Středoškolská technika 2022

Setkání a prezentace prací středoškolských studentů na ČVUT

PROGRAMOVÁNÍ MIKROKONTROLÉRŮ STM32 V JAZYCE MICROPYTHON

Pavel Váňa, Petr Nahodil, Tomáš Novák, Ondřej Pavlík

Střední průmyslová škola elektrotechnická
Ječná 30, Praha 2

1.Úvod

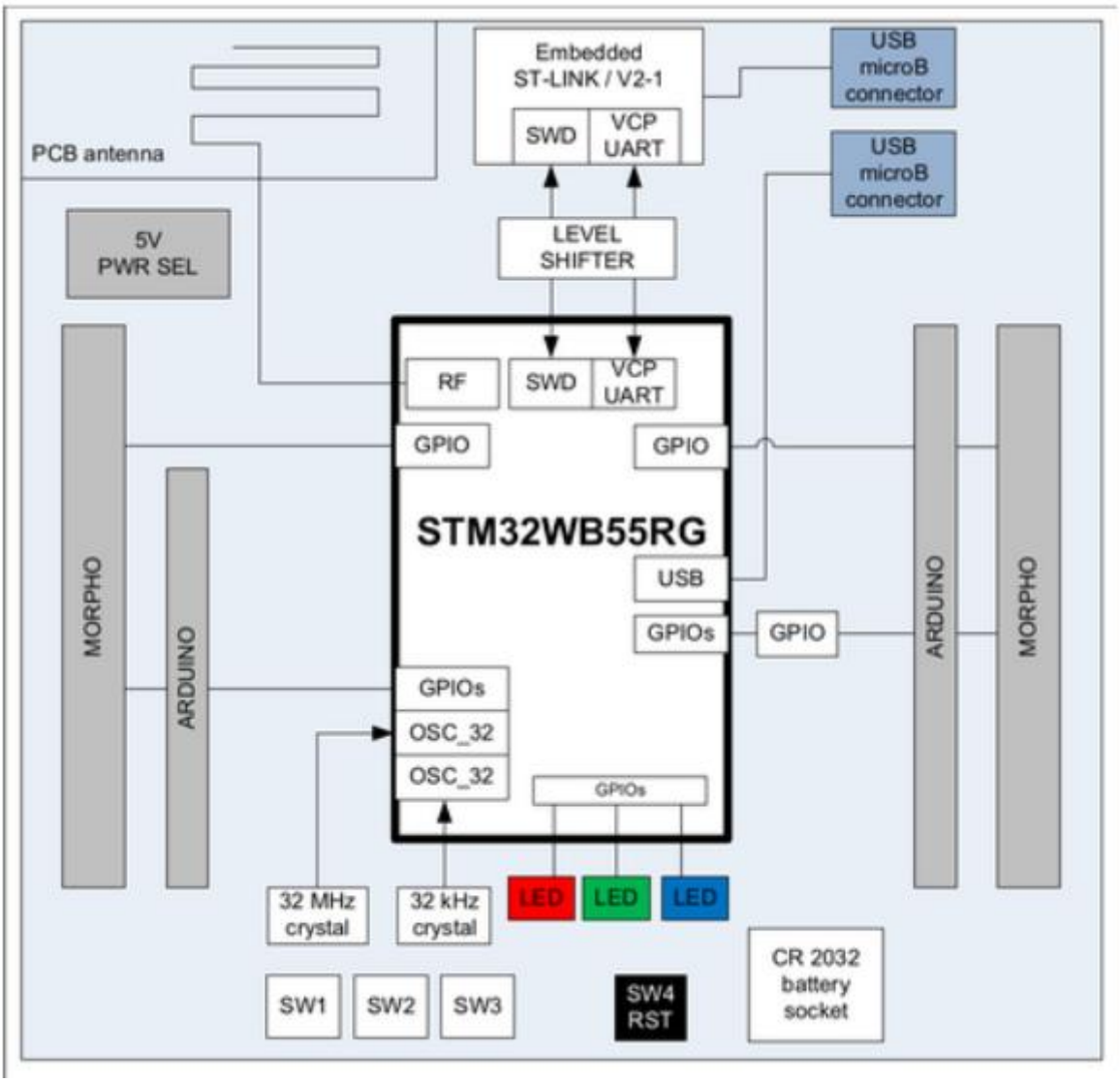
Python je moderní programovací jazyk, který umožňuje navrhovat jak jednoduché programy, tak i poměrně rozsáhlé. Je pro něj vyvinuto množství knihoven a frameworků. Je to jazyk, který se lze rychle naučit a patří proto mezi oblíbené jazyky. Existuje dokonce i jeho implementace microPython pro jednočipové počítače/mikrokontrolery, což je oblast, kde se používají především C/C++ popř. asembler. Rozhodli jsme se programování jednočipových počítačů v microPythonu odzkoušet na startkitu NUCLEO STM32WB firmy STMicroelectronic. Startkity s STM32 programujeme v jazyce C++ při výuce programování jednočipových počítačů. Jejich programování v microPythonu by mohlo programování jednočipů přiblížit i studentům, kteří preferují spíše programování PC, mobilů či tvorbu www stránek. Navíc z praktického hlediska při vývoji oceníme možnost přímo komunikovat s komplikovanými periferiemi připojenými (pomocí SPI, I2C, CAN …) bez potřeby ztráty času při vývoji pomocného komunikačního software.

2.MicroPython

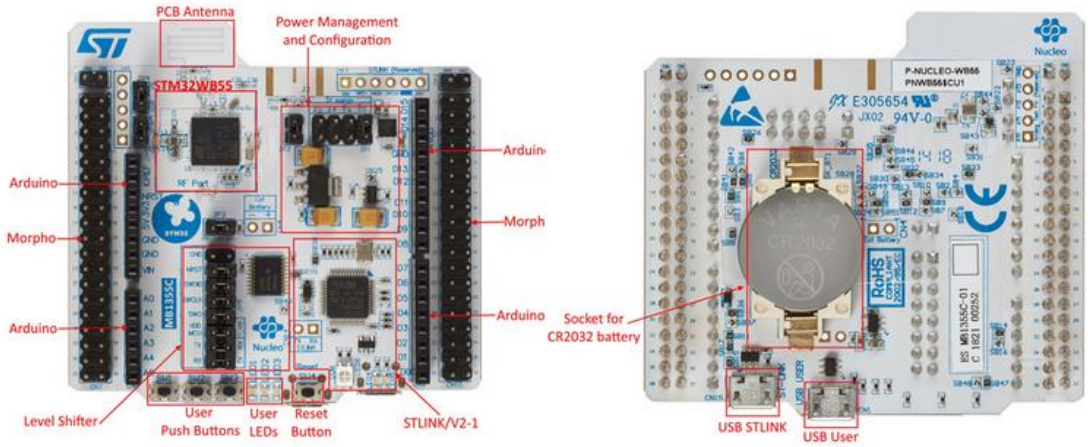
Implementace MicroPythonu vznikla v roce 2015 jako implementace Pythonu pro ARM-Cortex-M procesory. Její autoři se snažili v rámci možností dodržet kompatibilitu s Python3. Oficiálně jsou podle dokumentace podporované platformy pyboard (vlastní vývoj), ESP32, ESP8266 a WiPy/CC3200. Platforma STM32 oficiálně není podporovaná , ale prolistovaním zdrojových kódů na GitHube zjistíme, že MicroPython je portovaný i na vybrané desky Discovery a Nucleo od STM s podporou integrovaného hardware.
Implementace MicroPython se chová téměř stejně jako Python3. Hlavní omezení MicroPythonu je to, že chybí většina standardní knihovny. Některé části (např math) jsou k dispozici, spousta ne. Několik funkcí je na nestandardních místech se změněným rozhraním, např. generátor náhodných čísel. Co má MicroPython navíc je přístup k hardwaru. API MicroPythonu často používá metody tam, kde bychom čekali atributy, proto ne pin.value ale pin.value().

3.Použitý startkit

Využili jsme startkit STM32WB55 Nucleo [2] firmy STMicroelectronic, protože tyto startkity využíváme při výuce, kdy je programujeme v jazycích C/C++ v prostředí MBED a STM32CubeIDE. Blokové schéma startkitu:



Pohled na vlastní startkit:



4.Práce se startkitem pomocí microPythonu

Nejprve si z domovské stránky microPythonu stáhneme firmware pro náš startkit https://micropython.org/download/

https://micropython.org/download/NUCLEO_WB55/

STM32 via DFU

Boards with USB support can also be programmed via the ST DFU bootloader, using e.g. dfu-util or pydfu.py.

To enter the bootloader the `BOOT0` pin can be connected to `VCC` during reset, or you can use `machine.bootloader()` from the MicroPython REPL.

```
dfu-util --alt 0 -D firmware.dfu
```

Firmware

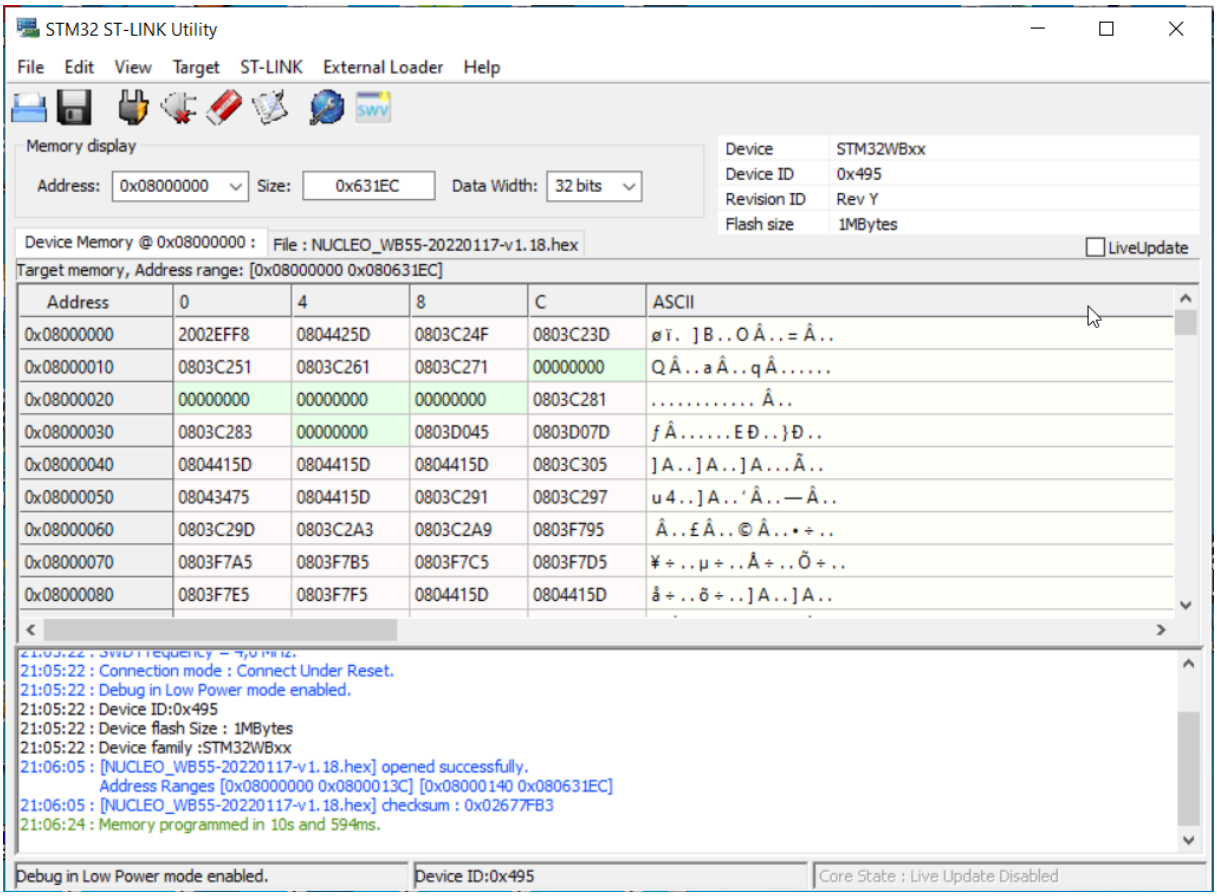
Releases

v1.18 (2022-01-17) .dfu [.hex] [Release notes] (latest)
v1.17 (2021-09-02) .dfu [Release notes]
v1.16 (2021-06-18) .dfu [Release notes]
v1.15 (2021-04-18) .dfu [Release notes]
v1.14 (2021-02-22) .dfu [Release notes]
v1.13 (2020-09-02) .dfu [Release notes]

Nightly builds

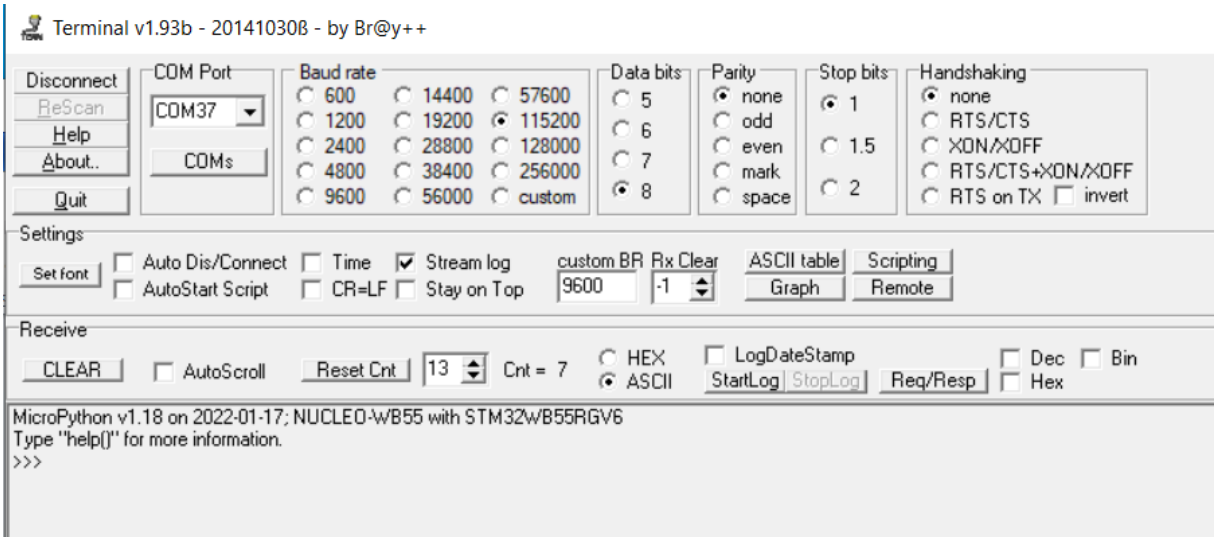
v1.18-392-g44186ef59 (2022-04-28) .dfu [.hex]
v1.18-389-g40047823b (2022-04-28) .dfu [.hex]
v1.18-388-g9d08eb024 (2022-04-28) .dfu [.hex]
v1.18-382-g014912daa (2022-04-27) .dfu [.hex]

Máme možnost získat jak dfu, tak hex soubor. Použití hex je jednodužší, neboť pro jeho nahrání do programové paměti STM32WB můžeme použít programátor ST-LINK 2.0, který je součástí startkitu NUCLEO a software STM32 ST-Link Utility nebo STM32CubeProgrammer.

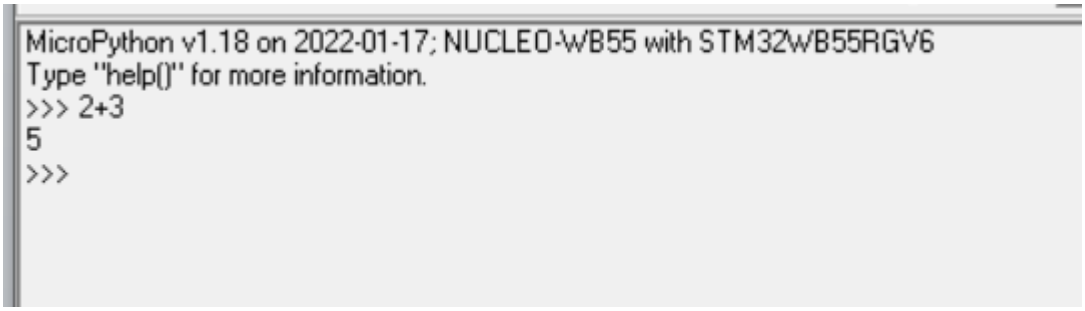


Po nahrání souboru NUCLEO_WB55-20220117-v.1.18.hex již můžeme pracovat s microPythonem. Pro nahrání tohoto souboru jsme použili usb konektor na NUCLEU označený ST-LINK. Startkit má ještě další usb konektor označený USB-USER. I ten nyní připojíme k PC a tím získáme přístup k (virtuálnímu) disku obsahujícímu adresář **pybflash** se soubory README,txt , pybcdc,inf , main.py a boot.py. Do souboru main.py můžeme umisťovat vlastní program v microPythonu.

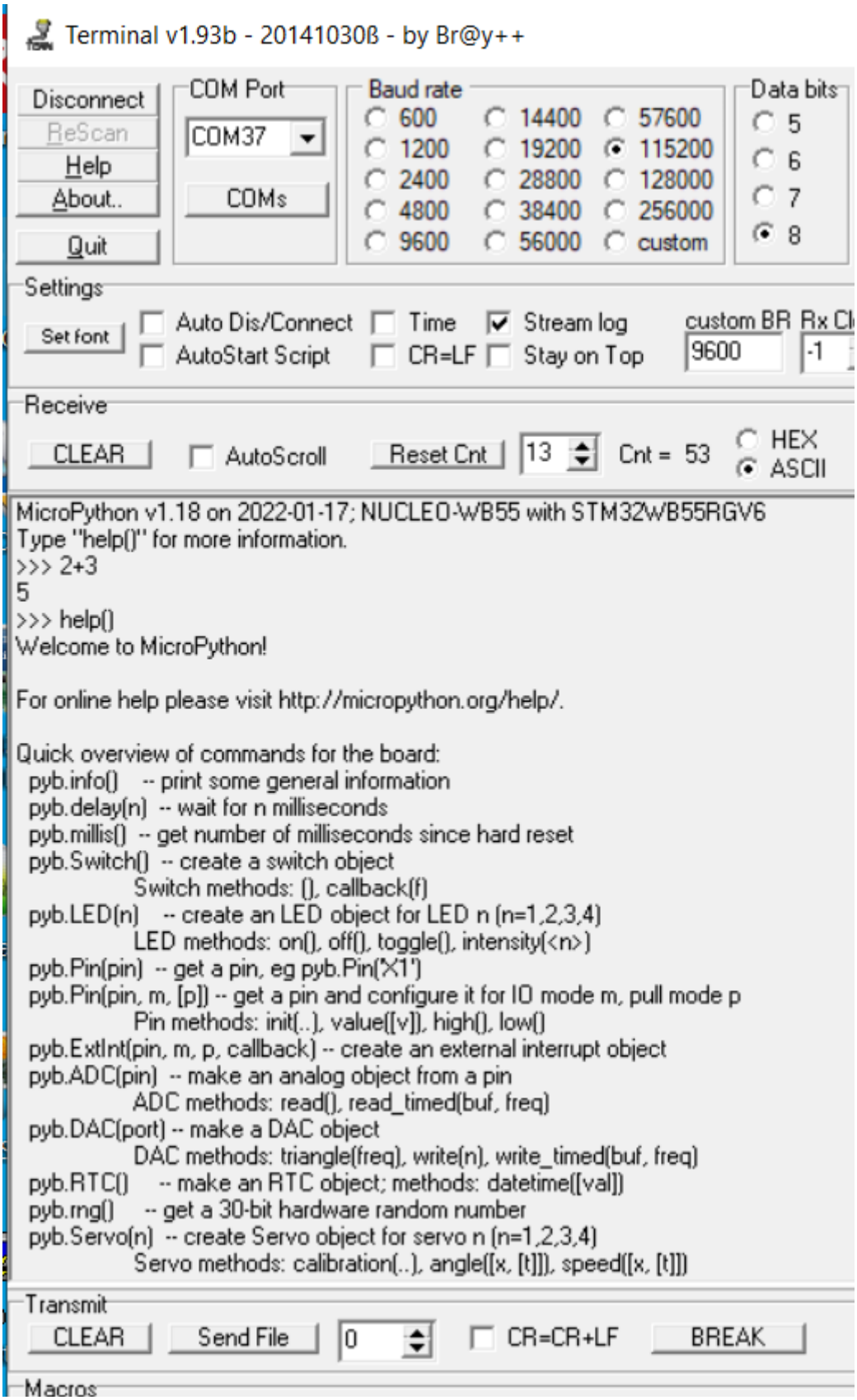
USB konektor ST-LINK používáme pak ke komunikaci. Na terminálovém programu nastavíme komunikační rychlost 115200Bd, 8bitová data, bez parity a s jedním stop bitem. Po připojení k terminálovému programu a resetování NUCLEA se na terminálovém programu ohlásí microPython:



Nyní můžeme microPython vyzkoušet. Zkusíme například odeslat 2+3. Dostaneme:



Nyní odešleme help() a dostaneme:



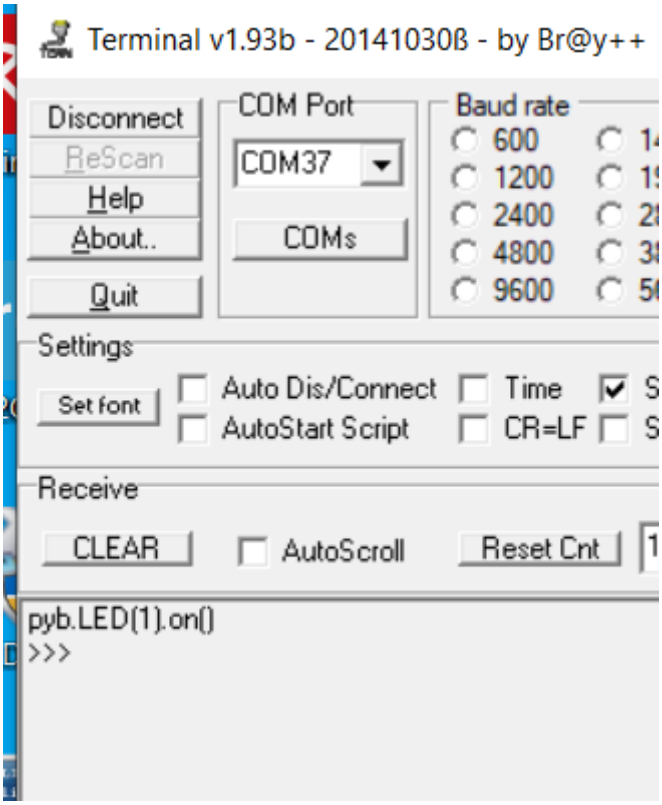
Kompletní výpis odpovědí helpu je

>>> help()

```
Welcome to MicroPython!

For online help please visit http://micropython.org/help/.

Quick overview of commands for the board:
  pyb.info()    -- print some general information
  pyb.delay(n)  -- wait for n milliseconds
  pyb.millis()  -- get number of milliseconds since hard reset
  pyb.Switch()  -- create a switch object
                   Switch methods: (), callback(f)
  pyb.LED(n)    -- create an LED object for LED n (n=1,2,3,4)
                   LED methods: on(), off(), toggle(), intensity(<n>)
  pyb.Pin(pin)  -- get a pin, eg pyb.Pin('X1')
  pyb.Pin(pin, m, [p]) -- get a pin and configure it for IO mode m, pull mode p
                   Pin methods: init(..), value([v]), high(), low()
  pyb.ExtInt(pin, m, p, callback) -- create an external interrupt object
  pyb.ADC(pin)  -- make an analog object from a pin
                   ADC methods: read(), read_timed(buf, freq)
  pyb.DAC(port) -- make a DAC object
                   DAC methods: triangle(freq), write(n), write_timed(buf, freq)
  pyb.RTC()     -- make an RTC object; methods: datetime([val])
  pyb.rng()     -- get a 30-bit hardware random number
  pyb.Servo(n)  -- create Servo object for servo n (n=1,2,3,4)
                   Servo methods: calibration(..), angle([x, [t]]), speed([x, [t]])
  pyb.Accel()   -- create an Accelerometer object
                   Accelerometer methods: x(), y(), z(), tilt(), filtered_xyz()
Pins are numbered X1-X12, X17-X22, Y1-Y12, or by their MCU name
Pin IO modes are: pyb.Pin.IN, pyb.Pin.OUT_PP, pyb.Pin.OUT_OD
Pin pull modes are: pyb.Pin.PULL_NONE, pyb.Pin.PULL_UP, pyb.Pin.PULL_DOWN
Additional serial bus objects: pyb.I2C(n), pyb.SPI(n), pyb.UART(n)

Control commands:
  CTRL-A        -- on a blank line, enter raw REPL mode
  CTRL-B        -- on a blank line, enter normal REPL mode
  CTRL-C        -- interrupt a running program
  CTRL- Welcome to MicroPython!

For online help please visit http://micropython.org/help/.

Quick overview of commands for the board:
  pyb.info()    -- print some general information
  pyb.delay(n)  -- wait for n milliseconds
  pyb.millis()  -- get number of milliseconds since hard reset
  pyb.Switch()  -- create a switch object
                   Switch methods: (), callback(f)
  pyb.LED(n)    -- create an LED object for LED n (n=1,2,3,4)
                   LED methods: on(), off(), toggle(), intensity(<n>)
  pyb.Pin(pin)  -- get a pin, eg pyb.Pin('X1')
  pyb.Pin(pin, m, [p]) -- get a pin and configure it for IO mode m, pull mode p
                   Pin methods: init(..), value([v]), high(), low()
```

```
  pyb.ExtInt(pin, m, p, callback) -- create an external interrupt object
  pyb.ADC(pin)  -- make an analog object from a pin
                ADC methods: read(), read_timed(buf, freq)
  pyb.DAC(port) -- make a DAC object
                DAC methods: triangle(freq), write(n), write_timed(buf, freq)
  pyb.RTC()     -- make an RTC object; methods: datetime([val])
  pyb.rng()     -- get a 30-bit hardware random number
  pyb.Servo(n)  -- create Servo object for servo n (n=1,2,3,4)
                Servo methods: calibration(..), angle([x, [t]]), speed([x, [t]])
  pyb.Accel()   -- create an Accelerometer object
                Accelerometer methods: x(), y(), z(), tilt(), filtered_xyz()

D       -- on a blank line, do a soft reset of the board
  CTRL-E        -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)
For a list of available modules, type help('modules')
>>>
```

Nyní odešleme např. **pyb.LED(1).on()**



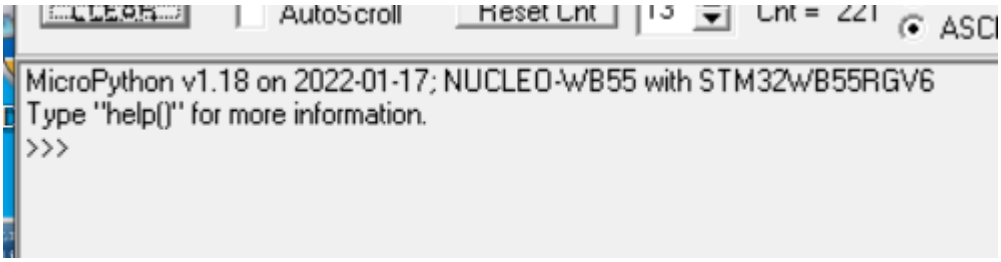
což způsobí rozsvícení modré LED. Poté příkazem pyb.LED(1).off() modrou LED zhasneme. Pokud u těchto dvou příkazů bude parametrem 2, budeme ovládat zelenou LED a v případě parametru 3 LED červenou.
Nyní napíšeme do souboru main.py v adresáři **pybflash** program

main – Poznámkový blok

Soubor Úpravy Formát Zobrazení Nápověda

```
# main.py -- put your code here!
import pyb
from utime import sleep_ms
pyb.LED(3).on()
sleep_ms(1000)
pyb.LED(3).off()
sleep_ms(1000)
pyb.LED(2).on()
sleep_ms(1000)
pyb.LED(2).off()
sleep_ms(1000)
pyb.LED(1).on()
sleep_ms(1000)
pyb.LED(1).off()
```

Napsaný program ještě uložíme a startkit zresetujeme. Nyní se postupně rozsvítí a zhasnou červená, zelená a modrá LED s prodlevami 1s a teprve po ukončení tohoto programu se na konzoli vypíše



5.Ukázky programů

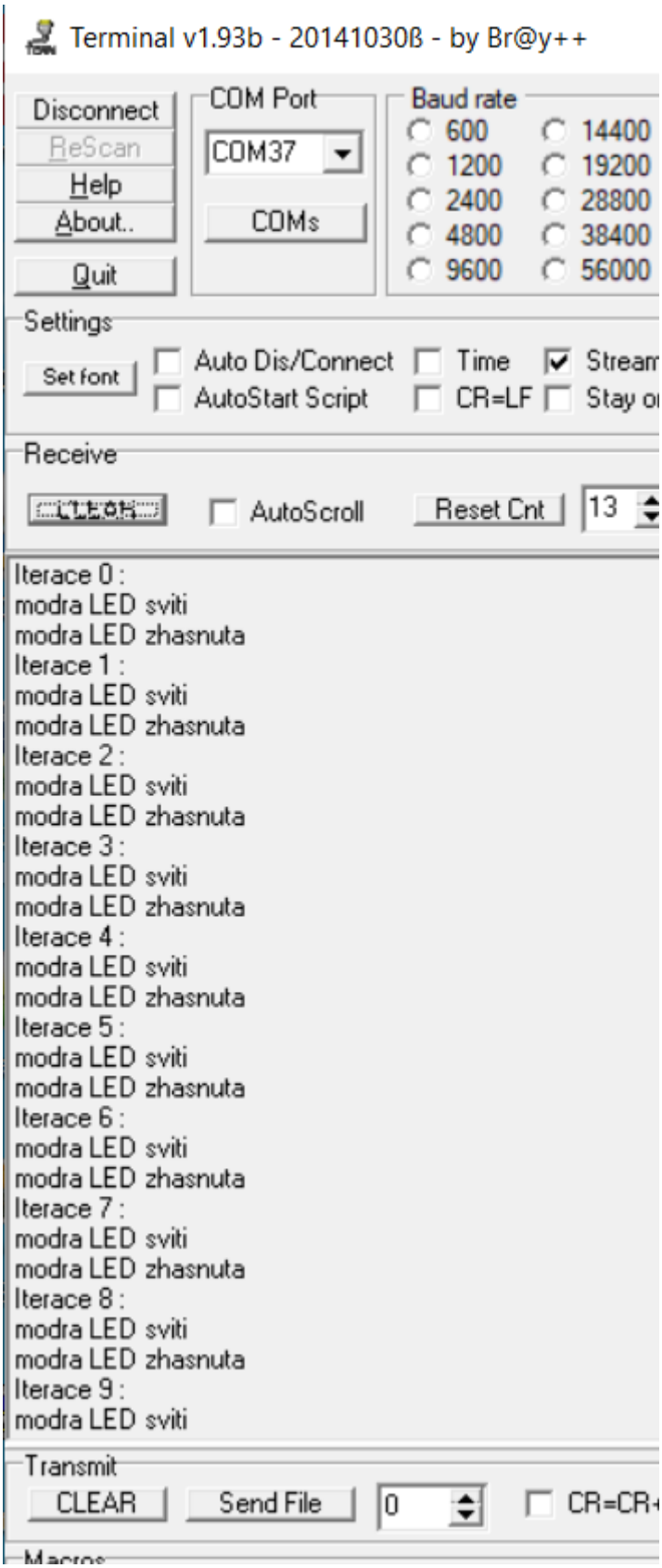
5.1 zablikání modrou LED 10x

Do souboru **main.py** v adresáři PYBFLASH zapíšeme

```
# main.py -- put your code here!
# Script object:
# Example of flashing the blue NUCLEO-WB55 LED at a given frequency.

import pyb # for device access (GPIO, LED, etc.)
from time import sleep # for system breaks (among others)

# Blue LED Initialization
led_bleue = pyb.LED(3) # LED3 screen-printed on the PCB
```

```
duration = 0.5 # Waiting time before changing LED status

# The loop will repeat ten times (for i from 0 to 9)
for i in range(10):

    # Displays the iteration index on the USB User serial port

    # print("Iterace {:2d}".format(i))


    print("Iterace %d : "%i)

    led_bleue.on() # Turns on the LED
    print("modra LED sviti")
    sleep(duration) # Waits for "duration" seconds

    led_bleue.off() # Turns off the LED
    print("modra LED zhasnuta")
    sleep(duration) # Waits for "duration" seconds
```

Po spuštění programu dostaneme

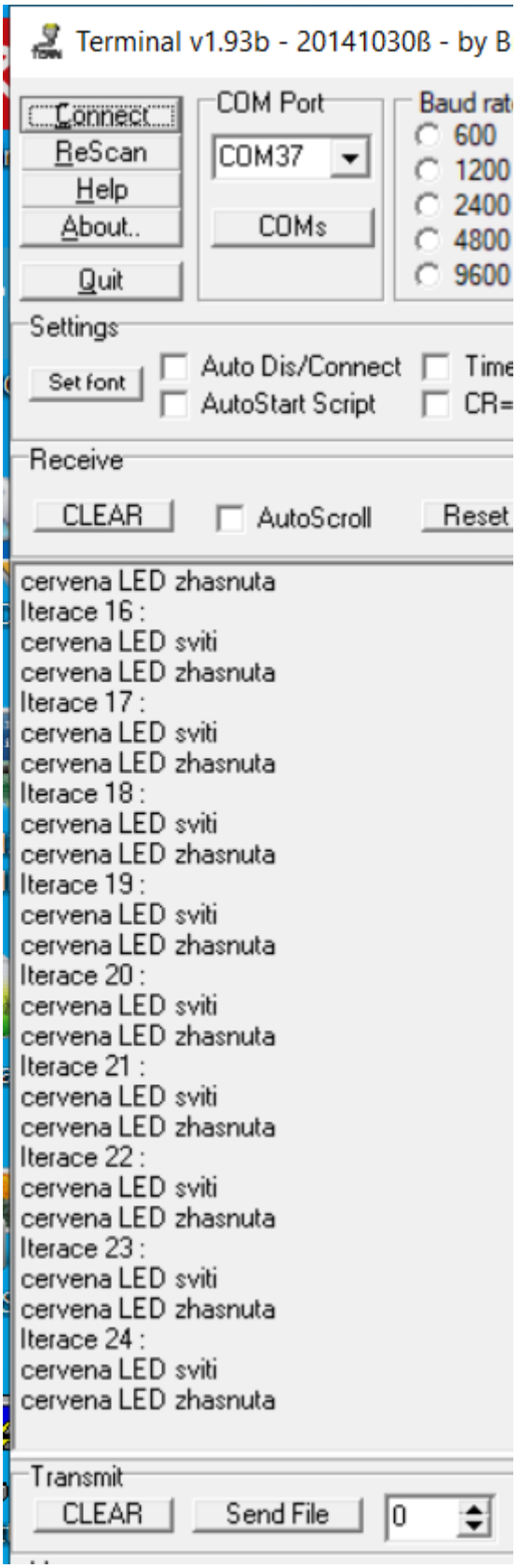


5.2 blikání LED v nekonečné smyčce

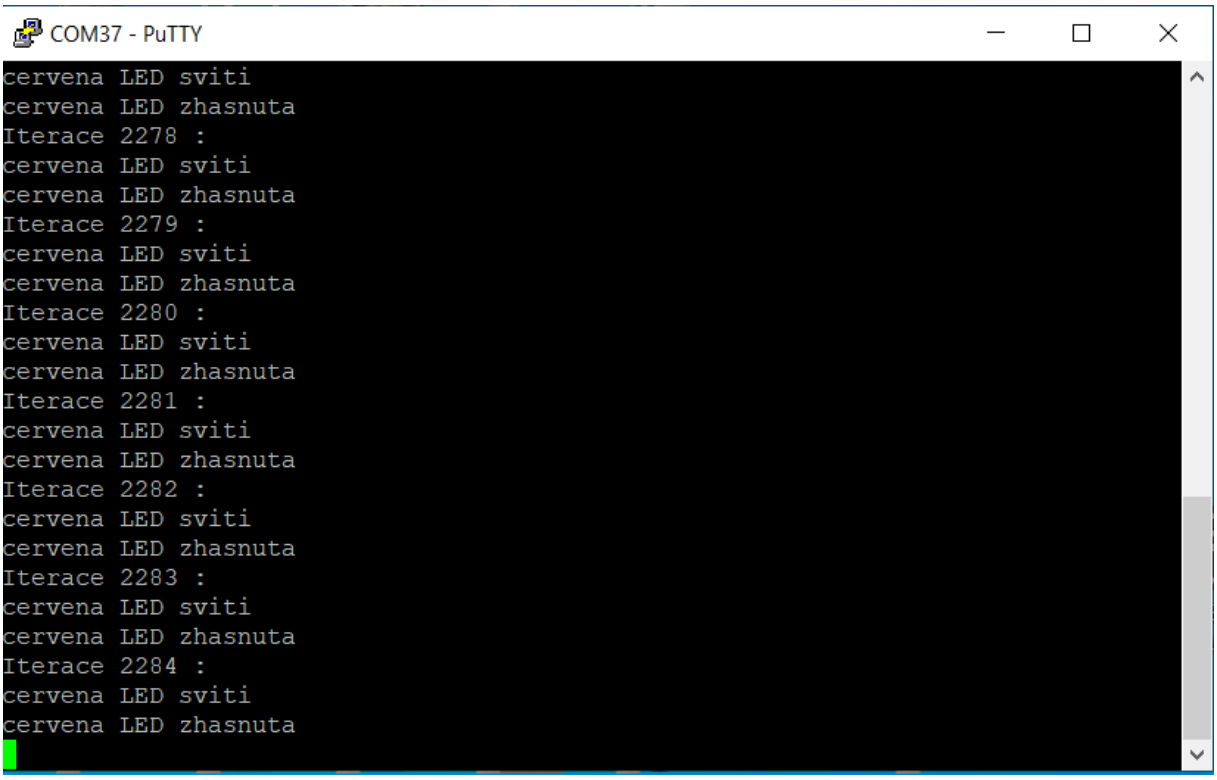
Předchozí program v souboru **main.py** v adresáři PYBFLASH modifikujeme

```
# Script object:
# Example of flashing the blue NUCLEO-WB55 LED at a given frequency.

import pyb # for device access (GPIO, LED, etc.)
from time import sleep # for system breaks (among others)

# Blue LED Initialization
led_bleue = pyb.LED(3) # LED3 screen-printed on the PCB

duration = 0.5 # Waiting time before changing LED statu
i=0
# The loop will repeat
while True:

    # Displays the iteration index on the USB User serial port

    # print("Iterace {:2d}".format(i))


    print("Iterace %d : "%i)
    i=i+1
    led_bleue.on() # Turns on the LED
    print("cervena LED sviti")
    sleep(duration) # Waits for "duration" seconds

    led_bleue.off() # Turns off the LED
    print("cervena LED zhasnuta")
    sleep(duration) # Waits for "duration" seconds
```

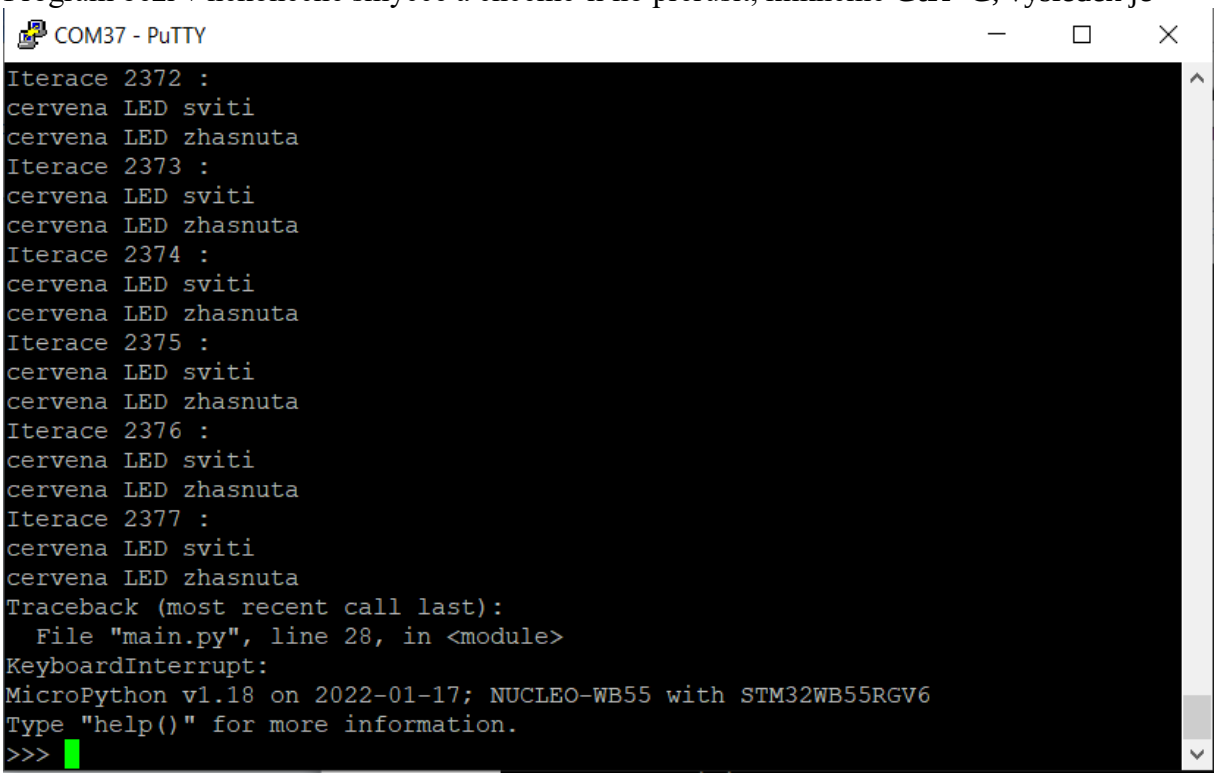
Po jeho spuštění LED trvale bliká a v terminálovém programu vidíme



Použijeme-li jako terminálový program **Putty**, vidíme



Program běží v nekonečné smyčce a chceme-li ho přerušit, klikneme **Ctrl+C**, výsledek je



6.Závěr

Ověřili jsme si, že programování MCU STM32WB v jazyce microPython je velice jednoduché. Ukázky programování komunikace I2C (30 příkladů) najdete v [7], [8]. Programování ADC, LCD displeje a BT třeba v [9] .

7.Zdroje

[1]domovská stránka microPythonu https://www.micropython.org/
[2] https://www.st.com/en/evaluation-tools/nucleo-wb55rg.html
[3] microPython tutorial https://docs.micropython.org/en/latest/pyboard/tutorial/index.html
[4] MicroPython documentation https://docs.micropython.org/en/latest/
[5] https://stm32python.gitlab.io/en/docs/Micropython/
[6] https://github.com/peterhinch/micropython-async/blob/master/v3/docs/TUTORIAL.md
[7] https://learn.sparkfun.com/tutorials/micropython-programming-tutorial-getting-started-with-the-esp32-thing/experiment-4-i2c
[8] Python machine.I2C Examples
https://www.programcreek.com/python/example/101407/machine.I2C
[9] https://stm32python.gitlab.io/en/docs/Micropython/exercises