

# **Přehled a použití mikropočítačových zařízení pro začátečníky**

Overview and usage of microcontroller kits suitable for beginners

Tomáš Bělaška



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2011/2012

## **ZADÁNÍ BAKALÁŘSKÉ PRÁCE**

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Tomáš BĚLAŠKA**  
Osobní číslo: **A09207**  
Studijní program: **B 3902 Inženýrská informatika**  
Studijní obor: **Bezpečnostní technologie, systémy a management**

Téma práce: **Přehled a použití mikropočítačových zařízení pro začátečníky**

Zásady pro vypracování:

1. Zpracujte literární rešerši na téma vývojových prostředků s mikropočítači se zaměřením na zařízení určená pro začátečníky.
2. Zvolte několik zástupců těchto prostředků a ověřte jejich praktické využití.
3. Zpracujte návody na práci se zařízeními vybranými v bodě 2 včetně jednoduchých ukázkových příkladů.
4. Zhodnoťte vlastnosti těchto prostředků z hlediska snadnosti použití i možností jejich uplatnění.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

1. MANN, Burkhard. C pro mikrokontroléry: ANSI-C, kompilátory C, spojovací programy – linkery, práce s ATMEL AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky. Vyd. 1. Praha: BEN, 2003, 279 s. ISBN 80-730-0077-6.
2. PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-730-0110-1.
3. CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol: O'Reilly, 2005, 377 s. ISBN 05-960-0755-8
4. ARDUINO. Arduino Home Page. 2011. Dostupné z: <http://www.arduino.cc>
5. HCS08 Microcontrollers. FREESCALE, Inc. Freescale. 2011.
6. Atmel AVR 8- and 32-bit Microcontrollers. ATMEL CORPORATION. Atmel. 2012.

Vedoucí bakalářské práce:

Ing. Jan Dolinay, Ph.D.

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce:

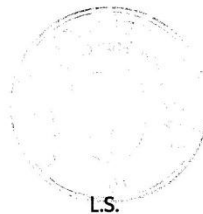
24. února 2012

Termín odevzdání bakalářské práce:

25. května 2012

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.  
děkan



L.S.

doc. Mgr. Milan Adámek, Ph.D.  
ředitel ústavu

## **ABSTRAKT**

Tato práce je věnována především začátečníkům v oblasti programování mikrokontrolerů. Začátečník se zorientuje v termínech dané problematiky, získá přehled o základních metodách a dostupných zařízeních, které může využít pro programování mikrokontrolerů.

Obsahem teoretické části je vysvětlení základních pojmů týkajících se mikrokontrolerů. Jsou popsány jejich jednotlivé části s větším důrazem na ty, se kterými přichází začátečník při programování do styku a jejich znalost je pro další práci nezbytná. Praktická část se věnuje konkrétním vývojovým prostředkům, které byly zvoleny jako nejvhodnější. Součástí praktické části jsou návody na zprovoznění vybraných zařízení a také ukázkové programy v jazycích určených pro tato zařízení.

Klíčová slova: mikrokontroler, vývojové prostředky, Arduino, Atmel, PICAXE, Freescale

## **ABSTRACT**

This thesis is mainly focused on the beginners in the area of programming of microcontrollers. Beginners will familiarize themselves with this issue, get knowledge about basic methods and available devices, which a beginner can use for programming of microcontroller.

The content of the theoretical part is explanation of basic terms relating to microcontrollers. Some components and circuits are explained in more detailed way, especially those with which beginners work most frequently and knowledge of them is necessary. The practical part is dedicated to particular development devices, which were selected as the most suitable ones. There are also instructions how to get the selected device to work. Several sample programmes in language intended for this device are also included.

Keywords: microcontroller, development devices, Arduino, Atmel, PICAXE, Freescale

Na tomto místě bych chtěl poděkovat vedoucímu práce, Ing. Janu Dolinayovi, Ph.D. za jeho čas, který mi věnoval při konzultacích, za cenné rady a připomínky, na základě kterých tato práce mohla vzniknout.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD.....</b>	<b>9</b>
<b>I TEORETICKÁ ČÁST.....</b>	<b>10</b>
<b>1 OBECNÉ POJMY.....</b>	<b>11</b>
1.1 MIKROPROCESOR, MIKROKONTROLER.....	11
1.2 KONCEPCE MIKROPROCESORU.....	12
1.3 ARCHITEKTURA MIKROPROCESORU.....	13
<b>2 MIKROKONTROLER.....</b>	<b>14</b>
2.1 MIKROPROCESOR.....	14
2.1.1 Aritmeticko-logická jednotka.....	14
2.1.2 Řadič.....	14
2.1.3 Registry.....	15
2.2 PAMĚTI.....	16
2.2.1 ROM.....	16
2.2.2 PROM.....	16
2.2.3 EPROM.....	16
2.2.4 EEPROM.....	16
2.2.5 FLASH.....	17
2.2.6 RAM.....	17
2.3 KOMUNIKAČNÍ ROZHRANÍ.....	17
2.3.1 USB.....	18
2.3.2 SPI.....	18
2.3.3 IIC.....	19
2.3.4 UART.....	20
2.3.5 CAN.....	21
2.4 PERIFERNÍ OBVODY.....	22
2.4.1 Zdroj hodinového signálu.....	22
2.4.2 Čítače a časovače.....	23
2.4.2.1 Režim čítání.....	23
2.4.2.2 Režim časování.....	23
2.4.3 PWM.....	23
2.4.4 A/D převodník.....	24
2.4.5 Watchdog.....	26
<b>3 ZPŮSOBY PROGRAMOVÁNÍ.....</b>	<b>27</b>
3.1 PROGRAMOVACÍ A LADICÍ ROZHRANÍ.....	27
3.1.1 BDM.....	27
3.1.2 JTAG.....	28
3.1.3 ISP.....	29
3.2 SIMULÁTORY.....	30
3.3 PROGRAMÁTORY.....	30
3.4 VÝVOJOVÉ KITY.....	31
<b>II PRAKTICKÁ ČÁST.....</b>	<b>32</b>
<b>4 ARDUINO UNO.....</b>	<b>34</b>

4.1	SPECIFIKACE KITU ARDUINO UNO.....	35
4.2	ARDUINO IDE .....	36
4.2.1	Vytvoření a přenesení programu do MCU .....	37
4.3	WIRING .....	37
4.4	ZPROVOZNĚNÍ KITU ARDUINO UNO.....	39
4.5	SCHÉMA ZAPOJENÍ.....	40
4.6	SROVNÁVACÍ PROGRAM PRO ARDUINO UNO .....	41
<b>5</b>	<b>PICAXE.....</b>	<b>43</b>
5.1	SPECIFIKACE MCU PICAXE 08M2 .....	44
5.2	ZPROVOZNĚNÍ MCU PICAXE 08M2.....	45
5.3	PICAXE PROGRAMMING EDITOR .....	45
5.3.1	Vytvoření a přenesení programu do MCU .....	46
5.4	BASIC .....	46
5.5	SCHÉMA ZAPOJENÍ.....	47
5.5.1	Schéma zapojení pro nahrávání programu .....	47
5.5.2	Schéma zapojení pro běh programu .....	48
5.6	SROVNÁVACÍ ZDROJOVÝ KÓD .....	49
<b>6</b>	<b>FREESCALE M68EVB08GB60 .....</b>	<b>51</b>
6.1	SPECIFIKACE VÝVOJOVÉHO KITU .....	52
6.2	ZPROVOZNĚNÍ VÝVOJOVÉHO KITU .....	52
6.3	FREESCALE CODEWARRIOR .....	53
6.3.1	Vytvoření a přenesení programu do MCU .....	53
6.4	JAZYK C .....	54
6.5	SROVNÁVACÍ ZDROJOVÝ KÓD .....	54
<b>7</b>	<b>SROVNÁNÍ TESTOVANÝCH PROSTŘEDKŮ .....</b>	<b>57</b>
	<b>ZÁVĚR .....</b>	<b>60</b>
	<b>ZÁVĚR V ANGLIČTINĚ.....</b>	<b>61</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>62</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>65</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>67</b>
	<b>SEZNAM TABULEK.....</b>	<b>68</b>



## ÚVOD

Elektronická zařízení nás v současné době obklopují prakticky neustále. Tato zařízení mají jednu společnou věc a to tu, že jsou řízeny či ovládány mikroprocesorem, mikropočítačem nebo mikrokontrolerem, záleží na druhu a složitosti aplikace, ve které součástku použijeme. Řízení pomocí těchto miniaturních elektronických „mozků“ dnes není výsadou pouze aut, strojů, inteligentních domů, notebooků, praček či mobilních telefonů, ale naprosto běžně se využívají například i v hračkách, což svědčí o několika aspektech.

V první řadě je to samotný pokrok v oblasti vývoje integrovaných obvodů, který ve 20. století nastartoval rozšíření těchto součástek do prakticky všech oblastí našeho života bez rozdílu věkové kategorie. S tím souvisí jejich cena. To, že se dnes tyto miniaturní obvody o vysoké integraci mohly takovýmto způsobem rozšířit, je jejich cena, která se pohybuje v řádech jednotek korun, a jsou tím pádem dostupné prakticky komukoliv. Další z mnoha aspektů, který vedl k tak masovému rozšíření, je jejich rozměr. Díky minimálním rozměrům je lze využívat v nejmenších aplikacích.

Cílem práce je seznámit začátečníka se základními pojmy týkajícími se mikrokontrolerů, ale především mu ukázat dostupné metody a možnosti jejich programování a ladění. Po přečtení práce by si měl být začátečník schopen vybrat nejvhodnější způsob vývoje mikrokontroleru, z hlediska vhodnosti pro jeho budoucí aplikaci či na základě jeho předchozích zkušeností s programováním.

Teoretická část bude v první řadě věnována zejména obecným pojmům, které je potřebné rozlišit pro další práci a vyhledávání informací. V druhé části budou popsány jednotlivé komponenty a obvody, které současné mikrokontrolery obsahují a o které mohou být doplněny. Poslední část teoretické práce bude věnována vývojovému softwaru a hardwaru, který lze použít. Pozornost bude věnována zejména vývojovým kitům, ale neopomenou popsat ani alternativní metody. Praktická část se již bude věnovat konkrétním vývojovým kitům, které budou shledány pro začátečníka jako nejvhodnější. Ty budou voleny na základě dostupnosti samotného vývojového kitu, dostupnosti a kvalitě dokumentací od výrobců, kvalitu softwarových vývojových prostředí a také přihlídnou k rozšířenosti a oblíbenosti mezi uživateli v České Republice. Je to z důvodu jazykové bariéry, jelikož většina tzv. datasheetů či manuálů se do češtiny vůbec nepřekládá, což by mohlo určitou část lidí odradit.

## **I. TEORETICKÁ ČÁST**

## 1 OBECNÉ POJMY

Pro každého člověka, který začíná nejenom s některou z oblastí elektroniky, je nezbytné, aby znal základní pojmy a termíny. Správné vyjadřování může výrazně ušetřit čas při učení a také při vyhledávání dalších informací k danému problému, v opačném případě hrozí odrazení uživatele od používání konkrétního výrobku, komponenty či metody, což ale není žádoucí. Výrobci si to dobře uvědomují, a proto jsou ve většině případů dostupné velmi dobře a podrobně zpracované dokumentace většinou však v anglickém jazyce.

### 1.1 Mikroprocesor, mikrokontroler

Velmi důležité je rozlišit tyto dva základní prvky. Mikroprocesor, označován také jako jádro či CPU (Central Processing Unit), je složitý logický obvod, který vykonává sled aritmetických a logických operací na základě instrukcí, které jsou obsaženy v programu. Důležitou informací je, že bez dalších podpůrných obvodů není schopen samostatně pracovat. Základními částmi mikroprocesoru jsou:

- aritmeticko-logická jednotka (ALU)
- řadič (Control Unit)
- registry

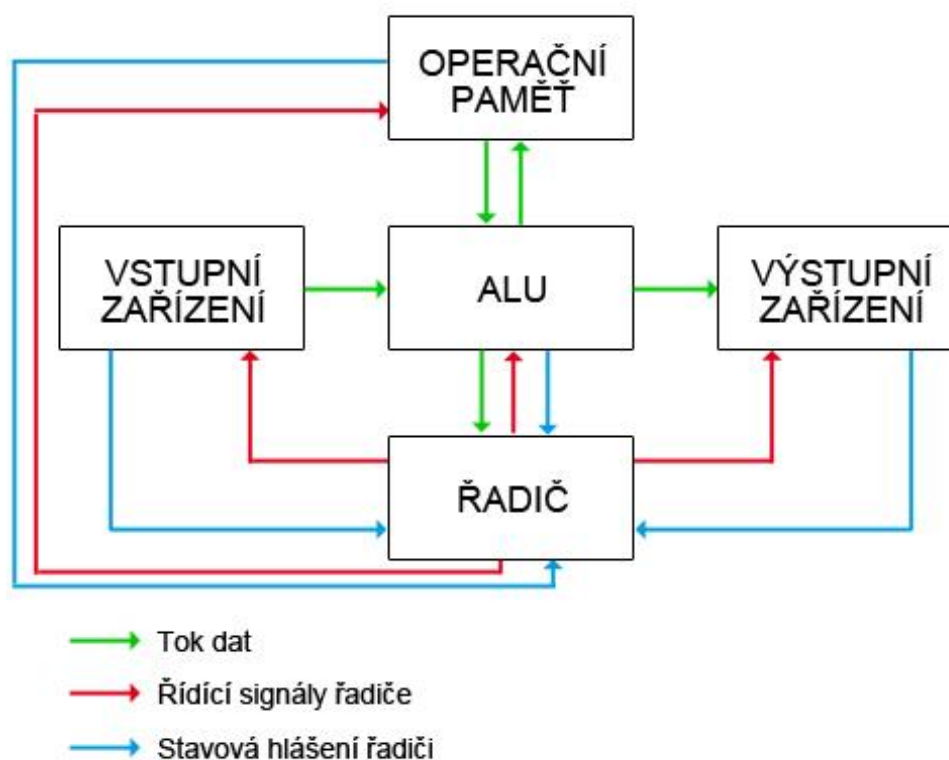
Pokroky v technologii integrovaných obvodů (dále IO) umožnily zmenšení rozměrů a koncentraci mnoha funkcí do jednoho IO. Vzniká tak nový pojem jednočipový mikropočítač. Jinak řečeno, jedná se o mikropočítač, který integruje velké množství podpůrných obvodů do jednoho čipu. Obecně se v anglické literatuře, na webových stránkách, datasheetech a katalozích nepoužívá pojem jednočipový mikropočítač, nýbrž mikrokontroler (angl. microcontroller nebo MCU). V této práci je z tohoto důvodu využíván termín mikrokontroler (dále MCU).

Mikrokontroler je tedy obvod o vysoké integraci, který dokáže, na rozdíl od CPU, samostatně pracovat a obsahuje tyto komponenty a obvody, které budou v druhé kapitole dále rozvedeny:

- mikroprocesor (jádro, CPU)
- paměť dat
- paměť programu
- vstupní a výstupní obvody a rozhraní pro komunikaci

## 1.2 Koncepce mikroprocesoru

Rozeznáváme dvě základní koncepce mikroprocesoru, které se týkají jeho paměťového prostoru. Von Neumannova koncepce byla navržena ve 40. letech 20. století a její blokové schéma je znázorněno na obrázku č. 1. Hlavním znakem této koncepce je společná paměť programu a dat. Podle von Neumannovi koncepce se tedy mikroprocesor skládá z aritmeticko-logické jednotky, řadiče a operační paměti, která slouží jak k uchování zpracovávaného programu, tak ke zpracovávání dat a výsledků výpočtů. Můžeme říci, že procesor zpracovává data pomocí předepsaného programu, který je sám o sobě reprezentován daty. Z toho plyne, že programy resp. data lze libovolně měnit a modifikovat a měnit funkce počítače. [2]



Obr. 1: Koncepce podle von Neumanna

Harvardská koncepce se začala objevovat v 60. letech 20. století a je specifická tím, že dělí paměťový prostor na prostor zvlášť pro data a pro program. Je to výhodnější z hlediska rychlosti procesoru, jelikož lze zároveň číst instrukce programu a pracovat s pamětí dat. [2]

### 1.3 Architektura mikroprocesoru

Známe základní dva druhy architektur mikroprocesorů. Jsou to CISC (Complex Instruction Set Computer) a architektura typu RISC (Reduced Instruction Set Computer). První známou a používanou je CISC, což označuje architekturu, která obsahuje velmi rozsáhlou sadu instrukcí. Věřilo se, že zjednoduší práci jak programátorům při psaní, tak i překladačům instrukcí. Postupem času se ale ukázalo, že rozsáhlá instrukční sada nebývá využívána, takže výhody, které hovořily pro tuto architekturu, již nebyly namístě. Z tohoto důvodu se začala tvořit architektura s pečlivě vybranými instrukcemi, která dostala název RISC, tedy sada s redukováním počtem instrukcí. V současné době je dnes používanějším typem architektury.

Postupem času se začínaly tyto architektury různými způsoby kombinovat a jako alternativa k CISC a RISC vznikla architektura MISC (Minimum Instruction Set Computer). MISC se vyznačovala použitím funkcí bez operandů. Operand je část instrukce, která je uložena v datové paměti a může například přesunout obsah vnitřního registru do datové paměti na nějakou adresu. Operandy mohou být vypuštěny díky tomu, že jsou známy implicitně, jelikož se většina operací provádí s hodnotami uloženými na interním či externím zásobníku (stack), jehož funkce bude vysvětlena v jiné kapitole. Instrukce bez operandů byly díky tomuto kratší, tím pádem není architektura tak náročná na rychlost operační paměti, často odpadala nutnost použití cache paměti a rychlejší byly také reakce na přerušení. Použití MISC je vhodné například v oblasti řízení či real-time systémech.

Poslední základní architekturou je VLIW (Very Long Instruction Word). Ta se používá pouze ve specializovaných aplikacích (výpočetní servery či grafické akcelerátory) kvůli velkým nárokům na rychlost operační paměti. U VLIW se používají instrukce pevné délky, podobně jako je tomu u RISC, ovšem délka těchto instrukcí je mnohem delší než 32 bitů (délka instrukce u RISC je většinou 32 bitů), dosahuje šířky řádově několik desítek i stovek bitů. Je tomu tak z toho důvodu, že každé instrukční slovo je rozděleno do většího množství polí (fields), přičemž v každém poli je umístěn kód jedné operace, instrukce.

## 2 MIKROKONTROLER

V předchozí kapitole byly uvedeny základní pojmy, které budou v této kapitole dále podrobněji popsány.

Díky skutečnosti, že jsou doplňující obvody MCU, které jsou v této kapitole popsány, integrovány v jednom pouzdře a není nutné je realizovat externě, často disponují MCU méně výstupy. Mohlo by se to zdát jako nevýhoda z důvodu toho, že zde není taková možnost dalšího rozšíření, na druhou stranu dnes ale každý výrobce nabízí tak širokou škálu MCU, že si můžeme zvolit mikrokontroler přesně podle našich požadavků. MCU se díky těmto vlastnostem využívají jako části nějakých dalších zařízení či aplikací, označované také jako vestavěné, angl. *embedded*. V dalších kapitolách budou tedy podrobněji popsány jednotlivé části, které MCU zpravidla obsahuje.

### 2.1 Mikroprocesor

Označován také jako CPU (Central Processing Unit), jádro či mozek. Mikroprocesor provádí instrukce uložené v paměti programu ve správném pořadí a řídí ostatní části mikrokontroleru. Dále řídí tok dat dovnitř a ven pomocí vstupních a výstupních obvodů. Podle šířky datové sběrnice se CPU dělí na osmibitové, šestnáctibitové atd.

#### 2.1.1 Aritmeticko-logická jednotka

Je označovaná také jako ALU (Arithmetic-Logic Unit). V této jednotce jsou prováděny aritmetické (sčítání, násobení...) a logické (logický součet a součin, exkluzivní součet, negace...) operace s daty. Výsledky, ve tvaru binárního čísla, těchto operací jsou dále předávány a zpracovávány v příslušných registrech.

Některé procesory mají ALU doplněnou ještě o další obvody, například FPU (Floating Point Unit), což je obvod, který slouží pro práci s čísly s plovoucí čárkou (tento typ čísel se označuje jako *float*). Tyto doplňky se používají z důvodu celkové rychlejší činnosti počítače, na druhou stranu je takto rozšířená ALU mnohem složitější. [1]

#### 2.1.2 Řadič

Označován jako CU (Control Unit). Jedná se o složitý sekvenční obvod, který řídí jednotlivé části a obvody počítače. Toto řízení je prováděno na základě programu, který je uložen v operační paměti a přes interní sběrnici přenesení právě do řadiče. Ten instrukci rozloží na takzvané mikroinstrukce a jednotlivé bity mikroinstrukce následně zasílá na

interní sběrnici. Tyto bity pak řídí všechny další bloky mikroprocesoru, tj. například určují, jakou operaci má provést ALU. [2]

### 2.1.3 Registry

Registry jsou velmi důležitou součástí a v jednom mikroprocesoru jich najdeme hned několik. Kapacita u těchto registrů je obecně velmi malá, pohybuje se od 4 do 128 bitů, nejčastěji používanými jsou 8, 16, 32 a 64-bitové registry. Pro tento velmi rychlý typ paměti se často používají klopné obvody typu D nebo JK. Některé registry jsou uživateli či programátorovi přístupné celé nebo alespoň jejich část což znamená, že může ovlivňovat jejich obsah a označujeme je jako *viditelné*. Druhou skupinou jsou registry neviditelné či nepřístupné, u kterých programátor nemůže jejich obsah nijak ovlivnit. Zde uvedu nejběžnější registry, které může mikroprocesor obsahovat: [3]

#### a) Základní registry přístupné programátorovi

- **universální registry** – tyto jsou určeny pro ukládání dat, která slouží jako operandy při provádění některých instrukcí a jejich velikost se běžně pohybuje o délce jednoho slova.
- **čítač instrukcí** (*Program Counter*) – označován také jako programový čítač – obsahuje adresu instrukce, která má být vyjmuta z hlavní paměti a zpracována mikroprocesorem.
- **ukazatel vrcholu zásobníku SP** (*Stack Pointer*) – obsahuje aktuální adresu vrcholu zásobníku. SP je paměť typu LIFO (LastIn – FirstOut), což znamená, že data, která jsou uložena jako poslední, odcházejí ze zásobníku jako první.
- **příznakový registr CCR** (*Condition Code Register*) – jsou zde uloženy tzv. příznakové bity (angl. flags), které nám poskytují dodatečné informace, např:
  - *OF (Overflow Flag)* – příznak přetečení nastaven na 1, pokud je výsledek operace vyšší než cílová hodnota.
  - *DF (Direction Flag)* – určuje směr při zpracovávání řetězových operací.
  - *IF (Interrupt Flag)* – pokud je nastaven na hodnotu 1, je přerušení povoleno.
  - *ZF (Zero Flag)* – pokud je výsledek nulový, nastaví se na 1.
  - *PF (Parity Flag)* – pokud je hodnota 1, dolních 8 bitů mělo sudý počet „1“.
  - *CF (Carry Flag)* – tzv. příznak přenosu, který slouží k indikaci přenosu mezi čísly o více slovech.

**b) Registry nepřístupné programátorovi**

- **instrukční registr (IR)** – registr je využíván pro ukládání operačního kódu právě prováděné instrukce.
- **adresový registr paměti** – obsahuje adresu místa paměti, s níž se právě pracuje.
- **datový registr paměti** – slouží pro krátkodobé uložení dat přenášených mezi hlavní pamětí a CPU.

**2.2 Paměti**

Paměť je další neoddělitelná součást mikrokontroleru. Slouží jak pro ukládání dat se kterými MCU pracuje, tak pro uložení programu. Pro přehlednost jsou uvedeny všechny i dříve používané typy pamětí.

**2.2.1 ROM**

ROM (Read Only Memory) je typ paměti, do které je obsah nahrán již při výrobě a jednou zapsanou hodnotu již nelze měnit. Tato paměť je určena pouze pro čtení a není závislá na napájení, z toho plyne, že si informaci uchovává i při odpojení napájení. [1]

**2.2.2 PROM**

Jedná se o paměti, které je možné naprogramovat pouze jedenkrát resp. hodnoty bitů se mohou změnit pouze z logické 1 na logickou 0, naopak to není možné. U paměti PROM (Programmable ROM) odpadla nutnost programovat obsah přímo ve výrobě a byla zde možnost programování až u zákazníka za pomoci speciálního přístroje. [1] [11]

**2.2.3 EPROM**

EPROM (Erasable Programmable ROM) je mazatelná za pomoci ultrafialového záření a poté lze opětovně naprogramovat. Počet takto provedených mazání a znovu naprogramování je ovšem omezen na desítky. [1]

**2.2.4 EEPROM**

EEPROM (Electrically Erasable Programmable ROM) již dovoluje elektricky mazat svůj obsah a opětovně programovat. Lze ji programovat buď pomocí programátoru, nebo přímo v aplikaci. Fyzikální děje jsou shodné pro programování stavu 1 i stavu 0 v jednotlivých buňkách, proto odpadá nutnost mazání celého obsahu, jako je tomu u paměti typu EPROM. Počet mazání a zápisů se u EEPROM pamětí pohybuje v řádech stovek tisíc. [1]



### 2.2.5 FLASH

Jedná se opět o elektricky mazatelné programovatelné paměti. Jejich struktura byla změněna za účelem dosažení co nejvyšší hustoty záznamu tak, že jedna paměťová buňka je představována jedním tranzistorem. Mazání a zápis FLASH je organizován po blocích, tudíž odpadá problém nutnosti vymazání celého obsahu paměti. Stačí pouze před zápisem vymazat příslušný blok. Garantovaný počet přepisů/vymazání každého bloku se udává v řádu stovek tisíc. [11]

### 2.2.6 RAM

RAM (Random Access Memory) je druh polovodičové paměti, ze které lze jak číst, tak do ní zapisovat. Nutno podotknout, že se můžeme setkat i s označení RWM (Read/WriteMemory) a jedná se o totéž. U tohoto typu paměti dochází po odpojení napájení ke ztrátě uložených informací. Výhodou tohoto typu je její rychlost. Podle technologie uchování informace dělíme RAM paměti následovně:

- **Statická RAM (SRAM)** – realizována bistabilním klopným obvodem (např. RS, JK, D, T). Používá se CMOS technologie – malý příkon, krátká přístupová doba. Pro realizaci je nutné použít alespoň dva tranzistory, tudíž je poměr cena/kapacita poměrně vysoká. SRAM jsou využívány pro svou rychlost jako cache paměti.
- **Dynamická RAM (DRAM)** – Je založena na fyzikálním principu nabíjení kondenzátoru, kdy je jedna buňka tvořena kondenzátorem, což je ve srovnání s SRAM levnější. Kondenzátor se v čase samovolně vybíjí, tudíž musí být buňka obnovována. Při čtení se kondenzátor vybije, musí se tedy i po každém čtení obnovit, což zajišťuje speciální obvod. DRAM je používána v modulech DIMM a DDR DIMM. [1]

Z uvedených typů pamětí se dnes většinou využívají FLASH paměti pro program, RAM paměti se díky své rychlosti používají jako paměť dat a paměti EEPROM se stále v mikrokontrolerech využívají taktéž pro paměť dat, které se mají ponechat i po odpojení napájení.

## 2.3 Komunikační rozhraní

Komunikační rozhraní zprostředkovávají spojení a komunikaci mezi MCU a okolím. Mezi ně řadíme rozhraní pro sériovou a paralelní komunikaci.

### 2.3.1 USB

USB (Universal Serial Bus), česky univerzální sériová sběrnice, je v současné době jedno z nejpoužívanějších rozhraní pro připojení periférií, kterých může být až 127. Velkou výhodou je možnost připojení za chodu, tzv. Plug&Play, což znamená, že po připojení začne periférie automaticky fungovat a nemusí docházet k žádným restartům atp. Pro připojení více zařízení je USB vybaveno rozbočovači.

Verze USB 1.1, vydána v roce 1998, umožňovala rychlost přenosu 12 Mb/s. Verze 2.0 umožňuje rychlost přenosu 480 Mb/s a výrazně rozšiřuje škálu použitelných zařízení.

Ke komunikaci používá USB čtyři vodiče – dva slouží k napájení (GND a +5 VDC) a dva vodiče slouží pro přenos dat.

Poslední verze USB 3.0, uváděná na trh v roce 2010, disponuje přenosovou rychlostí až 5 Gb/s. Co je velmi důležité u tohoto standardu, nové verze byly vždy zpětně kompatibilní s verzí předchozí. [4]

Na obrázku č. 2 jsou uvedeny typy konektorů používané pro USB, například vývojový kit Arduino Uno, kterému bude věnována kapitola, využívá konektor USB typu „B“.



Obr. 2: Konektory USB

Komunikace probíhá na základě rámců, které jsou generovány každou 1  $\mu$ s. Každý rámec začíná SOF bitem (Start Of Frame), který slouží jako synchronizační signál. Poté jsou odeslána data, dochází k tzv. transakcím. Transakcí rozumíme výměnu paketů mezi hostitelem a periferním zařízením. Konec transakce je označen bitem EOF (End Of Frame). [4]

### 2.3.2 SPI

SPI (Serial Peripheral Interface) je jednoduché sériové synchronní rozhraní. Toto rozhraní se vyznačuje svou rychlostí a obsahuje ho téměř každý mikrokontroler. Využívá se pro komunikaci mezi řadiči, paměťmi a periferními obvody. Během každého hodinového

impulzu lze současně vysílat a přijímat datový bit, to znamená, že se jedná o synchronní obousměrný přenos. Rychlost tohoto rozhraní se pohybuje v řádech jednotek Mb/s.

SPI rozhraní rozlišuje maximálně jedno hlavní či vedoucí zařízení, tzv. *master*, které řídí jedno nebo více podřízených zařízení, tzv. *slave*. Rozhraní má čtyři povinné signály:

- SCK (Seríál Clock – sériové hodiny) – synchronizační signál, kterým hlavní zařízení vzorkuje jednotlivé bity.
- MOSI (Master Output Slave Input) – data výstupu vedoucího zařízení a vstupu podřízeného zařízení.
- MISO (Master Input Slave Output) – data vstupu vedoucího zařízení a výstupu podřízeného zařízení
- GND (Ground) – společná signálová zem

V případě více podřízených zařízení lze použít signál SS# (Slave Select), kterým vybereme podřízené zařízení.

Data jsou přenášena nejprve bitem s nejvyšší vahou, za ním následují ostatní. Zařízení mohou pracovat s jedním ze čtyř pracovních režimů výměny dat. Režimy, označované jako SPI 0 – SPI 3, se liší polaritou a fází synchronizačních signálů SCK.

Nevýhodou SPI je, že data mohou být přenášena pouze na kratší vzdálenosti. Dalším negativem je nejednotnost způsobů synchronizace (SPI 0 – SPI 3) a s každým výrobcem se může lišit. Proto bývá rozhraní doplněno o konfigurační registry, pomocí kterých se synchronizace nastaví. Nevýhodou je neexistence signálu ACK (acknowledge), který slouží k potvrzení příjmu dat. Díky tomu nemůžeme řídit rychlost přenosu dat v případě, že by některé zařízení nedokázalo data v takové rychlosti zpracovávat, nebo naopak, aby data mohla být vysílána v kratších intervalech. [4] [5]

### 2.3.3 IIC

Sběrnice IIC nebo-li I<sup>2</sup>C (Inter-Integrated Circuit) je synchronní sériová obousměrná sběrnice. IIC používá pouze tři vodiče:

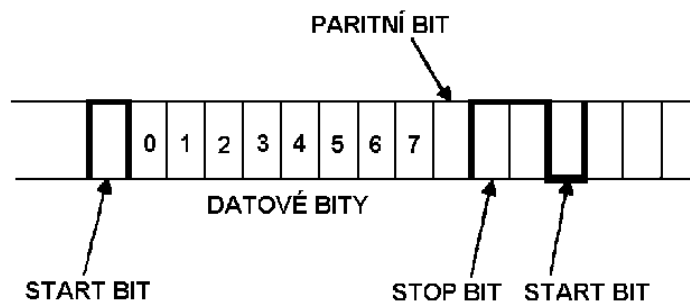
- SDA (Serial Data) – přenos bitů
- SCL (Serial Clock) – zdroj hodinového signálu
- GND (Ground) – společná signálová zem

Z uvedeného vyplývá, že rozhraní IIC umožňuje pouze poloduplexní provoz, což znamená, že můžeme data vysílat, přijímat, ale nelze tyto dvě činnosti provádět zároveň. Lze použít i na delší vzdálenosti, řádově metry. IIC nepotřebuje signál SS#, kterým u SPI probíhal výběr podřizovaného zařízení, jelikož každé zařízení na sběrnici má vlastní adresu. Velkou výhodou je, že IIC má přesně definovaný komunikační protokol, který umožňuje existenci více vedoucích zařízení a dále má jednoduchý mechanismus detekce kolizí. [4] [6]

### 2.3.4 UART

UART (Universal Asynchronous Receiver-Transmitter) je univerzální sériové asynchronní rozhraní, které umožňuje plně duplexní provoz. Je hojně využíváno ke komunikaci mezi různými zařízeními, jako např. mikrokontrolery, GSM, GPS moduly nebo PC. Pokud je rozhraní konfigurováno pro synchronní přenos, označuje se jako USART a taktéž podporuje plně duplexní provoz.

Asynchronní režim sériového přenosu je orientován na byty, což znamená, že nejmenší přenesená jednotka je jeden byte. Přenos každého bytu začíná start bitem, který přijímači signalizuje začátek přenosu. Průběh signálu u UART začíná start bitem v logické úrovni 0. V klidu je signál v logické 1. Poté následují datové bity a za nimi volitelný paritní bit. Přenos je ukončen stop bitem. Startovací bit dalšího bytu je odeslán kdykoliv po stop bitu předchozího bytu. Mezi přenosy tedy mohou být libovolně dlouhé pauzy. Startovací bit, který má vždy přesně definovanou hodnotu (logická nula), umožňuje snadno synchronizovat přijímač pomocí signálu z vysílače. Z toho vyplývá podmínka, že přijímač i vysílač musí pracovat s předem definovanou přenosovou frekvencí. Základními signály, které postačují pro komunikaci pomocí UART, jsou dva - Tx (Output) a Rx (Input). [1]



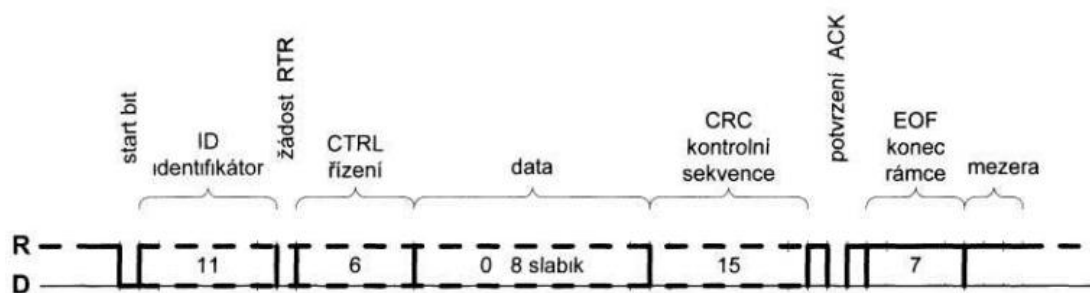
Obr. 3: Asynchronní přenos dat

### 2.3.5 CAN

Sběrnice CAN (Controller Area Network) je vhodná především pro složitější a náročnější úlohy. Původně byla sběrnice vyvinuta pro účely průmyslové automatizace, v současné době se ale stává standardním prvkem mikrokontrolerů.

Sběrnice CAN nezná dělení zařízení na *master* a *slave*, ale používá při komunikaci protokol CSMA/CD. Jde o protokol s tzv. detekcí kolizí. Zařízení, které vysílá, zároveň sleduje provoz na sběrnici, zda nezachytí jiné vysílání, které by mohlo s jeho vysíláním kolidovat. Pokud takové vysílání zařízení zachytí, pozastaví své vysílání a začne opět vysílat po náhodné době.

Komunikace je zahájena start bitem v logické úrovni 0. Následuje 11bitový nebo 29bitový identifikátor. Tento identifikátor, v případě zjištěné kolize, určuje priority jednotlivých vysílání stanic. Čím menší hodnotu identifikátor má, tím vyšší prioritu označuje. Za identifikátorem následuje RTR (Remote Transmission Request) bit, který nabývá stavů R (log. 1) nebo D (log. 0). Stavem D se sděluje, že data budou vysílána, stavem R budou data přijímána. Vysílání má prioritu před příjmem. Poté jsou zasílána data, byl-li RTR = D, pokud byl RTR = R, tak je délka dat nulová. Bit s nejnižší vahou jde jako první. Poté následuje 16bitový CRC kód, který kontroluje předchozí bity. Za CRC kódem následuje potvrzení správného příjmu dat ACK bitem (acknowledge). Poté už následuje jen označení konce rámce EOF (End Of Frame) a povinná mezera o délce alespoň tří taktů. Graficky je tento průběh zobrazen na obrázku č. 4.[1]



Obr. 4: Průběh signálu CAN

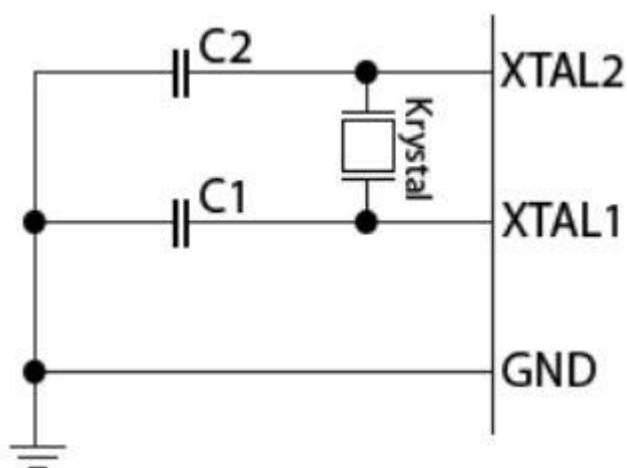
## 2.4 Periferní obvody

### 2.4.1 Zdroj hodinového signálu

Generátory hodinových impulsů či taktů jsou nezbytné pro funkci mikrokontroleru. Na základě těchto taktů pracuje každá součástka MCU. Hodinový signál lze vyrobit dvěma způsoby.

První možností je RC oscilátor. Ten bývá např. u MCU AVR od Atmelu zabudován přímo v pouzdře, tedy je interní, ale není to pravidlem. Tento interní RC oscilátor může být již kalibrovaný na určitou frekvenci, anebo může být laditelný, tzn., že je nutné teprve frekvenci změřit a poté RC oscilátor kalibrovat. Zda je oscilátor kalibrován či ne, najdeme ve specifikacích jednotlivého typu MCU.

Druhou možností je použití externího piezoelektrického krystalu. Ten se používá tehdy, vyžaduje-li aplikace přesnější časování a příklad jeho zapojení je na obrázku č.5.



Obr. 5: Zapojení externího krystalového oscilátoru

Tyto obvody, kterých mívá MCU hned několik, jsou velmi důležité i proto, že každá součást MCU pracuje jinou rychlostí, tudíž je nevyhnutelné použít nějakého zařízení, které tyto jednotlivé části MCU synchronizuje a ustálí veškeré signály před dalším zpracováním. S tím souvisí další problém. Jelikož každý obvod pracuje jinou rychlostí, obsahuje MCU tzv. děliče kmitočtu, které generovaný signál dělí, popř. násobí.

### 2.4.2 Čítače a časovače

Obvody čítačů a časovačů patří k jedním z nejčastěji používaným součástím mikrokontroleru. Jsou používány ke zpracování časových funkcí, ke generování signálů s přesným časováním nebo k synchronizaci mezi programem a vnějšími událostmi.

Obvody mohou být částečně nahrazeny programově, hlavně v levnějších variantách MCU, ale tím snižujeme výkonnost procesoru, jelikož ten je mnohem více zaměstnáván zpracováváním časových funkcí.

Čítání a časování je tedy prováděno jedním obvodem, který může pracovat v různých režimech a je řízen pomocí RC oscilátoru nebo pomocí krystalu. Krystal je vhodný tam, kde nám záleží na přesnosti generovaného kmitočtu. [1]

#### 2.4.2.1 Režim čítání

Jedná se o nejjednodušší režim, kdy jsou na vstup čítače přiváděny impulzy z vnějšího zařízení, generátoru a tyto vstupní impulzy jsou počítány. Aktivní hranou může být náběžná hrana, sestupná hrana anebo obě. Dále lze zpravidla volit směr čítání dopředu nebo dozadu. [1]

#### 2.4.2.2 Režim časování

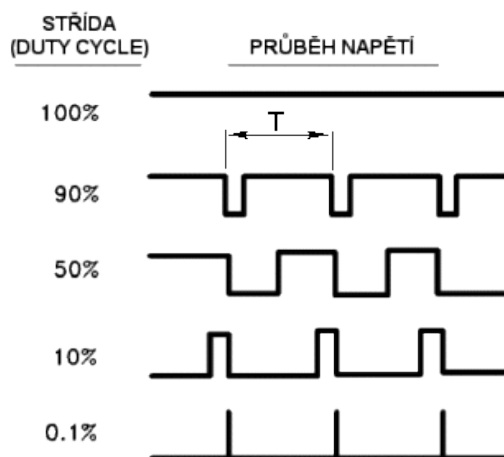
Časovače jsou označovány jako *timer* a obsahují tytéž obvody jako čítače. Timer umožňuje přesně odměřovat čas. Pracují tak, že je jim v inicializační části programu nastavena určitá hodnota. Při dopočítání do nastavené hodnoty, resp. do nuly je vyvoláno přerušení a s ním podprogram, který chceme na základě časování vykonat. Výhodou časovačů je, že je můžeme spustit, pozastavit a opět pustit, vynulovat, nastavit novou hodnotu nebo jeho aktuální hodnotu porovnávat. [1]

### 2.4.3 PWM

PWM (Pulse Width Modulation), pulzní šířkový modulátor, je obsahem téměř všech mikrokontrolerů a jedná se o funkci časovače, ale ne všechny vstupní a výstupní piny tuto funkci podporují. V datasheetu každého MCU údaje o tom, který pin je funkcí PWM vybaven, najdeme.

PWM je jednoduchý a vhodný pro přenos signálů i na delší vzdálenosti. Je vhodný pro řízení otáček elektromotorů, elektromagnetů, snímače teploty atp. Signál PWM má pevně danou periodu  $T$ , ve které se mění délka impulzu. Takový poměr mezi délkou impulzu a

délkou mezery, v jedné periodě, označujeme jako střídu. Střídu vyjadřujeme buď poměrem např. 1:1, 2:1 atp. nebo procentuálně, jak je naznačeno na obrázku č. 6.



Obr. 6: Průběh pulzní šířkové modulace

#### 2.4.4 A/D převodník

Analogově-digitální převodníky slouží k převodu analogových veličin (např. světlo, zvuk, teplota) na digitální signál. 8bitové MCU obsahují zpravidla jeden A/D převodník, který mívá ale větší počet vstupů, které jsou voleny vstupním multiplexerem (Input MUX na obr. 7). Hlavním ukazatelem, od kterého se odvíjí i jeho cena, je rozlišení převodníku, která se udává v bitech, a dále jeho rychlost. U mikrokontroleru mnohdy postačují 8 a 10bitové převodníky, používají se ale i 12 či 16tové převodníky. Čím citlivější převodník volíme, tím bude samotný výstup z převodu přesnější. Doba převodu se pohybuje v řádech stovek nanosekund.





### 2.4.5 Watchdog

Watchdog (Watchdog Timer - WDT) je nezávislý obvod založen na principu časovače s nastavitelnou hodnotou. Tento obvod může být realizován interně, tedy obsažen přímo v pouzdře mikrokontroleru, anebo jako samostatný integrovaný obvod. Jeho úkolem je hlídat správný běh a funkci programu a je často realizován tak, že po spuštění již nejde vypnout.

Watchdog funguje tak, že pokud přeteče předem definovanou hodnotu, resetuje mikrokontroler. Aby k resetu nedošlo, musíme zajistit jeho vynulování (inicializaci) před dosažením časového intervalu. To provedeme buď externím signálem, nebo znovu zapsáním hodnoty do řídicího registru watchdogu a nebo speciální instrukcí. První možnost platí pro externí watchdog, další dvě pro integrovaný watchdog. Pokud se v programu vyskytne chyba, která může být způsobena zacyklením programu, neošetřeným kombinacím vstupních dat, chyba při přenosu po sériové lince, vlivem elektromagnetického rušení atp., watchdog nebude vynulován, dojde k přetečení a následnému resetování mikrokontroleru. Povolení funkce watchdog lze v příslušném registru MCU. [8]

### 3 ZPŮSOBY PROGRAMOVÁNÍ

V této kapitole jsou uvedeny nejpoužívanější a nejdostupnější způsoby a metody, jakými lze program nebo zavaděč nahrát do mikrokontroleru, naprogramovat ho.

V první části jsou popsány rozhraní k tomuto určené. V dalších částech jsou pak popsány možnosti, jak a v čem program napsat, jak otestovat jeho správnost a následně jeho uložení do paměti mikrokontroleru.

#### 3.1 Programovací a ladicí rozhraní

Programovací a ladicí rozhraní jsou prvním prostředkem, který je nezbytný pro programování, nahrávání a ladění programů a bootloaderů (zavaděčů) do MCU.

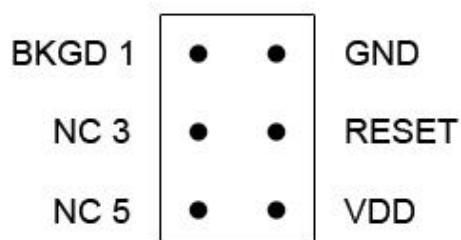
Bootloader (česky zavaděč) je program, který má dva úkoly. Bootloader buď přímo spouští uživatelský program anebo naslouchá na komunikační lince, zda nebude poslán nový program. V případě, že žádný program nepřichází, MCU pracuje podle toho stávajícího. Pokud je poslán nový program, bootloader jej přijímá z komunikační linky a zapisuje do paměti MCU. [16]

Programovací a ladicí rozhraní jsou nezbytná pro nahrání bootloaderu do paměti mikrokontroleru. MCU jsou dostupné samozřejmě včetně bootloaderů, takže při prvních krůčcích není znalost těchto rozhraní nezbytná. Pokročilejší uživatelé by však už tato rozhraní mohli využít pro nahrávání upravených či vlastních bootloaderů.

Pro upřesnění, bootloadery v MCU nejsou nezbytně nutné a nahrávat programy do paměti lze i bez nich, ovšem pak jsou zapotřebí dražší programátory.

##### 3.1.1 BDM

BDM (Background Debug Mode) je programovací a ladicí rozhraní umožňující obousměrnou komunikaci pomocí jednoho datového vodiče, které je využíváno u MCU firmy Freescale. BDM rozhraní nám umožňuje prostřednictvím BDC (Background Debug Controller) ladit v reálném čase. Můžeme pozastavit program a kontrolovat průběh jeho jednotlivých příkazů, můžeme pomocí něj přistupovat k RAM, FLASH nebo EEPROM paměti. Zapojení pinů je uvedeno na obrázku č. 8.



Obr. 8: Piny BDM rozhraní

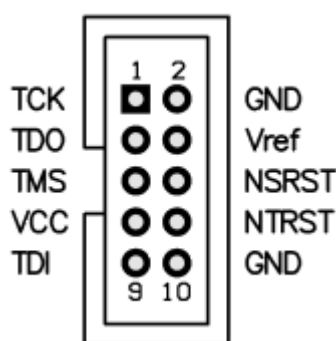
Pin BKGD umožňuje obousměrnou komunikaci. Dále je rozhraní doplněno o RESET a piny VDD a GND určené pro napájení. NC (no connected) jsou neaktivní piny.

### 3.1.2 JTAG

JTAG (Joint Group Action Group) je synchronní paralelní rozhraní definováno standardem IEEE 1149.1. Účelem tohoto rozhraní je testování programů běžících na různých zařízeních, v našem případě mikrokontroleru, to znamená, že obvody vyhovující normě JTAG umožňují, aby byly do mikrokontroleru nahrány instrukce, data a nějakým způsobem vyhodnocen výstup. Pro komunikaci používá pět signálů:

- TCK (Test Clock) – synchronizační signál sériových dat o frekvenci až 16 MHz
- TMS (Test Mode Select) – výběr testovacího režimu
- TDI (Test Data Input) - vstup pro data, nejnižší bit je přenášen jako první
- TDO (Test Data Output) – výstup pro data
- TRST (Test Logic Reset) – resetovací signál rozhraní

Tyto signály tvoří port TAP (Test Access Port), přes který je mikrokontroler připojen k testovacímu zařízení. Port TAP se může lišit počtem pinů, na obrázku č. 9 je zobrazen 10pinový port. Rozhraní JTAG se používá například u MCU firmy Atmel nebo Microchip. [4]



Obr. 9: JTAG konektor [23]

### 3.1.3 ISP

ISP (In-System Programming) je obecné označení techniky programování MCU, která umožňuje programování zařízení přímo v aplikaci. Jako programování technikou ISP je možno označit jak programování pomocí programovacích a ladících rozhraní (JTAG, BDM), tak i programování s využitím bootloaderu, kdy se program do MCU přenáší např. přes sériové rozhraní UART. [24]

Zkratka ICSP (In-Circuit Serial Programming) značí techniku, která definuje přenášení dat pomocí pěti vodičů. Tato technika byla implementována u MCU PIC firmy Microchip zhruba od roku 1992, ale v současné době se využívá například i u MCU od Atmelu. ICSP rozhraní umožňuje programovat jak paměť dat, tak i vnitřní EEPROM mikrokontroleru, aniž bychom museli MCU z aplikace vyjmát a vkládat do jiného zařízení, programátoru, a pak zase zpět do původní aplikace. Tato vlastnost logicky značně usnadňuje a urychluje samotné ladění a vývoj aplikace. [18]

Pro komunikaci rozhraní ICSP vyžaduje pět vodičů (označení pinů u MCU PIC):

- $V_{PP}$  – programovací napětí, pokud je přivedeno na pin, zařízení přejde do programovacího módu
- PGC (nebo ICSPCLK) – synchronizační signál
- PGD (nebo ICSPDAT) – pin pro obousměrný datový přenos
- $V_{DD}$  - +5V
- $V_{SS}$  – GND

[19]

U MCU od Atmelu se označení pinů liší a je následující:

- +5V
- GND
- SCK – synchronizační signál
- MISO – Master In Slave Out
- MOSI – Master Out Slave In
- RESET

[9]

ICSP je pouze jednou z mnoha modifikací ISP. Existují i další metody jako například IAP nebo UISP. Pro rozsah této práce je dostačující znalost modifikace ICSP, kterou obsahuje jak deska Arduino, tak mikrokontrolery firmy Microchip.

### 3.2 Simulátory

Simulátory slouží k otestování programu, aniž bychom potřebovali reálný mikroprocesor či mikrokontroler. Učit se programovat s pomocí simulátorů má výhody, nehledě na to, že takovýchto simulátorů existuje poměrně hodně, tudíž je z čeho vybírat. Simulátor je vhodné použít tehdy, nemá-li začínající uživatel potřebné znalosti z oblasti elektrických obvodů a tím pádem může snadno dojít ke zničení MCU. Velkým plus je to, že odladěné programy jsou schopny po přenesení do mikrokontroleru resp. vývojového kitu plnohodnotně pracovat.

V současné době je téměř pravidlem, že jsou simulátory součástí vývojových prostředí, tzv. IDE, to znamená, že pomocí jednoho softwaru můžeme psát program, přeložit do strojového kódu, debugovat a odladit jej a také rovnou nahrát do MCU. Jako příklad vestavěných simulátorů budou uvedeny softwary PICAXE Programming Editor od firmy Microchip a CodeWarrior od firmy Freescale u příslušných MCU.

### 3.3 Programátory

Tato zařízení zprostředkovávají komunikaci mezi PC a mikrokontrolerem. Programátory mohou fungovat dvěma způsoby:

- a) Mikrokontroler se vsune do patice umístěné na programátoru. Ten je spojen s PC pomocí sériového nebo paralelního portu. Do MCU se přenesení program a MCU se vloží zpět desky aplikace.
- b) MCU se do programátoru nezasunuje, ale zůstane umístěn v desce, která je opatřena některým rozhraním, např. JTAG, ICSP nebo BDM. Programátor se pomocí tohoto rozhraní připojí k desce s MCU a druhým konektorem se pomocí sériové nebo paralelní linky (RS 232C, USB) připojí k PC. Z uvedeného plyne, že programátor je zde ve funkci jakéhosi prostředníka mezi PC a deskou či kitem s MCU. Na obrázku č. 10 je uveden programátor pro MCU s architekturou AVR, kde je zprostředkováno spojení PC s aplikací pomocí JTAG rozhraní a komunikace s PC pomocí konektoru pro sériový přenos.



## **II. PRAKTICKÁ ČÁST**



V praktické části budou testovány tři vybraná zařízení s mikrokontrolery. Konkrétně se jedná o vývojový kit Arduino Uno s mikrokontrolerem ATmega328, dále pak samostatný mikrokontroler PICAXE 08M2 a posledním zařízením pro srovnání byl vybrán vývojový kit M68EVB908GB60 od firmy Freescale, který je mimo jiné využíván při výuce na naší univerzitě.

Tato tři zařízení byla vybrána na základě toho, aby se jejich základní vlastnosti a způsoby programování co nejvíce lišily. Důvodem je, aby byla zabráněna co možná nejširší oblast programování mikrokontrolerů.

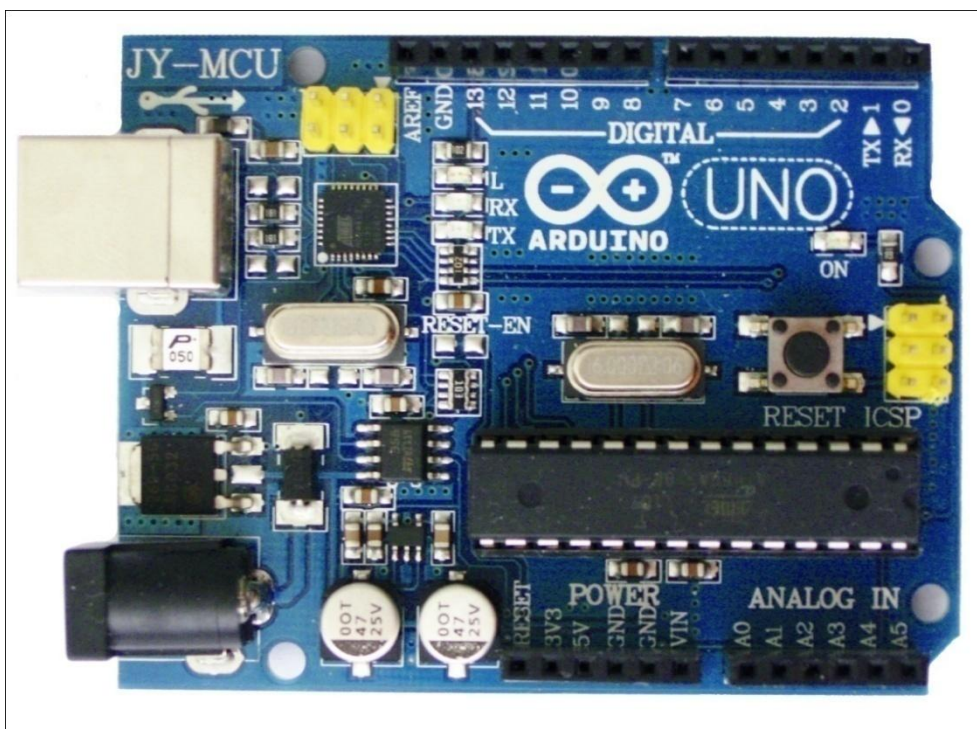
U každé testované platformy je úvod a základní informace o původu, vlastnostech a parametrech. Dále jsou popsány kroky pro jejich oživení a uvedení do chodu tak, aby bylo s nimi možné dále pracovat. V další kapitole je popsáno vývojové prostředí tzv. IDE a také programovací jazyk, který bude pro danou platformu nejvhodnější. Závěrem každé kapitoly je schéma zapojení pro správný běh programu a je uveden samotný srovnávací program s komentáři.

Závěrem praktické části je zhodnocení a srovnání testovaných zařízení, na základě kterého bude doporučena nejvhodnější platforma pro začátečníka.

## 4 ARDUINO UNO

Arduino je italská open-source platforma pro testování jak softwaru, tak hardwaru. Je určena nejenom pro kutily a umělce. V současné době se začíná Arduino využívat i v menších sériových výroбах. Arduino představuje komplexní nástroj, který umožňuje vše od samotného vývoje a testování programů až po umístění v cílové aplikaci.

Arduino je výjimkou mezi ostatními vývojovými kity. Většina vývojových kitů je osazena mikrokontrolery od té samé firmy tzn., že vývojový kit slouží na jednu stranu jako testovací prostředek. Na stranu druhou je v tomto případě vývojový kit jakousi přehlídkou vlastností a toho, co MCU dokáže a čeho je schopen. Arduino je výjimka v tom, že využívá mikrokontrolery od firmy Atmel a žádné MCU Arduino neexistují. To je výhoda v určité nezávislosti a nezájatosti vůči vlastnímu výrobku. Další výhodou je, že se Arduino stává velmi oblíbenou platformou a jeho výrobě se v různých modifikacích věnují další firmy, z čehož plyne budoucnost a další vývoj. U spousty vývojových kitů, které se dříve využívaly, se vývoj již zastavil, u Arduino to v současné době nehrozí a stále se vyvíjí nové modely, jak přímo Arduino, tak příbuzné a odvozené kity založené na této platformě. V této práci je věnována kapitola zatím poslednímu modelu, který byl uveden na trh v roce 2011 – Arduino Uno.



Obr. 11: Arduino Uno

## 4.1 Specifikace kitu Arduino Uno

- Mikrokontroler Atmel ATmega328
  - 8-bit CPU AVR, operační frekvence max. 20 MHz
  - 32KB FLASH paměti
  - 1024KB EEPROM paměti (čtení a zápis s EEPROM library)
  - 2KB SRAM paměti
  - 2x SPI
  - 1x UART
  - 1x I<sup>2</sup>C
  - 1x ICSP
  - 3x časovač
  - 6x PWM
  - 8 kanálový A/D převodník, rozlišení 10 bitů
  - 1x analogový komparátor
  - kalibrovaný RC oscilátor
  - 32kHz RTC (Real Time Clock)
- Napájení
  - 5V USB, 7 – 12V externí adaptér nebo baterie, mezní hranice 6-20V
  - max. 40mA SS na jeden I/O pin
  - max. 50mA SS pro pin 3,3V
- 14x digitální I/O pin
  - 0,1 - sériová linka (Tx, Rx)
  - 2,3 – externí přerušení
  - 3, 5, 6, 9, 10, 11 – PWM
  - 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) - SPI (s knihovnou SPI library)
  - 4, 5 – I<sup>2</sup>C (s knihovnou Wire library)
  - 13 – LED dioda
- 6x vstupní analogový pin

Vývojový kit Arduino Uno obsahuje dále MCU ATmega16U2, pomocí kterého je realizován USB-UART převodník. Lze tedy propojit desku a PC pomocí USB linky. Pokud by tímto obvodem nebylo Uno vybaveno, museli bychom použít externí převodník, který by umožnil nahrávat program přes sériovou linku.

## 4.2 Arduino IDE

Arduino IDE je jednoduché vývojové prostředí, určeno speciálně pro vývojové kity Arduino. Toto prostředí je zdarma a open-source, takže vývojář má k dispozici zdrojový kód a může jej modifikovat. Je naprogramováno kompletně v jazyku Java, což má obrovskou výhodu v nezávislosti na operačním systému. Java bez problému běží jak na Windows, Linuxu tak i Mac OS. Zda je systém 32 nebo 64 bitový nehraje roli. Na konci roku 2011 vyšla první stabilní verze Arduino IDE 1.0.

Funkce tohoto IDE jsou omezeny na nezbytné minimum. Umožní nám program přeložit a odeslat do MCU, neobsahuje debugger. Na obrázky č. 12 je zobrazen panel, který máme k dispozici se všemi dostupnými funkcemi.



Obr. 12: Hlavní nabídka Arduino IDE

Pole *File* obsahuje mimo klasické nabídky otevření a uložení programu, také položku *Examples*. Zde jsou napsány již předpřipravené odladěné ukázkové programy, které lze otevřít, zkompileovat a nahrát do MCU. Máme k dispozici například programy pro práci s digitálními, analogovými vstupy a výstupy, pro práci s reproduktory, displeji, paměti EEPROM, ukázky sériové komunikace mezi Arduinem a PC, servomotory, krokovými motory nebo ukázkové programy pro bezdrátovou komunikaci.

Pole *Edit* nám umožňuje běžné možnosti jako kopírovat obsah, vkládat jej a vyhledávat fráze ve zdrojovém kódu. Pole *Sketch* umožňuje kompilovat program a vkládat připravené knihovny, tzv. Library. Důležitou položkou je *Tools*. *Auto Format* nám upraví text do

stanovených pozic a odstraní přebytečné mezery, tím výrazně zpřehlední zdrojový kód, a tudíž to není nutné dělat ručně. *Serial Monitor* spustí okno, kde můžeme sledovat sériovou komunikaci mezi Arduinem a PC. Dále pak pomocí *Board* nastavujeme, jaký typ Arduino desky programujeme. V *Programmer* nastavujeme, jakým způsobem MCU programujeme. Poslední nabídkou položky *Tools* je *Burn Bootloader*, která slouží pro nahrání bootloaderu.

#### 4.2.1 Vytvoření a přenesení programu do MCU

V Arduino IDE je tento proces velmi jednoduchý. Stačí pouze spustit Arduino IDE a v položce *Tools > Board* vybrat typ desky, kterou programujeme, v tomto případě Arduino Uno.

Jakmile je program hotový, zkompilujeme jej pomocí *Sketch > Verify/Compile* nebo pomocí klávesové zkratky CTRL+R. Jakmile kontrola programu a kompilace proběhne úspěšně, v zelené dolní liště se objeví hláška *Done compiling*. V případě chyby je okno oranžové a jsou vypsány čísla řádků, kde se chyba vyskytla s krátkým popisem. Poté již stačí pouze odeslat program *File > Upload* (CTRL+U) a přenos je ukončen hláškou, opět v zeleném okně, *Done uploading*.

### 4.3 Wiring

Pro programování Arduina byl tvůrci vyvinut jazyk Arduino Programmable Language, který vychází z jazyka nazvaného Wiring. Wiring, podobající se jazyku C, byl vytvořen pro podobný vývojový kit jako Arduino, obecně lze ale říci, že Wiring je programovací framework pro mikrokontrolery. Jako alternativy lze samozřejmě použít assembler nebo jazyk C/C++.

Hlavním cílem vzniku tohoto jazyka bylo vytvoření jednoduché a maximálně přehledné syntaxe, což je pro začátečníka směřodonné. Pro názornost uvedu příklad jednoduchého programu, kterým pravidelně rozsvěcujeme a zhasínáme diodu. Na obrázku č. 13 jsou uvedeny dva zdrojové kódy, jejichž výsledkem je naprosto totožné chování. Vlevo je napsaný program pomocí jazyku Wiring. Kód psaný jazykem C je uveden vpravo.

<pre> 1 void setup() { 2   pinMode(13, OUTPUT); 3 } 4 5 void loop() { 6   digitalWrite(13, HIGH); 7   delay(1000); 8   digitalWrite(13, LOW); 9   delay(1000); 10 }</pre>	<pre> 1 #include &lt;avr/io.h&gt; 2 #include &lt;util/delay.h&gt; 3 4 // Nastaví bit 'b' bajtu 'B' na log. nulu. 5 #define LOW(B, b) (B &amp;= ~_BV(b)) 6 // Nastaví bit 'b' bajtu 'B' na log. jedničku. 7 #define HIGH(B, b) (B  = _BV(b)) 8 9 // Pin 13 je na 5. bitu portu B. 10 #define P13 (5) 11 12 void setup() { 13   // Všechny piny portu B budou výstupní. 14   DDRB = 0xFF; 15 } 16 17 /** 18  * Hlavní smyčka programu 19  */ 20 static void loop() { 21   LOW(PORTB, P13); 22   _delay_ms(500); 23   HIGH(PORTB, P13); 24   _delay_ms(500); 25 } 26 27 int main(void) { 28   setup(); 29 30   for (;;) 31     loop(); 32 33   return 0; 34 }</pre>
---	--

Obr. 13: Zdrojový kód psaný pomocí jazyku Wiring a C

U jazyka Wiring je zřejmá základní konstrukce, která se skládá ze dvou funkcí - *setup* a *loop*. Funkce *setup* je volána při startu programu. Inicializují se zde proměnné, módy pinů, potřebné knihovny atp. Tato funkce je spuštěna pouze jedenkrát, a to při zapnutí nebo resetu Arduina. V tomto konkrétním případě je ve zdrojovém kódu definován pin 13 jako výstup. [13]

Funkce *loop* je smyčka, ve které běží samotný program a na základě obsažených funkcí a různých podmínek MCU resp. vývojový kit pracuje. [14]

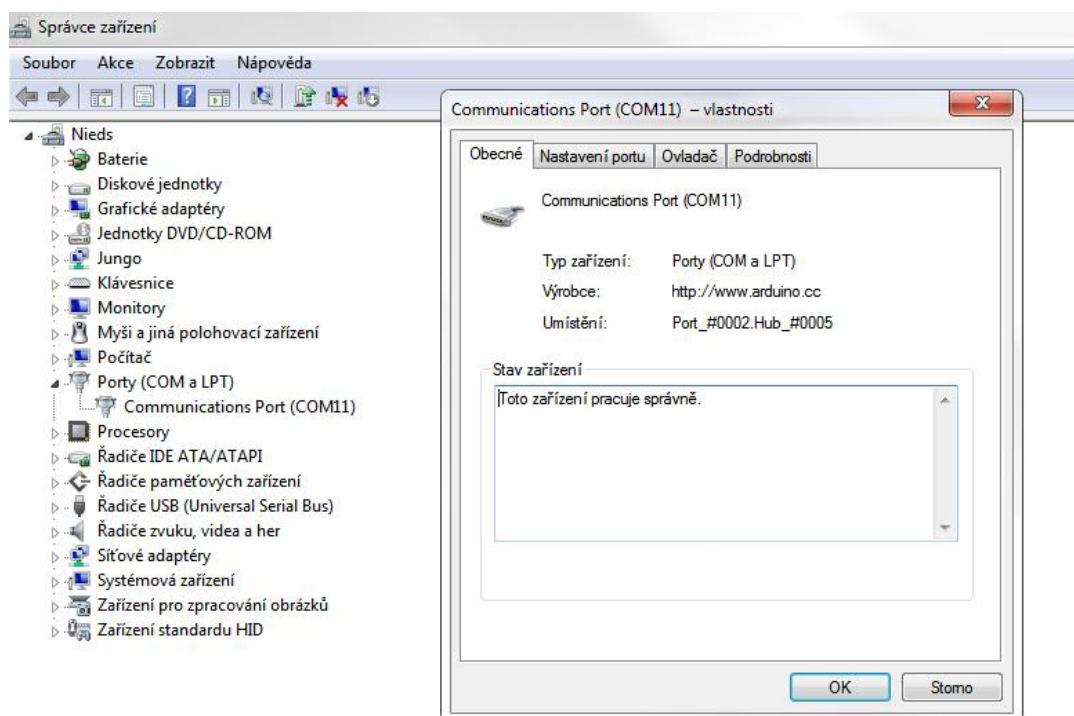
V programu uvedeném na obrázku č. 13 obsahuje smyčka *loop* časové a výstupní funkce, které v pravidelném časovém intervalu mění na pinu číslo 13 logickou 0 a logickou 1.

Program vpravo, psaný v jazyce C, je na první pohled o něco složitější a objemnější. Pro jeho psaní již nebyl použit software Arduino IDE a tudíž musí program obsahovat direktivy (*#include* a *#define*), které jsou uvedeny na prvních řádcích. Při použití Arduino IDE to není nutné, jelikož to tento software dělá automaticky na pozadí. Dále je nutné blíže znát hardware, který programujeme. V tomto případě konkrétně potřebujeme znát port,

který je označen jako pin 13. To samé platí pro označení vstupních a výstupních portů MCU. Z uvedeného vyplývá, že pro začátečníka je z hlediska přehlednosti a vyšší jednoduchosti zdrojového kódu vhodnější jazyk Wiring.

#### 4.4 Zprovoznění kitu Arduino Uno

- Stažení potřebného SW nejlépe z oficiální stránek Arduina. Stáhneme balíček, který obsahuje jak ovladače pro komunikaci s PC, tak samotné vývojové prostředí Arduino IDE.
- Připojíme Arduino Uno k PC pomocí USB kabelu.
- Instalace ovladačů pro Windows.
  - *Start > Ovládací panely > Správce zařízení > Další zařízení > Arduino Uno*
  - klikneme pravým tlačítkem a zvolíme *Aktualizovat software ovladače*, najdeme stažené soubory a vybereme složku *arduinoIDE\drivers*.
  - Po úspěšné instalaci by se nám mělo Arduino Uno objevit mezi porty.

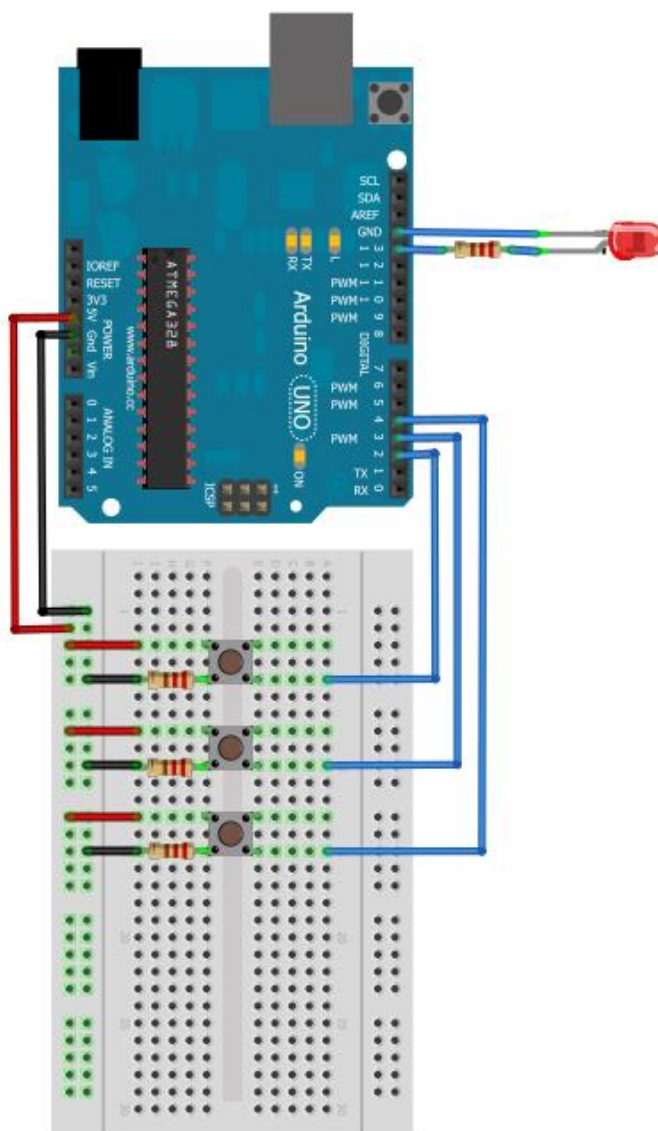


Obr. 14: Vlastnosti portu pro Arduino



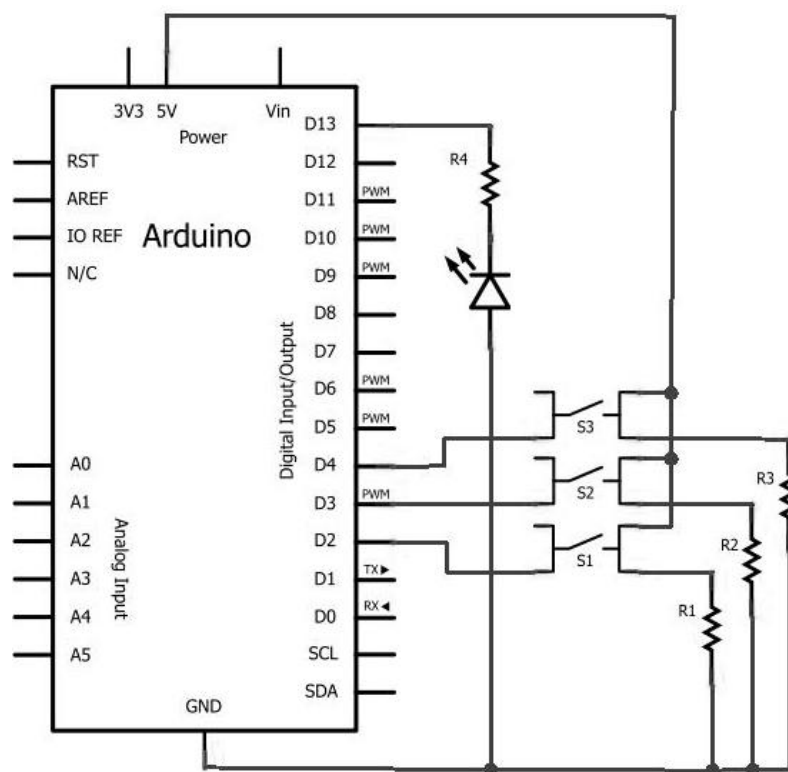
## 4.5 Schéma zapojení

Na obrázku č. 15 je grafické schéma zapojení pro jednoduchý srovnávací program, na obrázku č. 16 pak elektrotechnické schéma zapojení. Jsou použita tři tlačítka, kdy je každé připojeno jedním pinem na napájení 5V a druhým pinem přes odpor na zem (GND). Odporů R1, R2 a R3 mají hodnotu 10k $\Omega$ . Třetí pin je připojen na příslušný vstup vývojové desky, který je deklarován v programu. U programu pro Uno jsou to vstupy 2, 3 a 4. Směr tlačítek je zleva doprava a je vyznačen zelenou barvou na nepájivém poli. Zapojení je doplněno o signalizační diodu, která je připojena na zem a druhým koncem na pin číslo 13 přes odpor 330 $\Omega$ .



Obr. 15: Grafické schéma zapojení desky Arduino Uno





Obr. 16: Elektrotechnické schéma zapojení desky Arduino Uno

## 4.6 Srovnávací program pro Arduino Uno

Pro každou platformu je vytvořen program, který bude plnit totožnou funkci. Rozdíl bude v jednotlivých platformách a také v jazycích, v kterých budou napsány.

Jednoduchý srovnávací program bude mít za úkol kontrolovat stisk tří tlačítek a v případě, že bude některé stisknuto, bude následovat signalizace pomocí LED diody a zároveň bude odeslána zpráva do PC. Program pro Arduino Uno je psán ve vývojovém prostředí Arduino IDE jazykem Wiring a je uveden na obrázku č. 17.

```

// PROGRAM, KTERÝ REAGUJE NA STISK TLAČÍTEK.
// V PŘÍPADĚ STISKU TLAČÍTKA JE ZASLÁNA ZPRÁVA PROSTŘEDNICTVÍM USB
// POČET BLIKNUTÍ DIODY ODPOVÍDÁ OZNAČENÍ TLAČÍTKA
// BUTTON1 = 1X BLIKNUTÍ; BUTTON2 = 2X BLIKNUTÍ; BUTTON3 = 3X BLIKNUTÍ

// označení PINŮ, na které jsou přivedena TLAČÍTKA a LED dioda
int button1 = 2;    //BUTTON1 NA PINU 2
int button2 = 3;    //BUTTON2 NA PINU 3
int button3 = 4;    //BUTTON2 NA PINU 4
int ledPin = 13;    //LED NA PINU 13

//na základě proměnné STAV program vyhodnocuje, zda bylo stisknuto tlačítko
int stav = 0;

//SETUP je funkce, která je volána při startu. Inicializuje proměnné, definujeme v ní módy pinů...
void setup() {
    pinMode(ledPin, OUTPUT); //definice LED jako VÝSTUP
    pinMode(button1, INPUT); //tlačítka jako VSTUP
    pinMode(button2, INPUT);
    pinMode(button3, INPUT);
    Serial.begin(9600);      //ZAHÁJENÍ KOMUNIKACE S PC PŘES USB, RYCHLOST 9600 bps (bitů/s)
}

//podprogram blik, který provede zapnutí/vypnutí LED po půl vteřině
void blik() {
    digitalWrite(ledPin, HIGH); //NATAVENÍ PINU 13 NA LOG. 1
    delay(500);                 //PRODLEVA 0,5 S
    digitalWrite(ledPin, LOW);  //NATAVENÍ PINU 13 NA LOG. 0
    delay(500);                 //PRODLEVA 0,5 S
}

//hlavní smyčka programu, která čeká na stisk TLAČÍTKA
void loop()
{
    stav = digitalRead(button1);    //DETEKCE STISKU TLAČÍTKA 1
    if (stav == HIGH) {             //POKUD JE STISKNUTO
        Serial.println("Stisk_1!"); //POŠLI PŘES SÉRIOVOU LINKU USB ZPRÁVU O JEHO STISKU
        blik();                     //A ZÁROVEŇ BLIKNI DIODOU
    }

    stav = digitalRead(button2);
    if (stav == HIGH) {
        Serial.println("Stisk_2!");
        blik();
        blik();
    }

    stav = digitalRead(button3);
    if (stav == HIGH) {
        Serial.println("Stisk_3!");
        blik();
        blik();
        blik();
    }
}

```

Obr. 17: Srovnávací program pro Arduino Uno

## 5 PICAXE

PICAXE je v poslední době oblíbenou platformou. Původně bylo PICAXE navrženo pro výukové účely do škol, postupně se ale rozšířilo do široké oblasti použití. Jedná se o mikrokontrolery PIC firmy Microchip. PICAXE znamená, že mikrokontrolery PIC jsou již opatřeny bootloaderem, tudíž mohou být opět programovány pomocí sériové linky a není potřeba nákladnějších programátorů. [20]

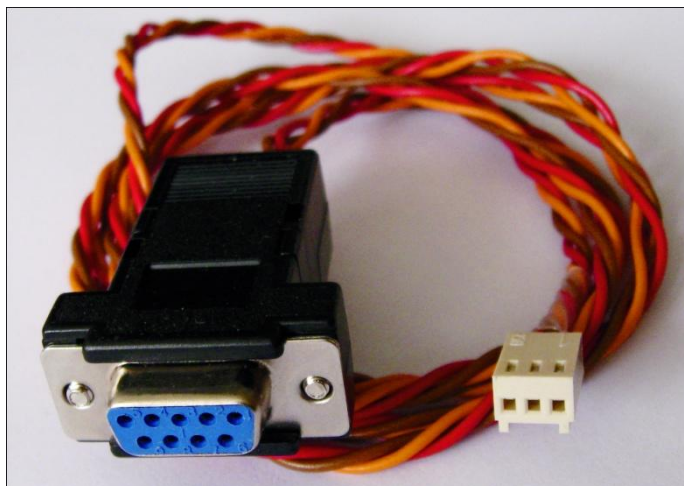
V této práci bude testován konkrétně typ PICAXE 08M2, který se programuje pomocí jazyka BASIC. Jedná se o malý MCU s osmi vývody, který byl uveden na trh v roce 2011 a jeho cena se pohybuje do sta korun. Cena použitého programovacího kabelu se pohybuje kolem 70 Kč.

V případě, že by byl z jakéhokoliv důvodu tento MCU nedostatečný, máme na výběr další a výkonnější varianty. Ty se od sebe liší zejména počtem vývodů, pracovní frekvencí, velikostmi pamětí nebo vyšším počtem vstupů s AD převodníkem. V tabulce číslo 1 jsou jednotlivé dostupné typy srovnány.

Tab. 1: Srovnání základních vlastností MCU PICAXE

	Typ PICAXE						
	<b>08M2</b>	<b>14M2</b>	<b>18M2</b>	<b>20M2</b>	<b>20X2</b>	<b>28X2</b>	<b>40X2</b>
Paměť (B)	2048	2048	2048	2048	4096	4096	4096
RAM (B)	128	512	256	512	256	1280	1280
Proměnné Byte (B)	28	28	28	28	56	56	56
Počet I/O Pinů	6	12	16	18	18	22	33
ADC/Touch Pins	3	7	10	11	11	16	27
Max. Freq. (MHz)	32	32	32	32	64	64	64
Serial In/Out	Ano	Ano	Ano	Ano	Ano	Ano	Ano
IR In/Out	Ano	Ano	Ano	Ano	Ano	Ano	Ano
I2C	Ano	Ano	Ano	Ano	Ano	Ano	Ano

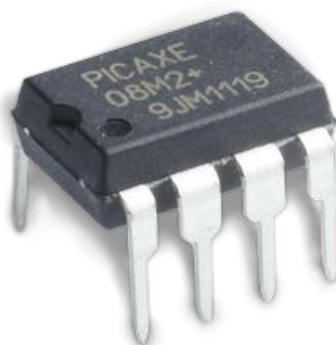
Pro mikrokontrolery PICAXE také existují vývojové desky, jejichž ceny se pohybují v řádech stovek korun. V této práci využita nebude. PICAXE 08M2 bude testováno na nepájivém poli a s PC bude MCU propojen pomocí programovacího kabelu s 9-ti pinovým CANON konektorem a převodníkem USB-RS 232.



Obr. 18: Programovací kabel pro PICAXE

## 5.1 Specifikace MCU PICAXE 08M2

- Mikrokontroler PICAXE 08M2
  - 8-bit PICmicro, operační frekvence max. 32 MHz
  - 2048B paměti
  - 256B EEPROM
  - 128B RAM
  - 1x I<sup>2</sup>C
  - 1x PWM
  - 3x A/D převodník
- Napájení
  - externě 4,5 - 5V



Obr. 19: PICAXE 08M2 [20]

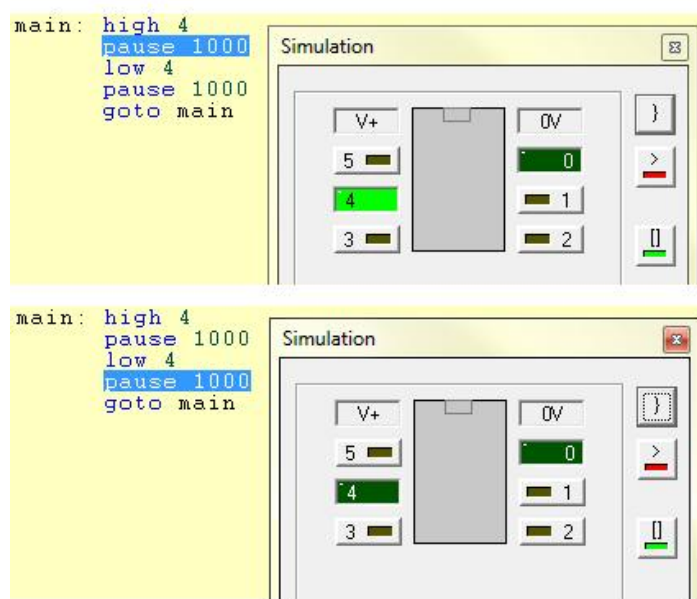
## 5.2 Zprovoznění MCU PICAXE 08M2

Pro zprovoznění tohoto příkladu nám postačí MCU PICAXE 08M2, který již obsahuje bootloader, tudíž není potřeba žádného programátoru. Bootloader nám zajistí správnou komunikaci mezi PC a samotným MCU. Dále je použit software PICAXE Programming Editor určen právě pro tyto MCU. Jelikož je komunikace mezi 08M2 zprostředkovávána přes sériový port, nikoliv USB, není zapotřebí doinstalování jakýkoliv driverů či ovladačů. Pro spojení s PC je použit univerzální programovací kabel, který je z jedné strany opatřen 9pinovým konektorem Canon a na straně druhé jsou vyvedeny 3 piny, určené pro zem (GND), pro příjem (Rx) a vysílání dat (Tx). Tento kabel je zapotřebí při nahrávání programu, při samotné funkci již není důležitý. Pokud však program obsahuje příkazy, které zasílají zprávy přes sériovou linku do PC, je pochopitelně nutný. V případě, že by kabel nebyl přítomný, nebude hlášena žádná chyba a program běží dále. Stejně to bude probíhat v případě, kdy program obsahuje příkazy, kdy naslouchá na sériové lince, zda jsou zadávány znaky z klávesnice. V takovém případě, opět při absenci propojovacího kabelu, bude program normálně běžet, ale nebude na tyto podněty reagovat.

## 5.3 PICAXE Programming Editor

Programovací prostředí PICAXE Programming Editor je určeno pro platformy Windows, uživatelé Linuxu nebo MacOS mohou využít program AXEpad, který je určen taktéž pro vývoj MCU PICAXE.

Editor má klasický vzhled, který se podobá programům pro Windows. Umožňuje samozřejmé akce jako ukládání, načítání, zakládání nových projektů a tisk. Velkou výhodou tohoto IDE je simulátor, kterým je vybaven. Tím můžeme testovat MCU aniž bychom ho měli fyzicky k dispozici. Na obrázku č. 20 je uveden krátký program, který zajišťuje blikání diodou, připojenou na pin 4, vždy s prodlevou jedné vteřiny.



Obr. 20: Simulátor v PICAXE Programming Editor

PICAXE Programming Editor nám dále umožňuje debugovat, kontrolovat správnost syntaxe, nahrávat program a mazat paměť připojeného MCU. Je zde také terminál, kde můžeme sledovat sériovou komunikaci.

### 5.3.1 Vytvoření a přenesení programu do MCU

Při spuštění PICAXE Programming Editor nám vyskočí okno, kam zadáme základní údaje o tom, jaký typ PICAXE budeme programovat a na kterém portu je PICAXE připojeno. V případě, že nám toto okno při spuštění nenajede (máme možnost vyskakování tohoto okno po startu zakázat), najdeme ho v horní liště, v položce *Options*.

Je-li program hotov stiskneme tlačítko *Syntax*, které zkontroluje správnost syntaxe programu, vypíše nám zda vše proběhlo v pořádku a informuje nás o velikosti zdrojového kódu. Poté program odešleme pomocí tlačítka *Program (F5)*.

## 5.4 BASIC

Mikrokontrolery PICAXE jsou programovány pomocí vyššího programovacího jazyku BASIC (Beginner's All-purpose Symbolic Instruction Code). V případě programování mikrokontrolerů PIC se jedná o PICAXE BASIC Language. Konstrukce tohoto jazyka je podstatně jednodušší, než například u Wiringu. Je to dáno především tím, že zdrojový kód neobsahuje žádné složené či kulaté závorky u funkcí a to značně snižuje vizuální objem zdrojového kódu, což zákonitě zvyšuje přehlednost kódu a pro začínajícího programátora

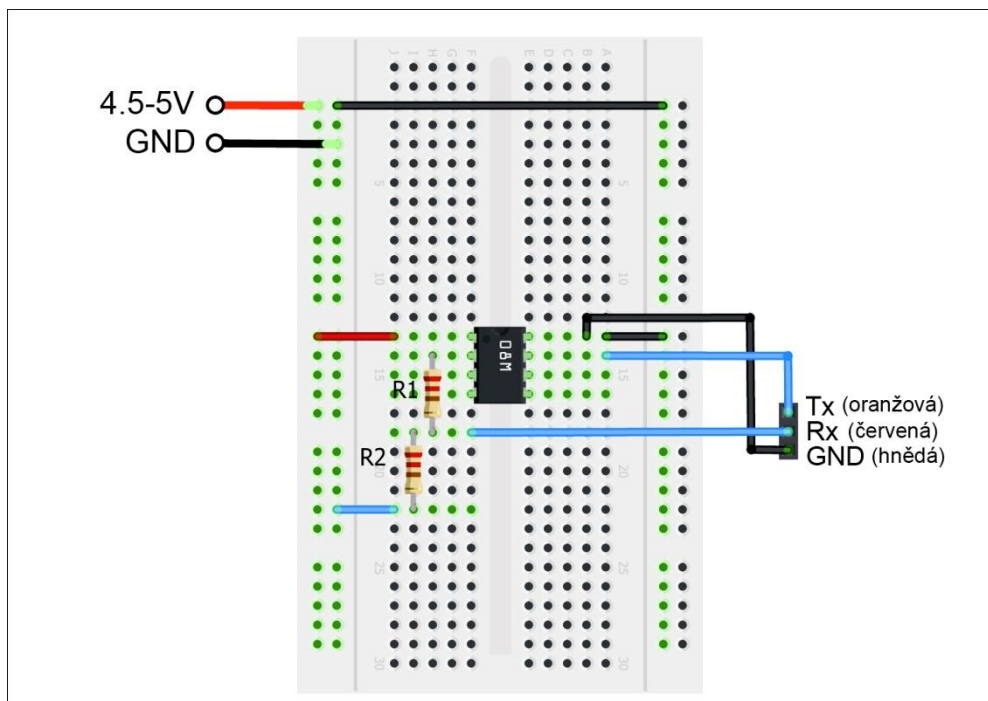
je to výhoda a další z faktorů, na základě kterého si může vybrat platformu, na které bude programovat a vyvíjet.

## 5.5 Schéma zapojení

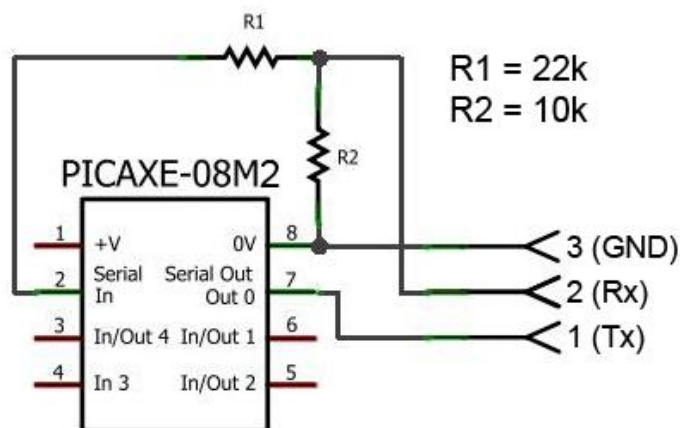
Zde budou uvedeny dva způsoby zapojení. První schéma je určeno pro přenesení programu z PC do MCU pomocí sériové linky. Druhé schéma odpovídá potřebnému zapojení pro srovnávací zdrojový kód psaný v jazyce BASIC.

### 5.5.1 Schéma zapojení pro nahrávání programu

Pro nahrání programu do MCU byl původně zvolen univerzální programovací kabel pro všechny typy PICAXE, který je na jedné straně osazen 9pinovým Canon konektorem a na jeho druhé straně jsou vyvedeny 3 piny (GND, Tx a Rx). Dále byl použit notebook, který neobsahoval sériový port RS 232, proto bylo nezbytné použití USB-COM převodníku, konkrétně se jednalo o Prolific USB to RS 232. Ten bohužel nefungoval správně a musel být použit starší PC s rozhraním RS 232, kde již probíhala komunikace bezproblémově. Na obrázku č. 21 je uvedeno schéma zapojení pro nahrání programu pomocí RS 232. Odpor  $R1 = 22k$  a  $R2 = 10k$ .



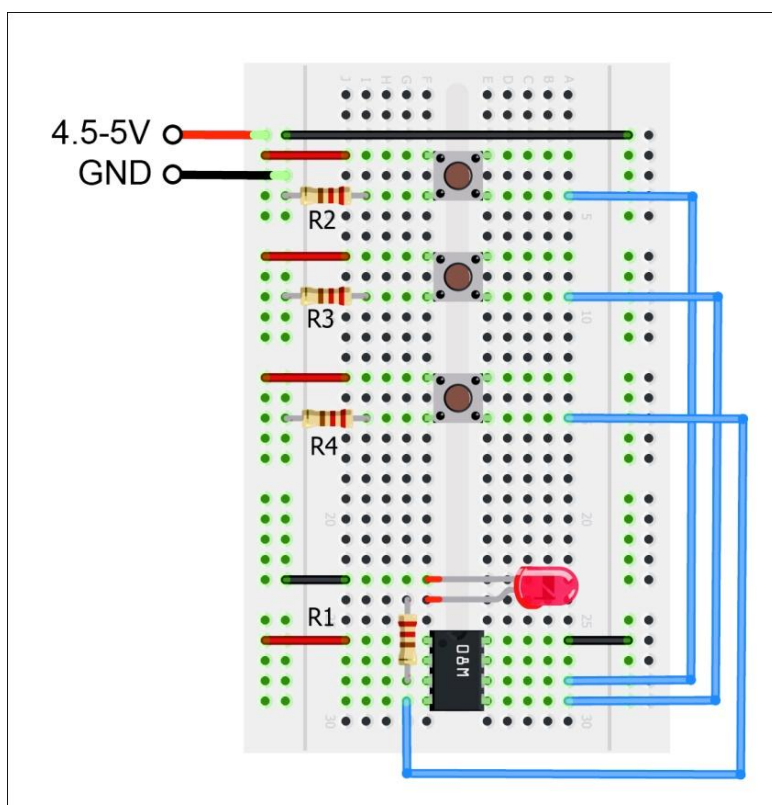
Obr. 21: Grafické schéma zapojení pro programování MCU PICAXE 08M2



Obr. 22: Schéma zapojení pro programování MCU PICAXE 08M2

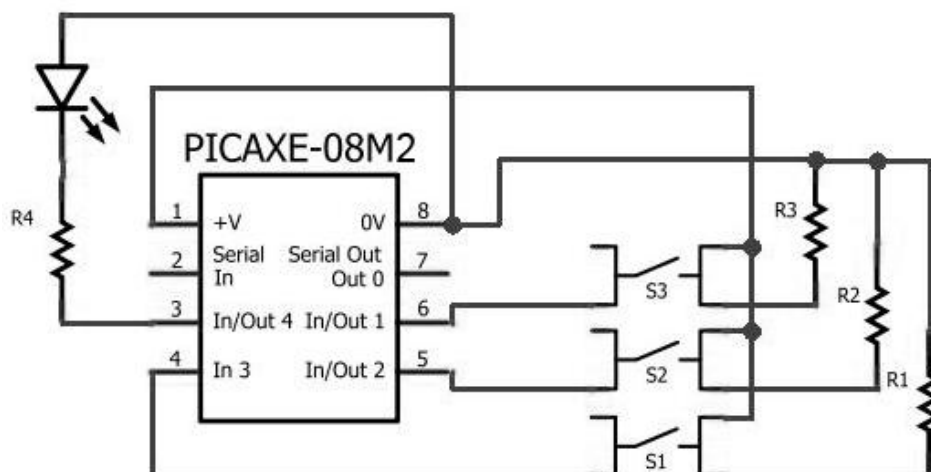
### 5.5.2 Schéma zapojení pro běh programu

Na obrázku č. 23 je uvedeno schéma zapojení, na kterém je testován srovnávací program psaný v BASICU. Odpor  $R1 = 330\Omega$  a odpory  $R2, R3, R4 = 10k\Omega$ . Směr zapojení tlačítek je zleva doprava a je označen zeleně.



Obr. 23: Grafické schéma zapojení pro běh zkušebního programu





Obr. 24: Schéma zapojení pro běh zkušebního programu

## 5.6 Srovnávací zdrojový kód

Na obrázku č. 24 je uveden srovnávací program, napsaný v jazyce BASIC pro MCU PICAXE 08M2. Jeho základní částí je návěští *main*, kde se vyčkává na stisk tlačítka. Program cyklicky kontroluje jednotlivá tlačítka, a když není žádné stisknuto, skočí zpět na návěští *main* a funkce se opakuje. Dojde-li ke stisku tlačítka, vykoná se příslušný podprogram. Pokud bylo tedy stisknuto například první tlačítko, provede se skok na návěští *blik 1*. Zde je první instrukce *gosub*, která provede podprogram *blik* a po jeho vykonání se pokračuje na další instrukci. Podprogram *blik* obsahuje instrukce *high*, *pause*, *low*, *sertxd* a *return*. *High* nastaví logickou 1 na pinu 4 a rozsvítí diodu, pomocí funkce *pause* vteřinu vyčkáme a funkcí *low* nastavíme logickou 0 a diodu zhasneme. Funkce *sertxd* vyšle po sériové lince zprávu do PC. Podprogram je ukončen instrukcí *return*, která zajistí návrat z podprogramu.

```
;PROGRAM REAGUJÍCÍ NA STISK TLAČÍTEK
;PICAXE 08M2
;BASIC

;ve smyčce main se neustále vyčkává na stisk tlačítka
main:
    if pin1 = 1 then blik1 ;pokud je stisknuto,skoč na blik1
    if pin2 = 1 then blik2
    if pin3 = 1 then blik3
    goto main ;vrať se zpět na návěští main

;podprogram, který 1x blikne diodou
blik1:
    gosub blik ;fce gosub vyvolá podprogram blik
    srtxd("Stisk tlačítka č. 1!")
    goto main

;podprogram, který 2x blikne diodou
blik2:
    gosub blik
    gosub blik
    srtxd("Stisk tlačítka č. 2!")
    goto main

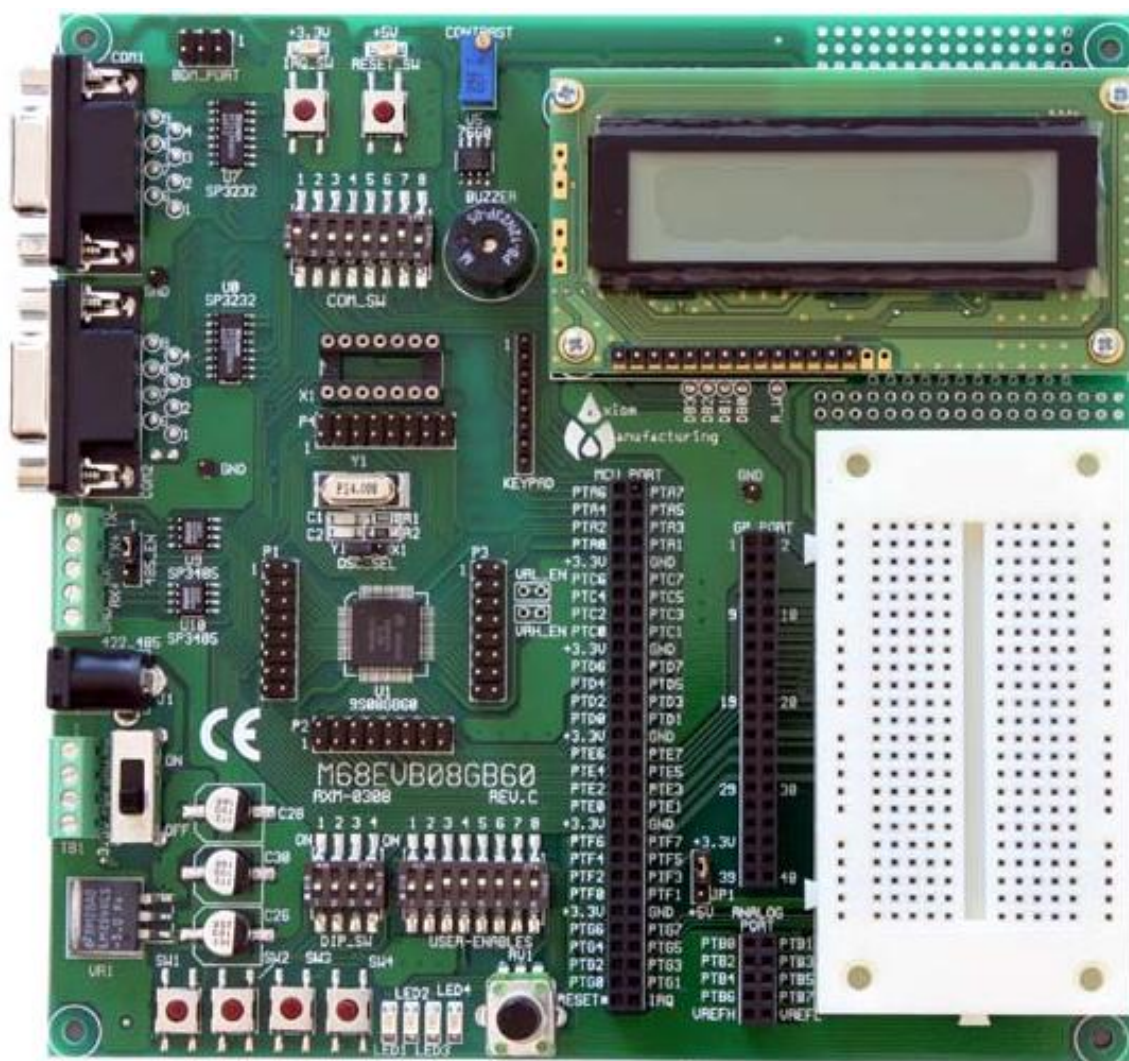
;podprogram, který 3x blikne diodou
blik3:
    gosub blik
    gosub blik
    gosub blik
    srtxd("Stisk tlačítka č. 3!")
    goto main

;podprogram, který rozsvítí diodu
blik: high 4
    pause 1000
    low 4
    pause 1000
    return ;funkce return ukončí podprogram
```

Obr. 25: Srovnávací program pro PICAXE 08M2 v jazyce BASIC

## 6 FREESCALE M68EVB08GB60

Tento vývojový kit určen pro výukové účely, pro vývoj a ladění aplikací a je osazen mikrokontrolerem Freescale M9S08GB60. Deska integruje kromě MCU také LCD displej, 4 tlačítka, 4 LED diody a kontaktní nepájivé pole. To má výhodu v tom, že již není potřeba tyto komponenty dokupovat samostatně a je zaručena jejich kompatibilita, na druhou stranu se tím cena vývojové desky výrazně zvyšuje.



Obr. 26: Vývojový kit Freescale M68EVB908GB60

## 6.1 Specifikace vývojového kitu

- Mikrokontroler M9S08GB60
  - 8bit CPU HCS08, max. frekvence 40MHz
  - 60kB FLASH paměti
  - 4kB RAM paměti
  - 1x SPI
  - 2x SCI
  - 1x I2C
  - 10-bit A/D převodník, 8 kanálový
  - BDM ladicí rozhraní
  - 2x časovač
  - 56 vstupně/výstupních pinů na 7 portech (značeny A-G)
  - krystalový a RC rezonátor
- 32kHz nebo 4MHz krystalový oscilátor
- +3.3V a 5V stabilizovaný zdroj
- Sériový port COM1 (RS 232), sériový port COM2 (RS 232, RS 422/485)
- Uživatelské komponenty
  - 4x LED diody (PTF0-3)
  - 4x tlačítko (PTA4-7)
  - 4x DIP přepínač (PTB4-7)
  - 2x16 LED displej (PTG3-7, PTE6-7)
  - Buzzer (PTD0)
- Napájení
  - 9V

[22]

## 6.2 Zprovoznění vývojového kitu

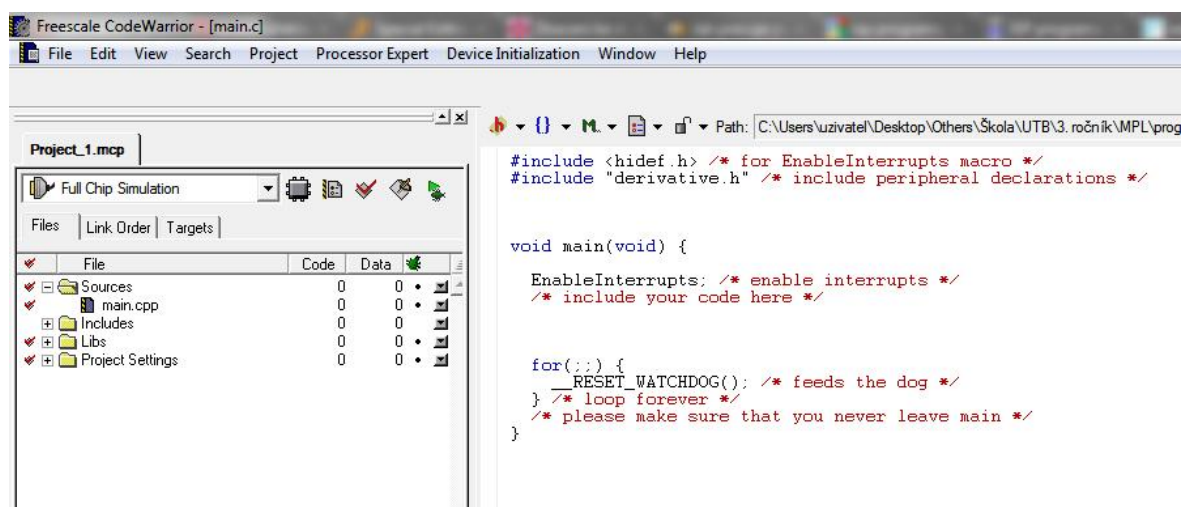
Zprovoznění tohoto vývojového kitu je jednoduchá záležitost. Důvodem je fakt, že komunikace probíhá pomocí sériového portu, podobně jako u testovaného PICAXE 08M2, tudíž není potřeba instalovat jakékoliv ovladače pro převodníky atp. Postačí standardní kabel pro sériovou komunikaci na propojení mezi vývojovou deskou a PC.

## 6.3 Freescale CodeWarrior

CodeWarrior je vývojový prostředek od firmy Freescale pro velmi širokou škálu mikroprocesorů i mikrokontrolerů počínaje 8bitovými rodinami HC08 a HCS08, dále pak RS08 a také 32bitové MCU ColdFire V1 a Flexis. Podporuje také většinu operačních systémů počínaje Windows (32 i 64bitové verze XP, Vista a Windows 7) a Linux (výrobce udává konkrétně Linux Red Hat Enterprise Edition 5.x 32/64 bit a Ubuntu 10.04 32/64bit). Obsahuje základní části jako textový editor pro psaní samotného zdrojového kódu a kompilátor pro překlad. Dále pak obsahuje simulátor a velmi dobrý debugger buď pro softwarové simulace nebo pro ladění přímo v mikrokontroleru resp. v aplikaci pomocí BDM rozhraní. Tento software je dostupný jak v plné verzi, tak také freewarové verzi. Rozdíl je především v tom, že neplacené verze mají omezenou velikost výsledného binárního kódu. Např. verze 10.1 má pro jádro HCS08 omezení 32kB, což ale pro většinu aplikací bohatě postačuje. CodeWarrior podporuje jazyky Assembler a C/C++. [21]

### 6.3.1 Vytvoření a přenesení programu do MCU

Při spuštění vývojového prostředí zvolíme *Create New Project*. Dále volíme zařízení a způsob připojení. Zařízení je *HCS08 > HCS08G Family > MC9S08GB60*. Způsob připojení zvolíme při testování na simulátoru *Full Chip Simulation*. Pokud máme vývojový kit fyzicky k dispozici, volíme *HCS08 SerialMonitor*. V dalším kroku volíme jazyk, ve kterém budeme programovat. Na výběr máme Assembler, C a C++. Testovací program je psán v jazyce C. Dále si ještě zvolíme jméno a umístění projektu v PC. Po úspěšném dokončení se nám objeví základní konstrukce programu uvedená na obrázku č. 27.



Obr. 27: Prostředí vývojového softwaru Freescale CodeWarrior

Je-li program hotov, zkompilujeme jej pomocí *Project > Compile (CTRL+F7)*. V případě chyb nebo varování nám CodeWarrior opět vypíše čísla řádků a jednotlivé chyby. Pokud se jedná o varování, ta můžeme ignorovat, program bude fungovat. Nahrání do MCU je zde specifické a postup je následující. Musíme na vývojovém kitu stisknout tlačítko SW4, poté stiskneme RESET a tlačítko SW4 uvolníme. V IDE klikneme na *Debug (F5)* a tím je zahájeno nahrávání programu do MCU. Tlačítka SW4 a RESET jsou na kitu označena.

## 6.4 Jazyk C

Jazyk C je vyšší programovací jazyk strukturovaný do bloků. Základním blokem je funkce a celý program se skládá právě z takovéto posloupnosti bloků, což je ostatně znakem vyšších programovacích jazyků.

Díky strukturovanosti je zdrojový kód přehledný, ovšem ve srovnání s testovanými programovacími jazyky Wiring a PICAXE Basic je o něco složitější a objemnější. Důvodem je větší univerzálnost jazyka C, což má výhody i nevýhody. Jednoznačným plus je lepší přístup k programovanému hardwaru, např. operace s registr. Nevýhodou je již zmíněný větší objem zdrojového kódu a tím pádem menší přehlednost, což je pro začátečníka problém a zákonitě delší doba, kterou bude potřebovat k napsání programu či k porozumění a orientaci v již hotovém programu.

U MCU se před vyvoláním programu v C spouští počáteční modul (tzv. startup), programovaný většinou v assembleru. Tento modul provádí základní inicializaci příslušných registrů MCU, oblasti paměti, ukazatelů zásobníku (Stack Pointer) a vektorů přerušení. Po tomto spouštěcím kódu začíná již program v jazyce C funkcí s názvem *main()*, která je vyvolána startup modulem a tím pádem je funkce *main()* povinná. Dále jsou již vykonávány příkazy podle zdrojového kódu. [26]

## 6.5 Srovnávací zdrojový kód

Z obrázku č. 27 je patrná složitější konstrukce. To je především dáno větší univerzálností a širšími možnostmi tohoto vývojového softwaru a pochopitelně i samotného programovacího jazyku. Je možné programovat v assembleru, C nebo C++.

Základní konstrukce je tvořena dvěma direktivami, dále hlavní částí programu, označenou jako *main* a poslední částí je kontrola správného běhu programu pomocí watchdogu.

Komentáře jsou odlišeny červenou barvou. Na obrázku č. 28 je pak uveden kompletní srovnávací zdrojový kód pro vývojový kit M68EVB908GB60.

Srovnávací zdrojový kód začíná již zmíněnými direktivami *#include*, kterými vložíme do zdrojového kódu definice základních portů a registrů konkrétního mikrokontroleru. Za nimi následuje prototyp funkce *cekej*. Následující funkce *main()* je již hlavní část, kde celý program probíhá. Jako první jsou nastaveny vlastnosti registrů:

- PTADD\_PTADDx – konfigurace pinů řídicího registru A jako vstupní (0) nebo výstupní (1)
- PTFPE – registr pro aktivaci (1) nebo deaktivaci (0) interních pull-up rezistorů
- PTFD – datový registr portu F, při jeho načtení získáme stav pinů portu a zápisem můžeme stav změnit

Následuje povolení přerušení pomocí *EnableInterrupts* a za touto funkcí již píšeme zdrojový kód. Program je vykonáván na základě podmínky *if*, kdy se vyčkává na stisk prvního, druhého nebo třetího tlačítka. V případě stisku některého z těchto tlačítek se vykoná příslušný počet bliknutí. Hodnotě logická 0 v tomto případě odpovídá rozsvícení diody a při logické 1 dioda zhasne. Je to dáno fyzickým zapojením LED diod. Funkce *cekej* je zde z důvodu ochrany proti opětovnému vyhodnocení stisku tlačítka.

Jedinou odlišností od ostatních srovnávacích programů je, že zde není přítomná komunikace mezi PC a vývojovou deskou, tudíž se nevypíše zpráva o stisku příslušného tlačítka na monitoru PC. Je to z důvodu toho, že jazyk C nezná žádnou konkrétní funkci pro přenos po sériové lince, jako u Arduino či PICAXE. Přenos zprávy je samozřejmě možný, ale složitějším způsobem.



```

#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */
#include <stdio.h>
#include "main_asm.h" /* interface to the assembly module */

void cekej(void);
void main(void) {
    PTADD_PTADD4 = 0;    // tlačítko 1
    PTADD_PTADD5 = 0;    // tlačítko 2
    PTADD_PTADD6 = 0;    // tlačítko 3
    PTFPE = 0;           // deaktivace pull-up rezistorů
    PTFDD_PTFDD0 = 1;    // dioda1
    PTFD = 0xFF;
    PTFPE = 0x00;

    EnableInterrupts; /* enable interrupts */
    /* include your code here */
    for(;;) {

        if ( PTAD_PTAD4 == 0 )    //pokud je tlačítko 1 stisknuto
        {
            PTFD_PTFD0 = 0;        //rozsvícení diody
            cekej();                //prodleva
            PTFD_PTFD0 = 1;        //zhasnutí diody
            cekej();                //prodleva
        }

        if ( PTAD_PTAD5 == 0 )
        {
            PTFD_PTFD0 = 0;
            cekej();
            PTFD_PTFD0 = 1;
            cekej();
            PTFD_PTFD0 = 0;
            cekej();
            PTFD_PTFD0 = 1;
            cekej();
        }

        if ( PTAD_PTAD6 == 0 )
        {
            PTFD_PTFD0 = 0;
            cekej();
            PTFD_PTFD0 = 1;
            cekej();
            PTFD_PTFD0 = 0;
            cekej();
            PTFD_PTFD0 = 1;
            cekej();
            PTFD_PTFD0 = 0;
            cekej();
            PTFD_PTFD0 = 1;
            cekej();
        }
    }

    __RESET_WATCHDOG(); /* feeds the dog */
} /* loop forever; please make sure that you never leave this function */

void cekej() {
    unsigned int i;
    for (i = 0; i < 0xFFFF; i++)
        __RESET_WATCHDOG();
}

```

Obr. 28: Srovnávací zdrojový kód pro Freescale M68EVB908GB60



## 7 SROVNÁNÍ TESTOVANÝCH PROSTŘEDKŮ

V této kapitole jsou srovnány nejpodstatnější vlastnosti jednotlivých vývojových prostředků, mikrokontrolerů a vývojových prostředí, na základě kterých byla vyhodnocena nejlepší varianta pro začátečníka.

Tab. 2: Srovnání testovaných zařízení

	Arduino Uno (MCU ATmega328)	PICAXE 08M2	M68EVB908GB60 (MCU M9S08GB60)
Max. frekvence	20 MHz	32 MHz	40 MHz
Počet I/O pinů	20	6	56
EEPROM	1024 kB	256 B	-
RAM	2 kB	128 B	4 kB
FLASH paměť	32 kB	2 kB	60 kB
Velikost srovnávacího programu	2,9 kB (cca 50 řádků)	0,182 kB (cca 45 řádků)	1,5 kB (cca 60 řádků)
Možnost ladění programu	NE	ANO	ANO
SPI	ANO	ANO	ANO
IIC	ANO	ANO	ANO
ISP	ANO	NE	ANO
USB	ANO	NE	NE
Časovač	ANO	ANO	ANO
PWM	ANO	ANO	ANO
A/D	ANO	ANO	ANO
D/A	ANO	ANO	ANO
Rozměr	2	1	3
Současná popularita	1	1	4
Cena	cca 500 Kč	cca 100 Kč	cca 3000 Kč

U položek, kde nelze jednoznačně odpovědět pomocí ANO/NE, byla zvolena stupnice od 1 do 4, kdy nejlepší ohodnocení je 1, hodnocení 4 pak nejhorší.

V tabulce č.2 jsou jako první srovnány základní parametry MCU počínaje jejich maximální pracovní frekvencí a velikostmi jednotlivých druhů pamětí. U velikosti srovnávacího programu jsou údaje, které byly uvedeny po dokončení kompilace příslušným IDE. Nejvíce omezeným prostředkem z tohoto pohledu je PICAXE 08M2, u kterého vyplývá a je uváděn počet řádků zdrojového kódu zhruba v rozmezí 600-1800 řádků. U platforem

Arduino a Freescale je velikost paměti mnohonásobně vyšší, takže odhad počtu řádků není zcela možný. Dále jsou srovnány komunikačních možnosti a možnosti zpracování analogového či digitálního signálu. Jako poslední jsou srovnány jejich rozměry z důvodu přímého použití v cílové aplikaci a také jejich současné oblíbenost mezi vývojáři. Jak Arduino Uno tak PICAXE 08M2 jsou prostředky přivedeny na trh v roce 2011 a těší se mezi vývojáři, programátory a nadšenci velké oblibě. Vývojový kit M68EVB908GB60 se již nevyrábí a zařazen byl pouze za účelem srovnání.

Tab. 3: Srovnání testovaných vývojových prostředí (IDE)

IDE	Arduino IDE	PICAXE Programming Editor	Freescale CodeWarrior
Jazyk	Wiring	BASIC	C/C++, Assembler
Debugování	NE	ANO	ANO
Simulátor	NE	ANO	ANO
Serial Monitor*	ANO	ANO	ANO
Ukázkové zdrojové kódy**	ANO	NE	NE
Přehlednost IDE	1	2	2
Podpora OS***	1	3	3
Přehlednost programu	2	1	2
Cena	ZDARMA	ZDARMA	ZDARMA

V tabulce č. 3 jsou srovnány vlastnosti vývojových prostředí, tzv. IDE a platí zde stejná pravidla jako u tabulky č. 2, to znamená, že pokud nebylo možné hodnotit parametr pomocí ANO/NE, byl ohodnocen číslem v rozmezí 1-4, kde 1 je nejlepší a 4 nejhorší ohodnocení.

\* Serial Monitor označuje funkci, pomocí které můžeme sledovat komunikaci mezi MCU a hostitelským počítačem, v našem případě zprávy o stisku příslušného tlačítka

\*\* Ukázkové zdrojové kódy jsou myšleny programy, které jsou obsaženy přímo v IDE a stačí je pouze nahrát, zařízení fyzicky zapojit, nahrát do MCU a vše bude fungovat. PICAXE Programming Editor ani CodeWarrior takto předepsané programy neobsahují.

\*\*\* Podpora operačního systému je u Arduina IDE perfektní a běží na všech základních platformách (Windows, Linux a MacOS). IDE pro PICAXE a CodeWarrior jiné platformy než Windows nepodporují. Existují ovšem další verze programů, které Linux či MacOS podporují a lze to tímto způsobem obejít. Jelikož ale konkrétní testovaná IDE jiné

platformy OS nepodporují a v případě změny operačního systému je nutné změnit i vývojové prostředí, byla z tohoto důvodu snížena známka.

Po komplexním zhodnocení a zvážení všech pro a proti, se jeví jako nejvhodnější varianta, pro začínajícího programátora či konstruktéra, italská platforma Arduino, v tomto konkrétním případě Arduino Uno s mikrokontrolerem ATmega328 doplněné o MCU ATmega16U2 pro převod USB na COM. Jedinými mínusy u této platformy je absence debuggeru a simulátoru, což lze ale obejít například použitím vývojového prostředí, přímo od firmy Atmel, AVR Studio. Naopak velkým plus je poměrně příznivá cena, rozměr a velmi široká základna lidí, kteří s touto platformou pracují. Další nesrovnatelnou výhodou oproti zbylým dvěma testovaným zařízením je open source licence, pod kterou je Arduino vydáváno. Zkušenější programátoři či konstruktéři si mohou vývojové nástroje, jak samotnou desku tak IDE, přizpůsobovat díky dostupnému zdrojovému kódu.

## ZÁVĚR

Cílem této práce je přiblížit začátečníkům dostupné způsoby a metody, které lze využít při programování mikrokontrolerů. Začátečník by si po přečtení práce měl být schopen vybrat nejvhodnější metodu a zařízení, ať už na základě potřebných vlastností mikrokontroleru nebo na základě předchozích programátorských či hardwarových zkušeností jedince.

V práci byly testovány tři druhy zařízení. Jednalo se o italskou platformu Arduino Uno, dále byl testován mikrokontroler PICAXE 08M2 od firmy Microchip a posledním zařízením pro srovnání byl využit vývojový kit Freescale M68EVB908GB60, který je na naší univerzitě využíván při výuce.

Tato tři zařízení byla vybrána na základě jejich odlišností, aby byla pokryta co největší část této problematiky. Prvním aspektem, podle kterého byly prostředky vybírány, je možnost využití v cílové aplikaci z hlediska jejich velikosti a ceny. Druhým aspektem pak byla odlišnost jazyků, pomocí kterých byly jednotlivé mikrokontrolery programovány. Konkrétně se jedná o jazyky Wiring, BASIC a jazyk C. Bylo přihlédnuto také k současné popularitě prostředků mezi programátory a konstruktéry.

Výsledkem této práce je vzájemné srovnání tří testovaných zařízení. Posuzováno bylo především snadnost zprovoznění, vlastnosti a chování jednotlivých prostředků, zpracování vývojových prostředí a vlastnosti programovacího jazyka, dále pak kvalita datasheetů, literatury a dostupnost dalších informací.

Po komplexním zhodnocení se jeví jako nejvhodnější varianta italská platforma Arduino Uno. Rozhodující bylo především jednoduchost zprovoznění, vlastnosti a chování vývojového kitu, přehlednost a zpracování vývojového prostředí, úspornost a přehlednost programovacího jazyka Wiring. Jedinou nevýhodou této platformy resp. Arduina IDE je absence simulátoru a debuggeru, což lze ovšem obejít jinými způsoby. Vše podtrhuje open source licence, konkrétně Creative Commons Attribution Share-Alike 2.5, pod kterou je software i hardware Arduino vydáván. To má za následek jistotu dalšího vývoje jak samotného Arduina, tak odvozených a příbuzných desek, jejichž vývoji se věnují další firmy.

## ZÁVĚR V ANGLIČTINĚ

The aim of this thesis is to introduce common ways and methods that can be used for programming microcontrollers to beginners. After having read this thesis a beginner should be able to choose the most suitable method and device either based on characteristic of microcontroller or previous programming or hardware experience.

In this thesis three kinds of device were tested. It was an Italian platform Arduino Uno, microcontroller PICAXE 08M2 by Microchip and the last one for comparison was development kit Freescale M68EVB908GB60, which is used at our university.

These devices were chosen due to their differences to cover utmost part of this issue. The first aspect is possible usage in final application based on their size and price. The second aspect was the difference in their programming language, mainly Wiring, BASIC and C. Present popularity of devices among programmers and constructors was also considered.

The output of this work is a comparison of three tested devices. What was considered was mainly simplicity of starting, properties and behaviour of devices, processing development environments, programming language properties, quality of datasheets, literature and accessibility of other information.

After the evaluation the Italian platform Arduino Uno appears as a suitable option. Determinative factors were mainly simplicity of starting, properties and behaviour of a development kit, transparency and design of development environment, clear arrangement of the programming language Wiring. The only disadvantage of this platform resp. Arduino IDE is the absence of simulator and debug mode, but this can be solved by other means. All is validated by open source licence, Creative Commons Attribution Share-Alike 2.5, because hardware and software is based on this licence. This is certainly the proof of further evolution of Arduino and derived and based boards on Arduino, whose development is dealt by other companies.

## SEZNAM POUŽITÉ LITERATURY

- [1] PINKER, Jiří. *Mikroprocesory a mikropočítače*. Praha: BEN - technická literatura, 2008. ISBN 978-80-7300-110-0.
- [2] TIŠNOVSKÝ, Pavel. Jak pracuje počítač?. *Root.cz* [online]. 27.2.2008 [cit. 2012-03-18]. Dostupné z: <http://www.root.cz/clanky/jak-pracuje-pocitac/>
- [3] TIŠNOVSKÝ, Pavel. Pohled do nitra mikroprocesoru. *Root.cz* [online]. 20. 3. 2008 [cit. 2012-04-03]. Dostupné z: <http://www.root.cz/clanky/pohled-do-nitra-mikroprocesoru/>
- [4] GOOK, Michael. *Hardwarová rozhraní: Průvodce programátora*. Brno: Computer Press, a.s., 2006. ISBN 80-251-1019-2.
- [5] TIŠNOVSKÝ, Pavel. Externí sériové sběrnice SPI a IIC. *Root.cz* [online]. 30. 12. 2008 [cit. 2012-04-03]. Dostupné z: <http://www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c/>
- [6] TIŠNOVSKÝ, Pavel. Komunikace po sériové sběrnici I2C. *Root.cz* [online]. 8. 1. 2009 [cit. 2012-04-03]. Dostupné z: <http://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/>
- [7] TIŠNOVSKÝ, Pavel. Architektura VLIW aneb pokus o překonání problémů architektur CISC a RISC. *Root.cz* [online]. 16.8.2011 [cit. 2012-04-06]. Dostupné z: <http://www.root.cz/clanky/architektura-vliw-aneb-pokus-o-prekonani-problemu-architektur-cisc-a-risc/>
- [8] FUKSA, Michal. Obvody watchdog a obvody hlídající napájení mikroprocesoru. Alespoň něco o 8051 [online]. © 2002 [cit. 2012-04-08]. Dostupné z: <http://www.volny.cz/fuksam/povidani/watchdog.htm>
- [9] 8-bit AVR Microcontroller with 4,8,16,32 KBytes In-System Programmable Flash. Atmel [online]. © 2009 [cit. 2012-04-08]. Dostupné z: <http://www.atmel.com/Images/doc8161.pdf>
- [10] ATmega328. Atmel Corporation [online]. © 2012 [cit. 2012-04-12]. Dostupné z: <http://www.atmel.com/devices/ATMEGA328.aspx?tab=parameters>
- [11] TIŠNOVSKÝ, Pavel. Nevolatilní paměti. *Root.cz* [online]. 18.9. 2008 [cit. 2012-04-17]. Dostupné z: <http://www.root.cz/clanky/nevolatilni-pameti/>

- [12] Arduino Uno. ARDUINO [online]. © 2011 [cit. 2012-04-12]. Dostupné z: <http://arduino.cc/en/Main/ArduinoBoardUno>
- [13] SMRŽ, Jan. Bootloader (zavaděč). *JS Homepage* [online]. 2006 [cit. 2012-04-21]. Dostupné z: <http://www.smrz.chrudim.cz/bootloader/>
- [14] Setup. Arduino [online]. 2011 [cit. 2012-04-24]. Dostupné z: <http://arduino.cc/en/Reference/Setup>
- [15] Loop. Arduino [online]. 2011 [cit. 2012-04-25]. Dostupné z: <http://arduino.cc/en/Reference/Loop>
- [16] DOLINAY, Jan. Platforma pro programování mikropočítače Atmel Mega8 s využitím bootladeru: Jednoduchý a levný vývojový kit. 2011.
- [17] Introduction to HCS08 Background Debug Mode. Freescale Semiconductor [online]. 2006 [cit. 2012-04-28]. Dostupné z: [http://www.freescale.com/files/microcontrollers/doc/app\\_note/AN3335.pdf](http://www.freescale.com/files/microcontrollers/doc/app_note/AN3335.pdf)
- [18] In-Circuit Serial Programming™ (ICSP™ ) Guide. U.S.A., 2003. Dostupné z: <http://ww1.microchip.com/downloads/en/devicedoc/30277d.pdf>
- [19] PICkit™ 2 Microcontroller Programmer USER'S GUIDE. U.S.A., 2005. Dostupné z: <http://www.modtronix.com/products/prog/pickit2/pickit2%20datasheet.pdf>
- [20] What is PICAXE?. PICAXE [online]. 2011 [cit. 2012-05-03]. Dostupné z: <http://www.picaxe.com/What-Is-PICAXE>
- [21] CodeWarrior for Microcontrollers (Eclipse IDE) - RS08/HCS08, ColdFire V1, Qorivva 56xx, PX Series, 56800/E DSC, Kinetis. Freescale Semiconductor [online]. ©2004 - 2012 [cit. 2012-05-07]. Dostupné z: [http://www.freescale.com/webapp/sps/site/prod\\_summary.jsp?code=CW-MCU10&tid=CWH](http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-MCU10&tid=CWH)
- [22] M68EVB908GB60: Development Board for Freescale MC9S08GB60 [online]. 2005 [cit. 2012-05-12]. Dostupné z: [http://cache.freescale.com/files/soft\\_dev\\_tools/doc/user\\_guide/M68EVB908GB60UM.pdf?fsrch=1&sr=5](http://cache.freescale.com/files/soft_dev_tools/doc/user_guide/M68EVB908GB60UM.pdf?fsrch=1&sr=5)
- [23] JTAGcable II. Propox [online]. 2002 [cit. 2012-05-14]. Dostupné z: [http://www.propox.com/products/t\\_117.html?lang=en](http://www.propox.com/products/t_117.html?lang=en)

- [24] ČELEDA, Pavel. UISP - AVR In-System Programmer. *Hw.cz* [online]. 26. září 2006 [cit. 2012-05-14]. Dostupné z: <http://www.hw.cz/navrh-obvodu/software/uisp-avr-in-system-programmer.html>
- [25] JTAG Programmer/Debugger for AVR. MDFLY: Your electronics supplier! [online]. © 2012 [cit. 2012-05-14]. Dostupné z: [http://www.mdfly.com/index.php?main\\_page=product\\_info&products\\_id=764](http://www.mdfly.com/index.php?main_page=product_info&products_id=764)
- [26] BURKHARD, Mann. C pro mikrokontrolery. Praha: BEN - technická literatura, 2003, s. 12. ISBN 80-7300-077-6.



**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

ACK	Acknowledge
A/D	Analog/Digital
ALU	Aritmeticko-logická jednotka
BDM	Background Debug Mode
CAN	Controller Area Network
CCR	Condition Code Register
CISC	Complex Instruction Set Computer
CMOS	Complementary Metal Oxide Semiconductor
CPU	Mikroprocesor (Central Processing Unit)
CRC	Kontrolní součet
CSMA/CD	Přístupová metoda pro komunikaci
CU	Řadič (Control Unit)
DRAM	Dynamic Random Access Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
EOF	End of Frame
EPROM	Erasable Programmable Read Only Memory
FPU	Floating Point Unit
GND	Nulový potenciál (Ground)
ICSP	In-Circuit Serial Programming
IDE	Integrated Development Environment
IIC	Inter-Integrated Circuit
IO	Integrovaný obvod
ISP	In-System Programming
JTAG	Joint Group Action Group
LCD	Liquid Crystal Display

MCU	Mikrokontroler
MISC	Minimum Instruction Set Computer
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
MUX	Multiplexer
PROM	Programmable Read Only Memory
PWM	Pulse Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RTC	Real Time Clock
RTR	Remote Transmission Request
RWM	Read Write Memory
RxD	Receive Data
SCK	Serial Clock
SDA	Serial Data
SOF	Start of Frame
SP	Stack Pointer
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TAP	Test Access Port
TxD	Transmit Data
USART	Universal Synchronous Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VLIW	Very Long Instruction Word
WDT	Watchdog Timer

**SEZNAM OBRÁZKŮ**

Obr. 1: Koncepce podle von Neumanna .....	12
Obr. 2: Konektory USB .....	18
Obr. 3: Asynchronní přenos dat .....	20
Obr. 4: Průběh signálu CAN .....	21
Obr. 5: Zapojení externího krystalového oscilátoru .....	22
Obr. 6: Průběh pulzní šířkové modulace .....	24
Obr. 7: Schema A/D převodníku používaného u mikrokontrolerů 8bit AVR .....	25
Obr. 8: Piny BDM rozhraní .....	28
Obr. 9: JTAG konektor [23] .....	28
Obr. 10: JTAG programátor pro AVR [25] .....	31
Obr. 11: Arduino Uno .....	34
Obr. 12: Hlavní nabídka Arduino IDE .....	36
Obr. 13: Zdrojový kód psaný pomocí jazyku Wiring a C .....	38
Obr. 14: Vlastnosti portu pro Arduino .....	39
Obr. 15: Grafické schéma zapojení desky Arduino Uno .....	40
Obr. 16: Elektrotechnické schéma zapojení desky Arduino Uno .....	41
Obr. 17: Srovnávací program pro Arduino Uno .....	42
Obr. 18: Programovací kabel pro PICAXE .....	44
Obr. 19: PICAXE 08M2 [20] .....	44
Obr. 20: Simulátor v PICAXE Programming Editor .....	46
Obr. 21: Grafické schéma zapojení pro programování MCU PICAXE 08M2 .....	47
Obr. 22: Schéma zapojení pro programování MCU PICAXE 08M2 .....	48
Obr. 23: Grafické schéma zapojení pro běh zkušebního programu .....	48
Obr. 24: Schéma zapojení pro běh zkušebního programu .....	49
Obr. 25: Srovnávací program pro PICAXE 08M2 v jazyce BASIC .....	50
Obr. 26: Vývojový kit Freescale M68EVB908GB60 .....	51
Obr. 27: Prostředí vývojového softwaru Freescale CodeWarrior .....	53
Obr. 28: Srovnávací zdrojový kód pro Freescale M68EVB908GB60 .....	56

**SEZNAM TABULEK**

Tab. 1: Srovnání základních vlastností MCU PICAXE .....	43
Tab. 2: Srovnání testovaných zařízení .....	57
Tab. 3: Srovnání testovaných vývojových prostředí (IDE) .....	58