

Počítačové systémy

Adam Šmelko

smelko@ksi.mff.cuni.cz

NSWI170 - cvičení

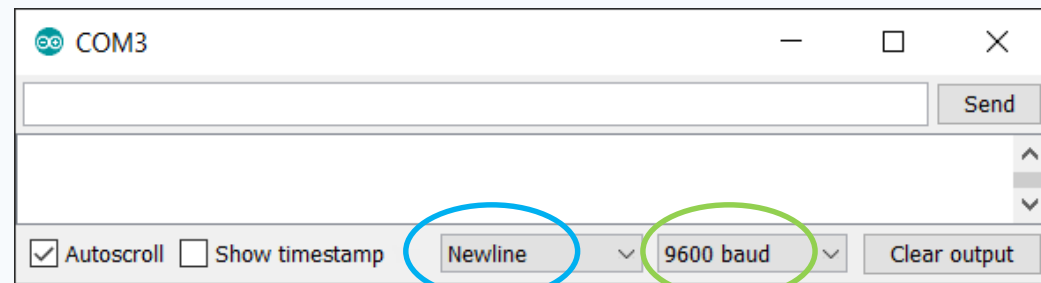
LS 2020/2021

Arduino

Sériová linka

Sériová linka - výstup

- komunikace s vnějším prostředím
- Arduino IDE
 - Tools / Serial monitor
 - nastavit **přesně** stejnou rychlost jako v setupu
 - 9600 baud
- setup
 - `Serial.begin( speed);`
- výstup
 - program v Arduino pošle text PC
 - PC to zobrazí v serial monitoru
 - `Serial.print(); .println();`
 - různé typy parametrů
 - int, string, float,
 - overloading / přetížení
- ➡ něco si z Arduina pošlete na serial monitor
- ❤ jednoduchý časovač
 - každou vteřinu vypsát uplynulý čas



```
Serial.begin( 9600);
```

```
Serial.print( "*");  
Serial.print( 1234);  
Serial.println( "whatever");
```

Debugging bez debuggeru

- užitečné např. pro ladění
 - možná se vám už někdy stalo, že program nedělá to, co by měl ☹️
 - ... a zbývá půl hodiny do termínu
- ladicí tisky
 - kudy běží program
 - hodnoty proměnných
 - je vhodné mít globální vypínač ladicích tisků
 - kdykoliv lze všechny ladicí tisky najednou vypnout
 - rozšíření: errorlevel

```
class Debug {  
public:  
    Debug( bool en=true) : enabled_( en) {}  
    void out( const char* str, int n) {  
        if( enabled_) {  
            Serial.print( "> ");  
            Serial.print( str);  
            Serial.println( n);  
        }  
    }  
private:  
    bool enabled_;  
};
```

Recodex
nepodporuje

```
auto current_time = millis();  
bool cond = button_pressed && current_time <= last_pressed+timeout;  
if( cond) {  
    whatever();  
}
```

problém
whatever se nevolá
proč?

```
Debug d;  
auto current_time = millis();  
bool cond = button_pressed && current_time <= last_pressed+timeout;  
d.out( "button_pressed: ", button_pressed);  
d.out( "last_pressed: ", last_pressed);  
d.out( "current_time: ", current_time);  
if( cond) {  
    d.out( "jsem vevnitř :-", 0);  
    whatever();  
}
```

HA! čas je větší než
last_pressed!
špatně napsaná podmínka

Sériová linka - vstup

- ♥ vstup

- PC (serial monitor) → Arduino
- Serial.available();
- Serial.read();
- Serial.readString();

vhodné
testovat

- vypište načtená data
- co se přesně načte
 - rozdíl read a readString
- zakončení řádky/vstupu
 - serial monitor: line ending
- užitečné např. pro zadávání paramterů a dat
 - konzolové ovládání programu

```
[co_mi_prislo][co_mi_prislo]....
```

```
auto x = Serial.readxxx();
```

- Arduino.cc / Resources / Reference / Serial

- www.arduino.cc/reference/en/language/functions/communication/serial
- ♥ mnoho dalších funkcí

- ♥ asynchronní čtení

- serialEvent()
- není nutné explicitně volat

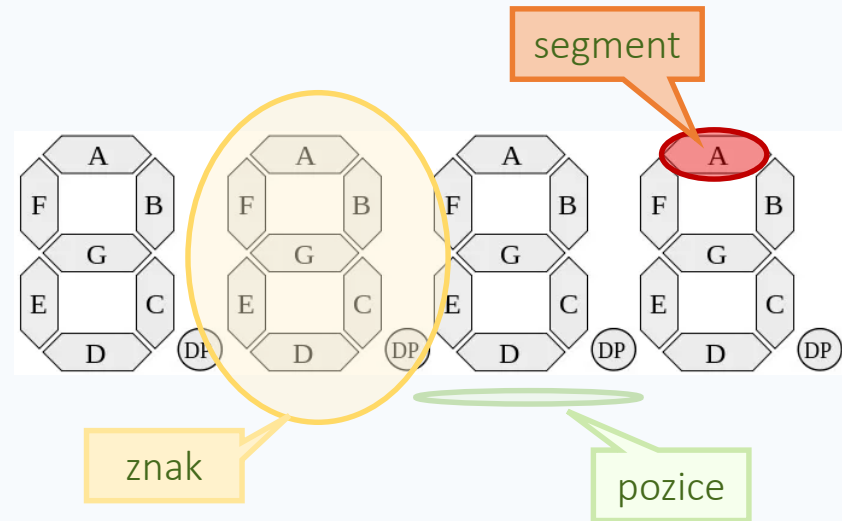
```
void serialEvent() {  
    auto s = Serial.readString();  
}  
  
void loop() { // nic :-)  
}
```

Arduino

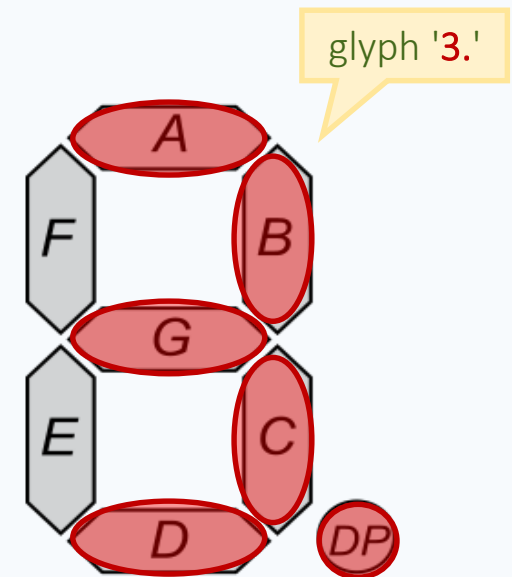
Segmentový displej

Segmentový displej

- 4-místný
- individuální ovládání segmentů
 - v rámci jednoho znaku
- glyph
 - svítící 'obrázek' v jednom znaku
 - množina rozsvícených segmentů
 - kódováno bity v jednom bajtu
- inverzní logika
 - bit 1 ≈ nesvítí, bit 0 ≈ svítí
 - '3.' ≈ 0b00110000 = 0x30 = 48



hodnota bitu	0	0	1	1	0	0	0	0
index bitu	7	6	5	4	3	2	1	0
(hex) hodnota	8	4	2	1	8	4	2	1
segment	DP	G	F	E	D	C	B	A



Segmentový displej - posuvný registr

- zápis na displej prostřednictvím posuvného registru

- shift register

- používá 3 piny

- latch, clock, data

- v setupu inicializovat

- pinMode(l/c/d _pin, OUTPUT)

- zápis

- zobrazení glyphu na pozici

- zavřít latch

- vsunout bity glyphu

- vsunout bitovou masku pozice

- lze i víc bitů

- na odpovídajících pozicích stejný glyph

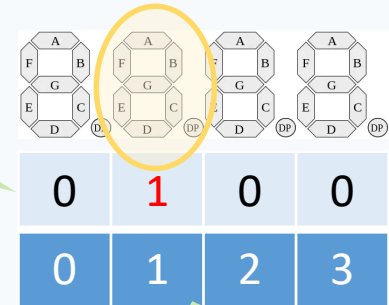
- otevřít latch

- **shiftOut**(data_pin, clock_pin, bitOrder, value)

↔ funshield.h

latch_pin
clock_pin
data_pin

kódování pozic
zleva



$1 \ll n$

↔ my lib

zavřít latch ≈ start zápisu

poslat data glyphu

poslat bitmasku pozice

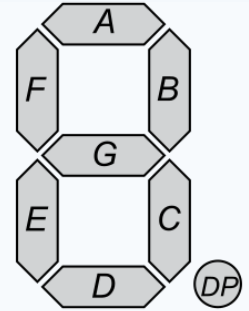
otevřít latch ≈ konec zápisu

```
writeGlyphBitmask( byte glyph, byte pos_bitmask) {  
    digitalWrite( latch_pin, LOW);  
    shiftOut( data_pin, clock_pin, MSBFIRST, glyph);  
    shiftOut( data_pin, clock_pin, MSBFIRST, pos_bitmask);  
    digitalWrite( latch_pin, HIGH);  
}
```

Segmentový displej - zobrazení

- setup
 - inicializovat piny
 - smazat displej
 - stejná hodnota (vše zhasnout) na všechny pozice
- ➞ 4.1 funkce pro zobrazení glyphu (obrázku) na konkrétní pozici

`writeGlyphBitmap( 0xFF, 0x0F);`



- `writeGlyphR( byte glyph, int pos), writeGlyphL`
- pozice 0 až 3
 - zprava - vhodné pro čísla
 - zleva - vhodné pro text

```
constexpr int glyphDigit[]  
{ 0xc0, 0xf9, 0xa4, 0xb0, 0x99,  
  0x92, 0x82, 0xf8, 0x80, 0x90  
};
```

- ➞ 4.2 funkce pro zobrazení libovolné jednociferné číslice na pozici
 - `writeDigit( int n, int pos)`
 - číslování pozic zprava
 - využijte `writeGlyphR`
 - ♥ skuste zadat číslo za pomoci `Serial.read()`
- ➞ 4.3 jednociferný čítač stisků tlačítka z 3.3
 - výstup na displej
 - jen jednociferně (modulo 10)
 - pozor na záporná čísla
 - ♥ pravé tlačítko cyklicky mění pozici zobrazení číslice

Segmentový displej - zobrazení

- ➡ 4.4  víceciferný čítač
 - vnitřní čítač 0-9999
 - zobrazena vždy pouze jedna cifra
 - b3: změna řádu
 - ovládání i zobrazení
 - b1/2: inc/decrement ve zvoleném řádu
 - $\pm 1 \rightarrow \pm 10 \rightarrow \pm 100 \rightarrow \pm 1000$



- ➡ zobrazení čísla
 - zobrazí max 4 cifry, bez úvodních 0
 - nezapomenout na číslo 0
- pozorujete něco zvláštního?
 - přidejte na konec loopu `delay(10);`
 -  zkuste vysvětlit

