

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 startkitu odzkoušet na NUCLEO STM32WB firmv 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 perifériemi 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/



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 BOOTO 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]
```

Ĵ

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.

5 STM32 ST-LIN	K Utility							—		\times
File Edit View Target ST-LINK External Loader Help										
🖴 🖥 👙	Ç 🏈 💱	s 🧝 🔜								
Memory display						Device	STM32WBxx			
Address: 0x08	000000 V Size	0x631EC	Data Wid	tth: 32 bits		Device ID	0x495			
						Revision ID	Rev Y			
Device Memory @	0x08000000 : F	ile : NUCLEO WE	355-20220117-v	1.18.bex		Flash size	1MBytes			Indate
Target memory, Add	dress range: [0x0	8000000 0x0806	31EC]							puate
Address	0	4	8	С	ASCII				•	^
0x08000000	2002EFF8	0804425D	0803C24F	0803C23D	øï.][3OÂ=Â			-13	_
0x08000010	0803C251	0803C261	0803C271	00000000	QÂa	ÂqÂ				
0x08000020	00000000	0000000	00000000	0803C281		Â				
0x08000030	0803C283	0000000	0803D045	0803D07D	f Â	EÐ}Ð				
0x08000040	0804415D	0804415D	0804415D	0803C305]A]	A]AÃ.				
0x08000050	08043475	0804415D	0803C291	0803C297	u 4]	A'—Â				
0x08000060	0803C29D	0803C2A3	0803C2A9	0803F795	£	À©Â∙÷				
0x08000070	0803F7A5	0803F7B5	0803F7C5	0803F7D5	¥÷μ	÷Â÷Õ÷	÷			
0x08000080	0803F7E5	0803F7F5	0804415D	0804415D	å ÷ õ	÷]A]A.				
<	1									>
21.03.22 . 300110	equency - 7,0 m	12.								^
21:05:22 : Connect	n Low Power mode	e enabled.								
21:05:22 : Device I 21:05:22 : Device f	D:0x495 lash Size : 1MByte	-c								- 10
21:05:22 : Device family :STM32WBxx										
21:06:05 : [NUCLEO_WB55-20220117-v1.18.hex] opened successfully. Address Ranges [0x08000000 0x0800013C] [0x08000140 0x080631EC]										
21:06:05 : [NUCLEO_WB55-20220117-v1.18.hex] checksum : 0x02677FB3										
21:00:24 : Memory	programmed in 1	us anu ogenis.								~
Debug in Low Power	r mode enabled.		Device ID:0x4	195		1	Core State : Live Update	Disabled		

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:

÷

	🧸 Terminal v1.93b - 20141030B - by Br@y++							
	Disconnect <u>R</u> eScan <u>H</u> elp <u>A</u> bout <u>Q</u> uit	COM Port	Baud rate C 600 C 14400 C 1200 C 19200 C 2400 C 28800 C 4800 C 38400 C 9600 C 56000	C 57600 C 5 C 115200 C 6 C 128000 C 7 C 256000 C 7 C custom C 8	bits Parity none Odd C even C mark C space	Stop bits	Handshaking none RTS/CTS XON/XOFF RTS/CTS+XON/XOFF RTS/CTS+XON/XOFF RTS on TX invert	
ľ	Settings					un fine a	r	
	Set font	Auto Dis/Connect AutoStart Script	CR=LF Stay o	n log <u>custom BR</u> n Top 9600	-1 Crap	ph Rem	note	
ł	Receive							
	CLEAR	AutoScroll	Reset Cnt 13 🚖	Cnt = 7	X Cline LogDate	Stamp topLog R	□ Dec □ Bin eq/Resp □ □ Hex	
	MicroPython v1 Type "help()" fo >>>	.18 on 2022-01-17; r more information.	NUCLEO-WB55 with S	TM32WB55RGV6				

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

MicroPython v1.18 on 2022-01-17; NUCLEO-WB55 with STM32WB55RGV6 Type "help()" for more information. >>> 2+3 5 >>>

Nyní odešleme help() a dostaneme:

arminal v1.93b - 20141030ß - by Br@y++				
Disconnect COM Port Baud rate Data bits BeScan COM37 ▼ 600 14400 57600 0 5 Help COMs COMs 0 19200 115200 0 6 Quit COMs 0 0 38400 256000 0 7				
Settings Set font Auto Dis/Connect Time Stream log custom BR Rx C AutoStart Script CR=LF Stay on Top 9600 1				
CLEAR AutoScroll Reset Cnt 13 Cnt = 53 ASCI				
 MicroPython v1.18 on 2022-01-17; NUCLEO-WB55 with STM32WB55RGV6 Type "help[)" for more information. >>> 2+3 >>> 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.witch() 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>)</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.RTC() make an ARC object DAC methods: triangle(freq), write(n), write_timed(buf, freq) pyb.RTC() get a 30-bit hardware random number pyb.Servo(n) create Servo object for servo n (n=1,2,3,4) Servo methods: calibration(), analef(x, Itil) 				
Transmit CLEAR Send File 0 🗢 CR=CR+LF BREAK				
Macros				

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()
```

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()

2	뢽 Terminal v1.93b - 20141030ß - by Br@y++						
ir	Disconnect <u>R</u> eScan <u>H</u> elp <u>A</u> bout	COM Port	Baud rate C 600 C 14 C 1200 C 15 C 2400 C 24 C 4800 C 35 C 9600 C 55				
	Settings						
20	Set font	Auto Dis/Connect AutoStart Script	t ☐ Time 🔽 S □ CR=LF □ S				
	Receive						
	CLEAR	AutoScroll	Reset Cnt 1				
	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

```
AutoScroll Reset Unt 13 Unt = 221 ASC

MicroPython v1.18 on 2022-01-17; NUCLEO-WB55 with STM32WB55RGV6

Type "help()" for more information.

>>>
```

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

🧸 Terminal v1.93b - 2014103	Oß - by Br@y++
Disconnect <u>ReScan</u> <u>H</u> elp <u>A</u> bout <u>Q</u> uit	Baud rate C 600 C 14400 C 1200 C 19200 C 2400 C 28800 C 4800 C 38400 C 9600 C 56000
Set font Auto Dis/Connect	☐ Time ▼ Stream ☐ CR=LF ☐ Stay of
Receive	Reset Cnt 13 🚖
Iterace U: modra LED sviti modra LED zhasnuta Iterace 1 : modra LED sviti modra LED zhasnuta Iterace 2 : modra LED zhasnuta Iterace 3 : modra LED zhasnuta Iterace 3 : modra LED zhasnuta Iterace 4 : modra LED zhasnuta Iterace 5 : modra LED zhasnuta Iterace 6 : modra LED sviti modra LED sviti	
CLEAR Send File 0	CR=CR+
Macros	

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

🧸 Terminal v1.93b - 20141030ß - by E
Connect COM Port Baud ra BeScan COM37 ▼ C 600 Help COM37 ▼ C 1200 About. COMs C 4800 Quit C 9600
Settings
Set font Auto Dis/Connect Tim AutoStart Script CR:
Receive
CLEAR AutoScroll Rese
cervena LED zhasnuta Iterace 16 : cervena LED sviti cervena LED zhasnuta Iterace 17 : cervena LED sviti cervena LED zhasnuta Iterace 18 : cervena LED zhasnuta Iterace 19 : cervena LED zhasnuta Iterace 20 : cervena LED zhasnuta Iterace 20 : cervena LED sviti cervena LED sviti
Transmit CLEAR Send File 0

Použijeme-li jako terminálový program Putty, vidíme

COM37 - PuTTY − □	×
cervena LED sviti	^
cervena LED zhasnuta	
Iterace 2278 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2279 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2280 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2281 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2282 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2283 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2284 :	
cervena LED sviti	
cervena LED zhasnuta	
	\sim

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

Port COM37 - Putty − □	×
Iterace 2372 :	~
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2373 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2374 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2375 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2376 :	
cervena LED sviti	
cervena LED zhasnuta	
Iterace 2377 :	
cervena LED sviti	
cervena LED zhasnuta	
Traceback (most recent call last):	
File "main.py", line 28, in <module></module>	
KeyboardInterrupt:	
MicroPython v1.18 on 2022-01-17; NUCLEO-WB55 with STM32WB55RGV6	
Type "help()" for more information.	
>>>	\sim

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] <u>https://learn.sparkfun.com/tutorials/micropython-programming-tutorial-getting-started-</u>

 $\underline{with-the-esp32-thing/experiment-4-i2c}$

[8] Python machine.I2C Examples

https://www.programcreek.com/python/example/101407/machine.I2C

[9] https://stm32python.gitlab.io/en/docs/Micropython/exercises