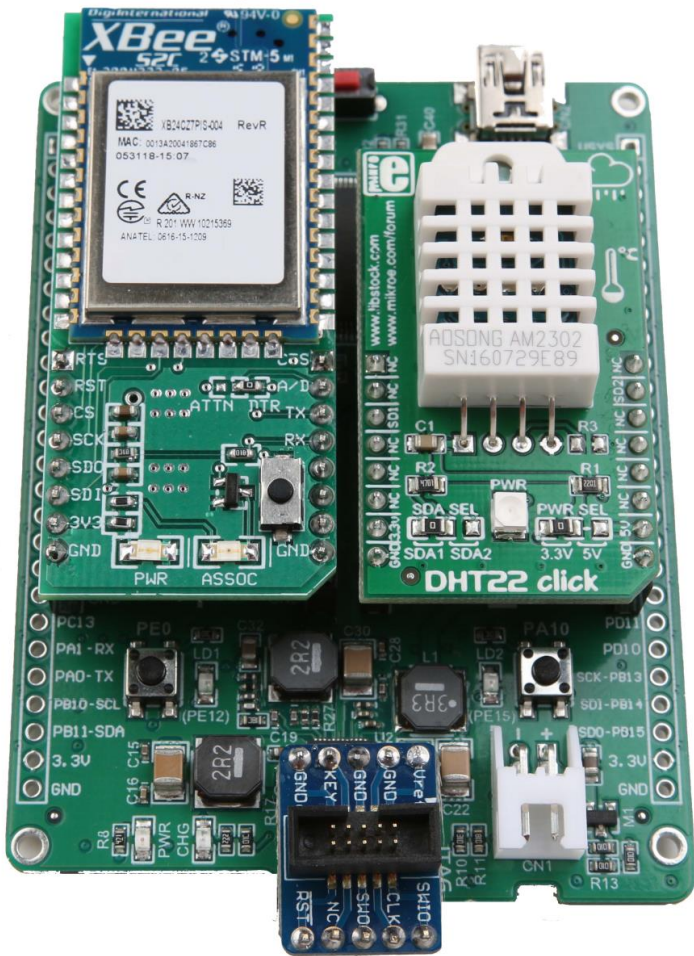
```
1 import pyb
2 import machine
3 import dht
4 from machine import Pin
5 from machine import UART
6
7 u = UART(2, 9600)
8 u.init(9600, bits=8, parity=None, stop=1)
9 rstPin = Pin(Pin.cpu.E7, mode=Pin.OUT)
10 LED_POD1 = Pin(Pin.cpu.E12, mode=Pin.OUT)
11 dhtPin = Pin(Pin.cpu.E11)
12 DHT1 = dht.DHT22(machine.Pin(dhtPin))
13
14 rstPin.value(1)
15
16 while True:
17     DHT1.measure()
18     tempVal = str(DHT1.temperature())
19     humVal = str(DHT1.humidity())
20     tempVal = 'Temperature = ' + tempVal + 'C' + '\r\n'
21     humVal = 'Humidity = ' + humVal + '% ' + '\r\n' + '\r\n'
22     u.write(tempVal)
23     u.write(humVal)
24     LED_POD1.value(1)
25     pyb.delay(1000)
26     LED_POD1.value(0)
27     pyb.delay(1000)
28
```

The interface also shows a project tree on the left with the file `main.py` selected. At the bottom, there is a terminal window with the message: "Packages installed successfully: Installed packages: 'pyserial'>=3.3,<4.0', 'docopt'>=0.6.2".

Presented by:

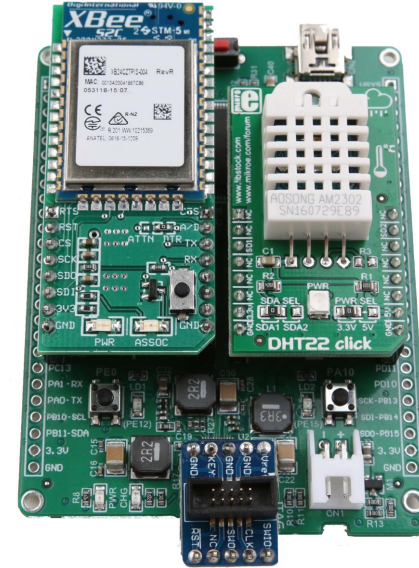
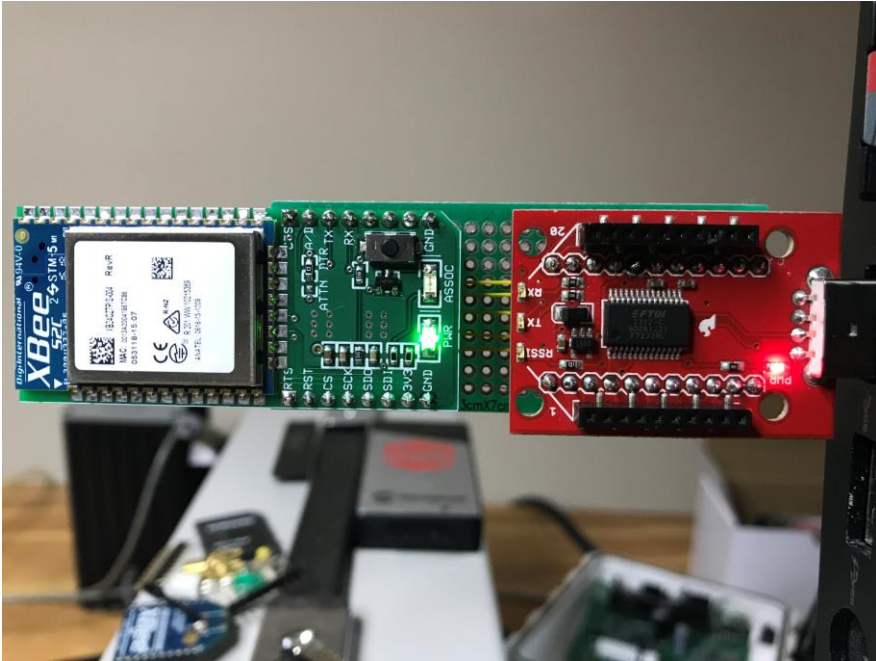
# XBee Radio Modules

## PyCharm XBee/ARM/MicroPython Application



# XBee Radio Modules

## Day 2 Summary



```
main.py X
1 import pyb
2 import machine
3 import dht
4 from machine import Pin
5 from machine import UART
6
7 u = UART(2, 9600)
8 u.init(9600, bits=8, parity=None, stop=1)
9 rstPin = Pin(Pin.cpu.E7, mode=Pin.OUT)
10 LED_POD1 = Pin(Pin.cpu.E12, mode=Pin.OUT)
11 dhtPin = Pin(Pin.cpu.E11)
12 DHT1 = dht.DHT22(machine.Pin(dhtPin))
13
14 rstPin.value(1)
15
16 while True:
17     DHT1.measure()
18     tempVal = str(DHT1.temperature())
19     humVal = str(DHT1.humidity())
20     tempVal = 'Temperature = ' + tempVal + 'C' + '\n\n'
21     humVal = 'Humidity = ' + humVal + '%' + '\n\n' + '\n\n'
22     u.write(tempVal)
23     u.write(humVal)
24     LED_POD1.value(1)
25     pyb.delay(1000)
26     LED_POD1.value(0)
27     pyb.delay(1000)
28
```

Presented by:

