

# **Použití platformy Wiring pro tvorbu programů pro mikropočítače**

The Use of the Wiring Framework for Microcontroller Programming

Pavel Stodulka

---

Bakalářská práce  
2013



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2012/2013

## **ZADÁNÍ BAKALÁŘSKÉ PRÁCE**

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel STODULKA**  
Osobní číslo: **A10061**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **prezenční**

Téma práce: **Použití platformy Wiring pro tvorbu programů pro mikropočítače**

Zásady pro vypracování:

1. **Popište platformu Wiring se zaměřením na softwarové funkce pro ovládání vstupů a výstupů.**
2. **Prostudujte a popište problematiku tvorby softwarových knihoven přenositelných mezi různými mikropočítači.**
3. **Zvolte vhodný mikropočítač z nabídky 8-bitových mikropočítačů Freescale.**
4. **Implementujte vybranou podmnožinu funkcí platformy Wiring pro zvolený mikropočítač.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. PINKER, Jiří. Mikroprocesory a mikropočítače. Praha: BEN – technická literatura, 2004. ISBN 80-730-0110-1.
2. HRBÁČEK, J. Komunikace mikrokontroléru s okolím. 1. vyd. Praha: BEN – technická literatura, 1999, 156 s. ISBN 80-860-5642-2.
3. 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. Praha: BEN, 2003. ISBN 80-730-0077-6.
4. BARRAGÁN H. Wiring [online]. 2013 [cit. 2013-01-28]. Dostupné z: <http://wiring.org.co/>

Vedoucí bakalářské práce:

**Ing. Jan Dolinay, Ph.D.**

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce:

**24. února 2013**


Termín odevzdání bakalářské práce:

**14. června 2013**

Ve Zlíně dne 24. února 2013

  
prof. Ing. Vladimír Vašek, CSc.  
*děkan*



  
prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## **ABSTRAKT**

Práce se zabývá možností přenositelnosti kódu mezi různými mikroočítači. Toto umožňuje platforma Wiring, ale pouze mezi podporovanými mikroočítači ATmega. V teoretické části je popsáno možné programování mikroočítačů, prostředí Wiring, pomocí něhož můžeme mít jeden program fungující na více mikroočítačích. V praktické části je provedena implementace některých funkcí Wiringu pro mikroočítač Freescale, který není Wiringem podporován.

Klíčová slova:

Wiring, mikroočítač, přenositelnost kódu, ATmega, Freescale

## **ABSTRACT**

Work is focused on code portability between different microcontrollers. This allows Wiring platform, but only between supported ATmega microcontrollers. The theoretical part describes the options for programming microcontrollers, Wiring environment with which we have a program running on multiple microcontrollers. In the practical part is described implementation of some Wiring functions for Freescale microcontroller, which is not supported by Wiring.

Keywords:

Wiring, microcontroller, code portability, ATmega, Freescale

Rád bych touto cestou poděkoval mému vedoucímu bakalářské práce, panu Ing. Janu Dolinayovi, Ph.D., za ochotu, odborné vedení, věcné připomínky a dobré nápady při řešení této práce.

**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 MIKROPOČÍTAČ A JEHO PROGRAMOVÁNÍ</b> .....	<b>11</b>
1.1 PROGRAMOVÁNÍ MIKROPOČÍTAČE.....	12
1.1.1 Assembler – jazyk symbolických adres.....	13
1.1.2 C/C++.....	13
<b>2 PŘENOSITELNOST KÓDU</b> .....	<b>14</b>
<b>3 WIRING</b> .....	<b>15</b>
3.1 SPECIFIKACE WIRINGU .....	16
3.2 FRAMEWORK.....	17
3.3 KNIHOVNY .....	17
3.4 VÝVOJOVÉ PROSTŘEDÍ .....	18
3.4.1 Menu .....	19
3.4.2 Textový editor .....	19
3.4.3 Okno zpráv .....	19
3.4.4 Konzole .....	19
3.5 PODPOROVANÉ MIKROPOČÍTAČE .....	19
3.6 PODPOROVANÉ DESKY (VÝVOJOVÉ KITY) .....	20
3.6.1 Parametry kitu Wiring S.....	20
3.6.2 Vývojové kity Arduino.....	20
3.7 ADRESOVÁNÍ PINŮ A PORTŮ .....	21
3.8 PŘENOSITELNOST KÓDU MEZI RŮZNÝMI DESKAMI.....	21
3.9 ADRESÁŘOVÁ STRUKTURA VÝVOJOVÉHO PROSTŘEDÍ.....	22
<b>4 POPIS FRAMEWORKU WIRING</b> .....	<b>23</b>
4.1 OBSLUHA VSTUPŮ A VÝSTUPŮ .....	23
4.1.1 Obsluha digitálních pinů .....	23
4.1.1.1 pinMode() .....	23
4.1.1.2 digitalWrite() .....	24
4.1.1.3 digitalRead() .....	24
4.1.1.4 pullup().....	24
4.1.1.5 noPullup() .....	24
4.1.2 Obsluha digitální portů.....	25
4.1.2.1 portMode() .....	25
4.1.2.2 portRead() .....	25
4.1.2.3 portWrite() .....	25
4.1.3 Obsluha analogových vstupů.....	26
4.1.3.1 analogRead() .....	26
4.1.3.2 analogReference() .....	26
4.1.3.3 analogWrite() .....	27
4.1.3.4 noAnalogWrite().....	27

<b>II</b>	<b>PRAKTICKÁ ČÁST</b> .....	<b>28</b>
<b>5</b>	<b>MIKROPOČÍTAČ FREESCALE HCS08GB60</b> .....	<b>29</b>
<b>6</b>	<b>IMPLEMENTACE PRO HCS08GB60</b> .....	<b>30</b>
6.1	ADRESOVÁNÍ PINŮ A PORTŮ .....	30
6.2	SLOŽKA S08 .....	30
6.3	SLOŽKA FREESCALE .....	31
6.4	SOUBORY POUŽITÉ PŘI PROGRAMOVÁNÍ .....	31
6.4.1	main.cpp .....	31
6.4.2	WMain.cpp .....	31
6.4.3	Wiring.h .....	31
6.4.4	BoardDefs.h .....	31
6.4.5	WDigital.c .....	32
6.4.6	WAnalog.c .....	33
<b>7</b>	<b>ZALOŽENÍ PROJEKTU A UKÁZKOVÝ PROGRAM</b> .....	<b>35</b>
7.1	ZALOŽENÍ PROJEKTU – METODA 1.....	35
7.2	ZALOŽENÍ PROJEKTU – METODA 2.....	37
7.3	UKÁZKOVÝ PROGRAM Č. 1 .....	39
7.4	UKÁZKOVÝ PROGRAM Č. 2 .....	40
	<b>ZÁVĚR</b> .....	<b>41</b>
	<b>ZÁVĚR V ANGLIČTINĚ</b> .....	<b>42</b>
	<b>SEZNAM POUŽITÉ LITERATURY</b> .....	<b>43</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK</b> .....	<b>44</b>
	<b>SEZNAM OBRÁZKŮ</b> .....	<b>45</b>
	<b>SEZNAM PŘÍLOH</b> .....	<b>46</b>

## ÚVOD

V dnešní době je téma přenositelnosti kódu mezi různými počítači nebo platformami velmi skloňované téma. Je stále důležitější, aby program nebo aplikace běžící na jednom stroji fungovala bez problému i na druhém, bez nutnosti provádět změny ve zdrojovém kódu.

Výjimku netvoří ani mikropočítače, u kterých je ale přenositelnost na horší úrovni než u osobních počítačů. Je to hlavně z důvodu použití programovacího jazyka blízcímu se strojovému kódu. Každý výrobce používá jiné instrukce pro tutéž práci, a to přenositelnému kódu neprospívá. Možností je používat vyšší programovací jazyk, dnes velmi rozšířený C. Bohužel ani ten neřeší přenositelnost kódu, ale výrazně tomu napomáhá.

Možností je použít programovací prostředí Wiring. Již dnes nabízí přenositelnost kódu mezi vybranými mikropočítači Atmel ATmega a deskami plošných spojů různých výrobců (Wiring, Arduino, Rouge Robotics nebo BDMicro) osazenými těmito mikropočítači.

Řešení spočívá ve vytvoření funkcí pro práci s mikropočítačem, které bude uživatel používat. Názvy těchto funkcí zůstávají pořád stejné, liší se pouze jejich vnitřní struktura v závislosti na zvoleném mikropočítači. Když tedy budeme chtít program nainstalovat do dvou různých mikropočítačů, stačí jej zkompileovat pro každý zvlášť a vše je hotovo.

Výsledkem této práce je rozšíření Wiringu i pro mikropočítače Freescale HCS08. Jednoduché programy pro práci se vstupy a výstupy napsané ve Wiringu tedy budou fungovat i na mikropočítači Freescale a opačně.

## **I. TEORETICKÁ ČÁST**

## 1 MIKROPOČÍTAČ A JEHO PROGRAMOVÁNÍ

Elektrický obvod je zapojení různých elektrických prvků (rezistor, kondenzátor, cívka, tranzistor aj.) takovým způsobem, aby vykonával námi požadovanou funkci, např. zesílení signálu nebo vykonávání logické funkce. Pokud chceme změnit výslednou funkci elektrického obvodu, musíme změnit zapojení prvků.

Mikropočítač je elektronický obvod integrovaný v jednom pouzdře (čipu). Jeho vnitřní zapojení je po celou dobu životnosti mikropočítače stejné a nedá se změnit. Jeho výsledná funkce není daná zapojením vnitřních elektrických prvků, ale programem, který jej řídí. Velkou výhodou mikropočítače je změna jeho činnosti pouhou výměnou programu, bez nutnosti zásahu do vnitřního zapojení. Tímto řešením odpadá pracný vývoj elektrických obvodů, desek plošných spojů a jejich složitá zapojení.

### **Mikropočítač se skládá z těchto základních komponent:**

- Aritmeticko-logická jednotka (ALU)

Provádí jednoduché aritmetické a logické operace. Mezi aritmetické operace můžeme zařadit sčítání, odečítání, násobení a dělení. Základními logickými operacemi jsou logický součet – OR, logický součin – AND, negace.

- Řadič

Řídí činnost celého mikropočítače.

- Paměť

U harvardské architektury, která je typická pro mikropočítače, se dělí na paměť programu a paměť dat. Tyto paměti jsou od sebe fyzicky oddělené. Výhodou je okamžité vykonávání programu po zapnutí napájení mikropočítače.

- Sběrnice

Spojují všechny části mikropočítače. Dělí se na sběrnice adresové, řídicí a datové.

- Čítače a časovače

Jejich úkolem je počítat impulzy. Čítače a časovače jsou stejné obvody, které se liší pouze v přiváděném signálu. U čítačů je vstupní signál náhodný v čase (např. můžeme počítat,

kolikrát bylo stisknuté tlačítko), časovačům se přivádí na vstup signál o pevně dané frekvenci (dokážeme měřit čas).

- Vstupní a výstupní obvody

Spojují mikropočítač s okolím. Dají se rozdělit na analogové a digitální, nebo paralelní a sériové.

Analogové obvody obsahují AD/DA převodník, který převádí analogový signál na digitální (AD) nebo opačně (DA). Digitální pracují s dvoustavovou logikou – pravda, nepravda (logická 1, logická 0).

Paralelní komunikace je taková, kdy se v jeden okamžik přenáší více bitů (b) najednou (většinou se přenášejí celé bajty (B);  $1B = 8b$ ). Na přenos 1b informace je potřeba jeden vodič. Při sériové komunikaci se přenášejí jednotlivé bity za sebou.

## 1.1 Programování mikropočítače

Programování je proces vytváření a implementace různých sad instrukcí, které umožňují počítači vykonat určitý úkol. Tyto instrukce jsou považovány za počítačové programy a umožňují plynulý chod počítače (Computer programming, 2013).

Programování v sobě zahrnuje tyto úkony:

- **Návrh algoritmu**

V návrhu algoritmu se řeší obecná funkce programu – teoretický postup řešení problému.

- **Přepis algoritmu do zvoleného programovacího jazyka**

Vytvořený algoritmus se přepíše s již konkrétními příkazy do programovacího jazyka. Výstup je zdrojový kód programu, např. v assembleru nebo C.

- **Testování a ladění programu**

Program se testuje a ladí na možné chybové nebo nedefinované stavy. Cílem je ošetřit tyto stavy a tím zvýšit jeho stabilitu.

### **1.1.1 Assembler – jazyk symbolických adres**

Patří do skupiny nižších programovacích jazyků. Jednotlivé příkazy assembleru odpovídají instrukcím mikro počítače (Assembler, [2013]). Jména instrukcí assembleru jsou tvořeny anglickou zkratkou (symbolem) jejich funkce (např. instrukce porovnání – compare – cmp). Tyto symboly si každý výrobce mikro počítačů volí sám.

### **1.1.2 C/C++**

Jazyk C je vyšší programovací jazyk. Znamená to, že jeden příkaz jazyka C se převede na několik po sobě jdoucích instrukcí assembleru. Mezi výhody použití jazyka C patří lepší orientace v kódu, kratší kód a snazší přenositelnost mezi počítači než u assembleru.

## 2 PŘENOSITELNOST KÓDU

V dnešní době existuje mnoho výrobců mikropočítačů. Každý výrobce dodává na trh mikropočítače s vlastním instrukčním souborem, který může být jiný i pro různé řady mikropočítačů.

Výsledkem je, že program napsaný např. pro konkrétní mikropočítač Freescale nemusí fungovat na jiném mikropočítači stejného výrobce, natož výrobce jiného. Tento problém řeší multiplatformní programování, u kterého nám stačí jeden kód, který dokážeme přeložit pro více mikropočítačů i různých výrobců.

Přenositelnost kódu je dána použitým programovacím jazykem a také dodržováním standardů. Některé překladače umí i něco více než co standardně potřebují – dokáží tak usnadnit programování (Bílek, 2008). Jako příklad je možno uvést překladač Wiring, který dokáže zjistit, jestli při volání funkce pro zápis bitové hodnoty na výstup je číslo pinu označeno konstantou nebo proměnnou.

Stejně je to také s knihovnami. Spousta z nich jsou standardně součástí daného programovacího jazyka a není problém je kdekoliv použít. Existuje ale i hodně knihoven, které si uživatelé napsali sami, aby si ulehčili práci. Tyto knihovny se sice rozšířily, ale není zajištěno, že se vyskytují na všech strojích. Pro správnou funkci programu potom potřebujeme tyto knihovny dodávat spolu s programem, aby se zajistila jeho funkčnost.

Vhodnými jazyky pro multiplatformní programování s přenositelným kódem je C nebo Java. V těchto jazycích můžou být napsány univerzální funkce pro práci s mikropočítači, se kterými pracuje uživatel a jsou stejné pro různé druhy mikropočítačů. Vnitřně ale tyto funkce pracují odlišně u každého mikropočítače.

### 3 WIRING

Wiring je platforma s otevřeným kódem, která se skládá z programovacího jazyku, integrovaného vývojového prostředí (Integrated Development Environment – IDE) a desky plošného spoje osazené mikropočítačem (vývojový kit). S jeho vývojem začal v roce 2003 Hernando Barragán.

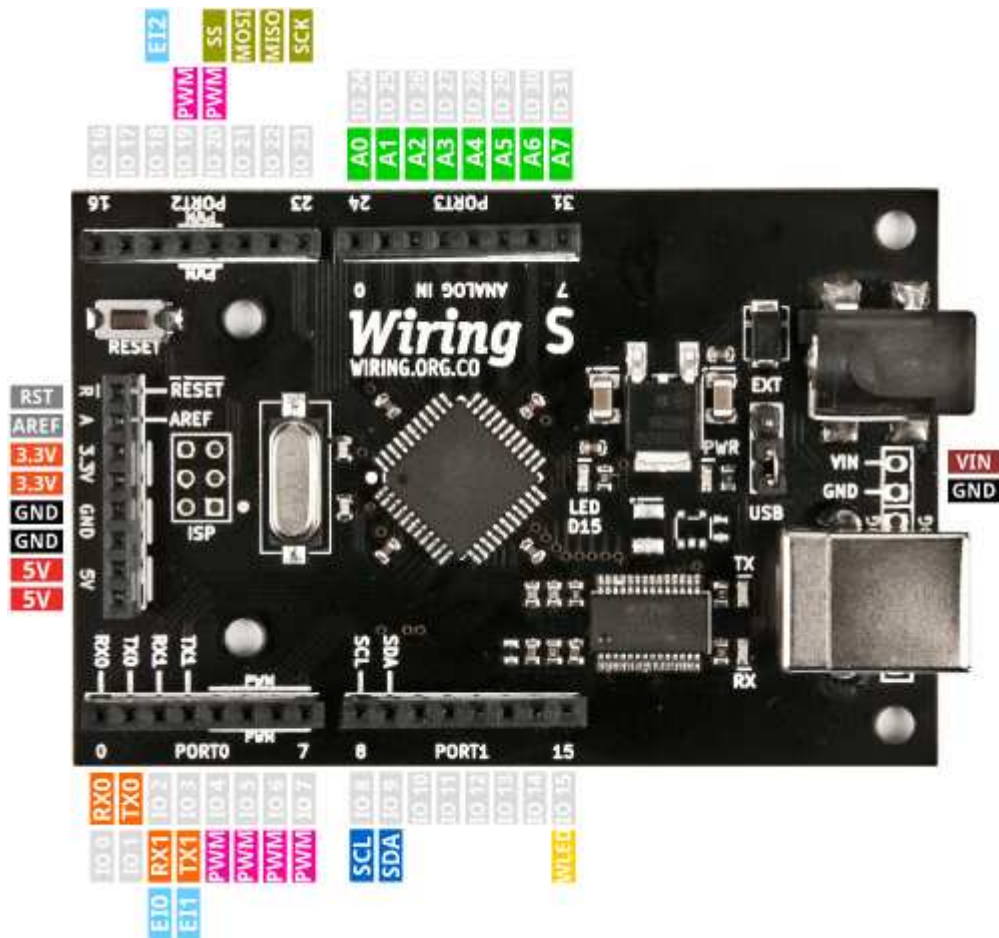
Wiring vznikl na základech programu Processing. Processing je programovací jazyk a vývojové prostředí pro vytváření obrázků, animací a interaktivního obsahu. Měl sloužit jako softwarový skicář a naučit základy programování. Výstup z programu je vizuální.

Později, na základech Wiringu, vznikl další projekt nazvaný Arduino. Arduino vyvíjí různé desky plošných spojů osazené mikropočítači, s různou hardwarovou výbavou. Programovací jazyk vychází z Wiringu a vývojové prostředí z Processingu. Pomocí Arduina si můžete naprogramovat jak desky od Arduina, tak i desky své.

Wiring umožňuje psát multiplatformní software k ovládání zařízení připojených k široké škále vývojových kitů a desek plošných spojů s mikropočítači.

Je vytvořen tak, že se každý programátor může podělit o své kódy nebo knihovny s ostatními, a tím přispět k rozvoji Wiringu. Tyto knihovny jsou k dispozici na webu [wiring.org.co](http://wiring.org.co).

Součástí Wiringu je i hardware. Můžeme si zakoupit několik druhů desek plošných spojů osazenými různými periferiemi a mikropočítači. Protože Arduino vychází z Wiringu, můžeme použít jakékoliv vývojové kity z obou projektů, nebo si vytvořit vlastní. Při vlastním zapojení ale musíme vybrat správný, podporovaný, mikropočítač.



Obr. 1: Vývojový kit Wiring S (Barragán a kol., 2013)

### 3.1 Specifikace Wiringu

- Aktuální jádro AVR8 podporuje kromě veškerého Wiring hardware (Obr. 1) i hardware založený na procesorech AVR ATmega.
- V plánu je podpora více hardwarových architektur.
- Brzy se rozšíří i o jiné procesory (Xmega, Microchip).
- Určený pro Linux, Macintosh i Windows.

## 3.2 Framework

Framework je ucelený soubor tématicky zaměřených knihoven (Majda, © 2000–2010).

Obsahuje tyto základní části:

- **Struktury** – funkce pro práci se strukturami, direktivy, poznámky
- **Data** – datové typy, konverze, ukazatele
- **Řízení** – porovnávací operátory, logické operátory, cykly, podmínky
- **Vstupy/výstupy** – čtení a zápis na piny, PWM modulace, přerušení, řízení napájení, sériová komunikace
- **Matematické operace** – operátory, matematické funkce (statistické, zaokrouhlování, trigonometrie, generování náhodných čísel)

## 3.3 Knihovny

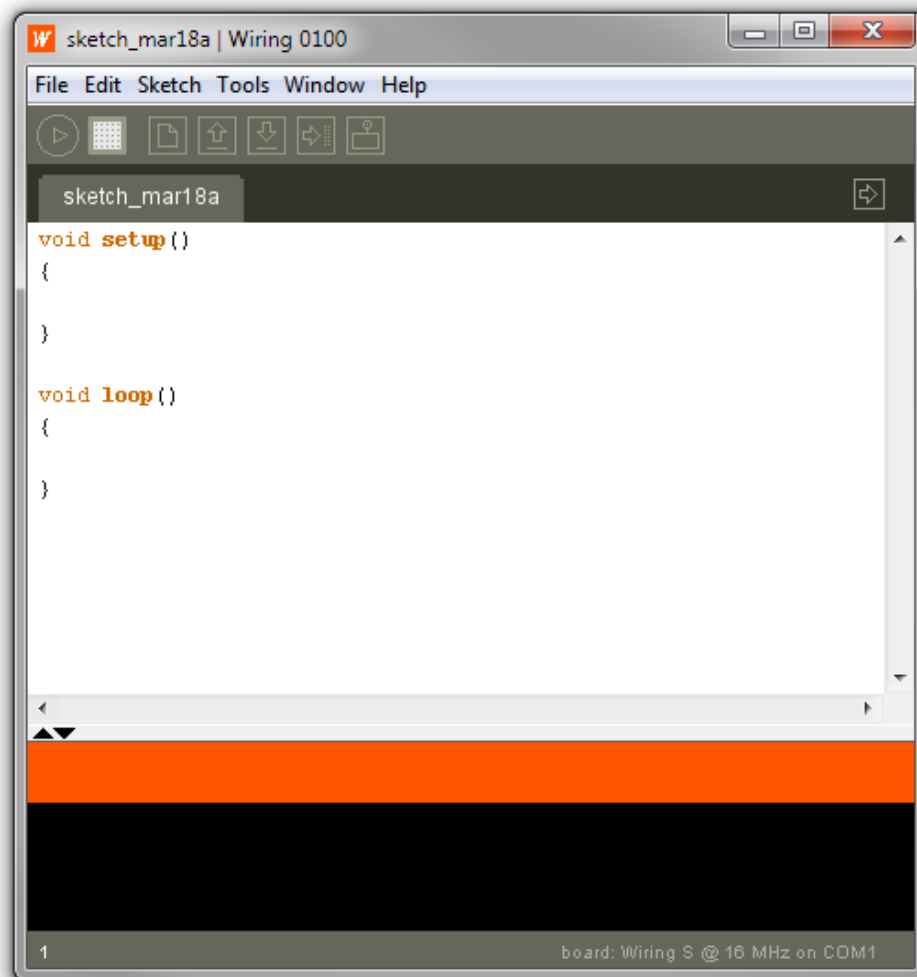
Obsahují v sobě metody a funkce pro práci s periferiemi. Knihovny, které jsou obsažené ve Wiringu, se dělí do dvou skupin:

- **Core Libraries** – základní knihovny, které jsou specifické pro danou platformu (obsluha paměti, základní práce s LCD displejem, komunikace po sériové lince). Do této skupiny knihoven patří např. LiquidCrystal, SPI, Servo.
- **Cross-platform Libraries** – multiplatformní knihovny, jsou nezávislé na dané platformě. Našli by jsme zde např. obsluhu tlačítek (Button), LED nebo potenciometru (Potentiometer).

### 3.4 Vývojové prostředí (IDE)

Vývojové prostředí je rozděleno do čtyř částí:

- Menu
- Textový editor
- Okno zpráv
- Konzole



Obr. 2: Vývojové prostředí Wiring

### 3.4.1 Menu

Obsahuje veškeré nástroje pro práci s projekty, nahrávání programů do mikropočítače, kompilace a sestavení programu, vkládání knihoven, výběr mikropočítače nebo vývojového kitu a nápovědu.

### 3.4.2 Textový editor

Slouží pro psaní programů. Již při založení projektu obsahuje náš program dvě základní funkce (Obr. 2), které jsou prázdné.

- void setup() – nastavení mikropočítače (vstupy a výstupy). Spouští se pouze jednou při zapnutí mikropočítače.
- void loop() – v této funkci se ve smyčce vykonává hlavní program.

### 3.4.3 Okno zpráv

Zobrazuje informace o stavu překladač programu nebo jednoduchá chybová hlášení spolu s jejich umístěním v textovém editoru.

### 3.4.4 Konzole

Výpis kompletních chybových hlášení a výsledné velikosti kódu spolu se zbývajícím volným místem v paměti mikropočítače.

## 3.5 Podporované mikropočítače

Po založení projektu si může uživatel vybrat, pro jaký mikropočítač bude svůj program tvořit. Na výběr má mnoho verzí mikropočítačů ATmega od firmy Atmel. Podporované mikropočítače jsou:

- ATmega168 - 16 MHz, 8 MHz
- ATmega168P(A) – 16 MHz, 8 MHz
- ATmega328P(A) – 16 MHz, 8 MHz
- ATmega644P(A) – 16 MHz, 8 MHz
- ATmega1284P(A) – 16 MHz, 8 MHz

### 3.6 Podporované desky (vývojové kity)

V nabídce originálního Wiring hardware najdeme několik vývojových kitů, včetně desky nové generace pod označením Wiring S. Brzo by se na trhu měly objevit další vývojové kity – Wiring XS, Wiring M, Wiring XL.

Nemusíme ale použít pouze Wiring hardware. Podporované jsou také desky RoqueRobotics (dvě desky + rozhraní na SD karty) (REDI Education, 2013), Arduino a BDMicro (dvě desky, obě osazené součástkami vhodné pro průmyslové použití) (MAVRIC-IIB, [2005]).

#### 3.6.1 Parametry kitu Wiring S

- Procesor Atmel ATmega644P, 16 MHz
- 64 KiB paměti
- USB
- 2 sériové porty
- 1x I2C port
- 8 analogových vstupů
- 6 hardwarových PWM pinů

#### 3.6.2 Vývojové kity Arduino

Arduino má v nabídce mnoho desek s bohatou výbavou periferií. V nabídce najdeme 17 desek plošných spojů osazených mikropočítači ATmega, vývojový kit a rozšiřující moduly.

#### Podporované desky Arduino:

- Uno – 16MHz
- Duemilanove, Nano – procesor ATmega328 – 16 MHz
- Mega 2560 – 16 MHz

- Mega (ATmega1280) – 16 MHz
- LilyPad – procesory ATmega 328 a ATmega168

### Arduino rozšiřující moduly

- GSM
- Ethernet
- WiFi
- Ovládání motorů
- USB/sériový adaptér

Rozšiřující moduly Arduino se dají oficiálně připojit i k deskám Wiring.

### 3.7 Adresování pinů a portů

Jednotlivé piny a porty jsou v prostředí Wiring označeny číslem, a to od 0 až maximální počet pinů/portů. Pokud si pořídíme desku Wiring S, která disponuje čtyřmi porty po osmi bitech, adresování je následující:

- Porty jsou označeny jako port 0, port 1, port 2, port 3
- Piny na portu 0 jsou očíslovány od 0 po 7
- Piny na portu 1 jsou číslovány od 8 do 15
- Další číslování pokračuje ve stejném duchu

### 3.8 Přenositelnost kódu mezi různými deskami

Pokud máme již napsaný program např. pro desku Wiring S, jednoduše si v nastavení vývojového prostředí změňme tuto desku na jinou. Program se zkompiluje a vytvoří. Pracuje stejně jako na předchozí desce.

Do každého projektu je vkládaný hlavičkový soubor „Wiring.h“. Jedná se o rozhraní pro programování aplikací (API – Application Programming Interface) (Orenstein, 2000). Obsahuje sadu standardů pro vývoj aplikace – funkce, knihovny aj.

Soubor „Wiring.h“ obsahuje mezi spousty vkládanými soubory také hlavičkový soubor „io.h“. Tento soubor dále volá odpovídající definice pro vstupy a výstupy různých mikropočítačů – podle toho, jaký jsme si zvolili ve vývojovém prostředí. Pokud by jsme porovnali definice různých mikropočítačů, uvidíme, že stejné funkce mají někdy různá jména (v závislosti na mikropočítači).

### 3.9 Adresářová struktura vývojového prostředí

Funkce a knihovny, které byly popsány v předchozích kapitolách, jsou různě umístěné v adresáři Wiring. Ty nejdůležitější najdete níže.

#### Organizace zdrojových kódů

- **cores**

V této složce jsou uloženy knihovny jádra AVR8 (core libraries), včetně knihovny „Wiring.h“.

- **examples**

Složka obsahuje již hotové a plně funkční příklady na používání různých funkcí (digitální vstupy a výstupy, analogové vstupy a výstupy, sériová linka, přerušení aj.).

- **hardware**

Najdeme zde konfigurační soubory pro mikropočítače a vývojové kity.

- **libraries**

Jedná se o knihovny, které nejsou závislé na architektuře mikropočítače (Cross-platform Libraries).

## 4 POPIS FRAMEWORKU WIRING

Framework Wiring obsahuje spoustu funkcí, pomocí kterých lze obsluhovat širokou škálu periférií, ať už digitálních nebo analogových. Periferie můžeme připojit buď na pin nebo na port.

- Pin je jeden výstup mikropočítače
- Sadu několika pinů nazýváme port
- Port zpravidla obsahuje osm pinů

### 4.1 Obsluha vstupů a výstupů

Součástí každého pinu je pullup rezistor. Pullup rezistory se v obvodech používají k zajištění potřebné logické hodnoty na vstupu, pokud je pin odpojený. Se zapnutým pullup rezistorem a odpojeným pinem je na vstupu logická jednička, s vypnutým pullup rezistorem a odpojeným pinem je na vstupu stav vysoké impedance. Ve výchozím nastavení jsou pullup rezistory vypnuty.

#### 4.1.1 Obsluha digitálních pinů

Následující funkce slouží k nastavení jednotlivých pinů buď na logickou jedničku, nebo logickou nulu.

##### 4.1.1.1 *pinMode(pin, režim)*

Před prací s piny mikropočítače jej musíme nastavit do požadovaného režimu. Existují dva režimy – vstup a výstup.

*pin*                    nabývá hodnot 0 až maximální počet pinů

*režim*                *INPUT* pro vstup, *OUTPUT* pro výstup

Příklad 1: Nastavení pinu č. 5 do režimu vstupu

```
pinMode(5, INPUT);
```

#### 4.1.1.2 *digitalWrite(pin, hodnota)*

Zapíše hodnotu logické jedničky nebo nuly na příslušný pin.

*pin*                   nabývá hodnot 0 až maximální počet pinů

*hodnota*            *HIGH* pro logickou jedničku, *LOW* pro logickou nulu

Příklad 2: Nastavení pinu č. 5 do logické nuly

```
digitalWrite(5, LOW);
```

#### 4.1.1.3 *digitalRead(pin)*

Vrací logickou hodnotu zvoleného *pinu*.

Příklad 3: Zjištění stavu 5 pinu

```
stav = digitalRead(5);
```

#### 4.1.1.4 *pullup(pin)*

Zapne pullup rezistor na daném *pinu*.

Příklad 4: Zapnutí pullup rezistoru na pinu 3

```
pullup(3);
```

#### 4.1.1.5 *noPullup(pin)*

Vypne pullup rezistor na daném *pinu*.

Příklad 5: Vypnutí pullup rezistoru na pinu 3

```
noPullup(3);
```

#### 4.1.2 Obsluha digitální portů

Následující funkce nám umožňují práci s osmi piny najednou (tj. s jedním portem). Zapsat lze pouze celé číslo typu integer.

##### 4.1.2.1 *portMode(port, režim)*

Nastaví port (všech osm pinů) do režimu vstupu nebo výstupu.

*port*            číslo portu

*režim*            *INPUT* pro vstup, *OUTPUT* pro výstup

Příklad 6: Nastavení portu 0 do režimu vstupu.

```
portMode(0,INPUT);
```

##### 4.1.2.2 *portRead(port)*

Přečte hodnoty na daném portu. Výsledkem je číslo datového typu int.

*port*            číslo portu

Příklad 7: Čtení hodnoty vstupů na portu 1

```
int hodnotaPortu = portRead(1);
```

##### 4.1.2.3 *portWrite(port, hodnota)*

Zapíše hodnotu typu integer na daný port.

*port*            číslo portu

*hodnota*        číslo, které chceme na port zapsat

Příklad 8: Zápis čísla 127 na port 1

```
portWrite(1,127);
```

### 4.1.3 Obsluha analogových vstupů

Pomocí těchto funkcí dokážeme číst napětí přiložené k určitým pinům, nebo na tyto piny přivést odpovídající napětí. Čtení z pinu je řešeno pomocí AD převodníků, zápis je prováděn pomocí PWM.

#### 4.1.3.1 *analogRead(pin)*

Přečte analogovou hodnotu napětí na zadaném pinu a výsledek vrátí jako číslo v rozsahu 0 (0V) až 1023 (5V).

*pin* zvolený pin

Příklad 9: Čtení napětí z pinu 6.

```
int hodnota = analogRead(6);
```

#### 4.1.3.2 *analogReference(mód)*

Nastaví referenční hodnotu napětí pro funkci *analogRead()*. Výchozí hodnota je 5V.

Možné hodnoty *módu*:

- DEFAULT – výchozí hodnota 5V
- INTERNAL1V1 – napětí 1,1 V
- INTERNAL2V56 – napětí 2,56 V
- INTERNAL – zabudované referenční napětí
- EXTERNAL – referenční napětí se zvolí takové, které se připojí na pin označený jako AREF. Smí se použít pouze rozsah 0 – 5 V.

Pokud připojíme napětí na pin AREF, nesmí se v aplikaci nastavit jiné referenční napětí. Mohl by vzniknout zkrat a ten poškodit mikropočítač.

Příklad 10: Nastavení výchozího referenčního napětí.

```
analogReference(DEFAULT);
```

#### 4.1.3.3 *analogWrite(pin, hodnota)*

Funkce zapne pulzní šířkovou modulaci (PWM) o požadované *hodnotě* na daný *pin*.

*pin* zvolený pin

*hodnota* celé číslo od 0 do 255

Příklad 11: Zápis čísla 100 na pin 6

```
analogWrite(6,100);
```

#### 4.1.3.4 *noAnalogWrite(pin)*

Vypne PWM na zadaném pinu.

*pin* zvolený pin

Příklad 12: Vypnutí PWM na pinu 6

```
noAnalogWrite(6);
```

## **II. PRAKTICKÁ ČÁST**

## 5 MIKROPOČÍTAČ FREESCALE HCS08GB60

Pro programování byl zvolen osmibitový mikropočítač od firmy Freescale. Jedná se o vývojový kit (M68EVB08GB60, revize C) používaný při výuce předmětu „Programování mikropočítačů“.

Technické vlastnosti vývojového kitu:

- 7 portů po osmi bitech (56 pinů)
- 1 port pro analogové vstupy
- 4 tlačítka
- 4 LED
- potenciometr připojený na 10-bitový převodník (s možností 8-bitového převodu), napájení potenciometru 3,3 V
- LCD display



Obr. 3: Freescale M68EVB08GB60

## 6 IMPLEMENTACE PRO HCS08GB60

Při programování zůstala zachována adresářová struktura Wiringu. V hlavním adresáři *Wiring* tak přibýly dvě složky:

- **S08** ve složce „cores“
- **Freescale** ve složce „hardware“

### 6.1 Adresování pinů a portů

Číslování pinů a portů je vytvořeno vzestupně od nuly, jak jdou za sebou porty A,B,C až G. Port A je označen jako 0, port B jako 1, atd.

Pin 0 najdeme na nejnižším bitu portu 0, pin 8 je na nejnižším bitu portu B, atd.

### 6.2 Složka S08

Obsahuje soubory pro práci s digitálními i analogovými vstupy/výstupy a hlavní soubor „*Wiring.h*“.

Protože Wiring používá pouze překladače pro mikropočítače které podporuje, musel jsem založit projekt ve vývojovém prostředí pro mikropočítače Freescale CodeWarrior a dále pracovat s ním. Tento projekt se nachází ve složce *CW\_build/M68EVB08GB60*. Kromě projektu zde najdeme i „*WMain.ccp*“, do kterého píše koncový uživatel své programy. „*WMain.ccp*“ obsahuje pouze funkce *setup()* a *loop()*, stejně jako nový projekt ve Wiringu.

*CW\_build* vyjadřuje složku s projekty pro jednotlivé mikropočítače Freescale. *M68EVB08GB60* je složka pojmenovaná přímo po vývojovém kitu, pro který jsou funkce pro práci s vstupy a výstupy naprogramovány.

### 6.3 Složka Freescale

V ní najdeme podadresář GB60, který obsahuje informace o připojeném mikropočítači a desce. V souboru „*BoardDefs.h*“ najdeme:

- Informace o celkovém počtu pinů a analogových vstupů
- *BoardInit()* – funkce pro inicializace desky (je prázdná)
- Makra pro mapování pinů na jednotlivé registry vstupů, výstupů a pull-up rezistorů

### 6.4 Soubory použité při programování

Pro programování mikropočítače Freescale v duchu Wiringu potřebujeme následující soubory.

#### 6.4.1 main.cpp

Jedná se o hlavní soubor CodeWarrior projektu. Obsahuje odkazy na „*Wiring.h*“ a „*WMain.h*“. Ve funkci *main()* je volána inicializace desky (*BoardInit*), nastavení mikropočítače pomocí funkce *setup()* a v nekonečné smyčce běží funkce *loop()* spolu s resetováním „*hlídacího psa (watchdog)*“.

#### 6.4.2 WMain.cpp

Jediný soubor, který potřebuje k programování běžný uživatel. Sem se zapisují kódy programu do dvou funkcí *setup()* a *loop()*.

#### 6.4.3 Wiring.h

Jsou v něm zapsány různé definice, odkazy na soubory „*BoardDefs.h*“, „*WDigital.h*“, „*WAnalog.h*“.

#### 6.4.4 BoardDefs.h

Obsahuje definice o maximálním počtu pinů a analogových vstupů, funkci pro inicializaci vývojového kitu a makra pro práci se vstupy a výstupy.

```
#define portInputRegister(PORT) \  
  ( ((PORT) == 0 ) ? &PTAD : \  
  ( ((PORT) == 1 ) ? &PTBD : \  
  ( ((PORT) == 2 ) ? &PTCD : \  
  ( ((PORT) == 3 ) ? &PTDD : \  
  ( ((PORT) == 4 ) ? &PTED : \  
  ( ((PORT) == 5 ) ? &PTFD : \  
  ( ((PORT) == 6 ) ? &PTGD : NOT_A_REG))))))
```

Obr. 4: Makro pro zjištění adresy vstupního registru podle portu

Na obrázku (Obr. 4) lze vidět makro, které vrací adresu registru příslušného portu. Pokud chceme port s číslem větším než 6, makro vrací hodnotu NOT\_A\_REG, které je totožné s NULL.

#### 6.4.5 WDigital.c

Obsahuje funkce pro práci s digitálními piny, porty a pull-up rezistory. Jsou zde dva typy funkcí – s podtržítkem a bez podtržítka.

V originálním Wiring kódu jsou ve funkci bez podtržítka (např. *pinRead*) rozhodnutí, ve kterém si překladač zjistí, zda-li je číslo, kterým označujeme jednotlivý pin, konstantní, nebo se při běhu programu mění. Pokud je konstanta, tak se automaticky do této funkce dosadí hotový kód pro čtení z daného pinu. V opačném případě se při každém volání této funkce zavolá funkce stejného jména, ale s podtržítkem (např. *\_pinRead*), která musí pokaždé zjistit adresu vstupně/výstupního registru a z něj přečíst hodnotu daného pinu.

Kdybychom porovnali rychlost programu, který obsahuje pouze konstantní hodnoty pro ovládání vstupů a výstupu, tak je rychlejší než stejný program s proměnnými hodnotami.

V provedení pro HCS08 se ve funkci *pinRead()* volá pouze *\_pinRead()*. Je to z důvodu, že překladač používaný v prostředí CodeWarrior nedokáže zjistit, jestli je daný pin konstanta nebo proměnná.

### Popis funkce `_pinRead()`

Jako první se vytvoří maska pro snímání pouze jednoho bitu z celého portu (Obr. 5). Tuto funkci zajistí makro `digitalPinToBitMask(pin)`, které se nachází v souboru „`BoardDefs.h`“. Následně se zjistí pomocí dalšího makra `port`, na kterém se daný pin nachází. V dalším kroku se do ukazatele `inputregister` uloží adresa registru vstupního portu obsahující daný pin (`portInputRegister(port)`). Zkontroluje se, jestli pin, který se má přečíst, vůbec existuje. Pokud ano, tak se pokračuje dál a funkce vrátí přečtenou hodnotu.

```
uint8_t _pinRead(uint8_t pin)
{
    uint8_t bitmask = digitalPinToBitMask(pin);
    uint8_t port = digitalPinToPort(pin);
    volatile uint8_t *inputregister;

    inputregister = portInputRegister(port);

    if (pin > ((TOTAL_PINS) - 1)) return LOW;

    if ((*inputregister) & bitmask)
        return HIGH;
    else
        return LOW;
}
```

Obr. 5: Funkce `_pinRead()`

### 6.4.6 WAnalog.c

Slouží pro čtení analogové hodnoty z pinů číslo 8 až 15. Obsahují celkem tři funkce:

#### a) `adcInit`

Provede Inicializaci AD převodníku – zapnutí převodníku na 10-bitový převod bez znaménka (0 – 1023).

**b) analogReference**

V implementaci pro HCS08 existuje pouze reference DEFAULT, ostatní nejsou podporovány.

**c) analogRead**

Pokud se funkce volá poprvé, tak proběhne inicializace. Následně se provede výběr pinu – nastavením jako analogový vstup a přepnutím multiplexeru na tento vstup. Pak se čeká na dokončení převodu a vrátí se výsledek.

## 7 ZALOŽENÍ PROJEKTU A UKÁZKOVÝ PROGRAM

Nejjednodušší způsob, jak vytvořit nový program, je zkopírovat složku „M68EVB08GB60“ z adresáře „Wiring/cores/S08/CW\_build“ do námi požadovaného adresáře. Zkopírovaná složka „M68EVB08GB60“ se může jakkoliv přejmenovat. Protože takto zkopírovaný projekt nezná aktuální cesty k potřebným souborům, přidají se do projektu cesty a dále už stačí psát samotný program do souboru „WMain.cpp“.

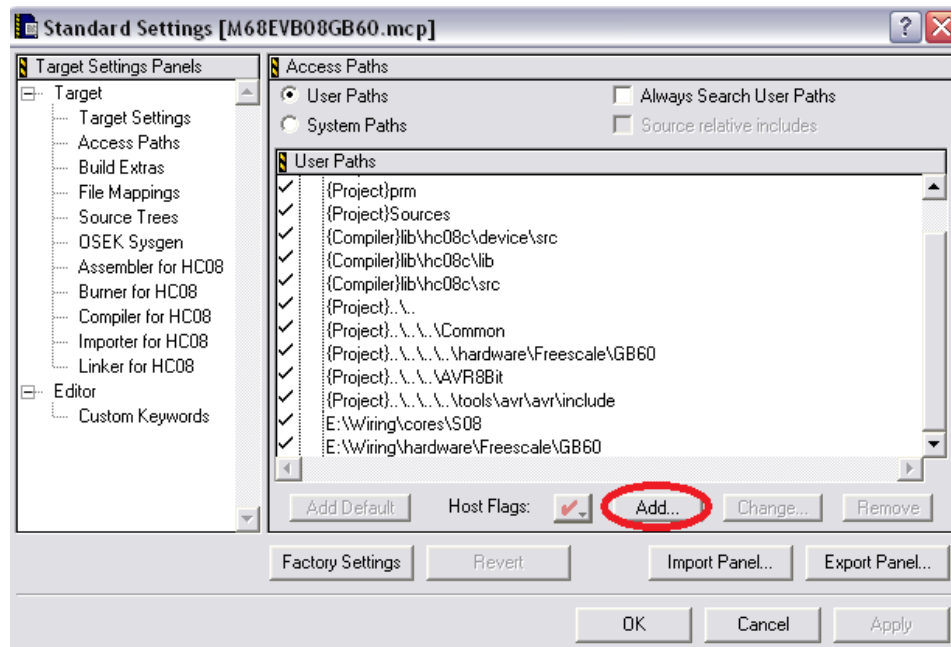
Druhou, složitější, možností je vytvořit nový projekt v prostředí CodeWarrior, do projektu připojit veškeré hlavičkové a zdrojové soubory potřebné pro správnou funkci programu. U tohoto způsobu ale vznikne spousta komplikací. Bude se muset přepsat soubor „main.cpp“ nově založeného projektu, který neobsahuje žádné odkazy na potřebné soubory a zkopírovat soubor WMain.h a WMain.cpp z adresáře „Wiring“ do složky s nově založeným projektem.

### 7.1 Založení projektu – metoda 1

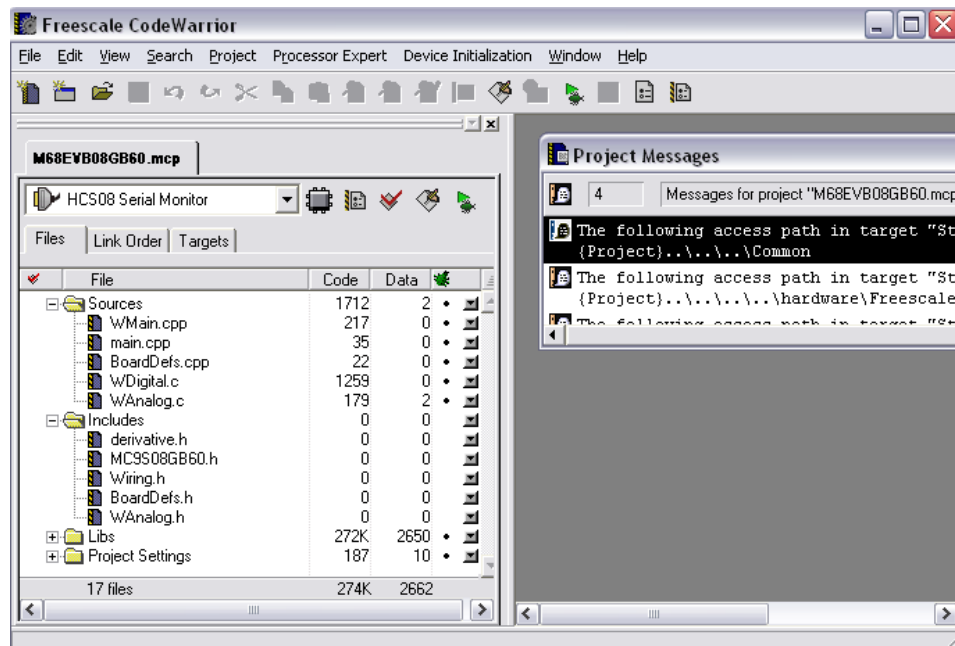
Jedná se o metodu zkopírování složky „M68EVB08GB60“.

- Zkopírování složky „M68EVB08GB60“ z adresáře „Wiring/cores/S08/CW\_build“ do námi zvoleného adresáře.
- Možnost přejmenování složky „M68EVB08GB60“ dle libosti.
- Otevření projektu v CodeWarrioru pomocí souboru „M68EVB08GB60.mcp“, který se nachází v námi zkopírované složce.
- Přidání cesty k souborům potřebným pro správnou funkci programu. V okně CodeWarrioru se klikne v liště menu na „Edit“ a vybere se „Standard Settings“. Otevře se dialogové okno (Obr. 6). Toto okno se může získat také pomocí klávesové zkratky „Alt+F7“
- V levé části okna se vybere „Access Paths“
- Kliknutím na tlačítko „Add“ (Obr. 6) se postupně přidají dvě cesty – jednu ke složce S08 v adresáři „Wiring/cores“, druhou ke složce GB60 v adresáři „Wiring/hardware/Freescale“.
- Okno se zavře kliknutím na tlačítko „OK“

- Rozkliknutím souboru „WMain.cpp“ v levé části CodeWarrioru (Obr. 7) se otevře textový editor, do kterého se může psát program.



Obr. 6: Standard Settings

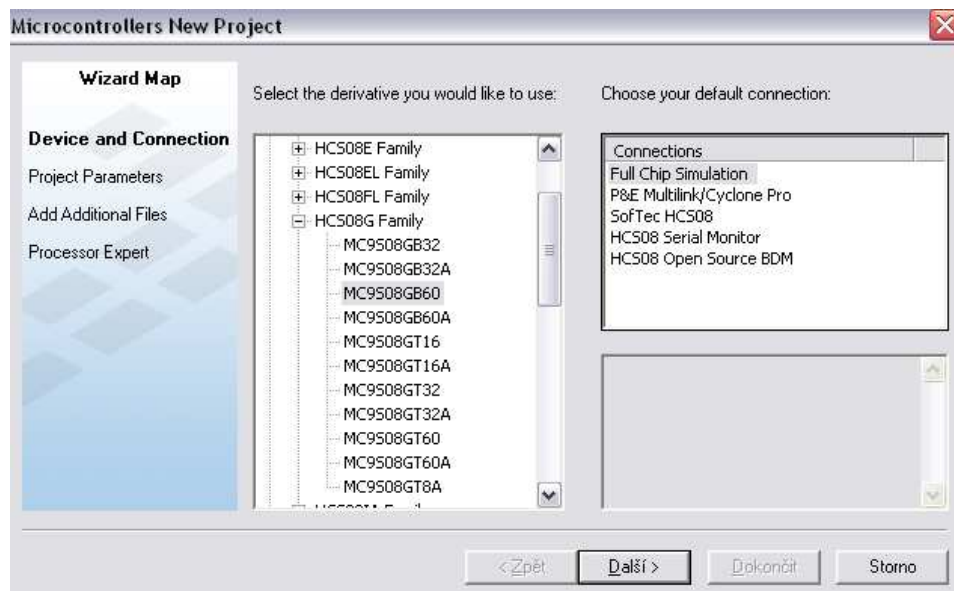


Obr. 7: Otevřený projekt v CodeWarrioru

## 7.2 Založení projektu – metoda 2

Jedná se o druhou, složitější, metodu.

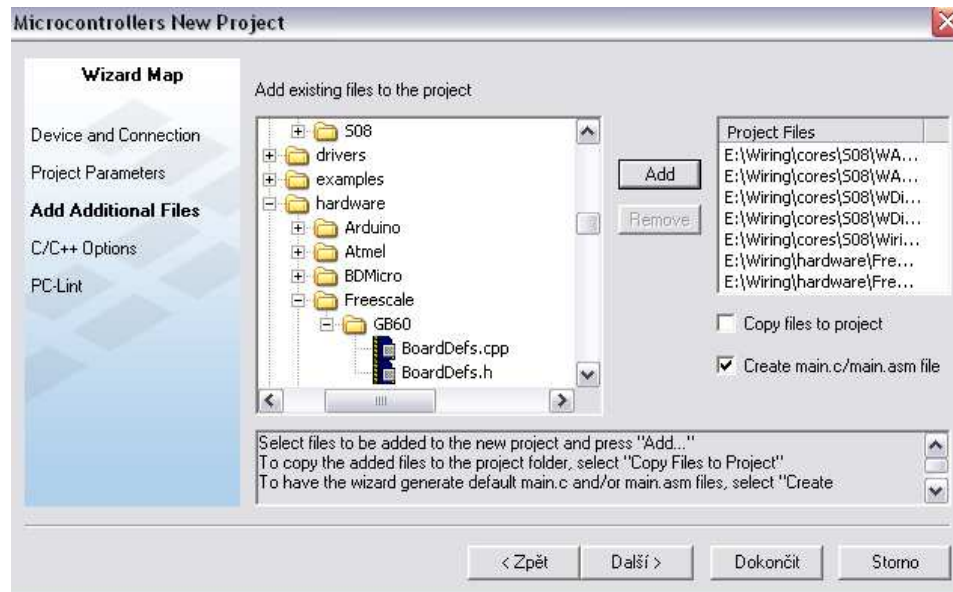
- Po otevření CodeWarrioru se v okně „Startup“ vybere možnost „Create New Project“. Pokud se toto okno neobjeví, projekt se založí kliknutím na „File“ a dále „New Project“.
- Vybere se správný mikropočítač (HCS08G Family -> MC9S08GB60) a zvolí způsob připojení („Full Chip Simulation“ pro simulaci nebo „HCS08 Serial Monitor“ pro reálně připojený mikropočítač), viz. Obr. 8. Klikne se na „Další“.



Obr. 8: Výběr mikropočítače

- Ve výběru programovacího jazyka se zatrhne možnost „C“ a „C++“, v pravé části okna se pojmenuje projekt a vybere se cesta, kam se uloží. Klikne se na „Další“.
- Do projektu se vloží pomocí tlačítka „Add“ celkem 7 souborů (Obr. 9). Ze složky „Wiring/cores/S08“ to jsou „WAnalog.c“, „WAnalog.h“, „WDigital.c“, „WDigital.h“ a „Wiring.h“. Ze složky „Wiring/hardware/Freescale/GB60“ to jsou soubory „BoardDefs.cpp“

a „BoardDefs.h“. Políčko „Copy files to project“ se nechá neoznačené. Pokračuje se stisknutím tlačítka „Další“.



Obr. 9: Vložení souborů do projektu

- Nyní už stačí dát pouze dokončit. Okno se zavře a zůstane pouze otevřený projekt.
- Přepíše se soubor „main.cpp“ ve složce „Sources“ nově založeného projektu souborem stejného jména ze složky „Wiring/cores/S08/CW\_build/M68EVB08GB60/Sources“.
- Zkopírují se soubory „WMain.h“ a „WMain.cpp“ ze složky „Wiring/cores/S08/CW\_build/M68EVB08GB60“ do adresáře nově založeného projektu. Tyto zkopírované soubory se následně přidají do projektu pomocí „Add Files“, které se zobrazí po kliknutí pravým tlačítkem myši na složku „Sources“ v levé části okna otevřeného projektu (Obr. 7)
- Nyní je již projekt připravený k programování. Rozkliknutím souboru „WMain.cpp“ v levé části CodeWarrioru (Obr. 7) se otevře textový editor, do kterého se může psát program.

### 7.3 Ukázkový program č. 1

Program na obrázku (Obr. 10) rozsvěcuje LED v závislosti na stisknutém tlačítku.

Ve funkci *setup* se nejprve nastaví piny s LED do režimu výstupu (piny 40 až 43), tlačítka připojená na piny 4 až 7 se nastaví do režimu vstupu a zapnou se na nich pullup rezistory.

Ve funkci *loop* se neustále přenáší hodnoty z tlačítek na diody.

```
#include "Wiring.h"

void setup(void){
    pinMode(40, OUTPUT);
    pinMode(41, OUTPUT);
    pinMode(42, OUTPUT);
    pinMode(43, OUTPUT);

    pinMode(4, INPUT); // tlacitko SW1 na portu A
    pinMode(5, INPUT); // tlacitko SW2 na portu A
    pinMode(6, INPUT); // tlacitko SW3 na portu A
    pinMode(7, INPUT); // tlacitko SW4 na portu A

    pullup(4);
    pullup(5);
    pullup(6);
    pullup(7);
}

void loop(void){
    digitalWrite(40, digitalRead(4));
    digitalWrite(41, digitalRead(5));
    digitalWrite(42, digitalRead(6));
    digitalWrite(43, digitalRead(7));
}
```

Obr. 10: Ukázkový program č. 1

## 7.4 Ukázkový program č. 2

Program na obrázku (Obr. 11) rozsvěcuje čtyři LED v závislosti na úhlu natočení potenciometru.

Nejprve je vytvořena čekací funkce *cekej*. Ve funkci *setup* jsou piny s LED nastaveny do režimu výstupu. Ve funkci *loop* se čte hodnota z pinu 8 a podle ní se rozsvěčují jednotlivé LED. Čím větší hodnota, tím více LED svítí. Na konci je zavoláno čekání.

```
#include "Wiring.h"

void cekej(unsigned int max){
  unsigned int i;
  int j;
  for (j = 0; j<= 2; j++){
    for (i = 0; i < max; i++ )
      i = i;
  }
}

void setup(void){
  pinMode(40, OUTPUT);
  pinMode(41, OUTPUT);
  pinMode(42, OUTPUT);
  pinMode(43, OUTPUT);
}

void loop(void){
  portWrite(5, 255);
  if ( analogRead(8) > 0 ){
    digitalWrite(40, LOW);
  };
  if ( analogRead(8) > 256 ){
    digitalWrite(41, LOW);
  };
  if ( analogRead(8) > 512 ){
    digitalWrite(42, LOW);
  };
  if ( analogRead(8) > 768 ){
    digitalWrite(43, LOW);
  };

  cekej(10000);
}
```

Obr. 11: Ukázkový program č. 2

## ZÁVĚR

Cílem této práce bylo získat informace o platformě Wiring a následně je uplatnit při rozšíření Wiringu o dalšího výrobce mikropočítačů – Freescale. Wiring je všestranný nástroj pro programování mikropočítačů, se spoustou knihoven rozšiřující jeho funkce. Mezi velkou výhodou bych zařadil, že je zdarma, má otevřený zdrojový kód a existuje k němu mnoho hardwaru.

O rostoucí popularitě Wiringu svědčí i jeho „sesterský“ projekt Arduino, který je ve světě ještě známější než jeho předchůdce Wiring. Arduino je kompatibilní s Wiringem, což ještě zvyšuje možnou uživatelskou základnu díky vyššímu výběru hardwaru.

Pokud je uživatel začátečník, může si vybrat na trénování několik základních vývojových desek osazenými různými mikropočítači. Pokud mu už základní vývojová deska stačit nebude, může si kdykoliv dokoupit rozšiřující moduly. Pomocí tohoto příslušenství můžete svůj mikropočítač ovládat pomocí webového prohlížeče, posílat SMS zprávy nebo pomocí SMS zpráv řídit svůj mikropočítač.

V praktické části jsem realizoval vstupně-výstupní funkce pro mikropočítač Freescale. To nám umožňuje psát programy pro ovládání digitálních pinů a portů naprosto stejně jako ve Wiringu. V oblasti analogových vstupů jsem implementoval funkci pro čtení analogové hodnoty připojené na daný pin. Zápis analogové hodnoty jsem nevytvořil, protože ve Wiringu je tento zápis prováděn pomocí pulzně šířkové modulace (PWM), ke které potřebuje navíc časovače a přerušení. Přesto si uživatel může rychle vytvořit svou vlastní PWM pomocí čekací smyčky a střídáním zápisu logické jedničky a nuly na výstup.

Tato práce má hlavní využití v záměně jednoduchých programů napsaných pro hardware Wiring a mikropočítač Freescale HCS08GB60. Do budoucna by se mohla rozšířit ještě o možnost použití časovačů a přerušení, pomocí kterých by se automaticky zpřístupnila funkce pro zápis analogové hodnoty na výstup.

## ZÁVĚR V ANGLIČTINĚ

The aim of this study was obtain information about Wiring platform and subsequently apply it at Wiring extension for next microcontrollers producer – Freescale. Wiring is versatile tool for programming microcontrollers, with a lots of libraries expanding its function. Among the great advantages I would place it is free, open-source and there are a lots of hardware.

The growing Wiring popularity is shown on its „sisters“ project called Arduino, which is in the world more famous then Wiring. Arduino is compatible with the Wiring, which increases possible users base due to higher hardware selection.

If the user is beginner, he can choose to train some basic development boards with different microcontrollers. If the basic development board is not enough, the user can buy expansion module for his microcontroller. With this accessories can control his microcontroller via a web browser, sending SMS via microcontroller or with SMS control it.

In the practical part I realized input/output function for Freescale microcontroller. This allow us writting programs for control digital pins and ports exactly like in the Wiring. In the area of analog inputs I implemented function for reading a analog values attached to the pin. I do not make a function for writting analog value to the pin, because this function is in the original Wiring realized by PWM, which needs a extra timers and interruptions. However, the user can quickly create his own PWM with waiting loop and writing alternate logical ones and zeraus on output.

This work have the main usage in swap programs written for Wiring hardware and microcontroller Freescale HCS08GB60. In the future could be this work extended timers and interruptions, which would automatically make available function for writting analog values to output pins.

## SEZNAM POUŽITÉ LITERATURY

- Arduino* [online]. [cit. 2013-06-03]. Dostupné z: <http://www.arduino.cc/>
- Assembler, [2005]. *Programování* [online]. [cit. 2013-03-16]. Dostupné z: <http://k-prog.wz.cz/progjaz/assembly.php>
- BARRAGÁN, Hernando, 2013. *Wiring* [online]. [cit. 2013-06-04]. Dostupné z: <http://wiring.org.co/>
- BÍLEK, Petr. Jak vzniká program. *Sally* [online]. 29.8.2003, 10.9.2008 [cit. 2013-06-04]. Dostupné z: <http://www.sallyx.org/sally/c/c03.php>
- Computer programming, © 2013. *Business Dictionary* [online]. [cit. 2013-04-07]. Dostupné z: <http://www.businessdictionary.com/definition/computer-programming.html>
- FREESCALE SEMICONDUCTOR, [2005]. *M68EVB908GB60*. Garland,. Dostupné z: [http://www.freescale.com/files/soft\\_dev\\_tools/doc/user\\_guide/M68EVB908GB60UM.pdf](http://www.freescale.com/files/soft_dev_tools/doc/user_guide/M68EVB908GB60UM.pdf)
- HRBÁČEK, Jiří, 1999. *Komunikace mikrokontroléru s okolím*. Praha, 156 s. ISBN 80-860-5642-2.
- JÁNEŠ, Vlastimil, 2012. Jednočipové mikropočítače. *České vysoké učení technické v Praze* [online]. [cit. 2013-03-16]. Dostupné z: [http://www.fd.cvut.cz/personal/janes/HWpocitacu/prednasky2012/prednaska9/HW\\_prednaska9.pdf](http://www.fd.cvut.cz/personal/janes/HWpocitacu/prednasky2012/prednaska9/HW_prednaska9.pdf)
- MAJDA, David, © 2000–2010. Knihovny vs. frameworky. *David Majda - osobní stránky* [online]. [cit. 2013-04-08]. Dostupné z: <http://majda.cz/zapisnik/265>
- MANN, Burkhard, 2003. *C pro mikrokontroléry: ANSI-C, kompilátory C, spojovací programy - linkery, práce s ATMELE AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky*. Vyd. 1. Praha: BEN, 279 s. ISBN 80-730-0077-6.
- MAVRIC-IIB, [2005]. *BDMICRO* [online]. [cit. 2013-04-09]. Dostupné z: <http://www.bdmicro.com/mavric-iib>
- ORENSTEIN, David, 2000. Application Programming Interface (API). *Computerworld* [online]. 10.1.2000 [cit. 2013-04-09]. Dostupné z: [http://www.computerworld.com/s/article/43487/Application\\_Programming\\_Interface](http://www.computerworld.com/s/article/43487/Application_Programming_Interface)
- PINKER, Jiří, 2004. *Mikroprocesory a mikropočítače*. 1. vyd. Praha: BEN - technická literatura, 159 s. ISBN 80-730-0110-1.
- Processing.org* [online]. [cit. 2013-06-03]. Dostupné z: <http://www.processing.org/>
- REDI Education Board, © 2013. *Rogue robotics* [online]. [cit. 2013-04-09]. Dostupné z: <http://www.roguerobotics.com/products/electronics/redi>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

ALU	Arithmetic and Logic Unit
B	Byte
b	bit
AD	Analog to Digital converter
DA	Digital to Analog converter
IDE	Integrated Development Environment
PWM	Pulse Wide Modulation
LCD	Liquid Crystal Display
SPI	Serial Peripheral Interface
LED	Light Emitting Diode
USB	Universal Serial Bus
I2C	Inter Integrated Circuit
GSM	Global System for Mobile communications
WiFi	Wireless Fidelity
API	Application Programming Interface

**SEZNAM OBRÁZKŮ**

Obr. 1: Vývojový kit Wiring S.....	16
Obr. 2: Vývojové prostředí Wiring.....	18
Obr. 3: Freescale M68EVB08GB60.....	29
Obr. 4: Makro pro zjištění adresy vstupního registru podle portu.....	32
Obr. 5: Funkce <code>_pinRead()</code> .....	33
Obr. 6: Standard Settings.....	36
Obr. 7: Otevřený projekt v CodeWarrioru.....	36
Obr. 8: Výběr mikropočítače.....	37
Obr. 9: Vložení souborů do projektu.....	38
Obr. 10: Ukázkový program č. 1.....	39
Obr. 11: Ukázkový program č. 2.....	40

## SEZNAM PŘÍLOH

P I Elektronická příloha zdrojových kódů a verze bakalářské práce na CD