ETC22 - **E**mbedded **T**echnology **C**lub

ETC22 - **E**mbedded **T**echnology **C**lub

Organizovaný ČVUT FEL v r. 2024 pro středoškolské studenty se zájmem o techniku a další její studium

Setkání 7 3. 6. 2024

7. ETC22- náplň

Čísla, reprezantace čísel

Hexadecimální, binární, oktalová čísla, reprezentace záporných čísel- dvojkový doplněk

STM32Duino, programování STM32 v prostředí Arduino s nástavbou pro STM32

Struktura pinu- stupně výstupní brány STM32

Čísla, - dekadická, hexadecimální oktalová čísla

Používáme dekadickou soustavu (máme deset prstů)základ =10
Číslice 0 až 9, pro zobrazení větších čísel
10 = jedna desítka, žádná jednotky
20 dvě desítky, žádná
23 = dvě desítky, tři jednotky
Číslo znázorněné počtem tisíců, stovek desítek, jednotek
p . 10^3 + x.10^2 + y .10^1 + z .10^0 (pozn. **10^0 =1)**
4196 = 4. 10^3 + 1.10^2 + 9 .10^1 + 6 .10^0

Fikce, co kdybychom počítali jen na 8 prstech (mimo palců)?
Číslice pouze 1 až 7 a reprezentace čísel, základ = 8
čísla 0, 1,2,3 4 ,5 ,6 ,7 a dál to najede
010, 011, 012, 013, až 017 (to odpovídá dekadicky je 8, 9, 10, 11, 12
0123 Bude odpovídat 1. 8^2 + 2. 8^1 + 3 . 8^0 = 64 dek+ 16 dek +3= 83 dek
oktalové číslo symbolizumme 0 – nulou na začátku (v programu dekadické číslo musí začít jinou číslicí než nulou

Hexadecimální čísla- základ 16 (začíná 0x)

Základ 16, počítáme 0, 1, 2,…..9 a jak dále?

Pro vyšší číslice zavedeme písmenné označení A, B, C, D, E, F

Základ 16, tedy hexadecimální soustava, hexadecimální čísla

Označíme 0x0, 0x1, 0x2,..0x9, 0xA, 0xB, 0xC, 0xD,0xE, 0xF

Dekad. to odpovídá 0, 1, 2, 3,…….9, 10, 11, 12, 13, 14, 15

0xA = 10

0xB = 11

0xC = 12

0xD = 13

0xE = 14

0xF = 15

$$p \,.\, 16^3 + x.16^2 + y\,.16^1 + z\,.16^0$$ **(znázorníme jen pro 4 řády)**

$16^3 = 4096$ $16^2 = 256$ $16^1 = 16$ $16^0 = 1$

Vybraná hexadecimální čísla

Jak větší čísla ?

0x10 = 16, 0x20 = 32, 0x30 = 48, 0x80= 128, 0xFF= 255

0x100 = 256, 0x101 = 257

Hex. Dek.

0x10 = 16

0x100 = 256

0x10 00 =4 096

0x10000 = 65 536

0x10 0000 = 104 8576

0x100 0000 = 16 777 216

0x1000 0000 = 268 435 456

0x 1000 0000 =4 294 967 296

Převod čísla hexadecimálního na dekadické

„Ruční převod“ hexadecimální na dekadické

Podle definice p . 16^3 + x.16^2 + y .16^1 + z .16^0 např.

0x70C5 = 7 x 4096 +0 +12 x 256 + 5 = 31 749 (dek.)

To ještě jde, ale jak obráceně z dek. na hex. ? Ručně to asi nebdme dělat, ale jen pro příklad pro tvorbu jednoduchého programu

Metody odečítání 400 000 - (6 x 65 536) = 400 000 - 3930216 = 6784

6784 – (1 x 4096) = 2688

0x10 = 16 2688 – (10 x 256)= 128 10 odpovídá 0xA

0x100 = 256 128 – (8 x 16) = 8

0x10 00 =4 096 0 – (0 x1)= 0

0x10000 = 65 536

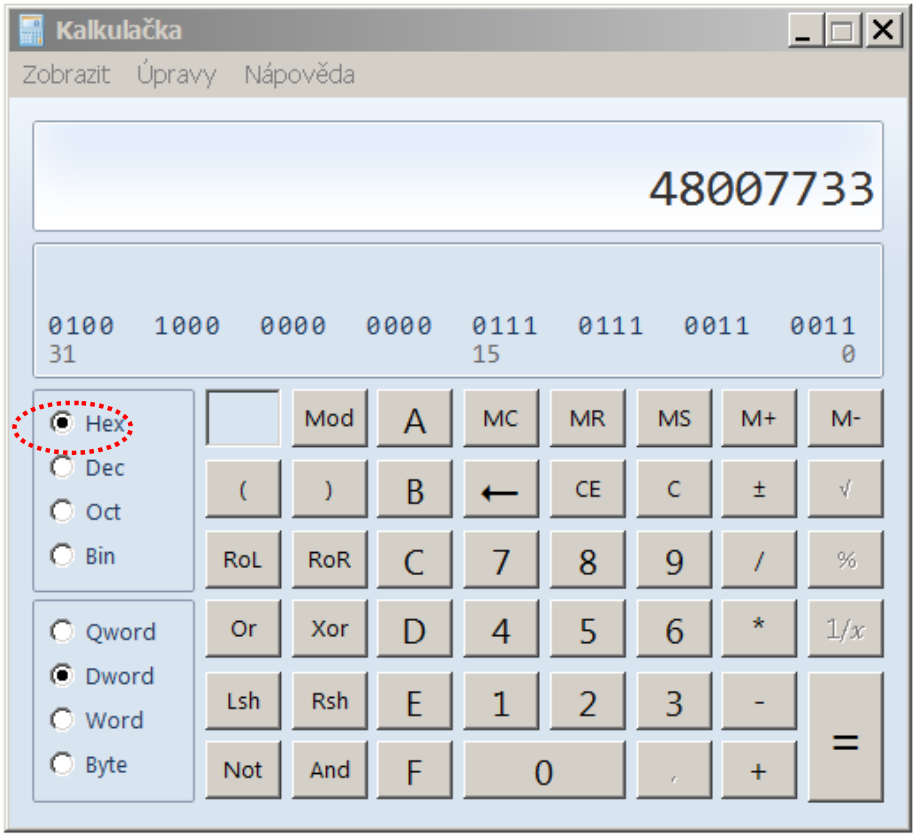
400 000 = 0x61A80

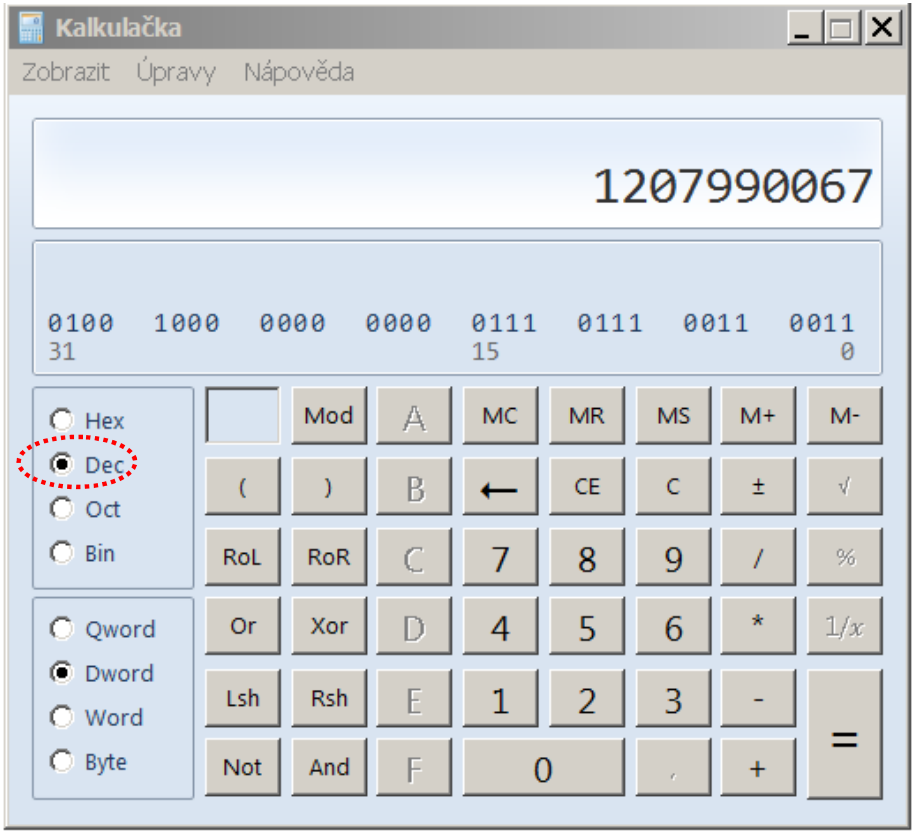
Další způsoby převodu hex- dec

Další způsoby převodu- dělení,….

Bude nás to zajímat, až když bychom to museli naprogramovat

Uživatelsky- kalkulačka ve Windows- převody dec. , hex. Bin čísla





Binární čísla - zásadní pro počítače a logické obvody

V počítači jsou jen dvě možnosti znázornění, dvouhodnotová logika

ANO, NE, (proud protéká, proud neprotéká, případně nebo napětí je, napětí není), Nebo vysoká úroveň „HIGH“ a nízká úroveň „LOW“

Budeme mít jen dvě číslice 0 a 1

Do dvou stavů zakódujeme pouze dvě číslice 0 a 1

Jak větší čísla ? podobně, jako v dek. soustavě

$p \cdot 2^3 + x. 2^2 + y \cdot 2^1 + z \cdot 2^0 = p \cdot 8 + x. 4 + y \cdot 2 + z \cdot 1$

Bin.	Hex.	Dek.	Bin.	Hex.	Dek.
0000	0x0	0	1000	0x8	8
0001	0x1	1	1001	0x9	9
0010	0x2	2	1010	0xA	10
0011	0x3	3	1011	0xB	11
0100	0x4	4	1100	0xC	12
0101	0x5	5	1101	0xD	13
0110	0x6	6	1110	0xE	14
0111	0x7	7	1111	0xF	15

Binární čísla a hexadecimální číla

8 bitová čísla, 16 bitová, 32 bitová

0000 0000 až 1111 1111 odpovídá 0x0 až 0xFF

16 bitová čísla

0000 0000 0000 0000 až 1111 1111 1111 1111 0x0 až 0x FF FF

Zde je vidět, že hexadecimální čísla výhodná pro znázornění reprezentace binárních čísel

Čtyři bity binárního číslo jsou znázorněny jendou hexadecimální číslicí.

(Pozn.: Mezi řády jsou pro čitlnost záznamu vloženy mezery, v normálním zápisu ta však není možné)

Zápis binárního čísla ve Wiring pro Arduino začíná 0b např. 0b10001111

Zobrazení kladných a zápor. čísel - dvojkový doplněk

Kladná čísla - nejvyšší bit 0

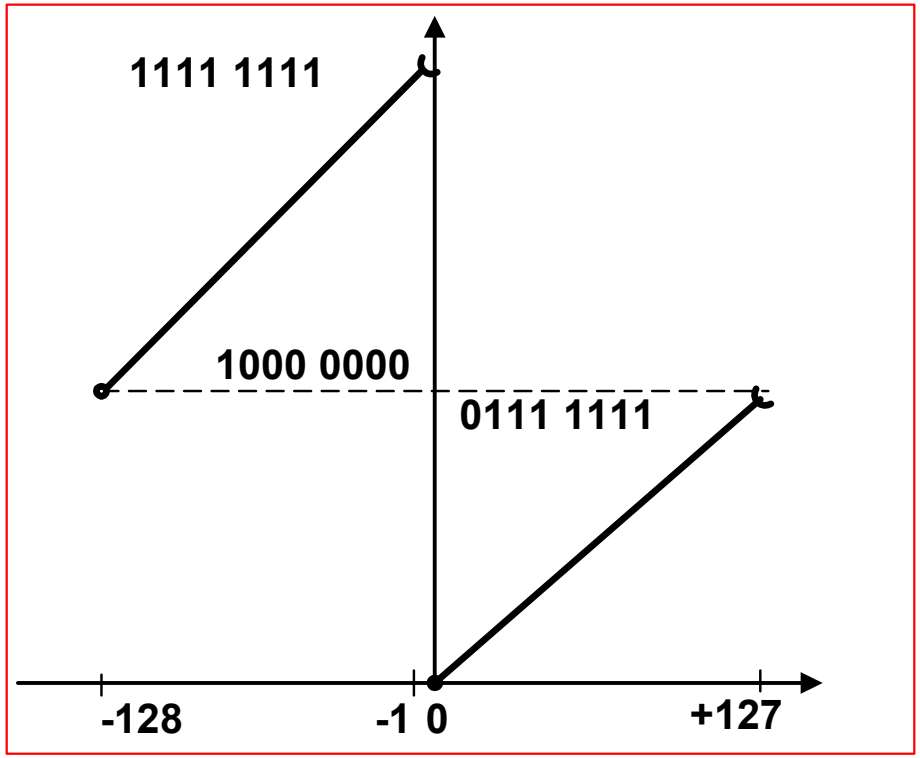
záporná čísla - nejvyšší bit 1

kladná čísla - přímo

záporná čísla - **dvojkový doplněk**

příklad pro 8 bitů

analogicky pro 16, 32, 64 bitů



možnost zobrazení -

kladná čísla - v rozsahu 0 až + $2^{(n-1)}$ -1

(+127; + 32 767; + 2 147 483 647; ....

záporná čísla - v rozsahu 0 až - $2^{(n-1)}$

(- 128; - 32 768; - 2 147 483 648)

Jádro ARM Cortex M **- 2 147 483 648 až + 2 147 483 647**

Čísla v dvojkovém doplňku - znázornění

Znázornění

analogie- hodiny

+ 5 min (po dvanácté 12 h + 5min)

11 h 55 min= za 5 minut dvanáct=

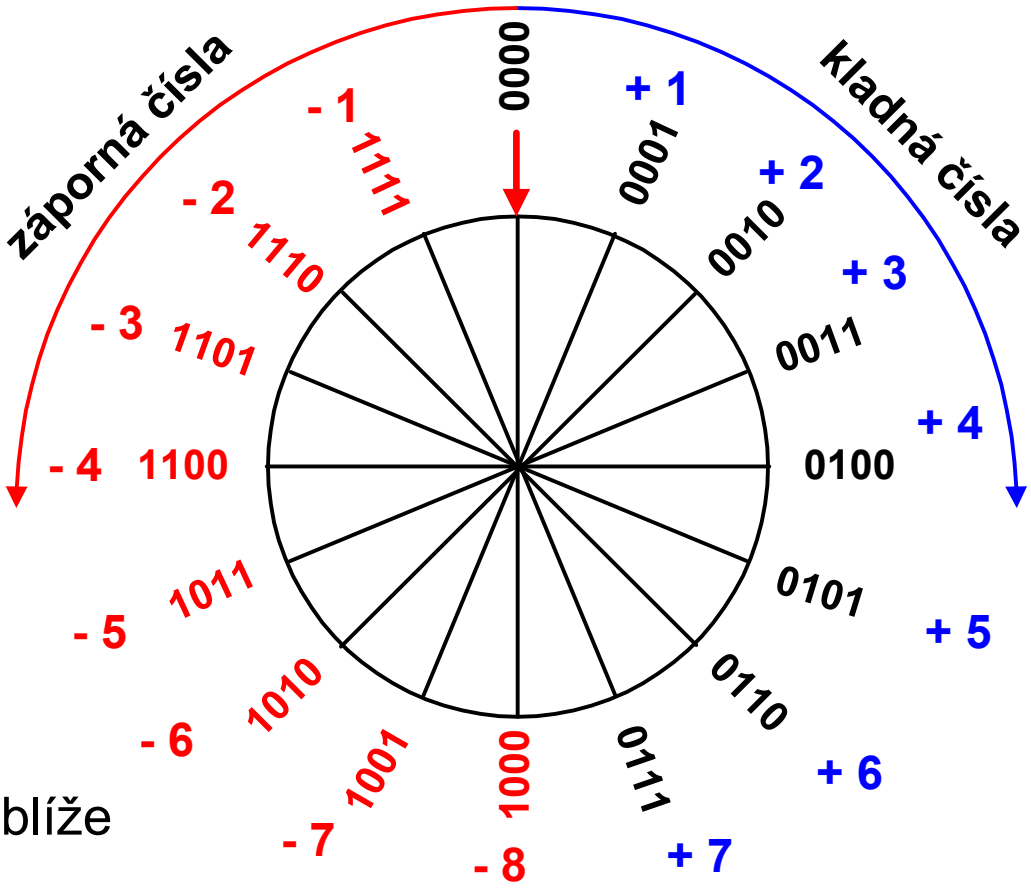
dvanáct – 5 min („záporný čas“)
vůči referenčnímu času

(„pět minut do dvanácté“)

padesátá pátá minuta =

- pátá minuta

(ale na ciferníku je číslo 11 dek.)



nejmenší záporné číslo samé 1111
(největší číslo bez znaménka – nejblíže „celé hodině“

rostoucí záporná čísla („rostoucí vzdálenost od počátku)

(čtvrthodina do celé)

největší záporné číslo - „nejvíce vzdálené celé hodině“ tedy počátku

Čísla v dvojkovém doplňku - převod

Převod čísel

převod kladné - na záporné

negace všech bitů a přičtení 1

příklad převod + 1 na – 1

převod záporné - na kladné

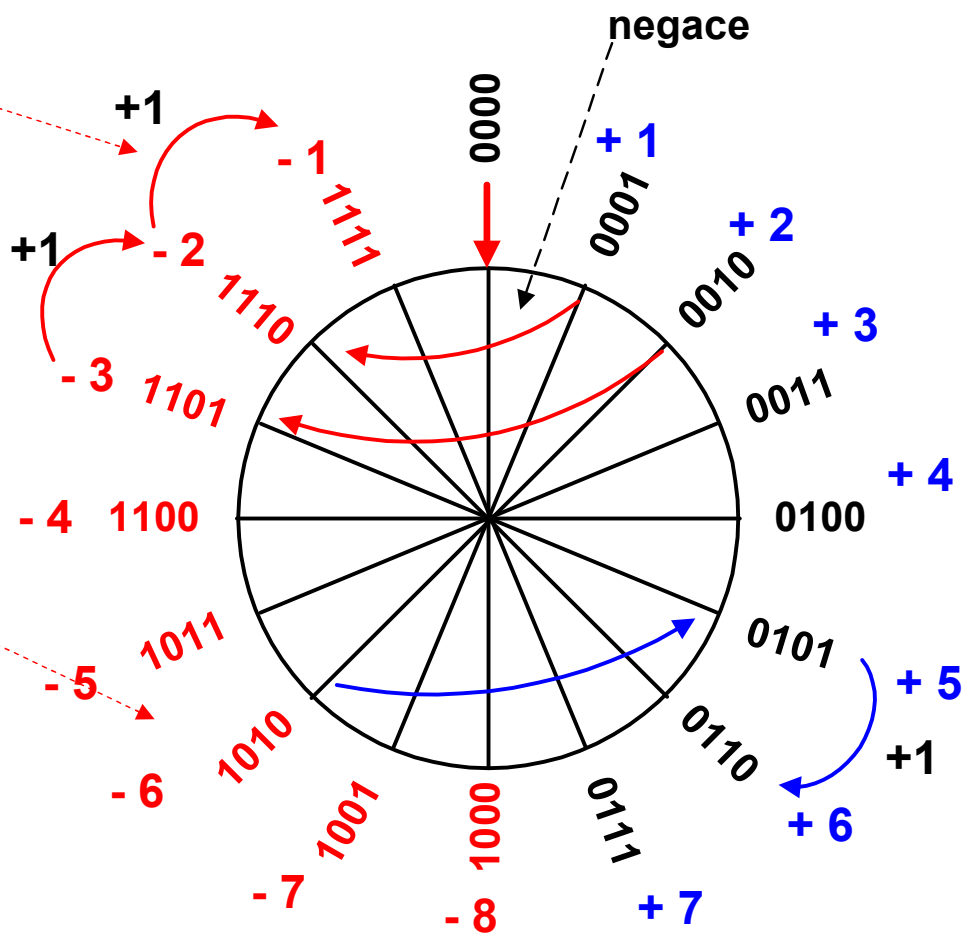
negace všech bitů a přičtení 1

příklad převod - 6 na + 6

-6 …negace = + 5 , + 5 +1 = 6

s rostoucí jemností dělení bude

- 1 symetricky (vedle) + 1 stále



Čísla- 8 bit reprezentace zápor. č. 8 bit - se znam.

8 bitové číslo se znaménkem – dvojkový doplněk

0, +1,+2 …. až +127 d 7Fh 0000 0000 b až 0111 1111 b

-1, -2, až -128 1111 1111 b až 1000 0000 b

Výpočet: kladná čísla - přímo binární ekvivalent

záporná - dvojkový doplněk

Určení dvojkového doplňku, negace všech bitů a přičtení 1

Příklad určení dvojkového doplňku pro čísla -1, -128,

1d	0000 0001 **b**	128 d	1000 0000 b
negace	**1111 1110 b**		**0111 1111 b**
+	**0000 0001**		**+ 0000 0001**
	1111 1111 b = -1 d		**1000 0000 = -128 d**
	FF h		80 h

Reprezentace čísel se zaménkem v počítači

Zatím jsem měli jen nezáporná čísla, jak znázornit záporná čísla

Metoda dvojkového doplňku

https://www.arduino.cc/reference/en/language/variables/constants/integerconstants/

BASE	EXAMPLE	FORMATTER	COMMENT
10 (decimal)	123	none	
2 (binary)	0b1111011	leading "0b"	characters 0&1 valid
8 (octal)	0173	leading "0"	characters 0-7 valid
16 (hexadecimal)	0x7B	leading "0x"	characters 0-9, A-F, a-f valid

Reprezenatace čísel – Arduino- wiring a další

8 bitové unsigned char 0 to 255

8 bitové se znaménkem *char* -128 to 127

byte (same as unsigned char)

int -32,768 to 32,767

unsigned int 0 to 65,535

word (same as unsigned int)

long (or *long int*) -2,147,483,648 to 2,147,483,647

unsigned long 0 to 4,294,967,295

float -3.4028235E+38 to 3.4028235E+38

double (same as float)

Čísla- typy

char	**-128 až 127 (8 bit.)**
unsigned char	**0 až 255 (8 bit.)**
byte (*jako unsig. char*)	**0 až 255 (8 bit.)**
int	**-32,768 to 32,767 (16 bit.)**
unsigned int	**0 až 65,535 (16 bit.)**
word (*stejné jako unsig. int*)	
long (long int)	-2,147,483,648 až 2,147,483,647 (32 bit.)
unsigned long	0 až 4,294,967,295 (32 bit.)

Doporučeno pro zjednodušení, náhrada

Bez znaménka		*se zaménkem*	
uint8_t	(unsigned char)	**int8_t**	(char)
uint16_t	(unsigned int)	**int16_t**	(int)
uint32_t	(unsigned long)	**int32_t**	(long)

Příklady použití zápisu čísel

```
uint8_t cis1 =  0b10001111;
uint8_t cis11 =  0x8F;                  uint8_t cis21 =  143;

int16_t cis2 =  -10000;                 int16_t cis21 = 0xD8F0 ;

uint16_t  cis3 = 0xC000;                uint16_t  cis31 = 49152 ;
uint16_t  cis32=  0b1100000000000000;

uint32_t  cis4  = 0x80000000            uint32_t  cis41=  2147483648 ;
uint32_t  cis42 = 0b10000000000000000000000000000000
```

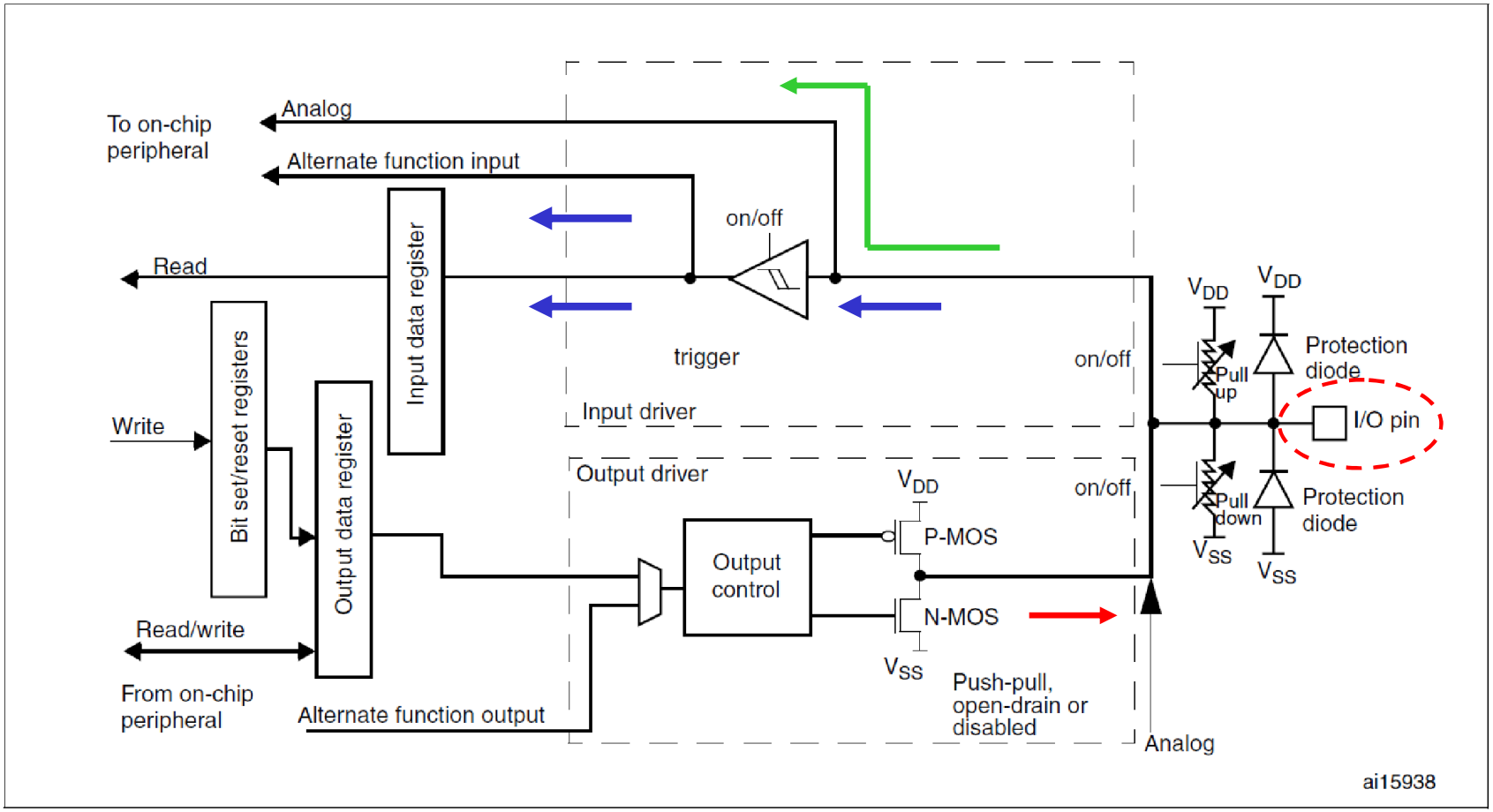
STM32 vstupně výstupní pin

Každý pin brány je samostatně **konfigurovatelný**

***Logické* signály**: brána vstup, výstup, *alternativní* funkce (UART, čítač,..)

***Analogové* signály** – vstup **ADC** , příp. výstup **DAC**

Pozor na záprné napětí a napětí větší , než je napětí na napájení (+3,3 V)



Ideové schéma spolupráce s GPIO v STM32 log. sig.

GPIO - General Purpose Input Output

brány – namapovány jako GPIO registry
do adr. prostoru jako paměti

GPIO prostor u STM32G030 od **0x5000 0000**
i STM32K031 od **0x5000 0000**

STM32F303RE od **0x4800 0000**

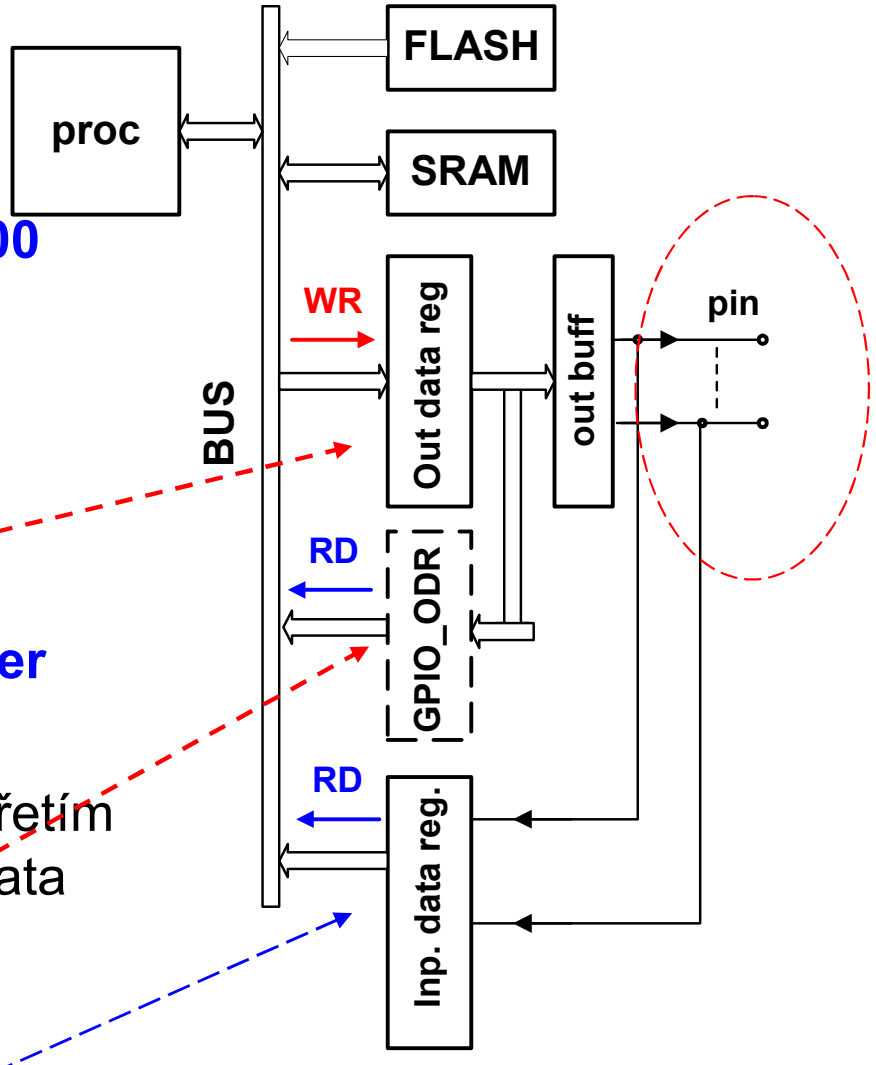
Každý pin s možností komunikace
s okolím může mít funkci vstupu
nebo výstupu

Výstup , regist – **ODR -Output Data Register**

Buffer- „*pustí*“ nebo „*nepustí*“ data ven

Pokud pin jako vstup, výstupní buffer je ve třetím
stavu (stavu vysoké impeance) Buffer data
skončí svou cestu v registru ODR
a „*dále nemohou postupovat“.*

Data z **ODR** lze však číst na stejné adrese,
jako se do ODR zapisovala



Vstupní brána **IDR - Input Data Register** – má jinou adresu, než ODR.

Připojení vstupně/výstupního pinu STM32Fxxx

Každý pin s možností komunikace s okolím může mít funkci **vstupu** nebo **výstupu**

Konfigurační registr- konfigurace vstupu/výstupu **MODE Register**,…

Změna stavu jednoho pinu:

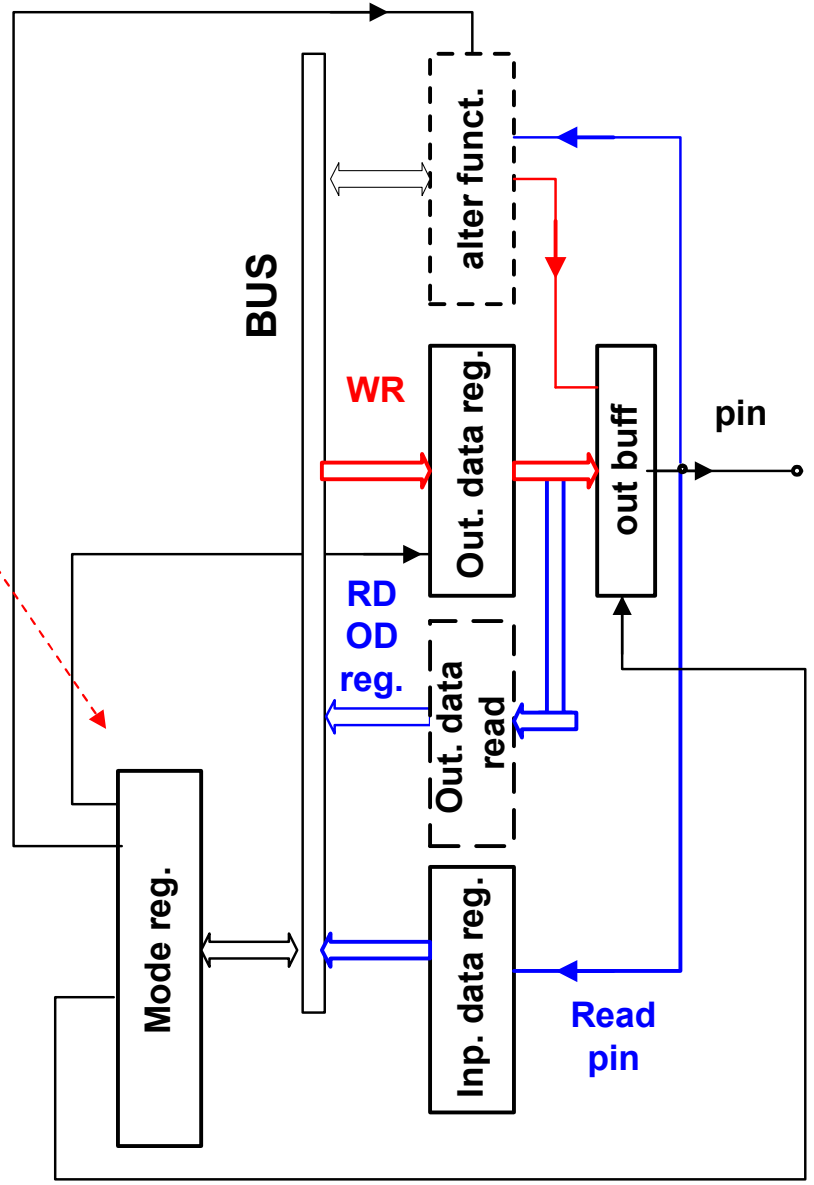
Přečíst obsah ODR (RD), do registru procesoru (např. R0)
modifikovat daný bit (dané bity),
zapsat nazpět do ODR (WR)

„*Read, Modify, Write*“- zabere minimálně. 3. instrukce

To je pomalé, jde to i jinak-registry

GPIOX_BSRR Set – reset registr

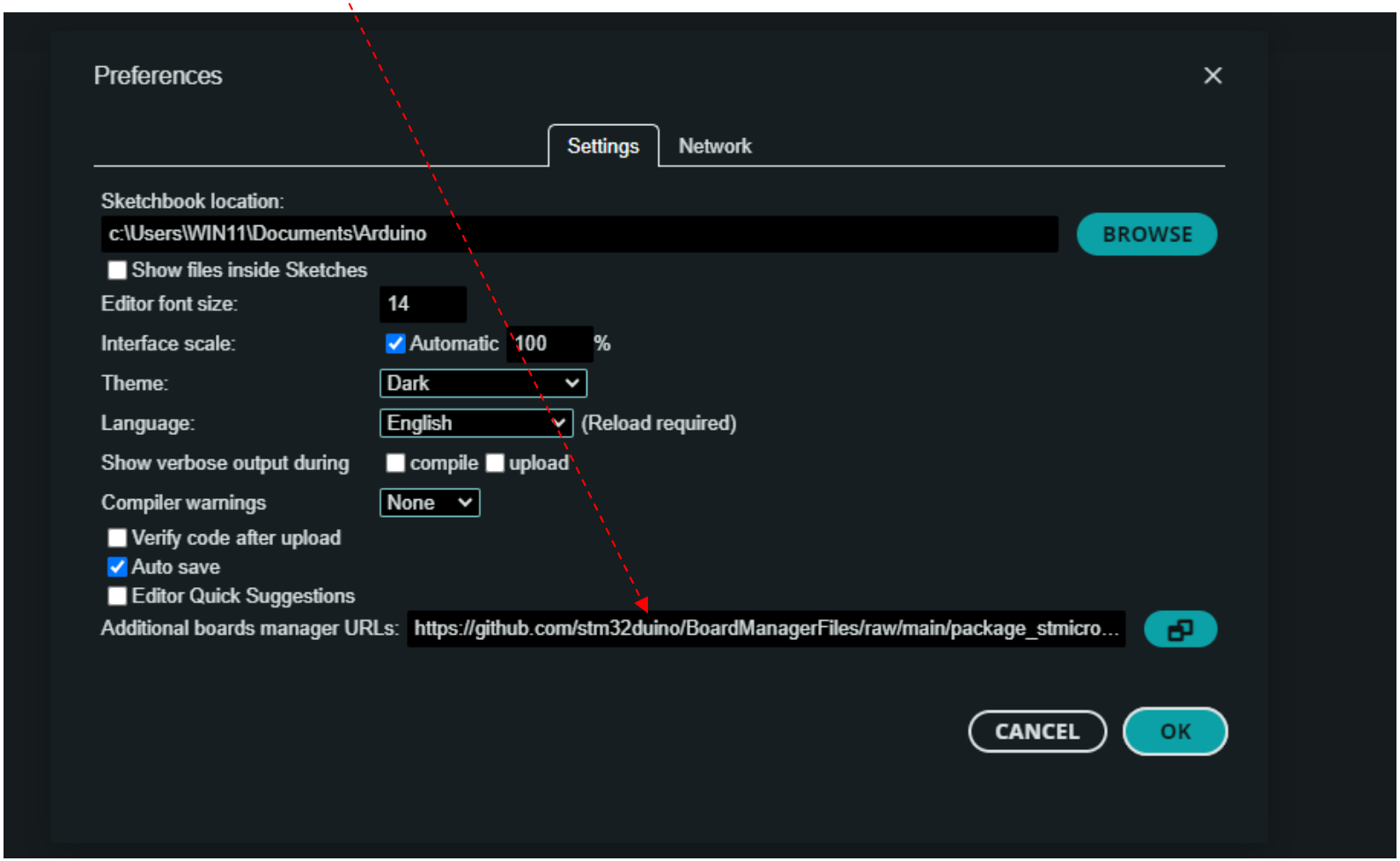
X- GPIO A, B, C, …..



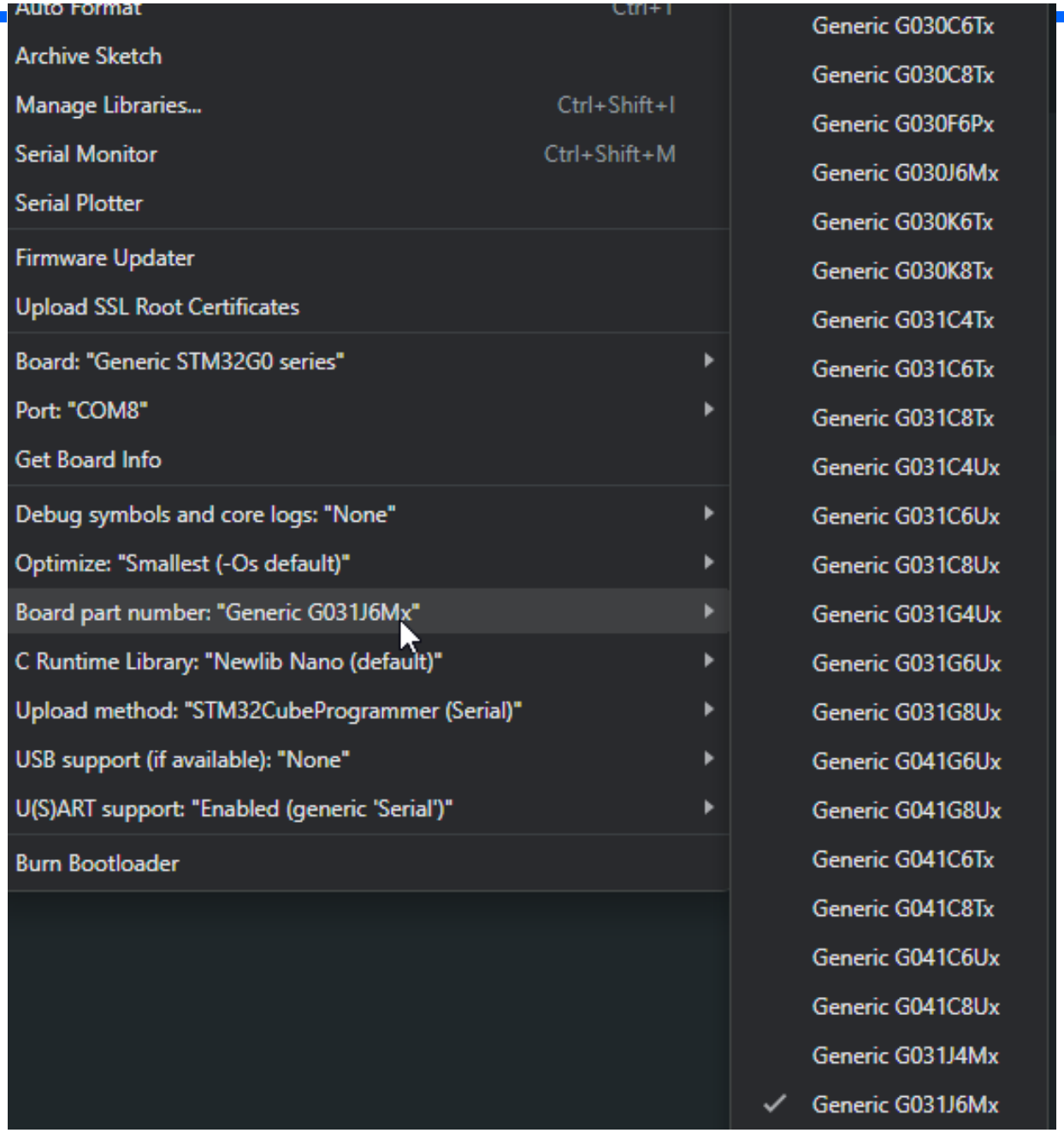
.

.adresa – doplnit, pokud není při instalaci

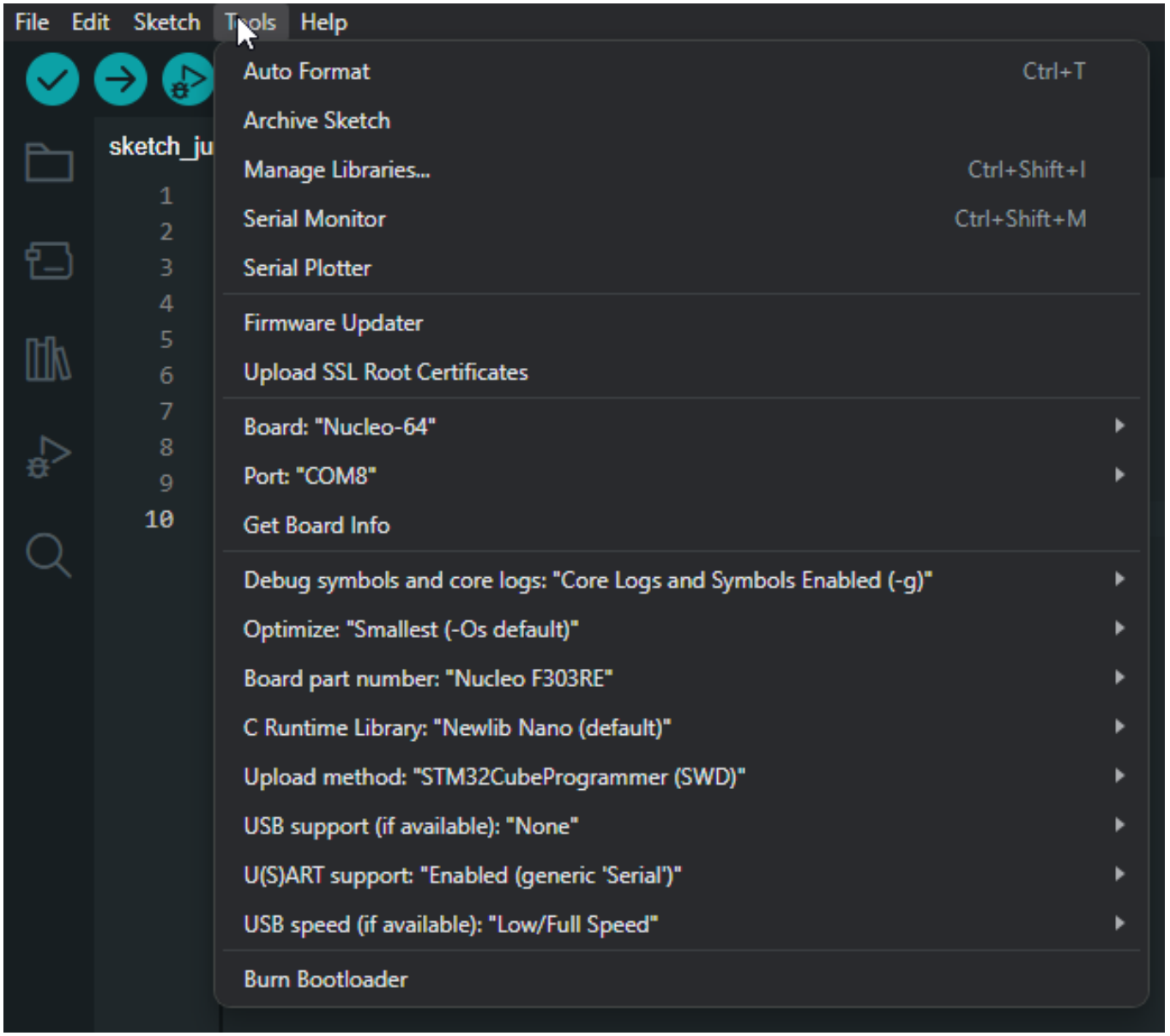
https://github.com/stm32duino/BoardManagerFiles/raw/main/package_stmicroelectronics_index.json

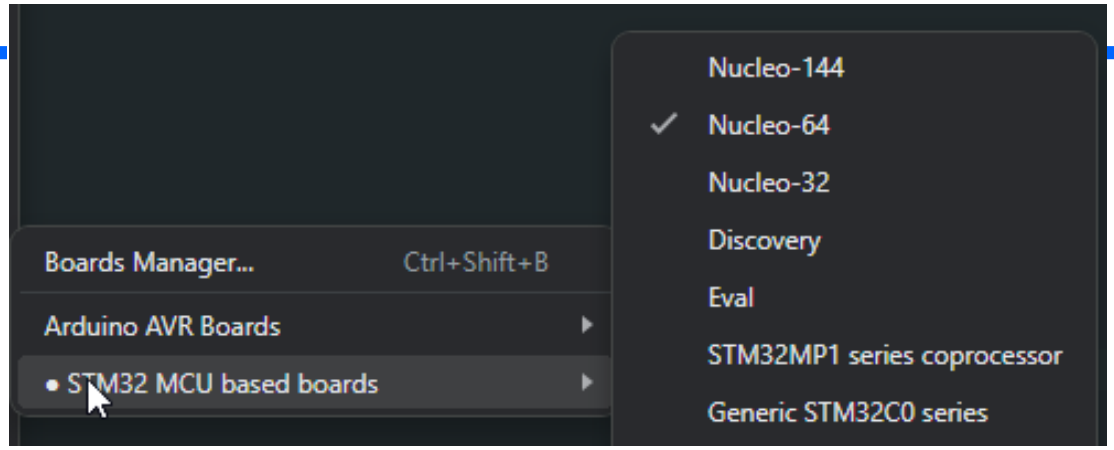


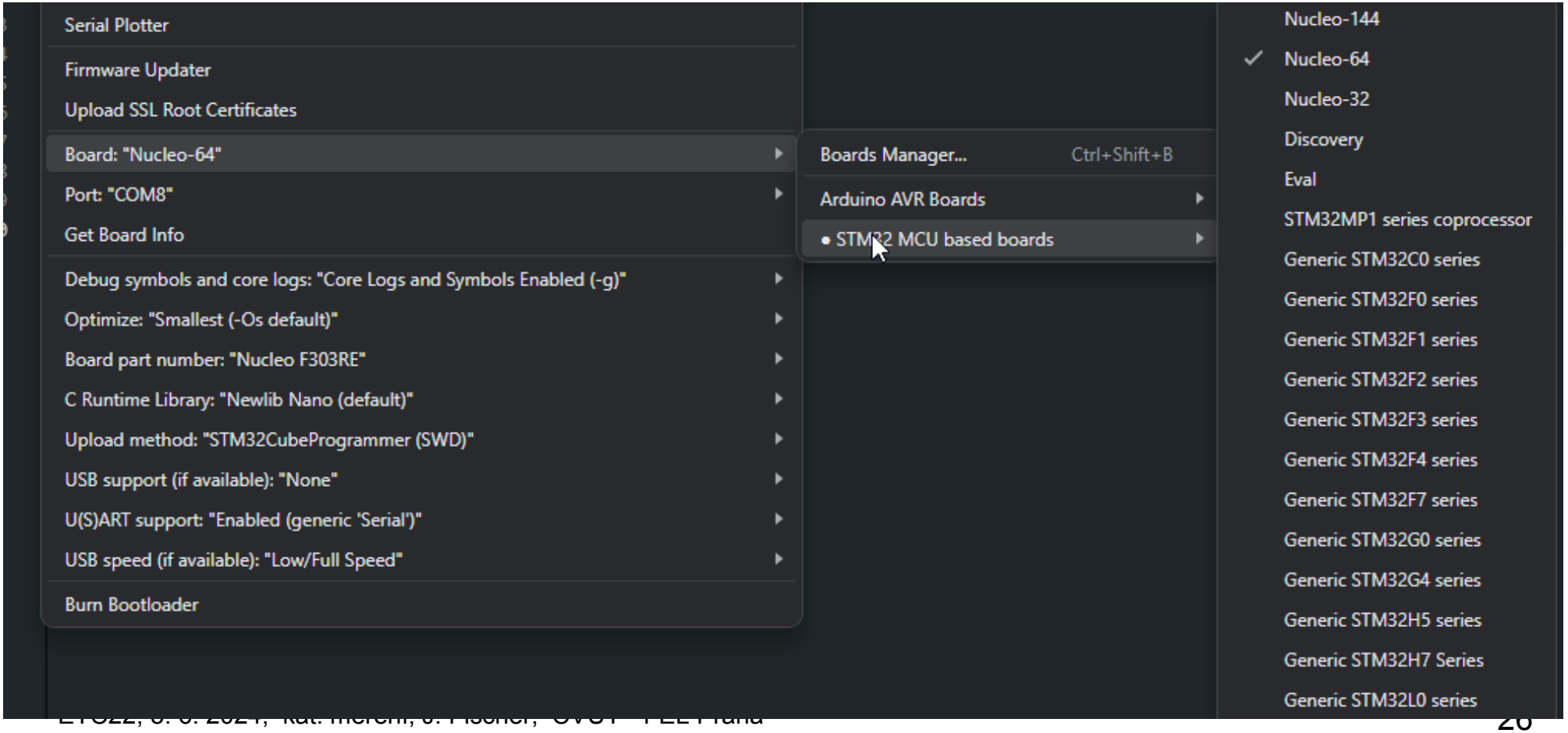
Nastavení- varianta pro STM32G031J6

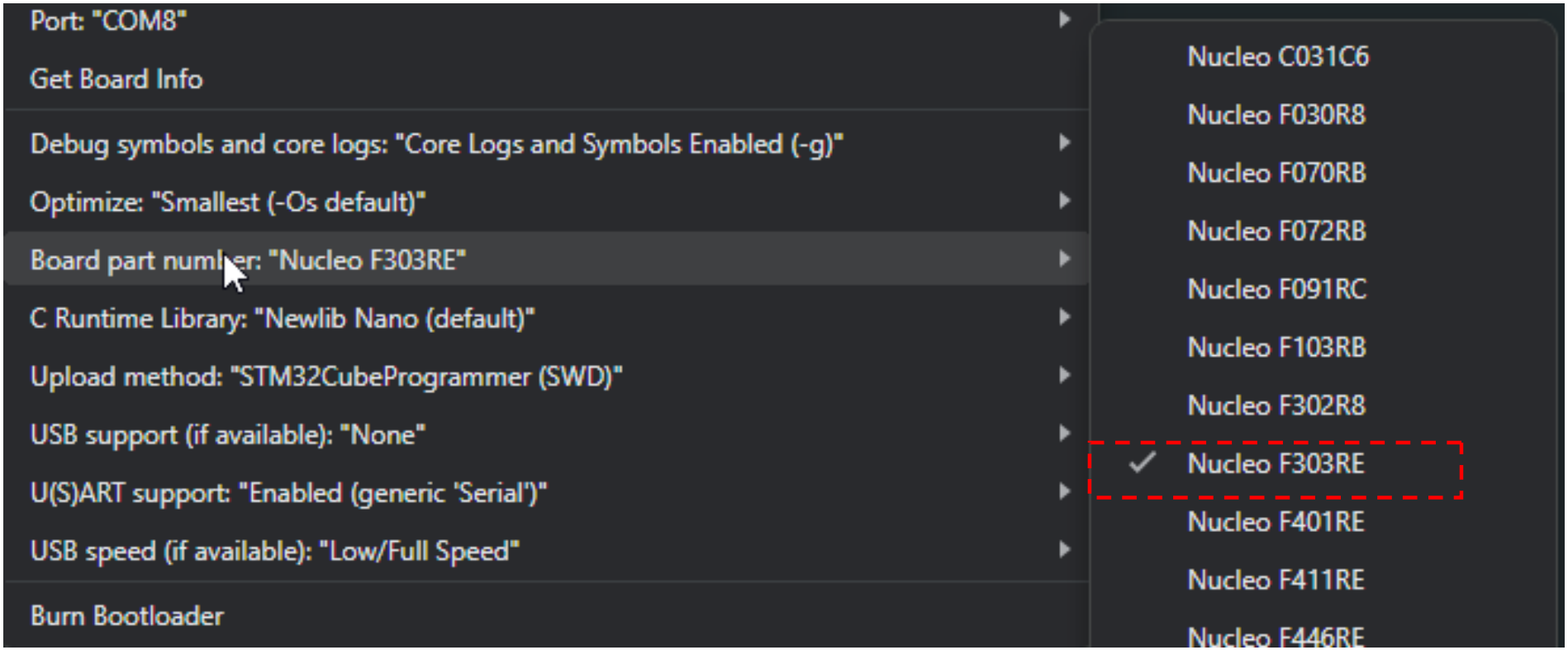


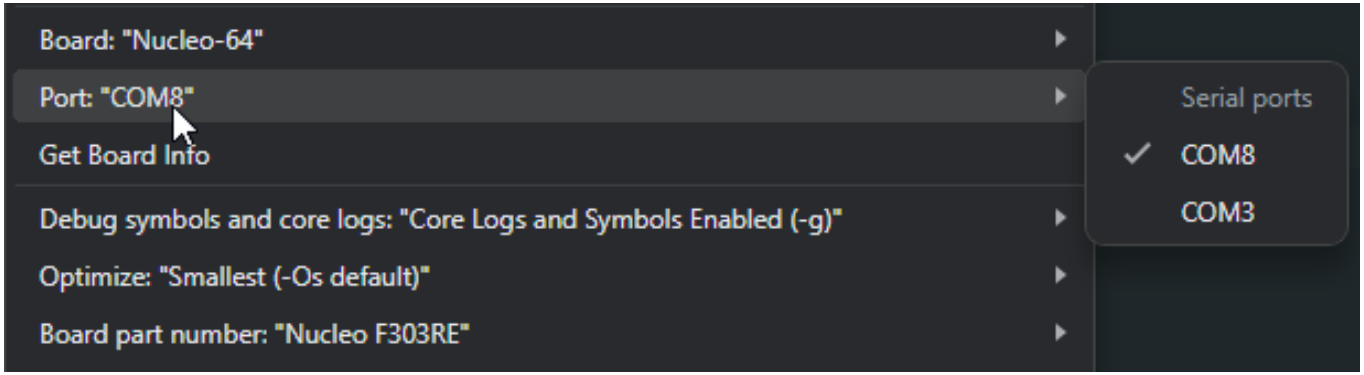
Nastavení pro Nucleo -64 („větší“)STM32G303RE)




Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
• STM32 MCU based boards
Nucleo-144
Nucleo-64
Nucleo-32
Discovery
Eval
STM32MP1 series coprocessor
Generic STM32C0 series

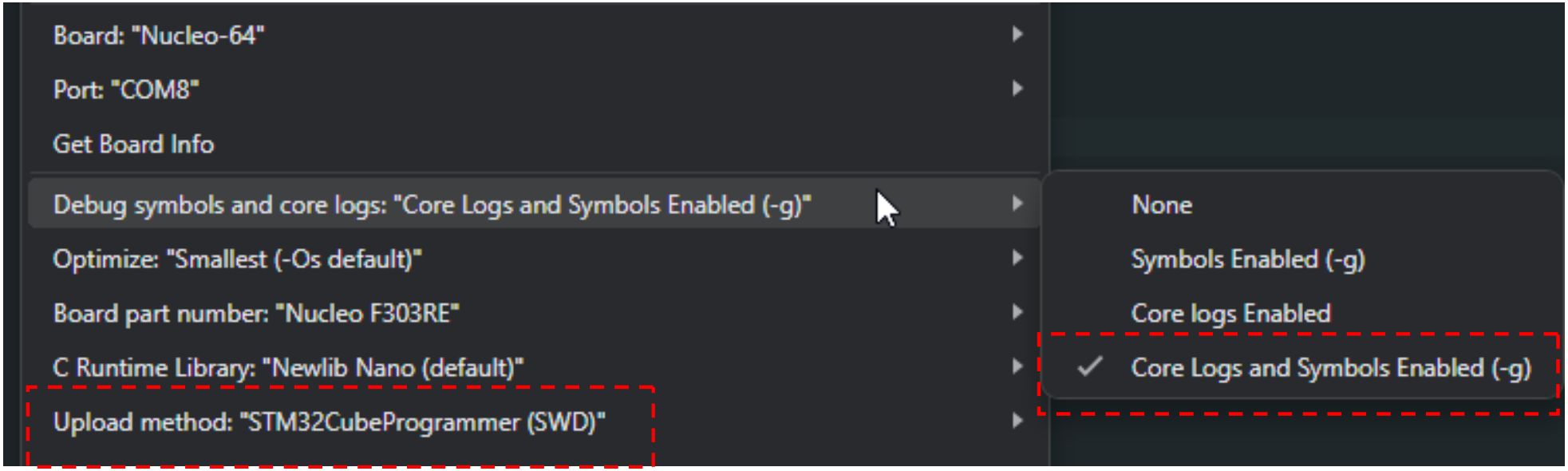

Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Nucleo-64"
Port: "COM8"
Get Board Info
Debug symbols and core logs: "Core Logs and Symbols Enabled (-g)"
Optimize: "Smallest (-Os default)"
Board part number: "Nucleo F303RE"
C Runtime Library: "Newlib Nano (default)"
Upload method: "STM32CubeProgrammer (SWD)"
USB support (if available): "None"
U(S)ART support: "Enabled (generic 'Serial')"
USB speed (if available): "Low/Full Speed"
Burn Bootloader
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
• STM32 MCU based boards
Nucleo-144
Nucleo-64
Nucleo-32
Discovery
Eval
STM32MP1 series coprocessor
Generic STM32C0 series
Generic STM32F0 series
Generic STM32F1 series
Generic STM32F2 series
Generic STM32F3 series
Generic STM32F4 series
Generic STM32F7 series
Generic STM32G0 series
Generic STM32G4 series
Generic STM32H5 series
Generic STM32H7 Series
Generic STM32L0 series

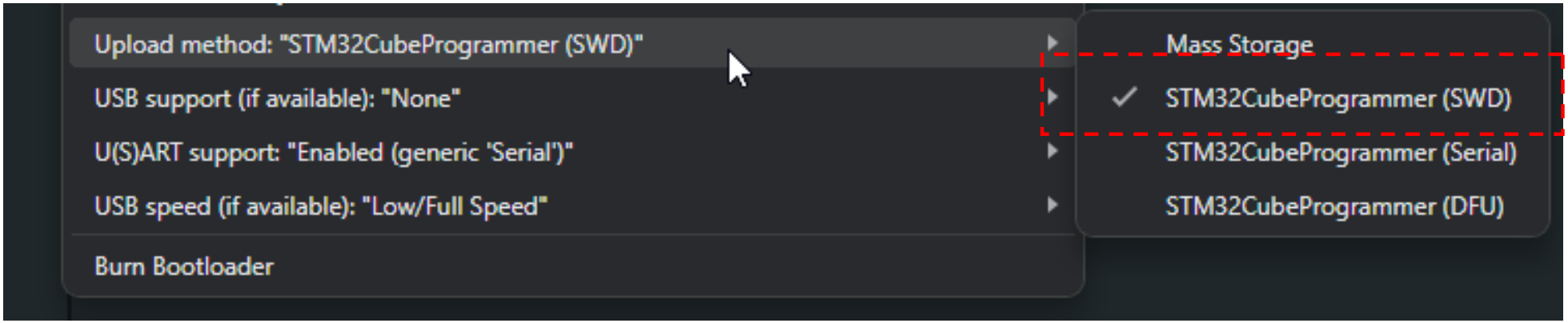

Port: "COM8"
Get Board Info
Debug symbols and core logs: "Core Logs and Symbols Enabled (-g)"
Optimize: "Smallest (-Os default)"
Board part number: "Nucleo F303RE"
C Runtime Library: "Newlib Nano (default)"
Upload method: "STM32CubeProgrammer (SWD)"
USB support (if available): "None"
U(S)ART support: "Enabled (generic 'Serial')"
USB speed (if available): "Low/Full Speed"
Burn Bootloader
Nucleo C031C6
Nucleo F030R8
Nucleo F070RB
Nucleo F072RB
Nucleo F091RC
Nucleo F103RB
Nucleo F302R8
Nucleo F303RE
Nucleo F401RE
Nucleo F411RE
Nucleo F446RE


Board: "Nucleo-64"
Port: "COM8"
Get Board Info
Debug symbols and core logs: "Core Logs and Symbols Enabled (-g)"
Optimize: "Smallest (-Os default)"
Board part number: "Nucleo F303RE"
Serial ports
COM8
COM3

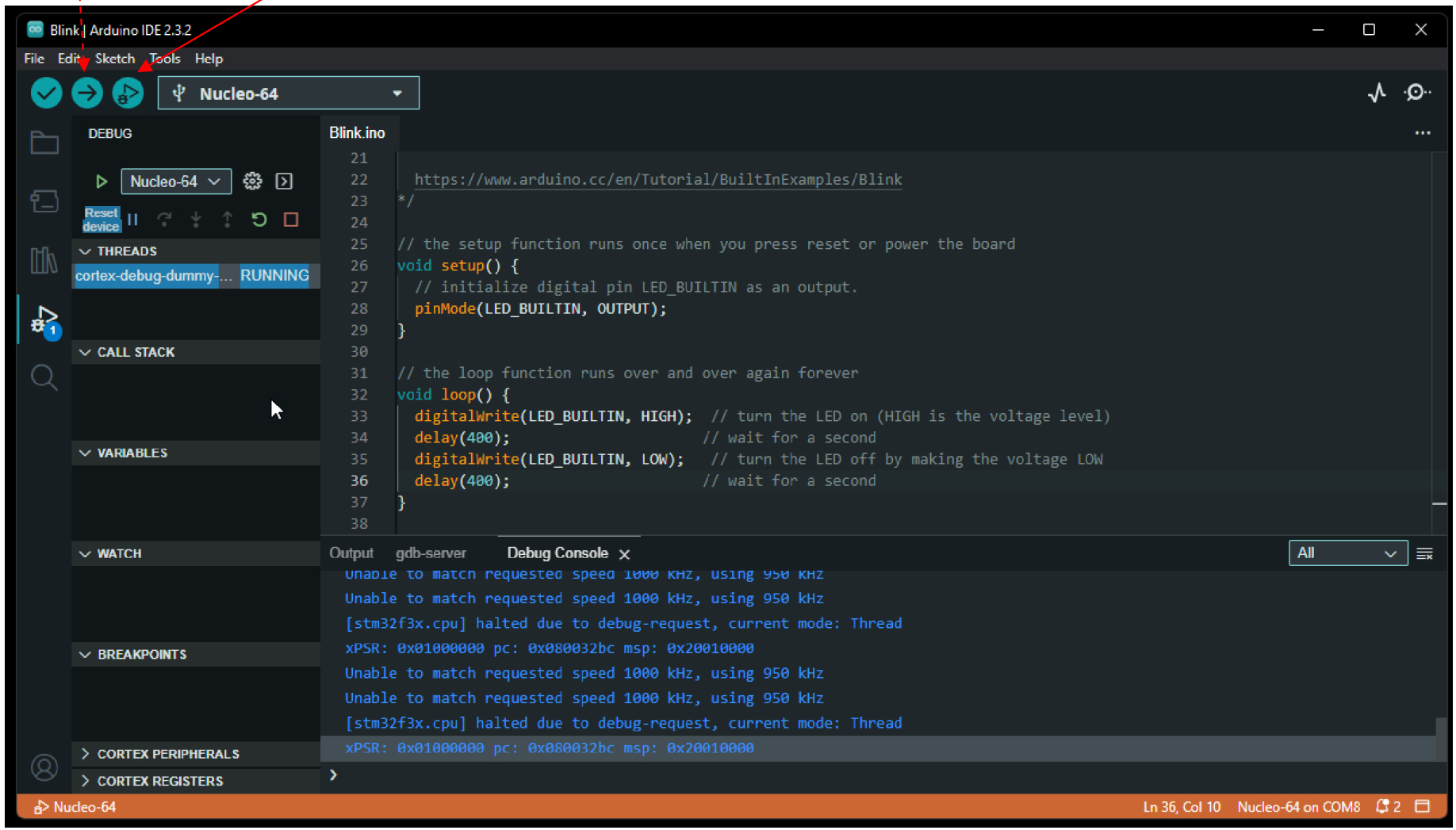
Nastavení pro debug

.



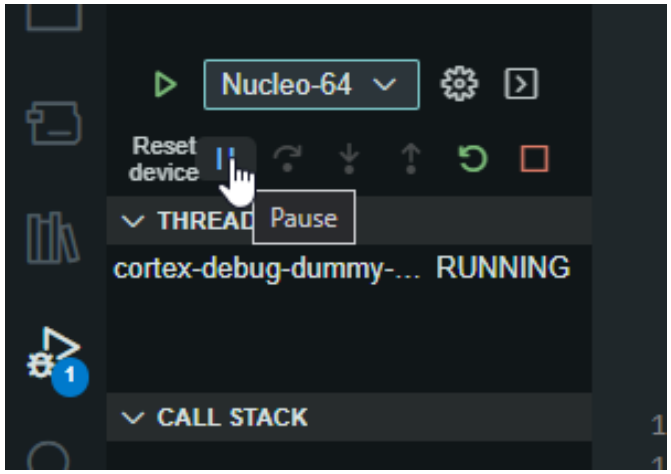


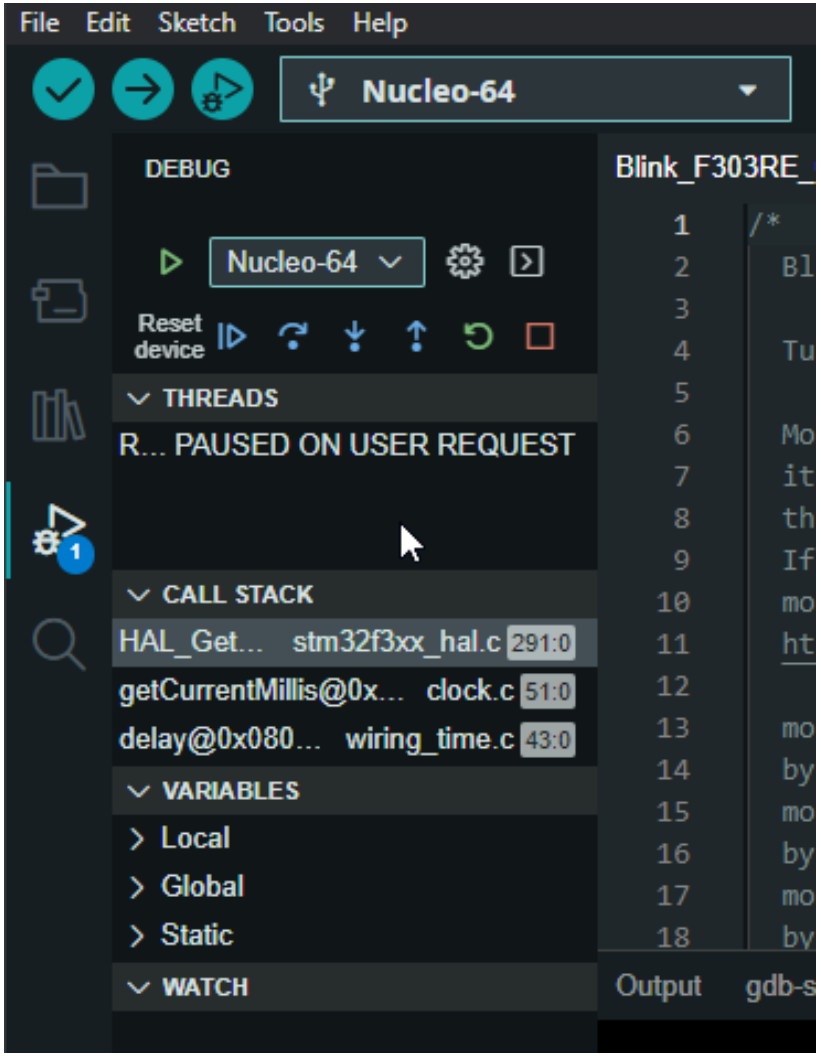
Přeložit a následně Debug



.

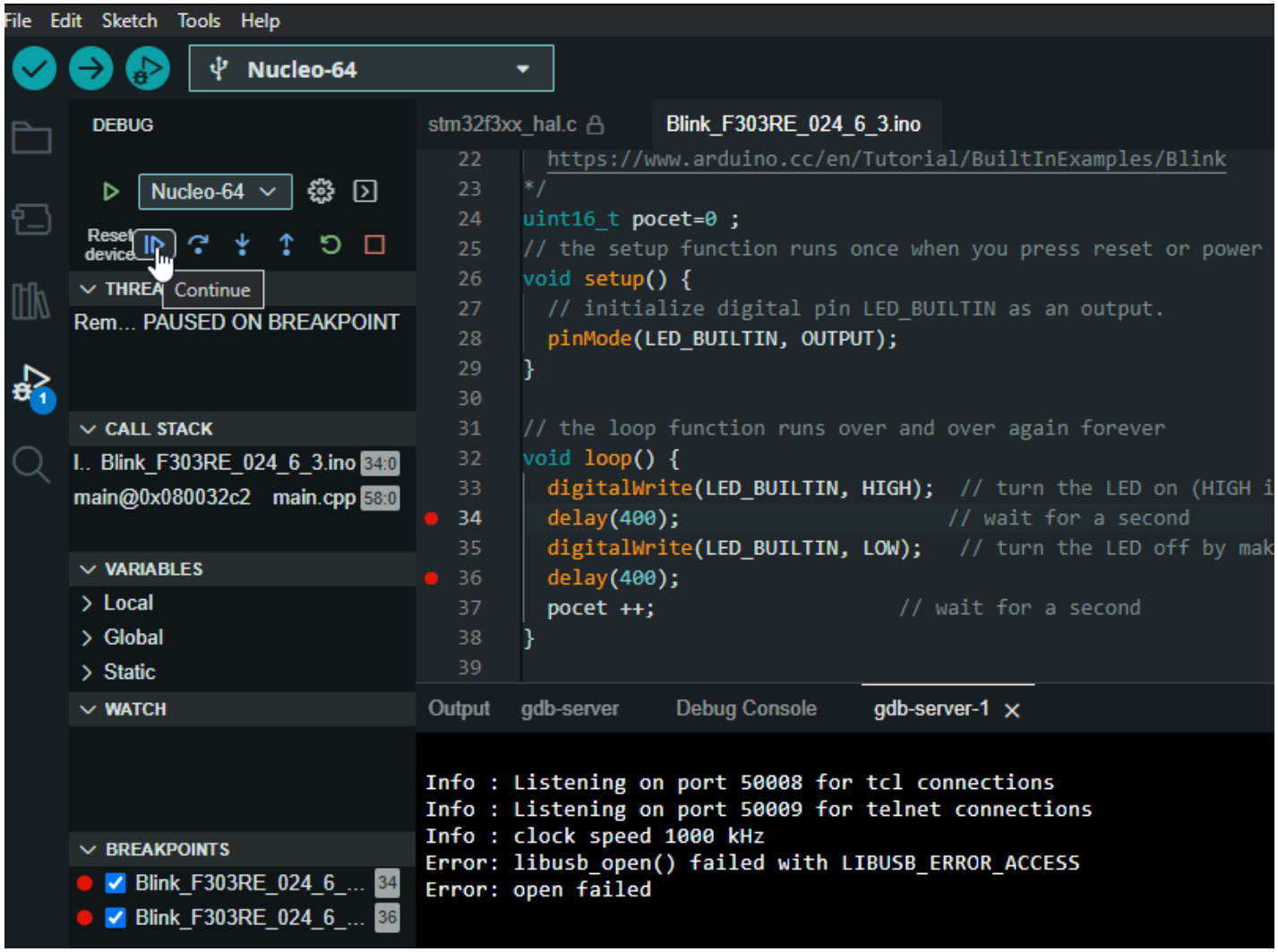
.klik na trojúhelník a spuštění



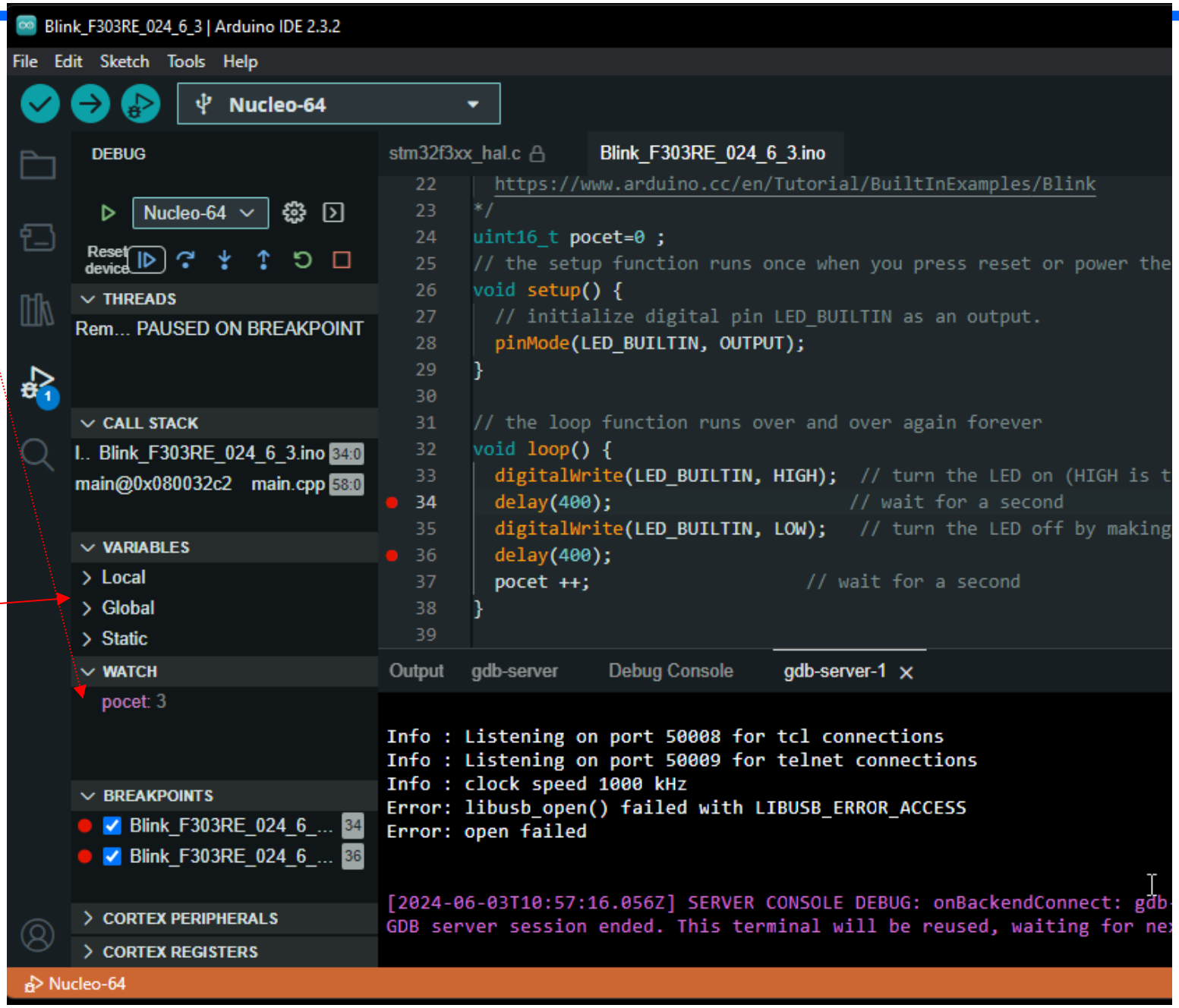


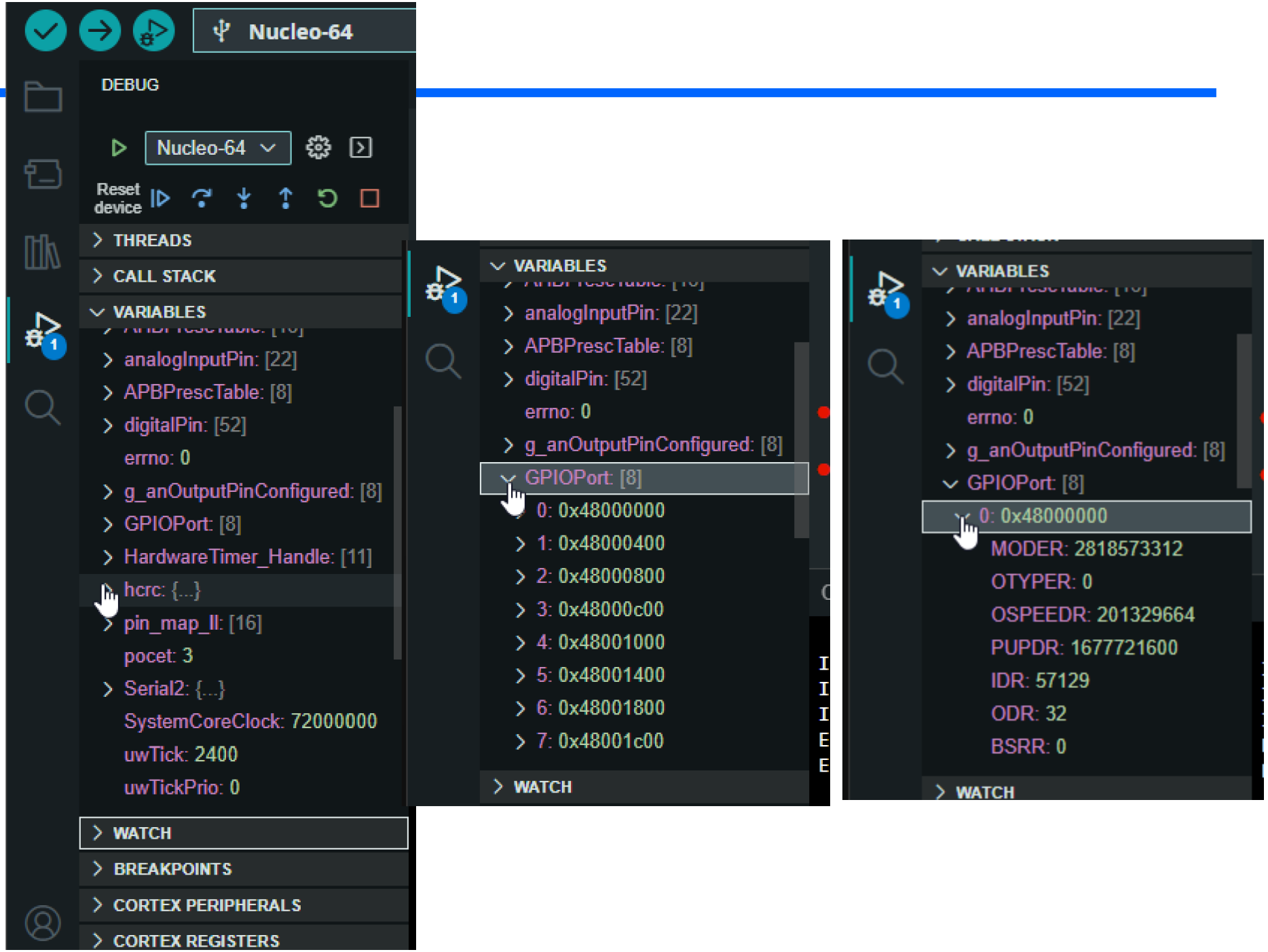
Zadání breakpointů

Klik vlevo od čísla řádku- zadání breakpointu



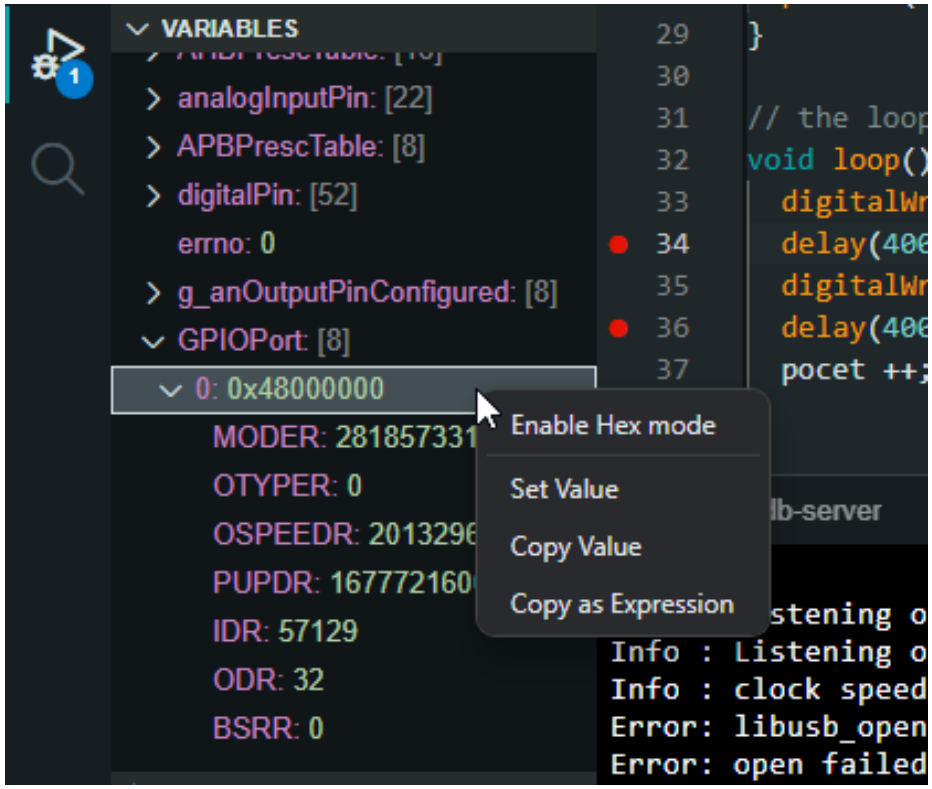
Zobrazení proměnné

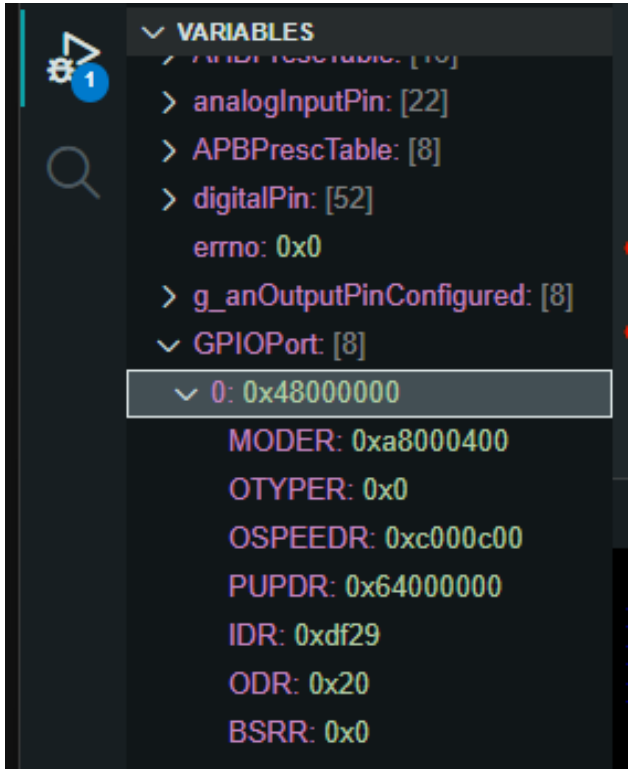



Nucleo-64
DEBUG
Nucleo-64
Reset device
THREADS
CALL STACK
VARIABLES
analogInputPin: [22]
APBPrescTable: [8]
digitalPin: [52]
errno: 0
g_anOutputPinConfigured: [8]
GPIOPort: [8]
HardwareTimer_Handle: [11]
hcrc: {...}
pin_map_II: [16]
pocet: 3
Serial2: {...}
SystemCoreClock: 72000000
uwTick: 2400
uwTickPrio: 0
WATCH
BREAKPOINTS
CORTEX PERIPHERALS
CORTEX REGISTERS
VARIABLES
analogInputPin: [22]
APBPrescTable: [8]
digitalPin: [52]
errno: 0
g_anOutputPinConfigured: [8]
GPIOPort: [8]
0: 0x48000000
1: 0x48000400
2: 0x48000800
3: 0x48000c00
4: 0x48001000
5: 0x48001400
6: 0x48001800
7: 0x48001c00
WATCH
VARIABLES
analogInputPin: [22]
APBPrescTable: [8]
digitalPin: [52]
errno: 0
g_anOutputPinConfigured: [8]
GPIOPort: [8]
0: 0x48000000
MODER: 2818573312
OTYPER: 0
OSPEEDR: 201329664
PUPDR: 1677721600
IDR: 57129
ODR: 32
BSRR: 0
WATCH

.

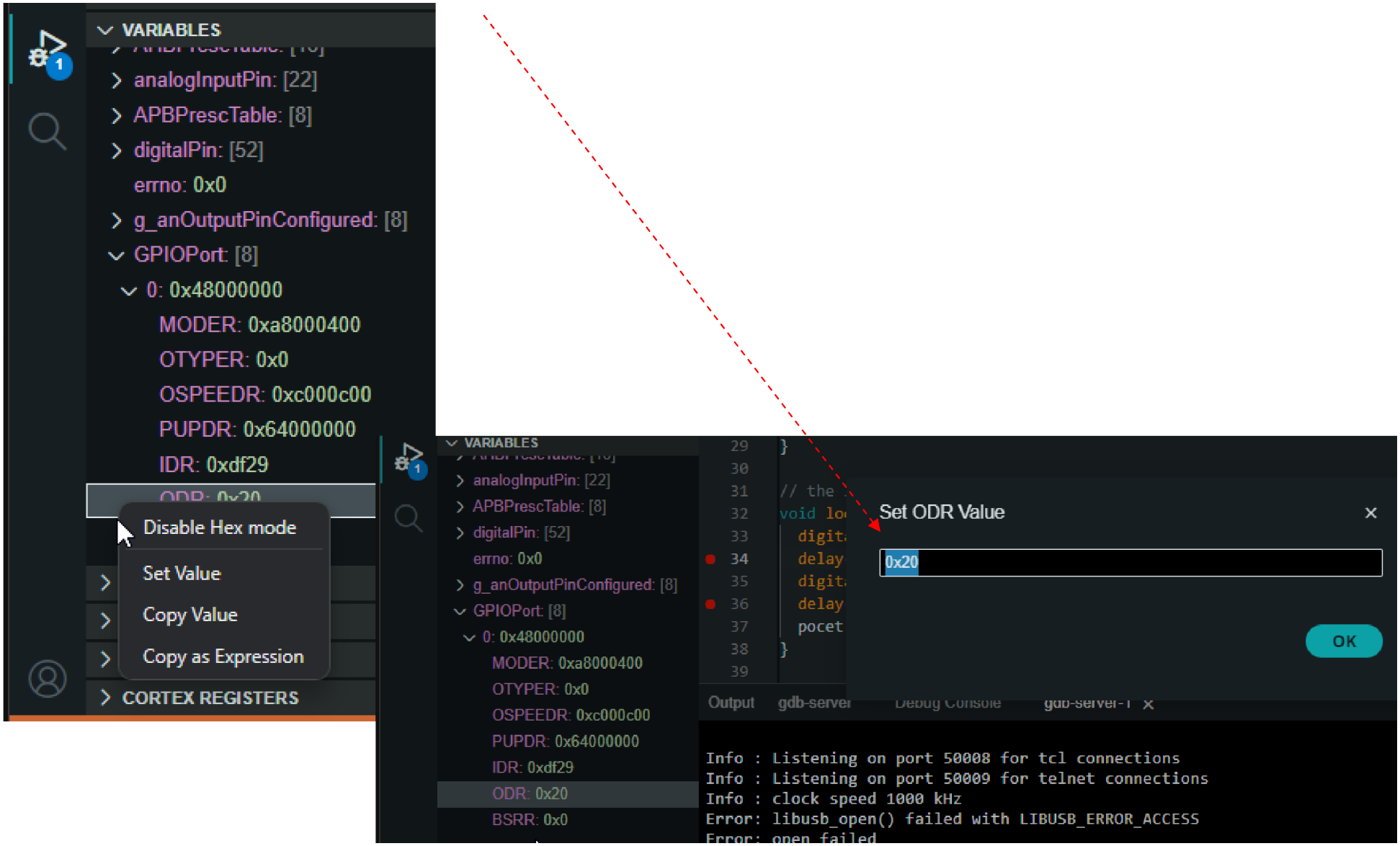
Vnitřní LED je na PA5 GPIOA_ODR 5





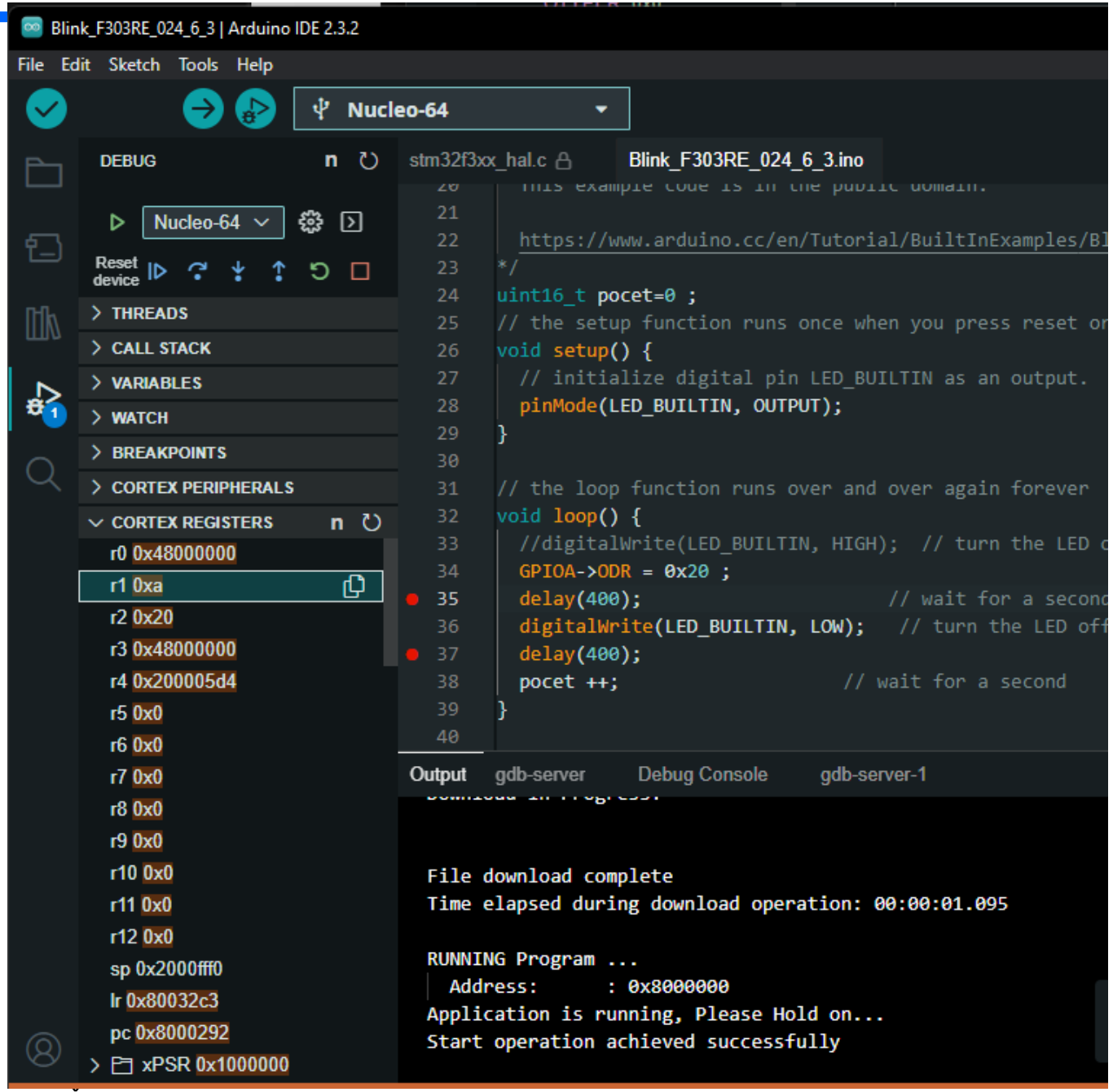
.

. pravá myš a „set value“



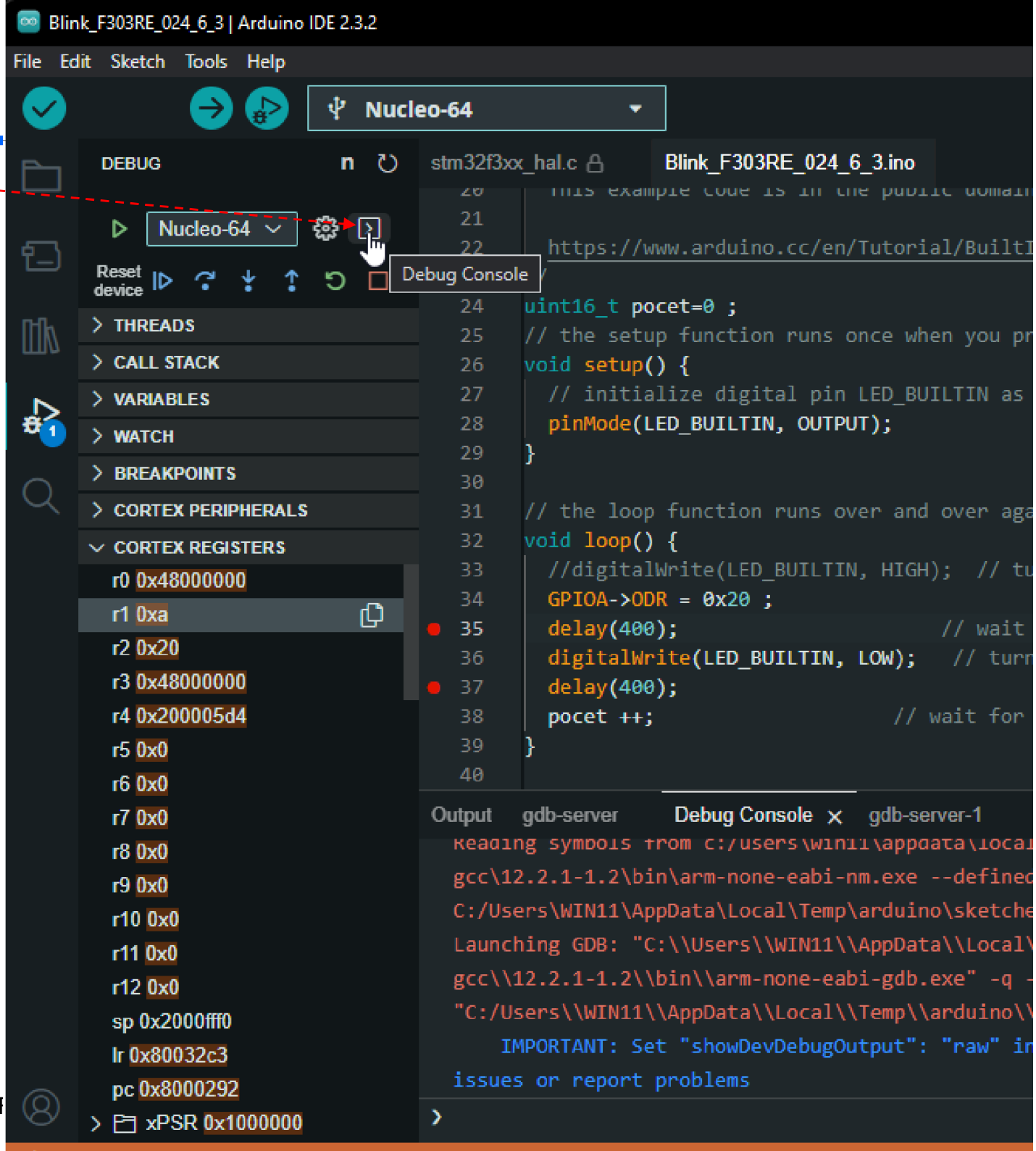
Registry procesoru

.



.

.Drebug konsole



Některé příkazy GDB

Zorazení nebo modifikace paměti nebo registrů v adr . prostoru
monitor mdw 0x20000400 0x80
monitor mdw 0x20000400 0x80
vypíše z paměti 0x80) tedy 128 wordů dat funguje
monitor mdw 0x20000400 0x80
monitor mdh 0x20000400 0x8 vypíše half wordy 16 bitů,
monitor mdb phys 0x20000400 0x8 vypíše Byty

onitor mwh 0x48000014 0x20
monitor mwh 0x48000014 0x0

.

.Konec