Arduino – úvod, LED

NSWI170: Lab 02

Patrik Dokoupil

Credit: Martin Kruliš

Shrnutí 1. úlohy

- Časté problémy:
  - Nedostatečná dekompozice:
    - 4x `for`-loop pro výpis znaků
    - Výpis i hledání minima v jedné dlouhé funkci
  - Funkce pro minimum sahající do glob. pole
  - Zdroják nebyl kódován v UTF-8
- Důraz na code review
  - Bílé komentáře není nutno opravovat

Pravidlo č. 2

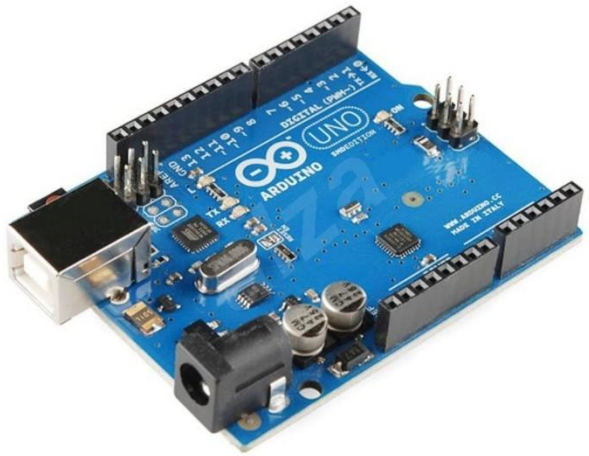
- # 2 Konstanty
  - Používejte konstanty místo číselných/textových literálů
    - Konstanty jsou pojmenované -> lepší srozumitelnost a čitelnost kódu
    - Hodnotu lze snadněji změnit (na jednom místě)
    - Víte, že se hodnota nemění
  - Všeho s mírou -> nemá smysl dělat konstanty ve smyslu
    - `constexpr int zero = 0;`
  - Konstanty lze používat i na jiných místech – např. vstupní parametry funkcí
    - `int min(const int arr[], int arr_size);`
  - `const` vs `constexpr` (hodně zjednodušeně)
    - `const` po inicializaci se hodnota nemění
    - `constexpr` hodnota kterou inicializujeme je známá během kompilace – pokud to jde, tak chcete použít spíše `constexpr`

Arduino

- Open source HW a SW projekt
- HW
  - Arduino deska - základní deska se sadou analogových a digitalních pinů
    - Existuje více typů desek, pro každou z nich existuje spousta klonů
    - My budeme používat Arduino UNO + FunShield
- SW
  - Arduino IDE

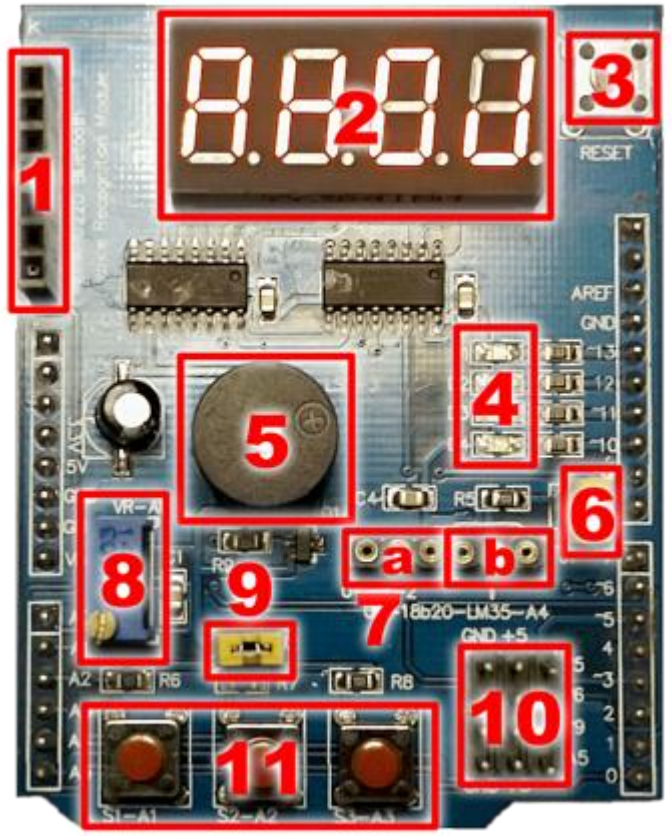
Arduino UNO

- CPU ATmega328P
- 14 digitálních I/O pinů
  - 6 z nich lze použít pro PWM
- 6 analogových vstupů
- Frekvence 16 MHz
- **FLASH memory 32 KB**
  - Kód programu
- **SRAM 2 KB**
  - Static data, heap, stack
- EEPROM 1 KB
- USB
- DC power



FunShield

- **3x tlačítka (obr. 11)**
- **4x LEDs (obr. 4)**
- **1x 4-místný display (obr. 2)**
- 1x „bzučák“ (nebudeme používat)
- **Tlačítko reset (obr. 3)**
- Detailní popis vč. schéma zde (budeme využívat jen základní funkcionalitu, tj. je to spíše pro ty, které to zajímá)



Zdroj, autor RNDr. Miroslav Panoš, Ph.D.

Arduino IDE

- Stáhnout, nainstalovat (na PC v Labu je již instalováno)
- „Sketch“ odpovídá aplikaci (nahraje se do Arduina a tam se spustí)
- „Library“ obsahují sdílenou funkcionalitu
- Nastavení:
  - Tools -> Board -> Arduino Uno
  - Tools -> Port -> COMx (Arduino Uno)
  - Sketch -> Include Library -> Add .ZIP Library -> vyberte Funshield.zip
  - Sketch -> Include Library -> vyberte Funshield
  - File -> Preferences -> Compiler Warnings = All

Arduino IDE

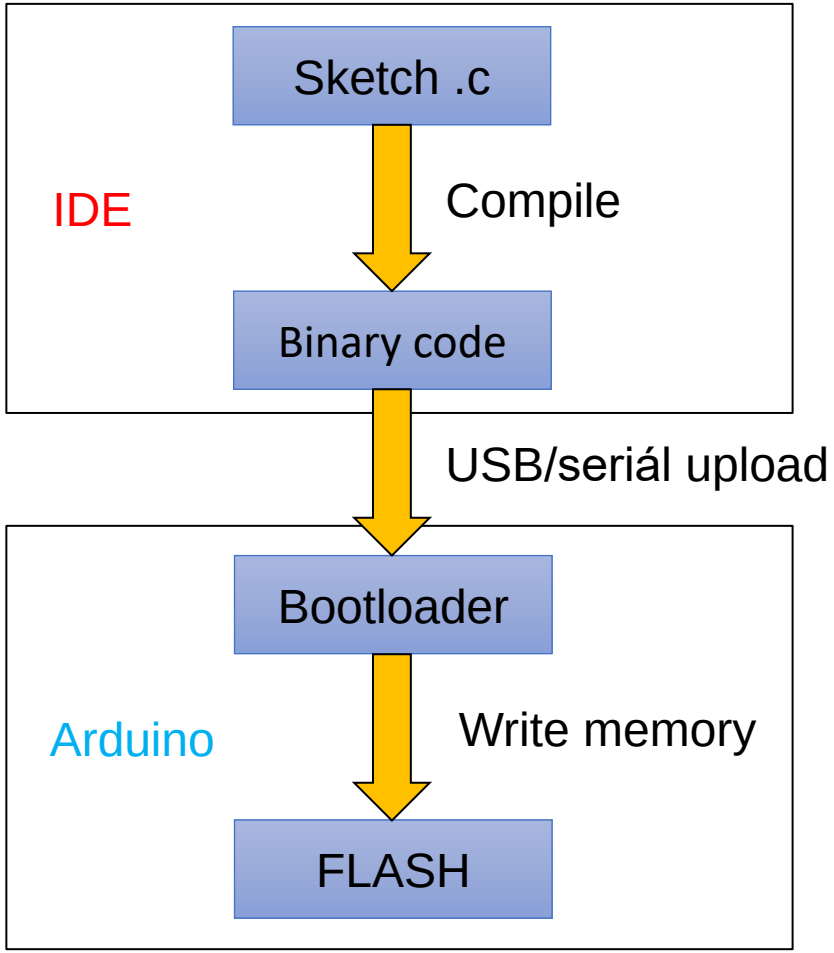
- 2 důležité funkce:
  - `void setup() { ... }`
    - Volá se pouze 1x při startu, hodí se pro inicializaci všeho možného (módy PINů apod.)
  - `void loop() { ... }`
    - Volá se opakovaně, zjednodušeně řekněme 1000x/s (ve skutečnosti závisí na frekvenci a počtu instrukcí v `loop()`)
  - `main()` nepíšete, dodá framework, velmi zjednodušeně získáte něco jako:

```
int main() {
    init();
    setup();
    for(;;) { // nekonečný cyklus
        loop();
    }
}
```

Pozn.: Rozdělení na `setup()` a `loop()` nás donutí občas použít globální proměnné (nebo statické), konkrétně tam kde bude nutné uchovat stav mezi jednotlivými voláními `loop()`

Kompilace a spuštění kódu

- Sketch se zkompiluje pro cílové CPU
- Výsledný binární kód se pomocí USB/serial nahraje na desku a ta se restartuje
- EEPROM na desce obsahuje boot loader který se po resetu desky spustí
- Pokud je tu po resetu nějaký USB/seriál „payload“ tak jej boot loader načte, uloží do FLASH paměti a spustí
- Jinak se spustí to co již bylo ve FLASH paměti



Cvičení

- T1: Rozsviťte LED
  - Nutno inicializovat LED pin (typicky v **`setup()`**)
    - **`pinMode(pin, OUTPUT/INPUT)`**
    - Používejte konstanty z **`funshield.h`**!
  - Zápisem na PIN nastavte LED
    - **`digitalWrite(pin, HIGH/LOW)`**
    - Na funshieldu z knihovny LOW odpovídá rozsvícené LED (jiné desky to mohou mít naopak) -> konstanty **`ON/OFF`** v funshield.h, nepoužívat přímo **`HIGH/LOW!`**
- T2: Rozblikejte LED
  - **`loop()`** běží příliš rychle, použijte **`delay(ms)`**

Cvičení

- T3: Rozblikejte LED bez **`delay()`**
  - **`delay()`** z T2 je aktivní (blokující) čekání což je něco, čemu se chceme obvykle vyhnout (během aktivního čekání totiž nelze dělat nic jiného)
  - Jako alternativu lze využít funkci **`millis()`** která vrací počet uplynulých ms (od začátku běhu programu). Vrací **`unsigned long`**
  - Můžeme tedy sledovat uplynulý čas a dle toho vypínat/zapínat LED
    - Uložit uplynulý čas do proměnné
    - V každé iteraci **`loop()`** se podívat kolik ms uběhlo
    - Pokud více než zadaný threshold tak proveďte požadovanou akci (zde LED ON/OFF)

Cvičení

- T4: Rozblikejte všechny LED najednou (bez **`delay()`**)
  - Použijte cyklus, nikoliv copy&paste
  - Kde je to vhodné, dekomponujte kód do funkcí (vzpomeňte si na minulé cvičení a feedback k první domácí úloze)
  - Nápověda: lze deklarovat pole s PINy, nezapomeňte inicializovat všechny LED

Domácí úloha

- Zobrazte „animovaný“ vzor
  - Rozsviťte LED sekvenčně (vzor „ping-pong“)
    - Od první k poslední LED, vždy svítí právě jedna
    - Jakmile se rozsvítí poslední, tak se „světlo“ odráží a vrací se zpět na začátek
    - Detaily (časování, která LED je první apod.) jsou v ReCodExu

Cvičení*

- T5: Podobně jako u domácí ulohy udělejte animovaný vzor, tentokrát však vzor „had“
  - Postupně rozsviťte LED jednu za druhou (svítí vždy právě 1)
  - Jakmile rozsvítíte poslední, tak se pokračuje opět první LED
- Extension 1: Zobrazte delšího hada
  - Ukázka pro délku hada = 3 je sekvence rozsvícených LED následující:

____ ___o __oo _ooo ooo_ oo__ o___ ____

Odevzdávání do ReCodExu

- Moccarduino – emulátor Arduina a FunShieldu
  - Emulace na úrovni funkcí, nikoliv emulace celé desky
  - Občas se chová odlišně (viz. tato stránka)
    - Speciálně pak kód nebude spouštěn na Arduinu ale na stroji s architekturou AMD64 (rozdílná velikost typů, např. int 16 vs 32b -> pozor na různé konstanty typu MAX_INT / volání std::max apod.)
  - Evaluace v ReCodExu je více striktní
    - Velmi citlivé na časování (velmi malé rozdíly v časování okem neodhalíte, ale ReCodEx ano)
    - Kompilátor v Arduino IDE je více tolerantní než GCC v ReCodExu