Designing Embedded Systems using Micro Python

Class 2: Getting Started with the Pyboard D-Series

June 11, 2019

Jacob Beningo

DesignNews

CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS

Course Overview

Topics:

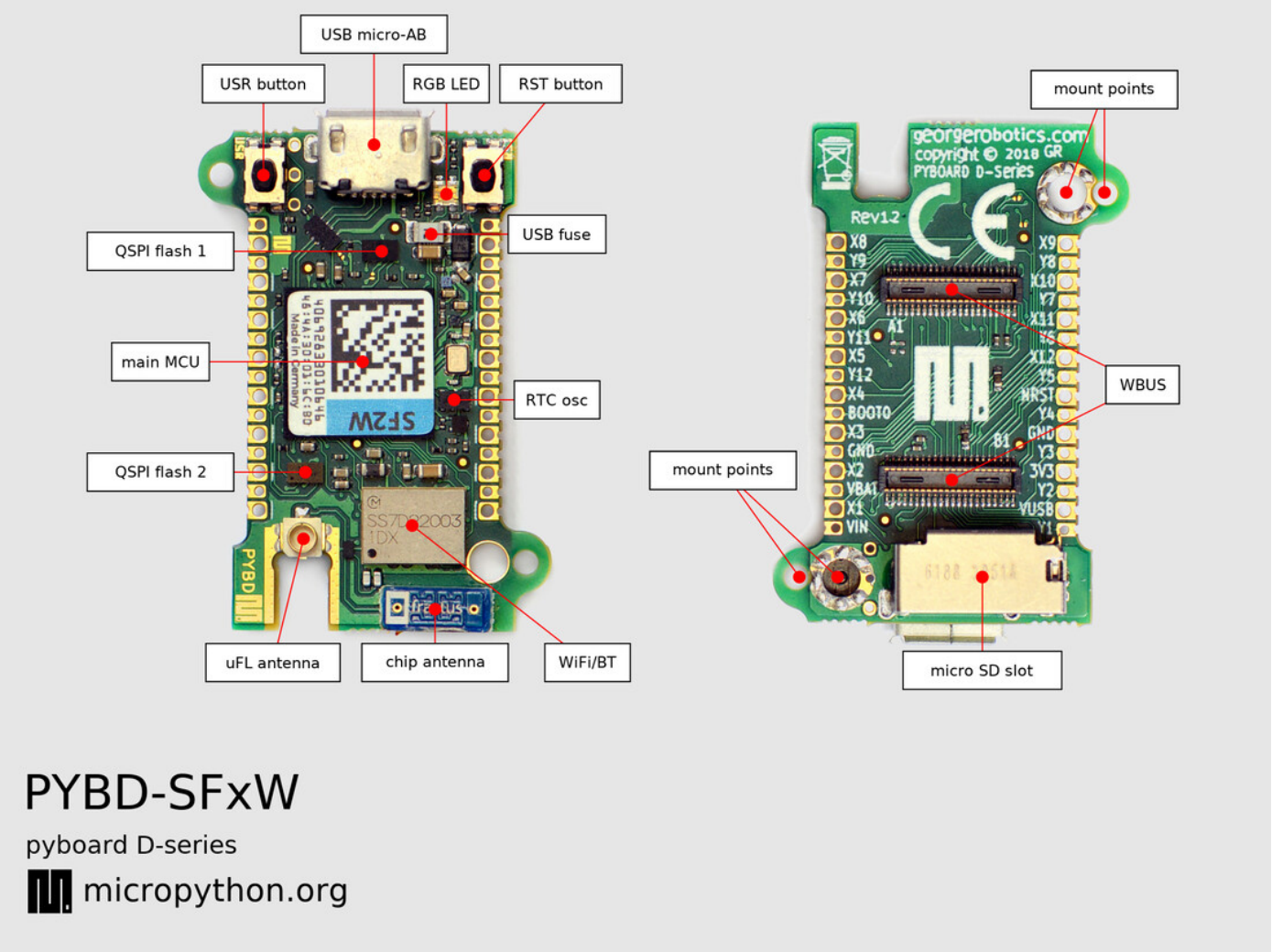
- Designing Products with MicroPython
- **Getting Started with the Pyboard D-Series**
- Customizing the MicroPython Kernel for Production
- Developing Real-time Application Projects
- Testing MicroPython Projects

Session Overview

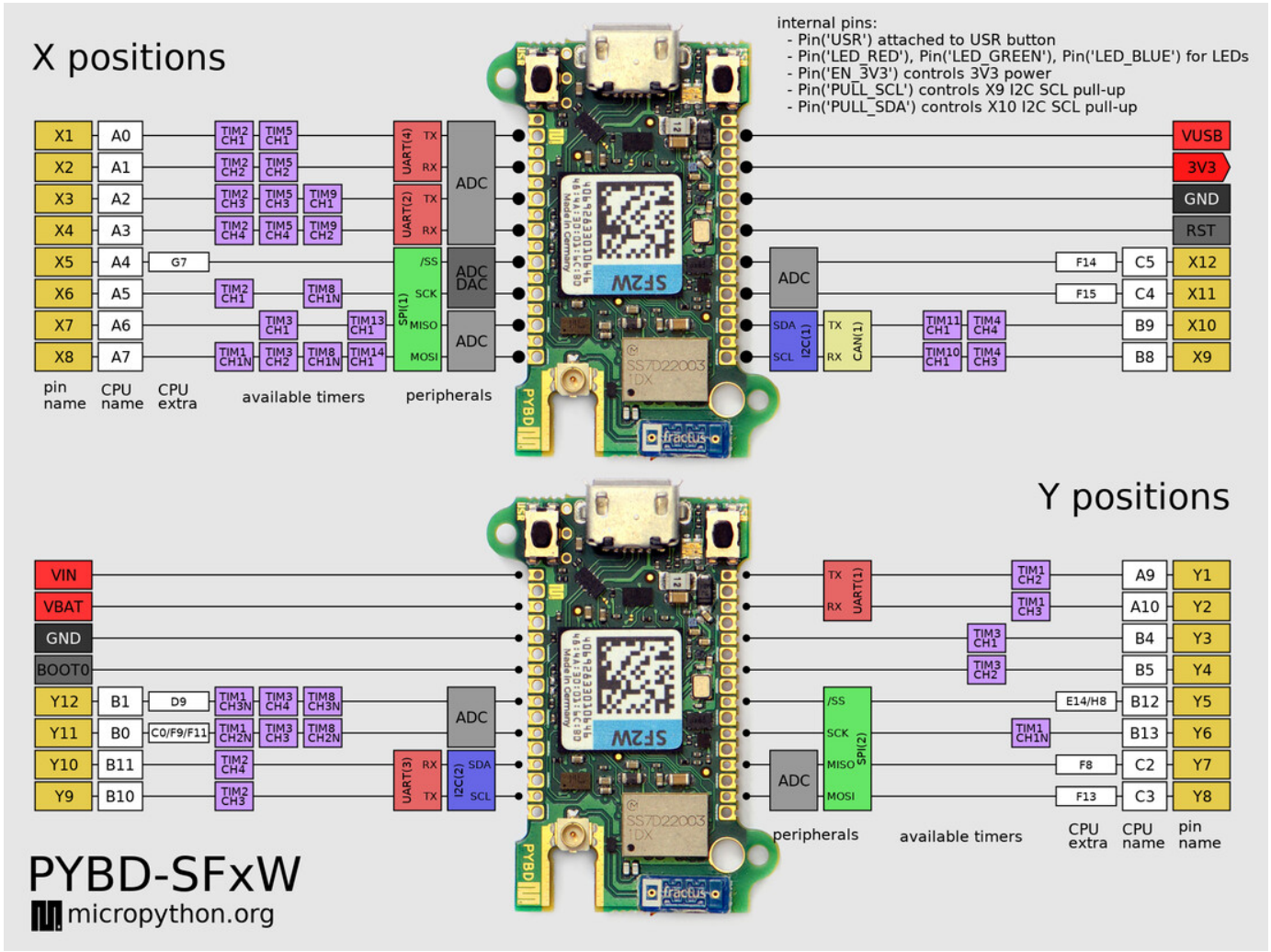
- Pyboard D-series Capabilities
- Running the Board
  - The four methods
- MicroPython Libraries
- Examples
- Best Practices

Presented by:

Pyboard D-series



Pyboard Series-D

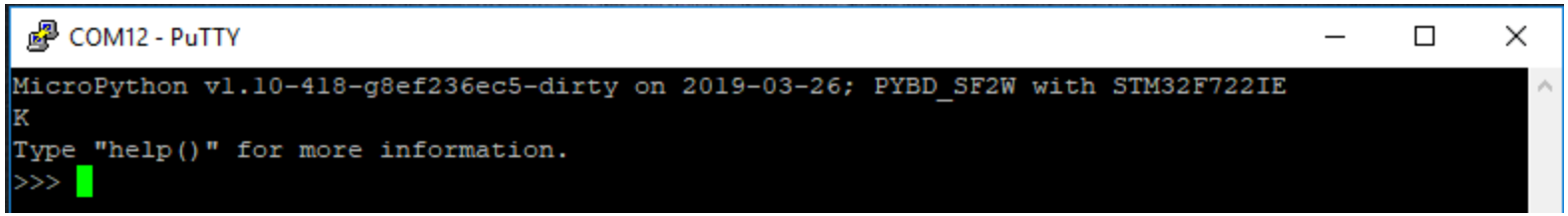


Running the Board

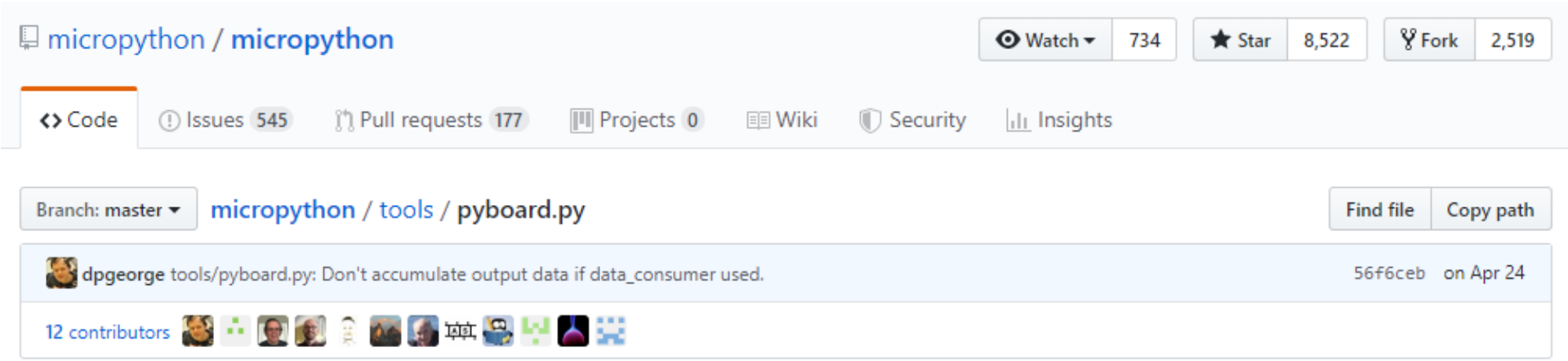
Four Methods to execute code

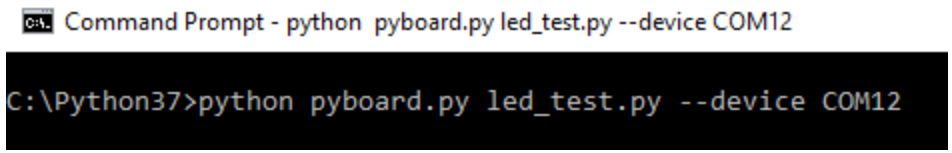
- REPL
- Remote Script
- From main.py
- Through “frozen code” (mpy files)

Running the Board – The REPL

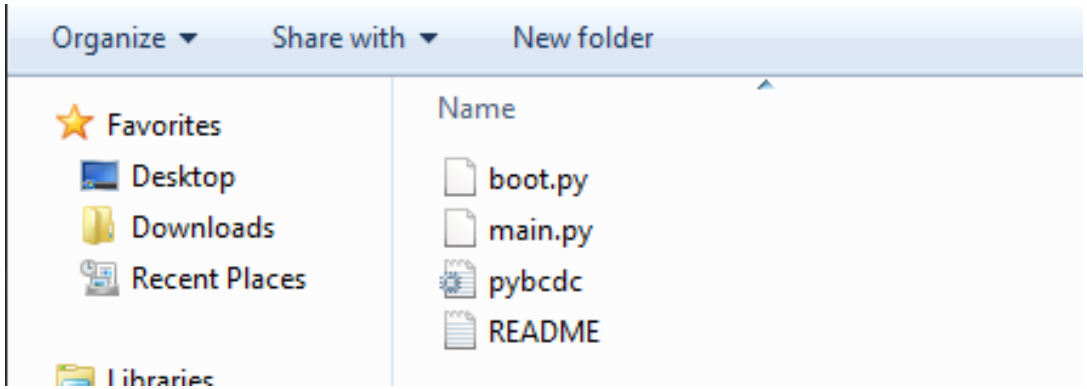


Running the Board – The RAW REPL





Running the Board - Scripts



- boot.py – defines scripts to run on startup
- main.py – start of python scripting program
- pybcdc – windows serial driver
- README – misc board information

Running the Board - Scripts

Libraries and external classes

Definitions and Initialization

Primary program loop

```
# main.py
import pyb
```

```
# define LED color constants
LED_RED = 1
LED_GREEN = 2
LED_YELLOW = 3
LED_BLUE = 4

# Defines the primary loop delay
DELAY_1000MS = 1000

# Create an Led object assigned to the green LED
Led = pyb.LED(LED_GREEN)
```

```
# Main execution loop
# Toggle the LED every
while True:
    Led.toggle()
    pyb.delay(DELAY_1000MS)
```

MicroPython Library Overview

- http://docs.micropython.org/en/latest/library/index.html

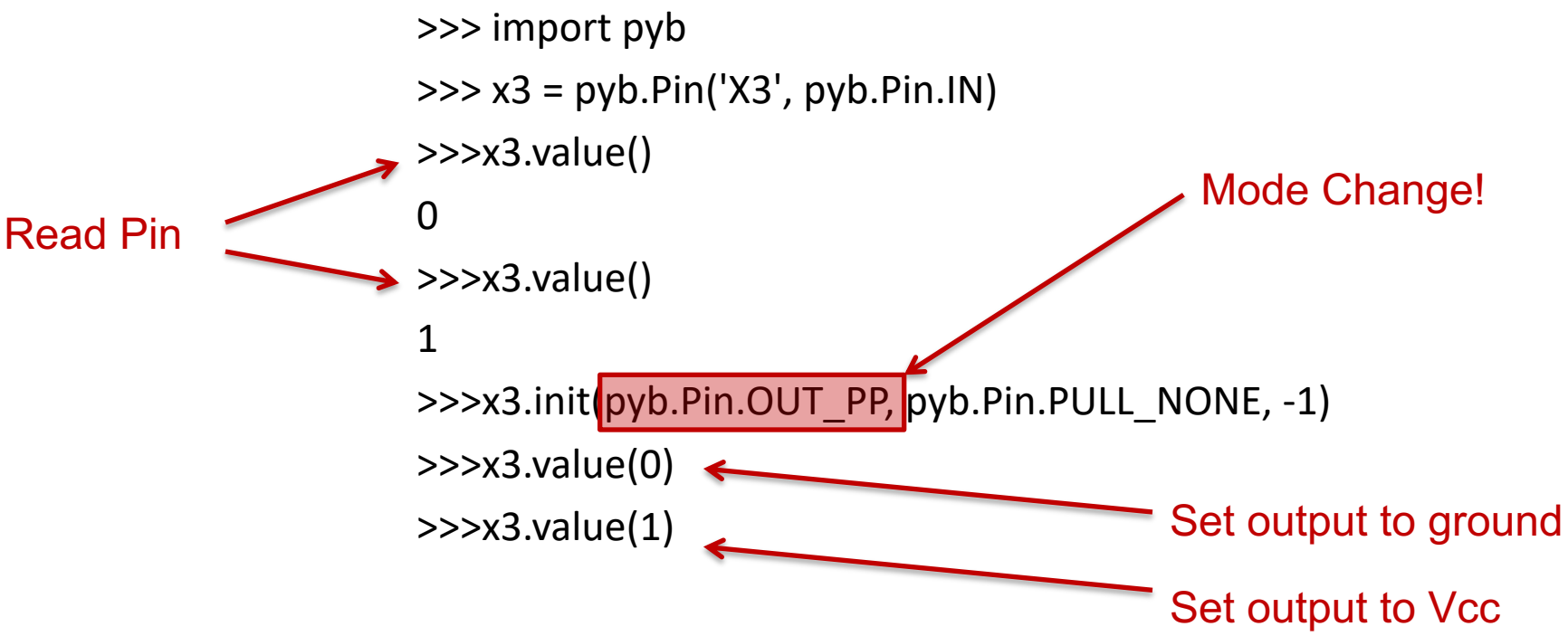
- Builtin functions and exceptions
- `array` – arrays of numeric data
- `cmath` – mathematical functions for complex numbers
- `gc` – control the garbage collector
- `math` – mathematical functions
- `sys` – system specific functions
- `ubinascii` – binary/ASCII conversions
- `ucollections` – collection and container types
- `uerrno` – system error codes
- `uhashlib` – hashing algorithms
- `uheapq` – heap queue algorithm
- `uio` – input/output streams
- `ujson` – JSON encoding and decoding
- `uos` – basic “operating system” services
- `ure` – simple regular expressions
- `uselect` – wait for events on a set of streams
- `usocket` – socket module
- `ussl` – SSL/TLS module
- `ustruct` – pack and unpack primitive data types
- `utime` – time related functions
- `uzlib` – zlib decompression
- `_thread` – multithreading support

MicroPython Libraries

- `btree` – simple BTree database
- `framebuf` — Frame buffer manipulation
- `machine` — functions related to the hardware
- `micropython` – access and control MicroPython internals
- `network` — network configuration
- `ucryptolib` – cryptographic ciphers
- `uctypes` – access binary data in a structured way

- class Accel – accelerometer control
- class ADC – analog to digital conversion
- class CAN – controller area network communication bus
- class DAC – digital to analog conversion
- class ExtInt – configure I/O pins to interrupt on external events
- class I2C – a two-wire serial protocol
- class LCD – LCD control for the LCD touch-sensor pyskin
- class LED – LED object
- class Pin – control I/O pins
- class PinAF – Pin Alternate Functions
- class RTC – real time clock
- class Servo – 3-wire hobby servo driver
- class SPI – a master-driven serial protocol
- class Switch – switch object
- class Timer – control internal timers
- class TimerChannel — setup a channel for a timer
- class UART – duplex serial communication bus
- class USB_HID – USB Human Interface Device (HID)
- class USB_VCP – USB virtual comm port

GPIO Example



UART Example

```
# Configure Uart6 for communication
Uart1 = pyb.UART(1,115200)
Uart1.init(115200, bits=8, parity=None, stop=1)

# define a receive task
def UartRx():

        # Have any characters been received?
        if Uart1.any():
            # Yes read the character
            temp = Uart1.readchar()
            print (chr(temp))

            if temp == ord('q'):
                return 1
            else:
                Uart1.writechar(temp)
```

Convert to char from int

Convert to int from char

Quit application

Using Threads

```
import micropython # For emergency exception buffer
import pyb              # For uPython MCU features
import _thread       # For thread support

# Buffer for interrupt error messages
micropython.alloc_emergency_exception_buf(100)

# Function that contains the task code for toggling the blue LED
def Led_BlueToggle():
    while True:
      pyb.LED(4).toggle()
      pyb.delay(250)

# Function that contains the task code for toggling the yellow LED
def Led_YellowToggle():
    while True:
      pyb.LED(3).toggle()
      pyb.delay(250)
```

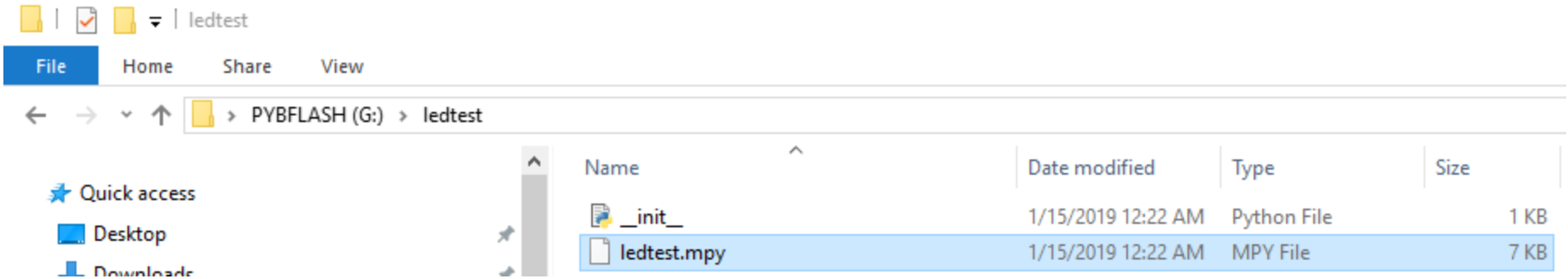
Using Threads

```
_thread.start_new_thread(Led_BlueToggle, ())
_thread.start_new_thread(Led_YellowToggle, ())

# Tracks seconds since program started
SecondsLive = 0

while True:
  pyb.delay(5000)
  SecondsLive = SecondsLive + 5
  print("Executing for ", SecondsLive, " seconds")
```

Running the Board - MPY



ledtest.mpy

```
4d03 021f 8209 0300 0000 0000 2d34 000c
0181 1580 0928 2830 3030 7060 6026 6660
2626 6646 2626 8608 6620 4626 6640 2626
2626 6620 2686 0800 00ff 8011 680d 0124
0d01 8011 680e 0124 0e01 8016 1101 5001
686d 0169 1101 2411 0132 8016 1301 5001
686e 0169 1301 2413 0132 8016 1601 5001
686f 0169 1601 2416 0132 8016 8700 5001
68ce 0069 8700 2487 0032 1481 0024 1701
1481 0124 1801 1481 0924 1901 1481 0b24
1a01 1481 0c24 1b01 1481 0f24 1c01 1481
```

Best Practices

- Take the time up front to develop a software architecture
- Experiment using scripts
- Use the raw mode for system recovery
- Compile your scripts to .mpy for production
- Read through the MicroPython tutorials and documentation
- Pick a simple project and develop it
- Master Python 3

Additional Resources

- Download Course Material for
  - http://bit.ly/MicroPythonProjects
  - Blog
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Designing Embedded Systems using MicroPython