6. Přerušení, sériová linka

Obsah

Časovač

Časovač umožňuje s vysokou přesností odměřovat čas. Obsahuje 8 nebo 16 bitový registr, který s příchodem hrany hodinového signálu změní svoji hodnotu o jedničku. Hodnota se může zvyšovat nebo snižovat (nejčastěji se zvyšuje).

Časovače jsou nejčastěji použité pro obsluhu funkcí `delay()`, `millis()`

V tomto cvičení časovače používat nebudete. Pokud byste si práci s nimi chtěli vyzkoušet, doporučujeme využít již hotové knihovny pro práci s časovači, př. https://www.arduino.cc/reference/en/libraries/arduino-timer/

Přerušení

Mechanismus přerušení (interrupt) zajišťuje vyvolání podprogramu (tzv. obsluhy přerušení) na základě vnější, nebo vnitřní události.

Přerušení je možné zakázat/(opětovně) povolit funkcemi `noInterrupts()` / `interrupts()`.

Podrobnější informace zde: https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/.

Vnější přerušení

Zdrojem vnějších přerušení jsou periferní obvody (časovač, COM port apod.). Přerušení jim umožňuje si asynchronně vyžádat pozornost procesoru a zajistit tak svoji obsluhu ve chvíli, kdy to právě potřebují bez ohledu na právě zpracovávanou úlohu.

Vnitřní přerušení

Vnitřní přerušení vyvolává sám procesor, který tak signalizuje problémy při zpracování strojových instrukcí. Jedná se například o pokus dělení nulou, porušení ochrany paměti, výpadek stránky.

Zapojení

V TÉTO ÚLOZE BUDETE POTŘEBOVAT:

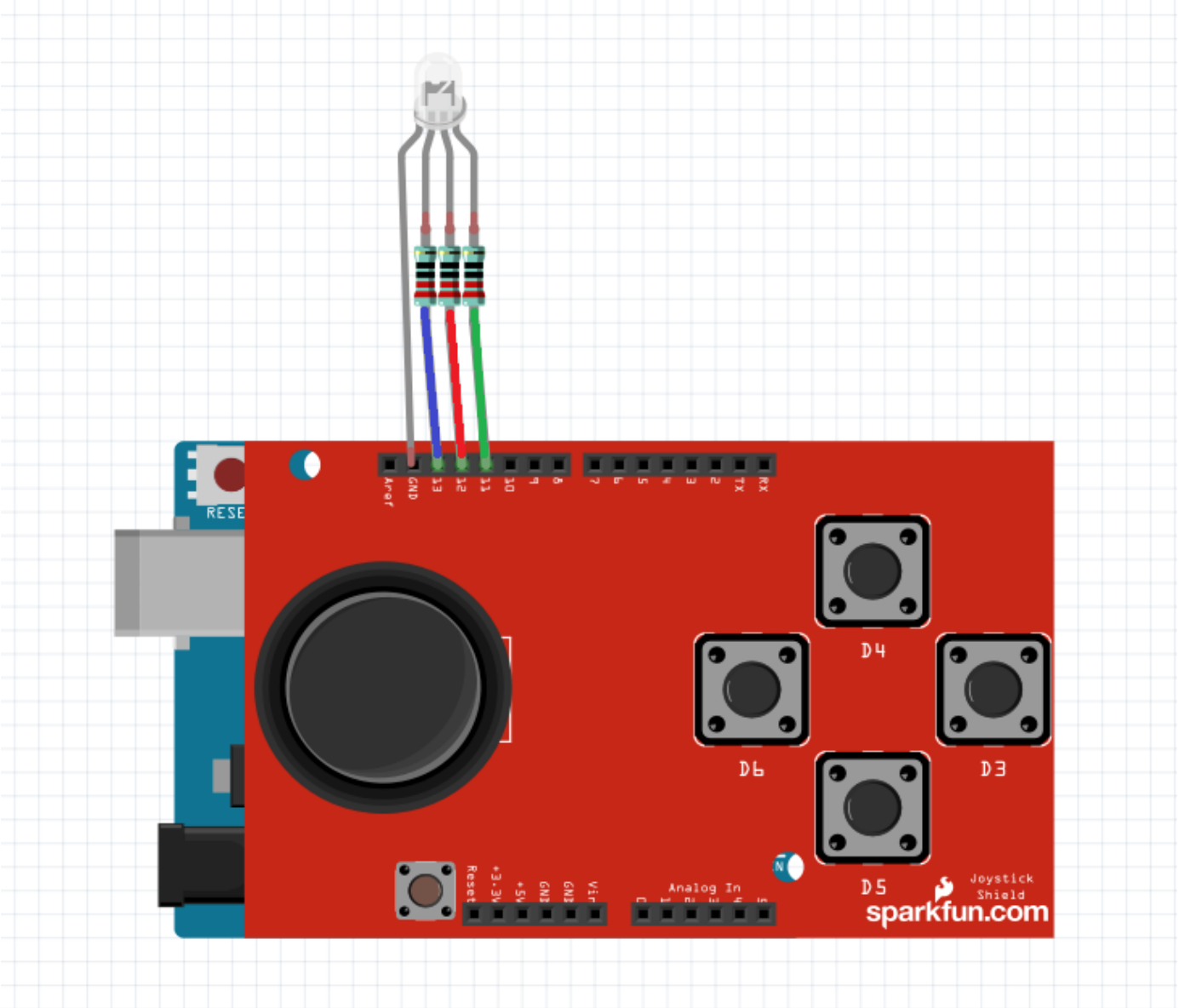
- 1 tlačítko
- RGB LED (nebo 3 LED)

Úlohy

1. úloha - Blikání RGB LED pomocí přerušení (4 b)

Rozblikejte RGB LED s frekvencí 1 Hz. Při každém zmáčknutí tlačítka změňte (jednou) barvu LED. Využijte vnějšího (externího) přerušení.

- Nejprve připojte RGB LED k joystick shieldu podle obrázku Obrázek 1



Obrázek 1. Schéma zapojení RGB LED k tlačítkovému shieldu

- Založte nový projekt v Arduino IDE.
- Zadefinujte piny připojené RGB LED například následovně:

```
#define BLUE_PIN  11
#define GREEN_PIN 12
#define RED_PIN   13
```

- Ve funkci `setup()` tyto piny nastavte jako výstupní pomocí funkce PinMode() (https://www.arduino.cc/en/Reference/PinMode):

```
pinMode(GREEN_PIN, OUTPUT);   // set GREEN_PIN as OUTPUT
```

- Je třeba zjistit, na které piny je možné nastavit přerušení. Podle dokumentace (https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/) naleznete na Arduino Uno (a dalších Arduinech založených na čipu ATmega328) právě dva takové piny a to pin 2 a pin 3. V případě Arduino Leonardo jde o piny 0, 1, 2, 3, 7.

- Připojení pinů na tlačítka je následující:

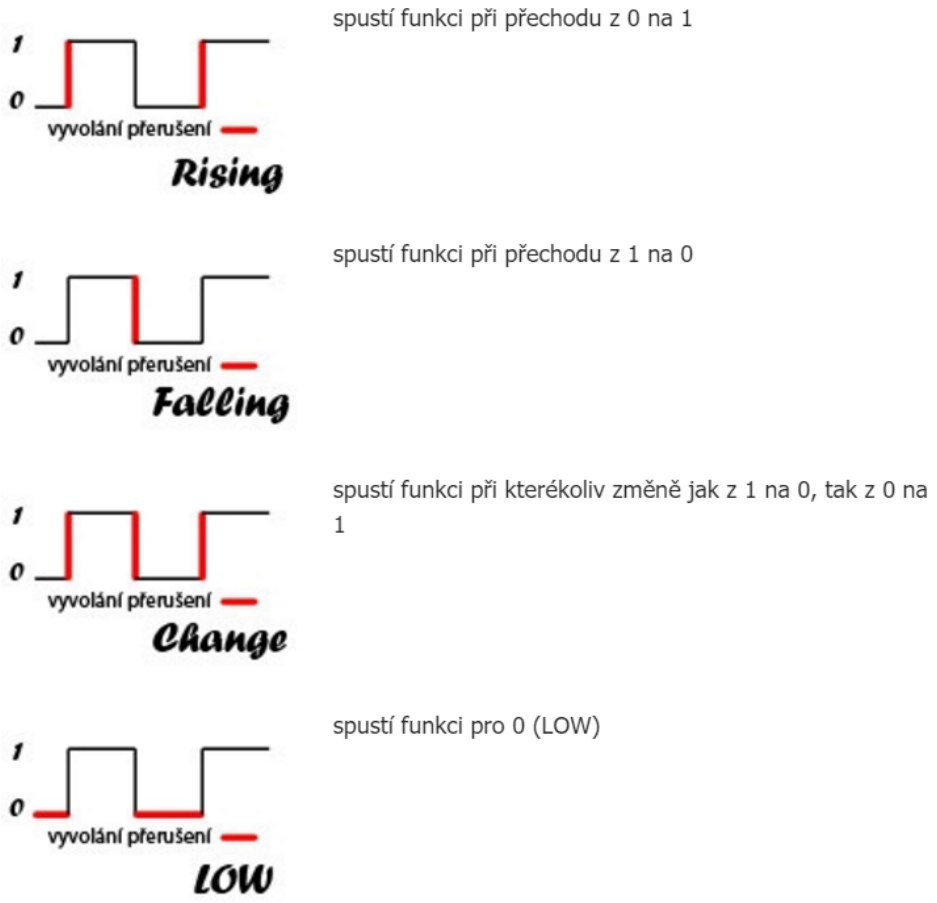
```
up button       = 2
down button     = 4
left button     = 5
right button    = 3
joystick button = 8
```

Externí přerušení tedy můžeme vyvolávat pomocí tlačítek **up** a **right**.

Pro práci s joystickem se můžete inspirovat zdařilou dokumentací k *SparkFun Joystick Shield*. Stránka z webu zmizela, je však stále dostupná v internetovém archivu (https://web.archive.org/web/20150911052342/https://www.sparkfun.com/tutorials/171) .

Dejte si ovšem pozor na jiné zapojení pinů!

- Pro nastavení přerušení využijte vestavěnou funkci `attachInterrupt()` (https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/) . Tato funkce v případě definované události (viz mode níže) vyvolá přerušení. Funkce má 3 parametry:
  i. `digitalPinToInterrupt(pin)` : funkce, která dle zadaného pinu identifikuje a vrátí konkrétní číslo přerušení
  ii. **obsluha přerušení (ISR)**: funkce volaná po vyvolání přerušení
  iii. **mode**: určuje, při které změně na pinu je vyvoláno přerušení - `CHANGE`, `RISING`, `LOW` nebo `FALLING`



Obrázek 2. Zdroj: Arduino a využití přerušení (interrupt) a ošetření záchvěvů při stisku tlačítka (http://arduino8.webnode.cz/news/lekce-21-arduino-a-vyuziti-preruseni-interrupt-a-osetreni-zachvevu-pri-stisku-tlacitka)

Volání funkce tedy vypadá například takto:

```
#define INTERRUPT_PIN 2

void setup()
{
  // ...
  pinMode(INTERRUPT_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), do_something, FALLING);
}

void do_something() {
}
```

V této ukázce tedy po stisknutí pravého tlačítka dojde k vyvolání přerušení a zavolá se funkce `do_something()`.

- K rozsvícení RGB LED využijte funkce `digitalWrite()`, viz dokumentace (https://www.arduino.cc/en/Reference/DigitalWrite).

> Proměnné, jejichž hodnotu měníte v obsluze přerušení je vhodné deklarovat jako tzv. volatile

(https://en.wikipedia.org/wiki/Volatile_(computer_programming)), př.:

```
volatile int i;
```

2. úloha - Vývoj aplikace komunikující s Arduinem (6 b)

V této úloze si vyzkoušíte naprogramovat aplikaci, která komunikuje s Arduinem pomocí sériové linky. Arduino bude aplikaci posílat údaje o teplotě získané z teploměru (v případě Una či Leonarda o poloze joysticku) a aplikace naopak zadá Arduinu, kterou barvu RGB LED má rozsvítit.

Python

- Spusťte VSCODE a otevřete workspace (adresář) ve kterém budete vytvářet PC apliakci.
- Do VSCODE nainstalujte rozšíření (v záložce Extensions) pro podporu Pythonu.
- Otevřete konzoli ve VSCODE, nainstalujte balíček pyserial: `pip install pyserial`
- Můžete použít a upravit následující kostru aplikace.
- Nezapomeňte změnit parametry sériové linky (především COM port).

```
import serial
import sys
from time import sleep


class Uart:
    def __init__(self):
        self._ser = serial.Serial()
        self._ser.port = 'COM6'  # TODO set the correct com port!
        self._ser.baudrate = 9600
        self._ser.timeout = 1
        try:
            self._ser.open()
        except serial.SerialException:
            print("Error while opening")
            sys.exit()

    def readData(self): # returns received data as string
        return self._ser.readline().decode("utf-8", errors="ignore")  # ignore invalid UTF-8 characters

    def writeData(self, data): # sends string data
        to_send = bytes(data,'ASCII')
        self._ser.write(to_send)

    def close(self):
        self._ser.close()
```

```
def main():

    ser = Uart()
    while True:
      pass # todo replace pass by actual code

if __name__ == "__main__":
    main()
```

starý návod v C/C++ (nedoporučujeme)

> Kód se týká vypracování úloze na Esploře. Pracujete-li na Unu či Leonardu, pozměňte jej dle toho (především co se týče čtení joysticku místo teploty).

- Založte v C/C++ IDE (př. v Netbeans) nový projekt.
- Implementaci samotné sériové komunikace máte předpřipravenou v souborech `SerialPort.h` a `SerialPort.c` : serialport.zip (../../media/tutorials/06/serialport.zip) . Připojte tyto soubory ke svému projektu.
- Pro zahájení komunikace je třeba nejdříve otevřít příslušný COM port pomocí funkce `connect()` , kterou naleznete v souboru `SerialPort.c` . Prostudujte si, jak tato funkce pracuje.
- Ve funkci `main()` tedy zavolejte funkci `connect()` a předejte jí příslušné parametry, handle (https://cs.wikipedia.org/wiki/Handle) a název COM portu. Handle deklarujete například takto:

```
HANDLE hFile;
```

- Zavolejte funkci `writeData()` , která pošle Arduinu informaci o barvě RGB LED k rozsvícení. Předejte handle jako parametr.
- Zavolejte funkci `readData()` , která přečte data poslaná z Arduina, konkrétně momentální teplotu ve stupních Celsia. Předejte handle jako parametr.
- Ošetřete návratové hodnoty všech tří volaných funkcí, tj. v případě, že vrátí false, nepokračujte dále v programu. Pozor! Před ukončením `main()` uvolněte handle pomocí funkce `CloseHandle()` :

```
CloseHandle(hFile);
```

- Nyní otevřete nový sketch v Arduino IDE. Zahajte sériovou komunikaci zavoláním funkce `Serial.begin()` .
- Ve smyčce čtěte hodnoty ze sériové linky a podle získaných hodnot, rozsviťte příslušnou RGB LED.
- Zároveň čtěte hodnotu teploměru pomocí `Esplora.readTemperature()` a posílejte tento údaj aplikaci například funkcí `Serial.println()` .
- Nakonec nahrajte program do Arduina a spusťte kód v IDE. Zkontrolujte, že svítí správná RGB LED a

na standardní výstup se vypisuje teplota.

Pro komunikaci s COM portem vyšším než 9, je třeba ve funkci `createFile()` zadat název portu takto: `\\\\.\\name`, tedy například: `\\\\.\\COM10` (narozdíl od portů 1 až 8, u kterých stačí zadat jméno, tedy například: `COM8`). Více informací zde: https://support.microsoft.com/en-us/kb/115831

Podrobnější informace o sériové komunikaci pod OS Windows naleznete na: https://msdn.microsoft.com/en-us/library/ff802693.aspx.

6. Přerušení, sériová linka
tutorials/06/index.adoc, poslední změna 52816e6b (24. 3. 2025 v 11:52, Robert Hülle)
Build status