# NSWI170 – Počítačové systémy

Tomáš Faltín

# Úkol 0: Jak ladit/odhmyzovat(=debugging) Arduino

- Arduino IDE
  - Tools → Serial Monitor → (nastavte rychlost) 9600 baund
- `setup()`
  - `Serial.begin(9600); // nastavte stejnou rychlost jako v IDE`
- Používání
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/
  - `print(), println()`
    - `Serial.print(„Hello world“); Serial.println(1234); ...`
- Vkládání ladících výpisů, kudy program běžel

# Úkol 1: Zápis čísla na displej

1. Vezměte si úkol 3 z minula (zápis čísla na libovolné pořadí)
2. Vytvořte funkci, která umí zapsat číslo mezi 0000-9999 na displej

# Pokus: delay() + segmentový displej

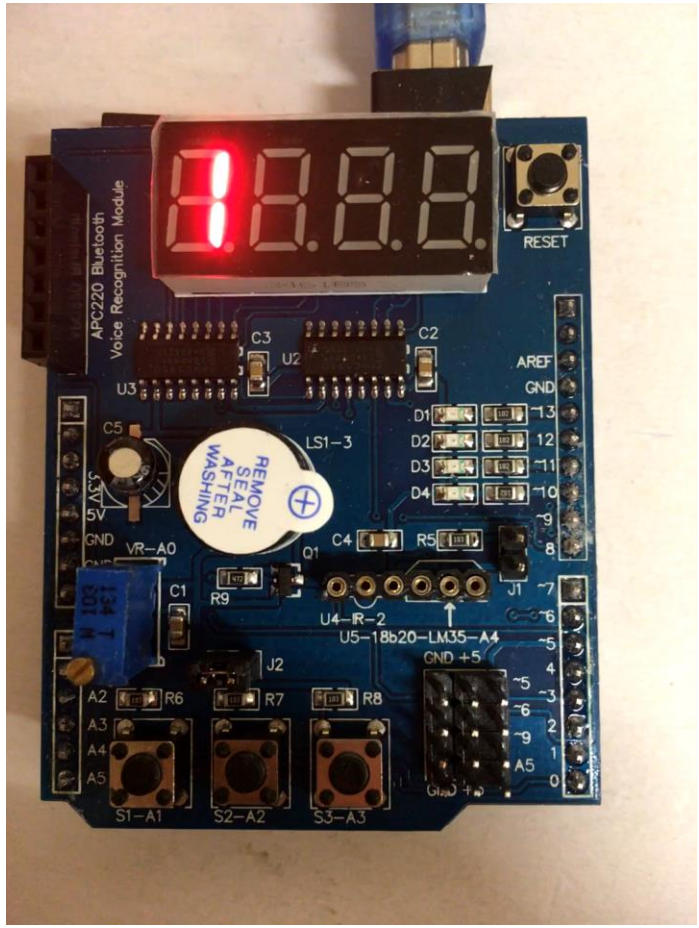
- Přidejte delay do hlavní smyčky hned za volání vaší funkce, např:

```
void loop() {
  seg_write_number(1234);
  delay(200);
}
```

- Zobrazuje displej číslo správně?
- Základní (naivní) implementace na videu na dalším slide

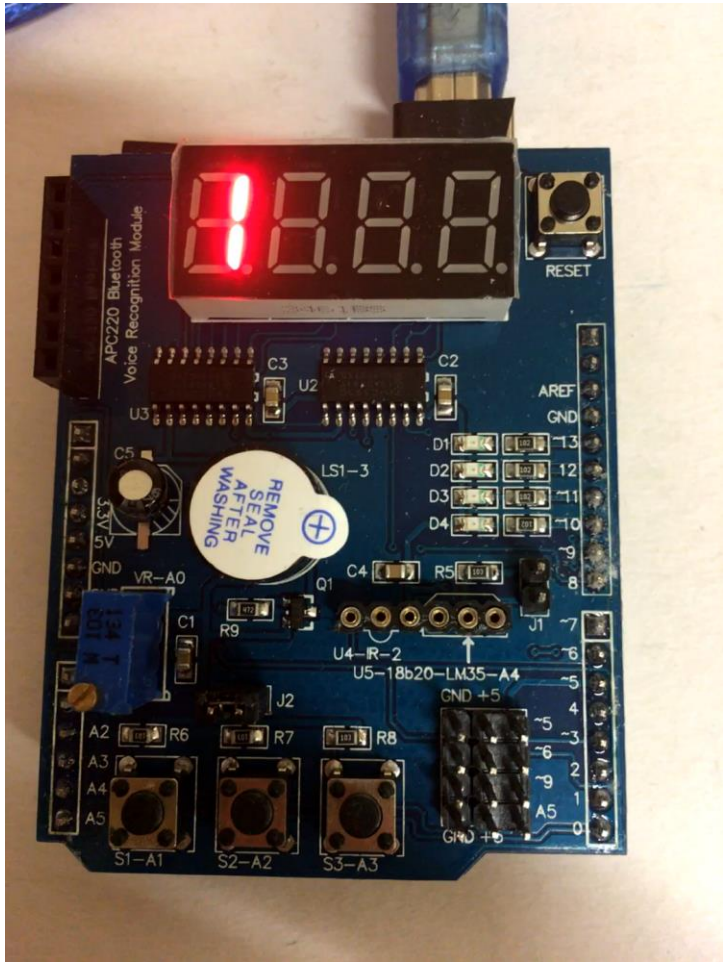
# Pokus: delay() + segmentový displej

- **Problém:**
  - Při zápisu jednoho znaku ostatní zhasnou
- Chceme dělat i další věci než jen zobrazovat věci na displeji



# Řešení: delay() + segmentový displej

- Časový multiplex
  - Pokud bude Arduino dostatečně rychle blikat, oko si bude myslet, že svítí
  - Můžu rychle za sebou zobrazovat znaky
    - 1 cyklus = zobrazení 1 znaku
- Zobrazení na displeji rozdělím na 2 funkce:
  1. Nastaví hodnotu
  2. V cyklu zobrazuje postupně znaky čísla



# Poznámka

- Hlavní funkce `loop()` by měla být co nejkratší, jelikož obsluhuje všechny funkce
- Složitější úlohy se podrozdělí na menší podúlohy
  - Když jsou dostatečně malé, vykonávají se v rámci jednoho volání funkce `loop()`
- Podobný princip/pattern (hlavní funkce která deleguje práci do menších funkcí) i u jiných (moderních) frameworků
  - Důraz na používání asynchronního volání
  - Např.: Node.js – JS runtime nad V8 enginem (engine, který používá Chrome)


# Úkol 2: Multiplexové zobrazení čísla na displeji

- Vytvořit funkce pro zápis čísel na displej
  1. Nastaví hodnotu (může se zavolat např. v `setup()`, případně ve funkci `loop()`)
  2. Volá se opakovaně v hlavní funkci a zobrazuje dané číslo
- Pomocný stav si držím v globálních proměnných

```
void loop() {
  display_loop(); // (2) zobrazuje číslo
  if (…)  {
    display_set(1234); // (1) uloží zadané číslo
  }
}
```

# Poznámka (objektový návrh)

- Věci, které k sobě logicky patří, je dobré umístit k sobě/označit, že k sobě patří (+ třeba limitovat přístup jen na některé funkce atd...)
  - encapsulace/zapouzdření
  - https://en.wikipedia.org/wiki/Encapsulation_(computer_programming)
- Vyšší programovací jazyky k tomu poskytují lepší nástroje (třídy, metody, viditelnost, ...)
- Lze programovat objektově i v nízkých programovacích jazycích, např. C, Pascal, ...

# Příklad: objektový návrh

```
// ukládá pozici v čísle zobrazovanou v loopu
int display_current_position;
// ukládá číslo, které právě zobrazujeme
int display_number;

// funkce inicializuje displej
display_setup()
// funkce starající se o zobrazení displeje
display_loop()
// funkce nastavující displej na dané číslo
display_set()
```

- Všimněte si, že všechny funkce/proměnné obsahují prefix `display_`
  - Je jasné, že funkce patří/provádí operace s displejem
- Označím stejně funkce/proměnné které k sobě logicky patří

# Úkol 3: Opravdové počítadlo

- Rozšíření úkolu 4 z minula
- Zobrazuje číslo na dipleji (tentokrát již 4-místné)
- Tlačítka mají stejnou funkci: Přičítání/odečítání 1, reset

# Úkol 4: Jednoduché stopky

- Přesnost na 0.1s
  - Zobrazují i desetinné místo – `0.0`
- Tlačítka:
  1. start/stop
  2. reset
     - Pouze pokud jsou stopky zastavené

# Úkol 5: Olympijské stopky

- Rozšíření úkolu 4, přídává funkcionalitu poslednímu tlačítku
- Nové tlačíko:
  3. okruh
     - Aktuální číslo na displeji se zastaví, ale stopky běží (interně) dál, po opětovném stisknutí stopky zobrazují opět aktuální hodnotu (ne tu zastavenou) a pokračují dál

# Domácí úkoly

- Nahrád do SISu zdroják obsahující funkce pro úkoly 3 a 5
- Do 14 dnů
- Podmínky:
  - Funkční
  - Rozdělené do funkcí
  - Srozumitelně pojmenováné konstanty/funkce
  - **Objektový návrh** funkcí = funkce, které k sobě patří mají stejný prefix