

```

*****
*      Tahak pro Arduino s Atmega 328/168  (UNO R3, Mini, Nano)      *
*      doplněk k referenční příručce      *
*      Nostalcomp 2024, verze 1.0 (pracovní verze)      *
*      *
*****

```

1) Dedikované piny (standardní využití, číslování dle Arduino):

PIN Funkce

=====

```

0  standard serial RX
1  standard serial TX

2  INT0 externí vstup prerušeni attachInterrupt()
3  INT1 externí vstup prerušeni attachInterrupt()

4  T0 externí hodiny pro Timer 0 (counter/timer mode)
5  T1 externí hodiny pro Timer 1 (counter/timer mode)
8  ICP1 vstup pro záchytný režim časovače Input Capture Pin

6  OC0A výstup compare timeru 0 ? (viz. datasheet ATmega)
9  OC1A výstup compare timeru 1 ? (viz. datasheet ATmega)
3  OC2B výstup compare timeru 2 ? (viz. datasheet ATmega)
4  XCK  ? (viz. datasheet ATmega)
8  CLK  ? (viz. datasheet ATmega)

6  AIN0 vstup 0 analogového komparátoru
7  AIN1 vstup 1 analogového komparátoru

10 SPI SS
11 SPI MOSI (tež ICSP)
12 SPI MISO (tež ICSP)
13 SPI SCK (tež ICSP)

3  PWM výstup AnalogWrite() - 490 Hz, Timer 2
5  PWM výstup AnalogWrite() - 980 Hz, Timer 0
6  PWM výstup AnalogWrite() - 980 Hz, Timer 0
9  PWM výstup AnalogWrite() - 490 Hz, Timer 1
10 PWM výstup AnalogWrite() - 490 Hz, Timer 1
11 PWM výstup AnalogWrite() - 490 Hz, Timer 2

13 LED 13 tradiční LED na Arduino. Svítí v HIGH

A4  I2C SCL
A5  I2C SDA

A6  pouze analogový vstup AnalogRead(), nema interní pullup!
A7  pouze analogový vstup AnalogRead(), nema interní pullup!

RESET      reset Arduina, aktivní v LOW
VIN, RAW   vstup napájecího nestab. napětí 7-12V Ideálně max. 9V!
5V         výstup stabilizovaného napětí 5V
3,3V       výstup stabilizovaného napětí 3,3V
GND        společná zem
AREF       vstup referenčního napětí pro AnalogRead, viz. analogReference()
IOREF      výstup 5V referenčního napětí pro některé shieldy
            - je spojen s 5V výstupem? Již se nepoužívá?

```

<https://deepbluembedded.com/arduino-uno-pinout/>
<https://www.engineersgarage.com/arduino-analog-comparator/>

2) Využití timerů 0/1/2 Arduino UNO/NANO/MINI:

```

TIMER0 (8b):  delay, millis, micros      PWM (AnalogWrite) pin 5,6      980 Hz
TIMER1 (16b):  servo                      PWM (AnalogWrite) pin 9,10     490 Hz

```

TIMER2 (8b): tone, IRremote PWM (AnalogWrite) pin 3,11 490 Hz

delaymicroseconds() - nevyužívá žádný timer!

shiftout, pulsein -- využívají nějaký Timer (0)?

A co I2C, LCD, Softwareserial, EEPROM ... ?

Piny použitelné pro serva se std. knihovnou - všechny?

- nelze pak použít PWM na pinech 9 a 10 (kolize Timeru 1)!

Funkce Millis, Delay, Micros, DelayMicroseconds při používání jiných přerušení:

Millis() - závisí na přerušení od Timeru 0.

V průběhu obsluhy jiného přerušení (ISR) nebude inkrementováno!

Delay() - závisí na přerušení od Timeru 0.

Nelze použít v jiné ISR, nebude fungovat!

Micros() - při využití jiných ISR zpočátku pracuje správně, ale po 1-2 mS se může začít chovat "zmateně"!

DelayMicroseconds() - nevyužívá žádný timer, pracuje vždy normálně.

3) Všeobecné užitečné informace:

Max. proud pinem: 40 mA

Max. proud všemi piny: 200 mA

Urovne digitalnich VSTUPU (TTL kompatibilni):

0 - 0,8V = LOW

2 - 5V = HIGH

Urovne digitalnich VYSTUPU (5V CMOS kompatibilni):

0V = LOW

5V = HIGH

pro Z (treti stav) nastavit pin jako INPUT! (například u charlieplexingu)

PWM hodnoty pro AnalogWrite:

hodnota 0 = log. 0 trvale (duty 0%)

hodnota 255 = log. 1 trvale (duty 100%)

A6 a A7 (u NANO):

pouze analogove vstupy!!

Není zde pullup. Ani po doplnění pullupu (10k) nelze využít jako digital input!

Funkce DigitalRead(A6/A7) vždy vrátí 0 (odzkoušeno)

- bez ohledu na nastavení pinu INPUT / INPUT_PULLUP

- bez ohledu na uroven na pinu (GND, +5V)

Je možno číst pouze pomocí AnalogRead:

Například čtení tlačítka s pullupem:

Stisknuto: analogread(A6/A7) < 101

Uvolněno: analogread(A6/A7) > 100

ATMEGA 168 versus 328 (např. v PRO MINI)

	Flash	SRAM	EEPROM	Osc	
168:	16KB	1KB	0,5KB	20MHz	(168V pouze 10 MHz)
328:	32KB	2KB	1KB	20MHz	

https://www.futurlec.com/ICAtmel_ATMega_Comparison.shtml

Nejčastěji používané 5V stabilizátory na UNO/MINI/NANO:

- může se měnit dle výrobce daného klonu!

Pro MINI: typicky stabilizátor MIC5205

- Uin max 16V
- Out 5V / 150mA
- low drop (17mV naprázdno, 165 mV při 150mA)

UNO/NANO: typicky stabilizátor LM1117

- Uin max 15V
- Out 5V / 800mA
- drop (1,1V/100mA , 1,2V/500mA , 1,3V/800mA)

Některé starší verze NANO (v2) využívaly též 78M05!

(Uin max 35V, Out 5V/500mA, drop 2V)

Maximální (trvalé) vstupní napětí pro LM1117 je 15V, pro MIC5205 je to 16V.

Bácha na nestabilizované adaptéry (hlavně 12V, ale někdy i 9V), které tato napětí při malém odběru překračují a stabilizátory spolehlivě odpálí!

U nestab. adaptérů doporučuji používat předstabilizátor 9V (např. s 7809).

Bitové posuvy >> a << (nejedná se o rotace!):

- nemění původní hodnotu proměnné (pokud ji znovu nepřiradíme)
- nejsou cyklické! Vysunuté bity jsou ztraceny, z druhé strany se doplní 0(*)!

```
x>>1;      hodnota x se nemění, pouze hodnota výrazu
x = x>>1;   hodnota x se změní
x>>0;      posuv se neprovádí, hodnota výrazu i proměnné zůstává x
```

(*) - POZOR! Neplatí u posuvu vpravo (>>), pokud je proměnná záporná a nejvyšší bit = 1 (znaménko -). Pak se zleva doplňují jedničky !!!

příklad:

```
int cislo1 = 128;
int cislo2 = -128 ;
Serial.print (cislo1, BIN);
Serial.print ("\t");
Serial.println (cislo2, BIN);
Serial.print (cislo1>>2, BIN);           //doplní 0 (kladné číslo)
Serial.print ("\t");
Serial.println (cislo2>>2, BIN);          //doplní 0 (záporné číslo)
Serial.print (cislo1<<2, BIN);           //doplní 0 (vždy)
Serial.print ("\t");
Serial.println (cislo2<<2, BIN);          //doplní 0 (vždy)
```

Logické operatory NOT, OR, AND, XOR:

***** NOT *****

```
~          logická negace všech bitů hodnoty (~128) = -129
           Bácha na typ proměnné!          byte(~128) = 127
           - pro výpočty
! = not     pouze v podmínkách          (!128) = (not 128) = 0
           = hodnota výrazu              (!0)  = (not 0)  = 1
NOT         nelze!
```

***** OR *****

```
|          logický součet hodnot (12 | 3) = 15
           - pro výpočty
|| = or     pouze v podmínkách          (12 || 3) = (12 or 3) = 1
           = hodnota výrazu             (12 || 0) = (12 or 0) = 1
```

```

                                (0 || 0) = (0 or 0) = 0
OR                                nelze!

***** AND *****
&                                logicky soucin hodnot (15 & 3) = 3
                                - pro vypocty
&& = and                         pouze v podminkach (15 && 3) = (15 and 3) = 1
                                = hodnota vyrazu (15 && 0) = (15 and 0) = 0!
AND                                nelze!

***** XOR *****
^ = xor                         logicky XOR hodnot (0x55 ^ 255) = (0x55 xor 255) = 170 (0AAh)
                                - pro vypocty
Verze pro podminky neni.
XOR                                nelze!
-----

```

Zapis binarnich, octalovych a hexadecimalnich cisel:

```

10  = 10          dekadicky
B10 = 2           binarne (b10 nelze!)
010 = 8           osmickove (prefix je NULA)
0x10 = 0x010 = 16 hexadecimalne
0x1f = 0x1F       nezalezi na velikosti pismen
-----

```

Funkce, které vraceji nejakou hodnotu (nejsou void) neni nutne prirazovat:

```

byte x,y,z,q;
x=10;
y=20;
pricti();
Serial.println (z);          //30
x--;
q=pricti();
Serial.println (q);          //29

int pricti()
{
z = x+y;
return (x+y);
}
-----

```

Poznamky k Serial.print(ln) a Serial.write:

Serial.print(ln) i Serial.write vraci pocet vypsanych znaku (bajtu).

```

Ale bacha: int poc = Serial.print("AH0J")    //poc == 4
            int poc = Serial.println("AH0J")   //poc == 6, protoze CR,LF!
            int poc = Serial.write("AH0J")      //poc == 4

```

Serial.print (x, y) x = cislo, y = pocet vypsanych desetinnnych mist

Serial.print (x, HEX) x = CELE cislo vypsane v soustavach DEC,BIN,OCT,HEX

Serial.print ("\t") TABulator = 8 mezer

Serial.print ("\t\t") 2x TABulator = 16 mezer

Serial.print ("AH0J") vypise retezec znaku AH0J

Serial.print ("A") = Serial.print ('A') vypise znak A

Serial.print ('AH0J') vypise hodnotu 20298

Serial.println('A'+ 'H'+ '0'+ 'J') vypise hodnotu 290 = int('A'+ 'H'+ '0'+ 'J')

Serial.println(int('A'), HEX) vypise hodnotu 41 (041h)

Serial.write - odesila "pure" bajty, napriklad odradkovani terminalu:

```

Serial.write (0xD)            - CR
Serial.write (0xA)            - LF

```

nebo
(const) byte buf[] = {0xD, 0xA} - definujeme pole bajtu
Serial.write (buf,2) - CR+LF. 2 je pocet poslaných bajtu z pole (povinne)

POZOR!

Sdruzeny vypis typu Serial.print (x," - ",y) nefunguje! (ani s teckou)

sizeof(pole) ==> vraci pocet bajtu pole! Nikoliv pocet polozek pole!

```
byte pole1[] = {0,1,2,3,4,5,6,7,8,9}; //10 polozek, 10 bajtu  
word pole2[] = {0,1,2,3,4,5,6,7,8,9}; //10 polozek, 20 bajtu!
```

```
Serial.begin(19200);
```

```
Serial.print("Pole byte = ");  
Serial.println (sizeof(pole1)); //vypise 10 (totez pro pole CHAR - retezec)
```

```
Serial.print("Pole word = ");  
Serial.println (sizeof(pole2)); //vypise 20 (totez pro pole INT)
```

Arduino potouchlost - funkce map()

<https://robodoupe.cz/2020/arduino-potouchlost-funkce-map/>

aneb neni nad to, prepocitat si to vzdy hezky rucanko...

Arduino potouchlost - serva:

<https://robodoupe.cz/2018/arduino-potouchlost-serva/>

Arduino a vice serv:

<https://robodoupe.cz/2016/arduino-a-vice-serv/>

Konstanty #define versus const:

```
#define tl_pocet      6  
#define tl_maxain    1023  
#define tl_necit     10
```

```
//DOBRE, ma-li byt konstanta stanovena vypoctem, nedavat #define!!
```

```
const int tl_prah = (tl_maxain / tl_pocet);  
const int tl_hyst = ((tl_prah / 2) - tl_necit);
```

```
//SPATNE
```

```
#define tl_prah (tl_maxain / tl_pocet) //toto sice jeste funguje...  
#define tl_hyst ((tl_prah / 2) - tl_necit) //ale toto uz NE!!!!  
// nelze pouzit drive vypoctenou konstantu tl_prah !!
```

Podmineny preklad:

```
#define klidova      29 //paka schovana v klidove poloze  
#define pracovni    131 //paka prave prepina packu spinace
```

```
#if klidova < pracovni //nastaveni kroku serva dle smeru otaceni  
#define krok 1 //krok serva (doporucoji ponechat na 1)  
#else  
#define krok -1 //jen pokud se paka ma tocit opacne  
#endif
```

```
#if krok > 0
  while (poloha > zcil);
#else
  while (poloha < zcil);
#endif
```

```
*****
                        Chovani overovano v IDE 1.0.5-r2
*****
```

EOT (End Of Tahak) :-))