FAKULTA
STROJNÍ
ČVUT V PRAZE

Ústav mechaniky, biomechaniky a mechatroniky

Využití platformy PSoC pro mechatronické modely

Using the PSoC platform for mechatronic models

BAKALÁŘSKÁ PRÁCE

2017

Marek Michalák

Studijní program: B2342 TEORETICKÝ ZÁKLAD STROJNÍHO INŽENÝRSTVÍ
Studijní obor: 2301R000 Studijní program je bezoborový
Vedoucí práce: Ing. Petr Beneš, Ph.D.

Anotační list

Jméno autora:	Marek Michalák
Název bakalářské práce:	Využití platformy PSoC pro mechatronické modely
Anglický název:	Using the PSoC platform for mechatronic models
Akademický rok:	2016/2017
Obor studia:	Bez oboru
Ústav/odbor:	Ústav mechaniky, biomechaniky a mechatroniky Odbor mechaniky a mechatroniky
Vedoucí bakalářské práce:	Ing. Petr Beneš Ph.D.
Bibliografické údaje:	Počet stran: 32 Počet obrázků: 24 Počet příloh: 1
Klíčová slova:	PSoC, jednočipový systém, rovinný manipulátor
Keywords:	PSoC, system on chip, planar manipulator

Anotace: Tato práce se zabývá platformou PSoC. Úvod je věnován seznámení se se společností Cypress, která PSoC vyvíjí, přiblížení struktury, výkonnostních řad a možností jejich čipů. Dále je uvedeno několik základních příkladů pro pochopení principu tvorby blokových schémat a programů. Konkrétně ovládání LED, použití displeje a integrovaného potenciometru. To vše spojeno v programech pro ovládání servomotoru a DC motoru. Poslední část je věnována experimentálnímu využití této platformy pro obsluhu rovinného manipulátoru. V tomto případě pro měření povrchu ultrazvukem. Je také zařazen popis jednotlivých částí kódu a blokových schémat. V závěru jsou zhodnoceny výsledky experimentu.

Abstract: The main focus of this thesis is on platform PSoC. In the first part, there is an introduction of Cypress company, which is a developer of PSoC platform, introduction of structure, performance lines and possible use of their chips. In the next part, there are shown some basic examples, which should help to understand the principles of creation of block schemes and programs. Among these examples can be found LED controlling, display and integrated potentiometer using. All of these principles are connected to the programs for control of the servomotor and the DC motor. The last part focuses on the experimental use of this platform for handling a planar manipulator – in this case for surface measurement via ultrasound. The description of individual parts of codes a block schemes can be also found in the same chapter. In the conclusion, the results of the experiment are evaluated.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

V Praze, dne …… .....

Podpis

Poděkování

Chtěl bych poděkovat panu Ing. Petru Benešovi Ph.D. za cenné rady a vedení při zpracování této práce. V neposlední řadě své rodině a přátelům za dosavadní podporu a motivaci při studiu.

Obsah

Seznam obrázků

1. Úvod

V dnešní době se požadavky na automatické řízení stále zvyšují, proto je zapotřebí inovovat řídicí platformy. Je požadován větší výkon, stabilita, realtimové zpracování informací, jednoduchost použití a nízká cena. Nízkonákladové platformy jako Arduino jsou pro takové požadavky nedostačující, a na druhou stranu výkonná industriální technika je velice drahá. PSoC je střední cestou, který zvládá složitější aplikace, ale cenově se blíží spíše k nízkonákladovým platformám. Jeho unikátní architektura čipu a propracovaného softwaru umožňuje tvořit programy rychle a efektivně. Existuje také řada modulů, které ještě rozšiřují možnosti využití. I přes to, že není tak známý jako jiné platformy, má širokou základnu uživatelů, kteří sdílí své projekty na oficiálních stránkách výrobce, vzájemně si radí a přinášejí nová řešení problémů, což programování a logické uspořádání kódů velice zjednodušuje. Veškerá dokumentace je nejsnáze dostupná online. Výrobce na to spoléhá, proto nabízí mimo jiné celou řadu návodů a video manuálů, ovšem jen v anglickém jazyce. V češtině není k této platformě žádná podpora.

Cílem mé práce je seznámení se s vývojovým kitem CY8CKIT-050 PSoC, provést jednoduché experimenty a následně je popsat. Dále vytvořit jednoduchý mechatronický model řízený tímto vývojovým kitem.

1.1. Cypress

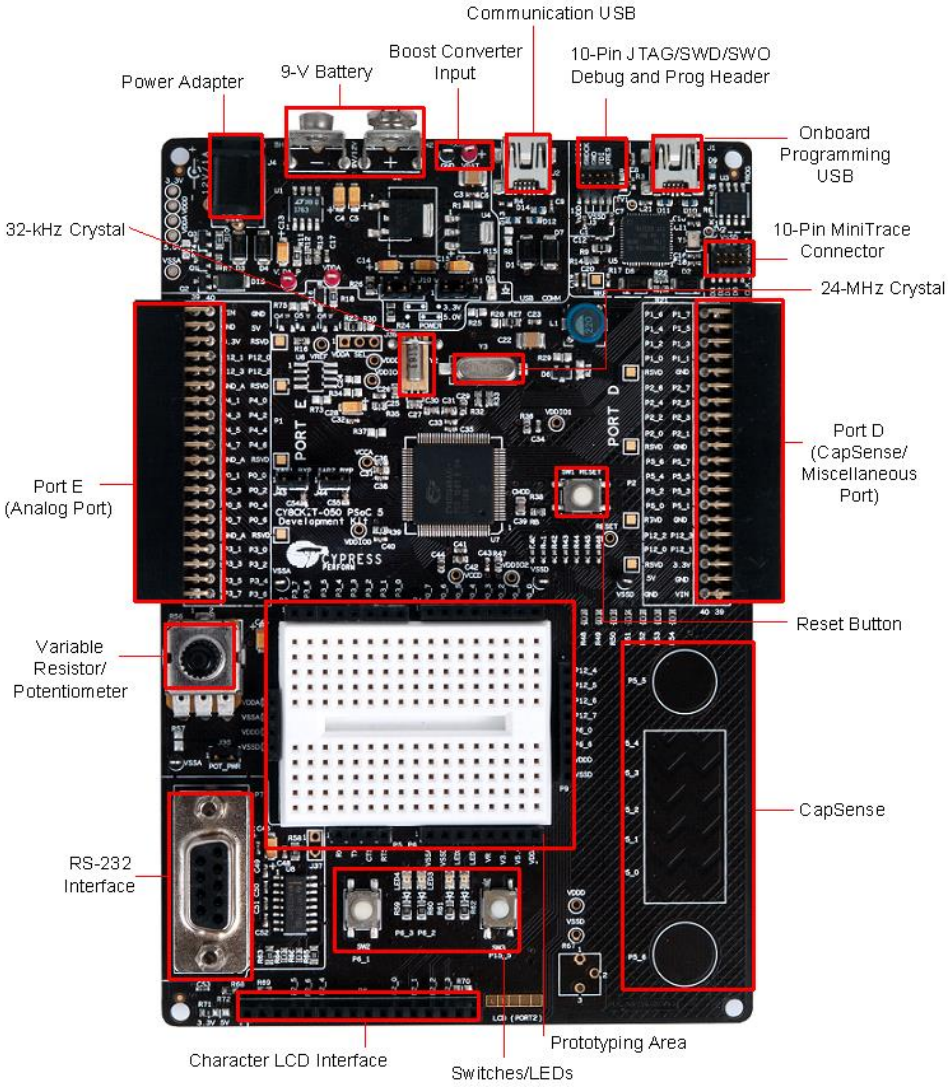
Platformu PSoC vyvinula společnost Cypress, Embedded in tomorrow, založená roku 1982 v Sillicon Valey, Kalifornii, Spojených státech amerických. Společnost se považuje za leadera na poli dotykových snímačů. Soustředí se na mnoho odvětví průmyslu a na výrobce spotřební elektroniky. Vyvíjí také NOR a RAM paměti, bezdrátová zařízení, a energeticky nenáročný hardware. Většinu z těchto věcí kombinuje ve svém produktu PSoC, což znamená Programmable System on Chip (Programovatelný systém na jednom čipu).[1]

1.2. PSoC

Jedná se o velmi dobře programovatelnou platformu, založenou na snadně konfigurovatelném a otevřeném hardwaru, který se dá jednoduše používat s volně dostupným softwarem. Je velice flexibilní, a neustále rozvíjen. Programuje se pomocí jazyka C.

Je určen široké škále programátorů, jako jsou studenti, domácí kutilové, ale také profesionální inženýři či výzkumníci.

Srdce PSoC je tvořeno mikrokontrolérem, které zahrnuje procesor, paměť, pole zpracování analogového signálu, digitální pole a programovatelné vstupy a výstupy. Může být využit pro všemožné aplikace kolem nás, jako například v domácí elektronice, mobilních zařízeních, industriální technice či pro medicínské účely [12]. Existuje v několika různých konfiguracích, a to od nejpomalejšího procesoru 8-bitů M8C – PSoC 1, až po nejrychlejší 32-bit Cortex M3 – PSoC 5LP, který mi byl zapůjčen (Obrázek 1)[2]



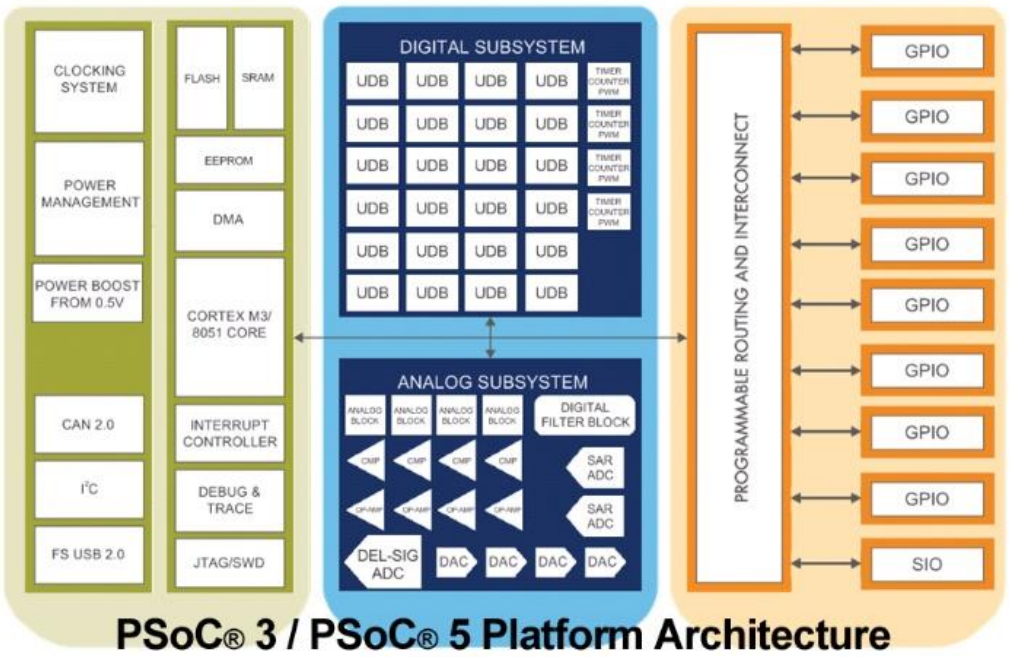
Obrázek 1 PSoC 5LP
(http://www.cypress.com/file/45276/download)

Největší výhodou je programovatelná logika. Spojením CPLD – Complex Programmable Logic Device (komplexní programovatelné logické obvody) a logických funkcí umí vytvořit digitální obvod. Toho docílí pomocí UDB – Universal Digital Block (univerzálních digitálních bloků), které po propojení umožnují vytvořit digitální funkci jen s využitím hardwaru. To znamená, že se pro každou aplikaci dokáží bloky tohoto pole přepsat pro odlišné účely. Jednou je například sled bloků využit jako časovač, a v dalším programu se z něj stane PWM blok nebo počítadlo, což je velice užitečné pro vytvoření vysoce konfigurovatelné a spolehlivé hardwarové aplikace.

Využívá také programovatelné analogové obvody, které dokáží interagovat se signály z okolí v reálném čase, a nabízí několik možností zpracování analogového signálu.

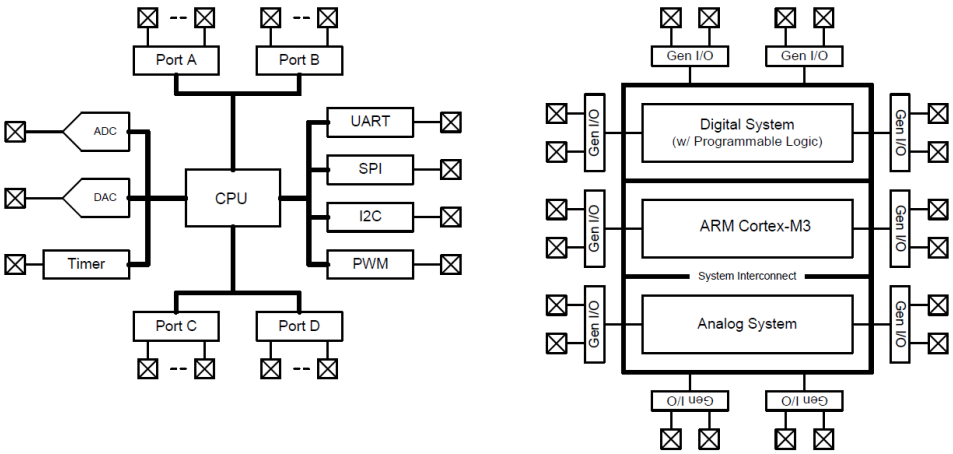
Systém GPIO (general purpouse IO – pin, který se stane vstupem či výstupem v závislosti na programu), umožňuje propojit a namapovat kteroukoliv funkci na jakýkoliv pin zařízení, což je velice výhodné, a zjednodušuje práci nejen při používání extérních hardwarových zařízení.

Zjednodušené schéma čipu PSoC (Obrázek 2). Podle řady procesoru se liší počet jednotlivých bloků, ale základní uspořádání je stejné.



Obrázek 2 Schéma čipu PSoC
(http://www.cypress.com/file/36086/download)

Všechny uvedené vlastnosti této platformy jí odlišují od ostatních, jako je Arduino, Rasbery Pi a další. Hlavní rozdíl je v tom, že PSoC dokáže zpracovávat signály, používat funkci PWM, různé protokoly jako I2C, CAN bus, a mnoho dalších věcí, a to vše bez použití procesoru. Na následujícím schématu (Obrázek 3) je vlevo vyobrazen obvod běžných platforem, vpravo PSoC. Je z něj patrné, že procesory ostatních platforem se musejí zaobírat každou akcí, protože jiná komunikace mezi prvky není možná, a tím jsou zpomalovány. Na rozdíl od nich umí PSoC vyřídit tyto záležitosti v periferních obvodech, a výkon procesoru je využit převážně na hlavní program [5].



Obrázek 3 Standardní platforma ve srovnání s čipem PSoC
(http://www.cypress.com/file/41436/download)

1.3. PSoC Creator

Součástí PSoC je také software nazvaný PSoC Creator, který je stěžejní pro vytvoření programu. Po krátkém seznámení se stane poměrně přehledným a jednoduchým nástrojem pro tvorbu programů. Jeho vývoj jde stále dopředu, a s každou novou verzí je vylepšován.

PSoC Creator je integrované prostředí, které umožnuje tvorbu firmwaru, kompilování programů a debugging systémů. V prostředí je možné využít mnoho z předdefinovaných programovacích bloků, kterých je v nabídce asi 200. Ty se velice jednoduše propojí, čímž vytvoří požadované schéma, logické zapojení, analogovou funkci a mnoho dalšího. Umožňuje exportovat projekty do univerzálních formátů pro softwary třetích stran.

Prostředí Creatoru dává prostor pro flexibilitu tím, že se dají vytvořit programy rozličnými způsoby, přičemž jsou k dispozici bloková schémata, nebo prostý editor C kódu. Při jejich zkombinování získáme jednoduché a přehledné schéma, kde na první pohled vidíme, co které prvky přímo či nepřímo ovlivňuje, a můžeme se místo zaobírání hardwarem soustředit na ladění softwaru [4].

2. Základní úlohy pro PSoC 5LP

V této kapitole se budu zabývat popisem jednoduchých úloh, které jsou základními kameny pro ovládání mikrokontrolérů, jako je čtení a zápis na piny, převod analogového vstupního signálu na digitální výstup, a v tomto případě výpis hodnot na připojený displeji.

Všechny popsané úlohy budou zpracovány v následujících 3 návodech s využitím platformy PSoC.

Veškeré obrázky týkající se kódu a blokových schémat pocházejí z prostředí PSoC Creator 3.0 a jsou vytvořeny autorem této práce.

2.1. Software a potřebná nastavení

Pro správné fungování je nutné si stáhnout ovládací software – PsoC Creator.

Nejnovější verze se dá stáhnout na stránce:

http://www.cypress.com/products/psoc-creator-integrated-design-environment-ide nebo použijeme starší software na přiloženém CD.

Po úspěšném nainstalování programu si sestavíme PSoC. Připojíme displej tak, aby směřoval ven, a zapojíme mini USB do konektoru blíže ke kraji (ten je určen pro programování, druhý konektor je pro čtení dat.). Poté připojíme pomocí USB do PC. Vedle zeleně svítící LED najdeme 2 spojky. Vzdálenější nastavíme podle nákresu na požadovaný režim napětí, ve kterém bude PSoC fungovat. Pro náš případ 5V. Pokud by nějaký připojovaný prvek vyžadoval napětí o maximální hodnotě 3,3V, nemusíme jej připojovat přes napěťové děliče. Jen přemístíme spojku do druhé polohy.

Spustíme program PSoC Creator. Po spuštění se dostaneme na hlavní stránku, kde můžeme vytvořit zcela nový projekt, nebo použít jeden z mnoha příkladových. Vždy je nutné v kategorii „Architecture“ zvolit „PSoC 5LP“.

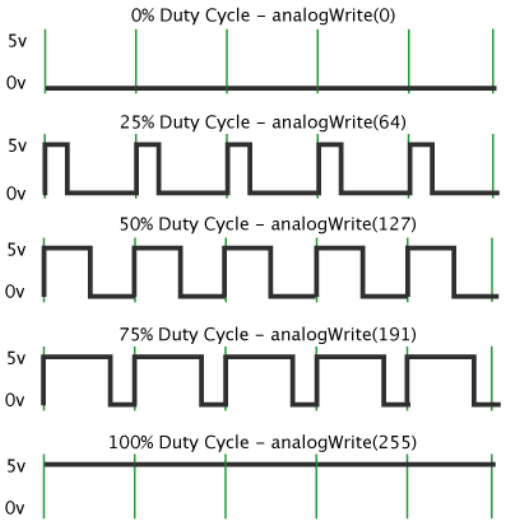
Po otevření projektu, ať už příkladového nebo nového, nás budou zajímat hlavně soubory main.c, TopDesign.cysch a „*název projektu*“.cydwr.

2.2. Blikání s LED

Nejjednodušší způsob blikaní s LED, kdy si přesně nastavíme dobu zapnutí a vypnutí je pomocí PWM. Tento způsob rovněž nezatěžuje procesor, jelikož ho zvládají periférie, jak již bylo řečeno v popisu platformy. Proto můžeme blikat v podstatě libovolným počtem LED, aniž bychom zatížili chod programu, ať už funkcí delay nebo jen samotným aktem blikání.

2.2.1. Popis PWM

PWM neboli Pulse Width Modulation (pulzně šířková modulace). Princip je takový, že máme obdélníkový signál, o jisté periodě. V signálu mohou nastat jen 2 stavy: +5V a 0V, neboli HIGH a LOW. Střídáním těchto stavů může simulovat napětí v tomto rozmezí, a to změnou poměru signálů HIGH/LOW v každé periodě. Čím delší bude úsek HIGH, tím vyšší bude průměrný proud a naopak. Úsek ve stavu HIGH je nazývá Duty Cycle, a v procentech vyjadřuje poměr ke stavu LOW (Obrázek 4). Využít se toho dá 2 způsoby. Buď bude perioda dlouhá (např. 1s), a LED bude přecházet mezi stavy zapnuto vypnuto nebo bude perioda velice krátká (např. 80-100ms), a tím budeme schopni řídit jas LED, v důsledku nižšího průměrného proudu. Musíme si ale uvědomit, že nižší průměrný proud je v určitém časovém úseku, ale okamžitá hodnota při stavu HIGH je 5V, takže i při použití PWM pro regulaci jasu LED je nutné použít předřadný odpor pro ty, které 5V nesnesou. V opačném případě bychom LED velmi rychle spálili. PWM má širokou škálu využití v mnoha aplikacích [6].

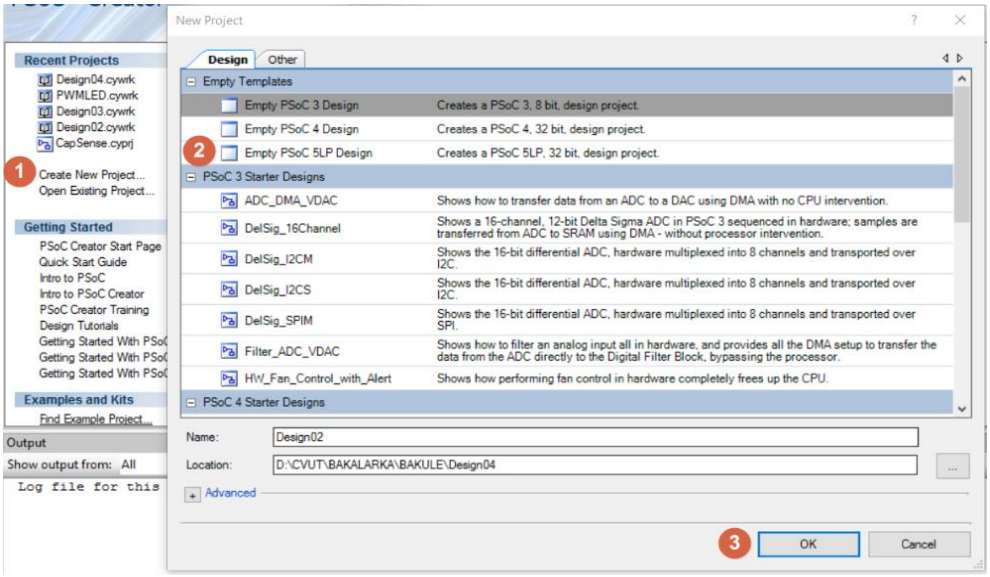


Obrázek 4 Ukázka signálu PWM s hodnotou Duty Cycle (https://www.arduino.cc/en/uploads/Tutorial/pwm.gif)

2.2.2. Návod na blikání s LED pomocí blokových schémat

Sestavíme PSoC jak bylo popsáno v úvodu a spustíme PSoC Creator.

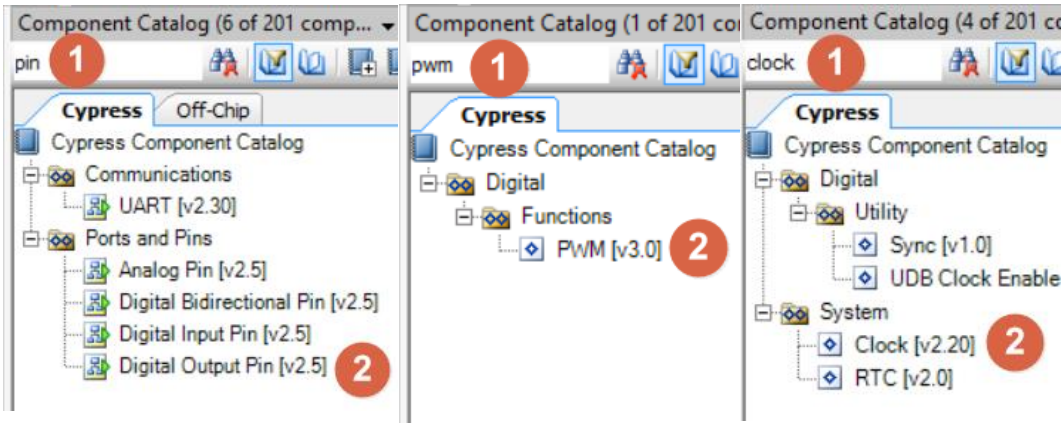
Po otevření programu vybereme možnost Create New Project, a dále Empty PSoC 5LP Design (Obrázek 5).



Obrázek 5 Otevření nového projektu

2.2.2.1. Nastavení bloků a pinů

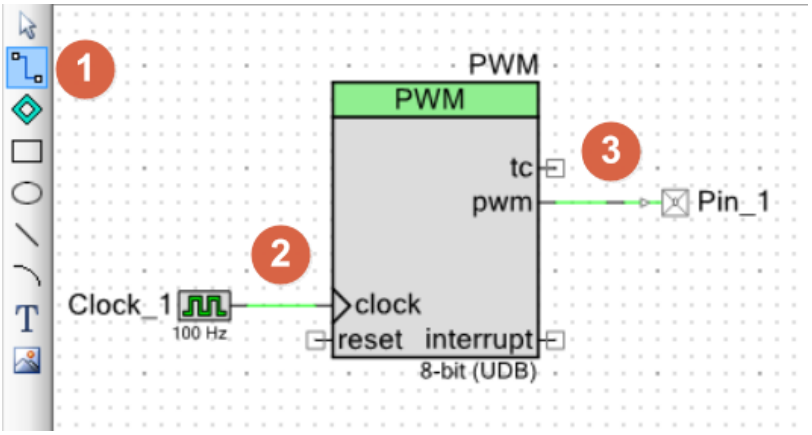
Otevře se nám prostředí blokových schémat TopDesign.cysch. V levé části máme menu, kde nalezneme všechny potřebné soubory, které se nám po otevření zobrazí také v horních záložkách. V pravé části se nachází Component Catalog, kde můžeme dohledat různé řídicí bloky. Do „Search“ napíšeme „pin“. V nabídce vybereme „Digital Output Pin“ (Obrázek 6), a přetáhneme jej na bílou pracovní plochu. Podobně přidáme „PWM“, a „Clock“.



Obrázek 6 Výběr bloků z katalogu

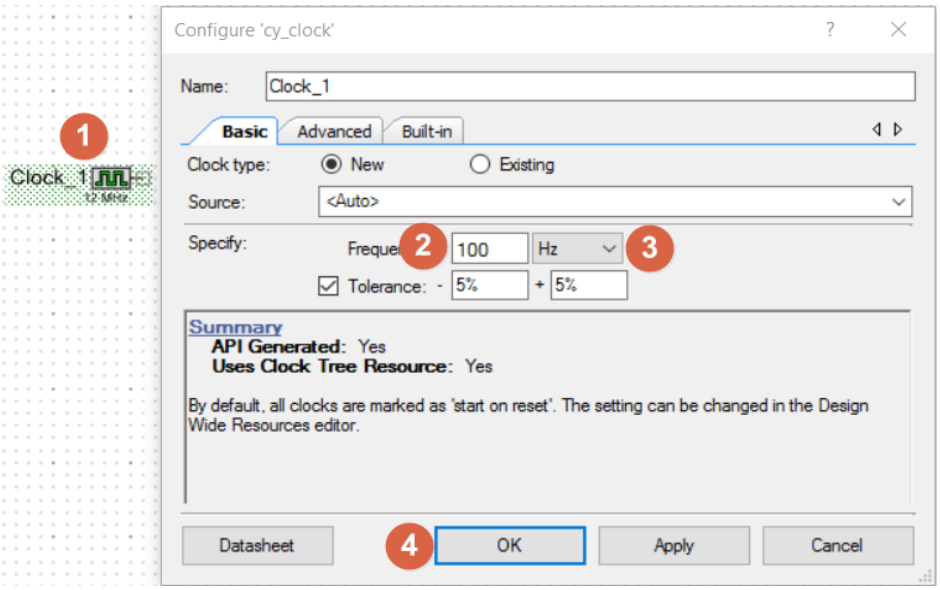
Pokud se budete chtít dozvědět více o daném bloku, stačí na něj kliknout pravým tlačítkem a otevřít si Datasheet nebo najít příkladový projekt s daným blokem (Find Example Project). Datasheety jsou poměrně obsáhlé a najdeme v nich vše potřebné od popisu fungování a zapojení, po způsob programování v C.

Po přidání bloků vybereme nástroj Wire Tool, a propojíme bloky naznačeným způsobem. (Obrázek 7)



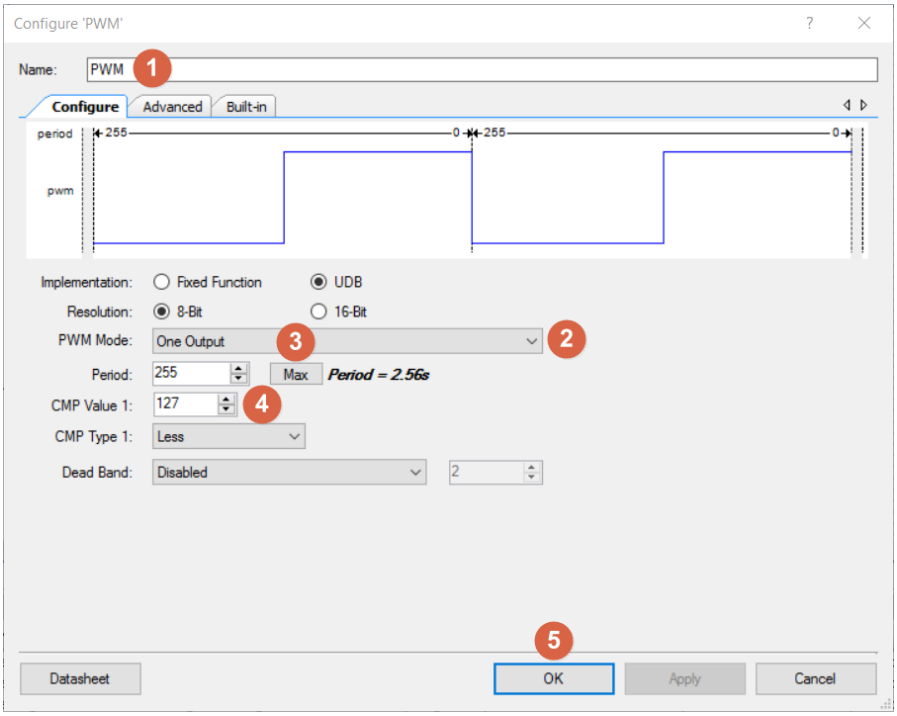
Obrázek 7 Vytvoření schématu pro LED

Nastavíme blok Clock_1 tak, že na něj 2x klikneme, a zvolíme hodnotu. Nám bude stačit 100 Hz (Obrázek 8).



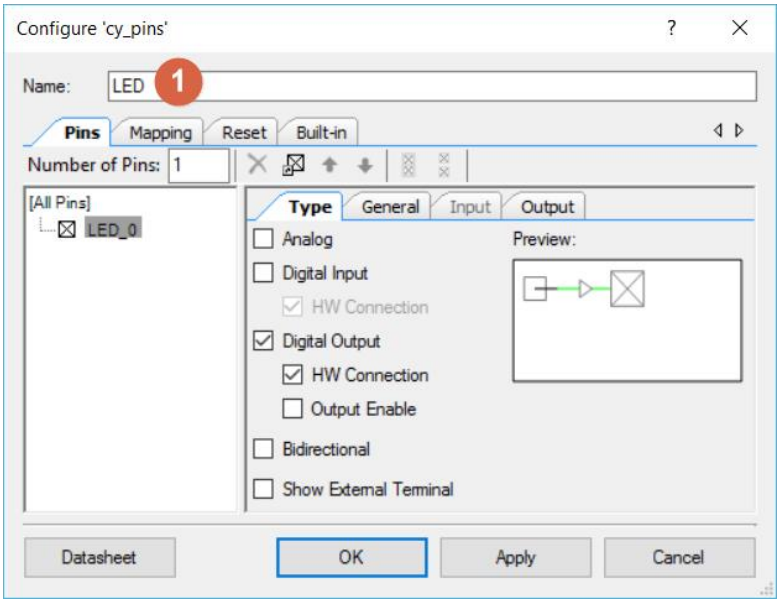
Obrázek 8 Nastavení Clock

Nyní nastavení bloku PWM (Obrázek 9). Zde si můžeme zvolit poměr HIGH-LOW, délku periody a počet výstupů. Nejprve přejmenujeme blok jen na PWM. V PWM Mode vybereme „One Output“. Dále si nastavíme periodu tlačítkem „Max“, a část periody, po kterou bude zapnutý. Pro nás 127.



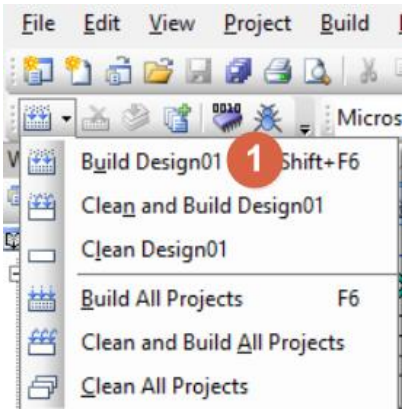
Obrázek 9 Nastavení bloku PWM pro LED

V nastavení Pin_1 pouze změníme jméno na LED (Obrázek 10).



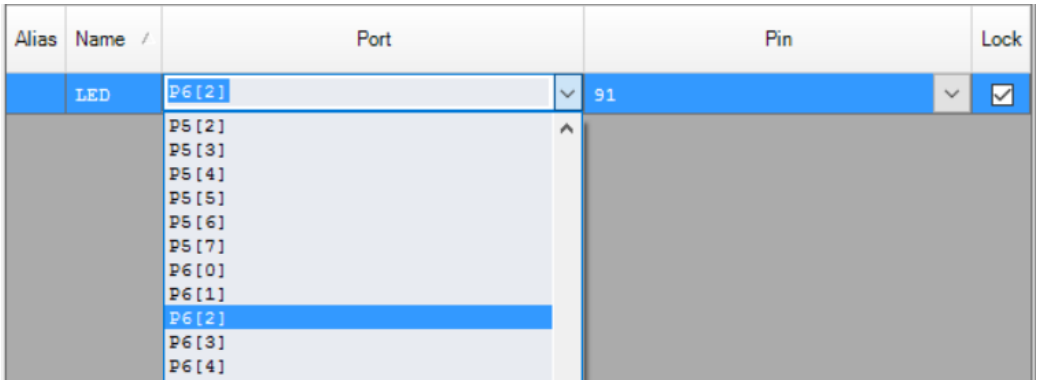
Obrázek 10 Nastavení Pin

V horním menu vybereme možnost Build „Design01“ (v závislosti na názvu projektu) a vyčkáme, dokud se nám v dolní konzoli nevypíše „Build Succeeded“ (Obrázek 11).



Obrázek 11 Build Design

Přepneme se do souboru Design01.cydwr, kde v pravém menu přiřadíme názvu LED Port P6 [2] (Pin 91) (Obrázek 12), na kterém je na desce připojena dioda.



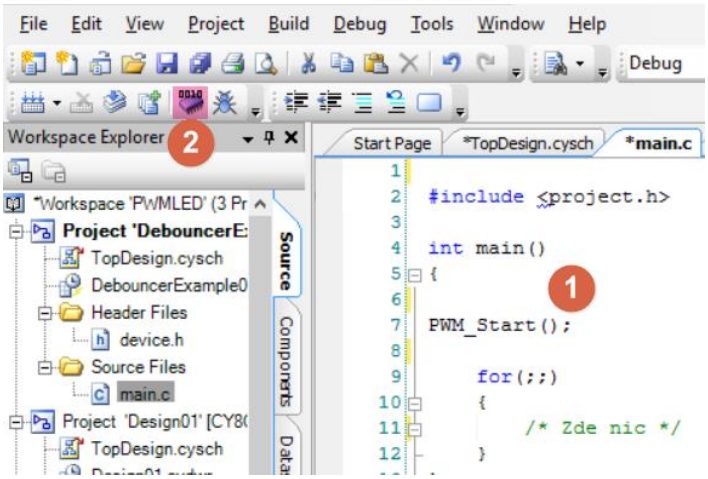
Obrázek 12 Přiřazení pinů

2.2.2.2. Kód

Nakonec do souboru *„main.c“* napíšeme následující kód. Jedná se jen o jeden řádek kódu navíc.

```
#include <project.h>
int main()
{
PWM_Start(); //Spustí blok PWM

    for(;;)
    {
 /* Zde nic */
    }
}
```

V menu zvolíme „Program“ a náš program se nahraje do PSoC (Obrázek 13).



Obrázek 13 Nahrání programu do PSoC

Po chvíli by měla začít blikat dioda s periodou 1,27 s

2.2.3. Návod na blikání s LED čistě pomocí kódu

Druhá metoda jen pomocí kódu je neefektivní, jelikož je třeba využít čas a výkon procesoru, přičemž funkce delay zastaví veškeré procesy. Dal by se vytvořit kód pomocí funkce rozdílů časů, ale PSoC nemá žádný jednoduchý časovač, ani nepočítá žádný jednoduše dostupný čas od začátku programu. Proto by se musel použít blok Timer, který se složitě nastavuje, nejednalo by se již o téměř čistý kód, a pro blikání s LED není tak efektivní jako první příklad s PWM.

2.2.3.1. Nastavení pinů

V panelu blokových schémat přidáme jen Output Pin. V nastavení jej přejmenujeme na PIN a odškrtneme políčko „HW conection“. Vybereme pin s LED, který přiřadíme v souboru „Design01“.cydwr

2.2.3.2. Kód

Do souboru main.c zkopírujeme tento kód a klikneme na Program.

```
#include <project.h>
int main()
{

    for(;;){
      LED_Write(~LED_Read());  //Zapíše opačný stav výstupu pinu LED
                               //"~" znamená NOT, opak (1=>0, 0=>1)
      CyDelay(1000);           //CyDelay(hodnota v ms)
      }}
```

LED bude blikat s intervalem 1 s. Ten určuje hodnota argumentu v CyDelay v ms.

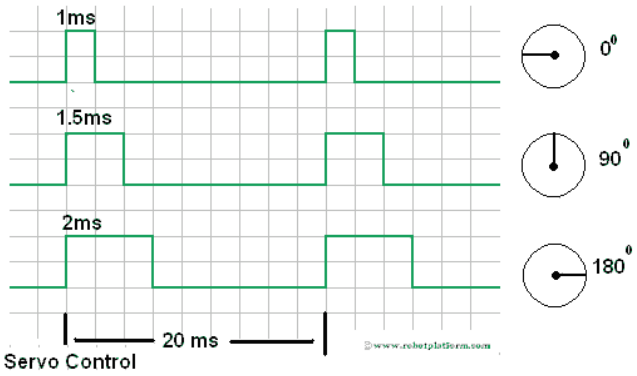
2.3. Ovládání servomotoru integrovaným potenciometrem s výpisem polohy na displeji

Ukážu, jak jde poměrně přesně ovládat pozice servomotoru pomocí potenciometru. Pokud bychom použili přesnější vstupní signál nebo generovali hodnoty na základě jiného podnětu než natočení potenciometru, mohli bychom dosáhnout přesnosti natočení až 0,06°.

2.3.1. Popis fungování servomotoru

Servomotor se skládá z motoru, který je připojen na převodovku dopomala. K výstupní hřídeli je také připojen potenciometr, který na základě změny odporu, která se projeví změnou napětí, dokáže odměřit, v jaké poloze se zrovna hřídel nachází.

Servomotor se ovládá pomocí délky impulzů PWM. Elektronika zaznamenává impulzy dlouhé 20 ms. Příchozí signál HIGH dlouhý 1,5 ms odpovídá úhlu 90°, co je neutrální úhel servomotoru (Obrázek 14). Pokud bude signál kratší, hřídelka se otočí směrem k minimu, pokud delší, k maximu. Každé servo má jiný rozsah od neutrální polohy, proto jsou v kódu zavedeny konstanty „Border1„ a „Border2“ na vykompenzování těchto lehkých rozdílů, aby měl servomotor plný rozsah otáčení. Každých 20 ms musí servo přijmou informaci o nastavované pozici, kterou po dosažení drží, i když na něj působí vnější zatížení. Krouticí moment, kterému dokáže servo odolat, popřípadě který dokáže servo vyvinout je jednou z jeho hlavních charakteristik.



Obrázek 14 Pozice servomotoru v závislosti na délce signálu

(http://www.robotplatform.com/knowledge/motion_control/servo-control.png)

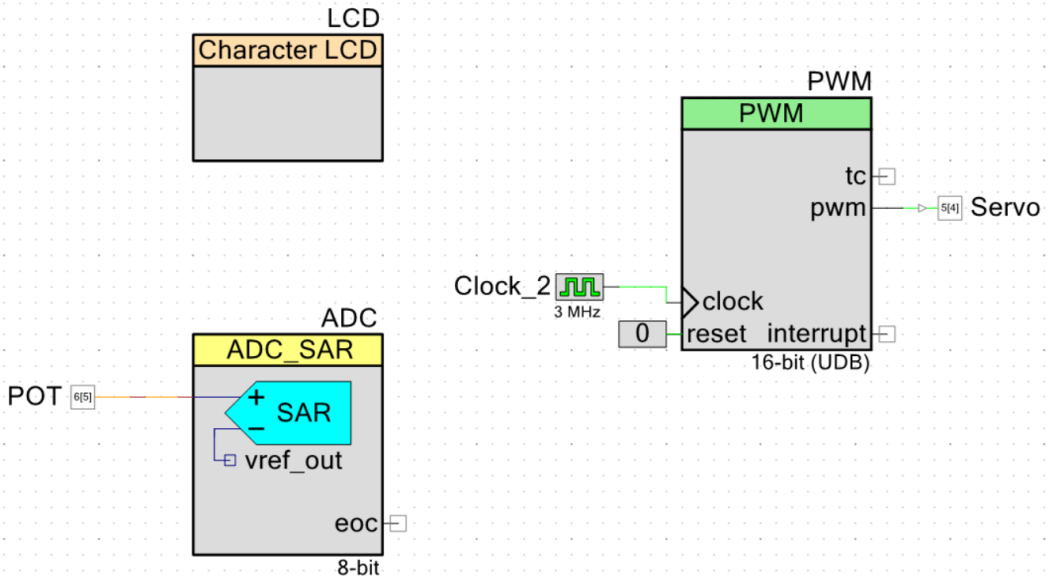
2.3.2. Zapojení servomotoru

Každý servomotor má 3 dráty. Černý nebo hnědý je ZEM (ground, GND), červený je pak napájení – pro malá serva standardně 5-12V pro použití s mikrokontroléry. Vstupní neboli řídící pak bílý nebo oranžový. Servo připojíme na PSoC podle uvedených barev nejlépe na piny GND a 5V. Řídicí pin zapojíme do námi definovaného Output Pinu. Viz níže.

2.3.3. Nastavení bloků a pinů

V této úloze bude zapotřebí větší počet bloků. Věnujte prosím pozornost názvům bloků a pinů, a jejich vzájemnému propojení.

Otevřeme si nový projekt a opět použijeme Component Catalog pro vyhledání bloků a vytvoříme si následující schéma (Obrázek 15):



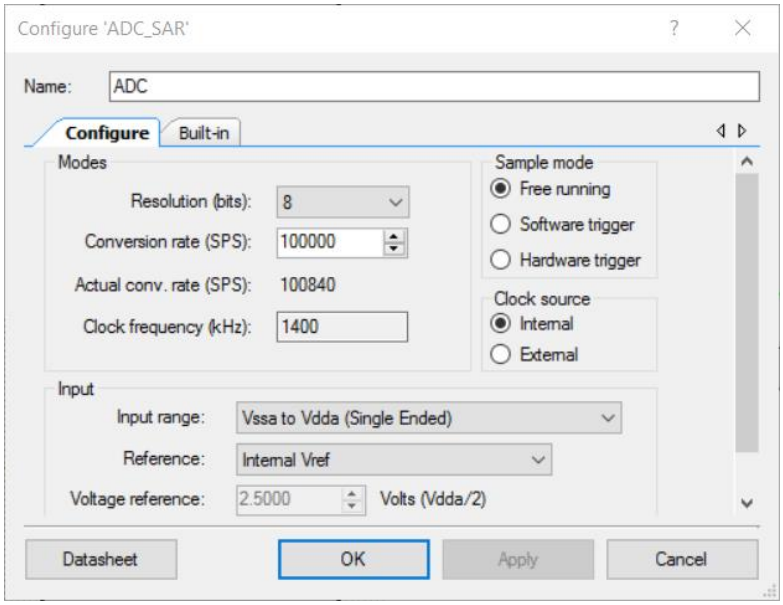
Obrázek 15 Schéma pro ovládání servomotoru

Pin POT je analog input pin a pin Servo je digital output pin. Hodnotu „0“ najdeme pod názvem „LOW“.

Blok Clock nastavíme na 3 MHz. U LCD_char_1 jen přepíšeme jméno na LCD.

V bloku PWM toho nastavíme více. Zvolíme UDB, 16-Bit, One Output, Period 60 000 (mělo by se nám zobrazit 20 ms) a změníme jméno pouze na PWM.

Prvek ADC_SAR nastavíme následovně (Obrázek 16):



Obrázek 16 Nastavení bloku ADC_SAR

Po sestavení bloku a nastavení všech hodnot dáme build a vyčkáme na kompilaci. V záložce „název“.cydwr přiřadíme LCD porty P2[6:0], Pin 95…99, 1…2 (velice důležité!), integrovaný PSoC potenciometr s názvem POT je hardwarově na pinu P[6]5. Servo můžeme připojit kamkoli. Například pin P1[6] (nejblíže k USB portu).

2.3.4. Kód

Jelikož není v PSoC integrovaná jednoduchá syntaxe či makro na analogový vstup jako je tomu například u Arduina, je kód poměrně dlouhý. Po jeho vložení do main.c klikneme na tlačítko program.

```
#include <device.h>
#include "stdio.h"
#define MAX_SAMPLE  8

void main()
{
       LCD_Start(); //Start LCD display block
       PWM_Start(); //Start PWM block
                           // Variable to set servo Borders of turn
   int Border1 = 1555;     // Change if servo is not at Border when angle is 0°
   float Border2 = 1.31;   // Change if servo is not at Border when angle is 210°

   int32 Count = 0;        // Variable to hold ADC count
   int32 mV = 0;           // Variable to hold the result in mV converted from ADC
   uint8 sampleCount = 0; //Variable to count number of collected samples
   int32 Samples = 0;      // Variable to hold cumulative samples
   uint32 average = 0;     // Variable to hold the average volts for 8 samples
   int32 ser=0;            // Variable to hold value for servo
   uint8 angle=0;          // Variable to hold value for LCD to print
   char displayStr[6] = {'\0'};  //String for LCD
   char displayStr1[6] = {'\0'}; //String for LCD

    ADC_Start(); //Start ADC and start conversion
    ADC_StartConvert();

       LCD_ClearDisplay();        //Clear Display
       LCD_Position(0,0);         //Write static text on Display
       LCD_PrintString("POT [mV] =");
       LCD_Position(1,0);
       LCD_PrintString("servo[o] =");

       while(1) //Loop
       {
    ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);//Read ADC count and convert to mV
    Count = ADC_GetResult16();
    mV = ADC_CountsTo_mVolts(Count);//Convert count to mV
    Samples = Samples + mV;// Add the current ADC reading to cumulated samples
    sampleCount++;

/* If 8 samples have been collected then average the samples and update the
display*/
       if(sampleCount == MAX_SAMPLE)
       {
         average = Samples >> 3;
         Samples = 0;
         sampleCount = 0;
         ser = average*Border2 + Border1;
         angle=(average)/(23.7);
       }
         PWM_WriteCompare(ser); //Set new Compare value for PWM
         sprintf(displayStr,"%4ld",average); //transform number to text line
         sprintf(displayStr1,"%4ld",angle);    //transform number to text line
         LCD_Position(0,12);            //Print Values on display
         LCD_PrintString(displayStr);
```

```
            LCD_Position(1,12);
            LCD_PrintString(displayStr1);
        }
}
```

Pomocí potenciometru ovládáme úhel natočení servomotoru ve stupních, který je zobrazován na druhém řádku displeje. Na prvním řádku můžeme vidět proud v mV, který měříme na potenciometru.

2.4. Ovládání motoru integrovaným potenciometrem s výpisem rychlosti na displeji

Ovládání DC motoru jen s využitím platformy PSoC je možné pouze v jednom smyslu otáčení. Pokud bychom chtěli motor ovládat v obou směrech, budeme potřebovat další externí hardware, a to tzv. H-Bridge, kde se o přepínání směrů stará čtveřice tranzistorů.

2.4.1. Zapojení motoru:

Jeden konektor z motoru zapojíme do portu označeného jako GND, druhý do námi zvoleného portu při volení v záložce „Název“.cydwr – Například pin P12_0, který je blízko GND. Pokud chceme, aby se motor točil obráceně, prohodíme kontakty.

2.4.2. Nastavení bloků a pinů

V našem případě se bude motor otáčet jen jedním směrem. Použijeme stejné sestavení bloků jako u ovládání servomotoru (Obrázek 15). Bude se ovšem lišit nastavení jednotlivých bloků.

Konkrétně PWM_1 přejmenujeme na PWM, dále hodnoty nastavíme na UDB, 8-bit, One Output, Period 255. Jinak vše necháme na předvoleném nastavení.

V bloku ADC_SAR nastavíme vše jako u servomotoru. Viz (Obrázek 16)

Clock změníme na 1 MHz a Output pin přejmenujeme na Motor.

V záložce „Název“.cydwr nastavíme piny, a to jako LCD P2[6:0], Pin 95…99, 1…2, POT jako P6[5] a poslední na libovolný port, například již zmiňovaný P12[0].

2.4.2.1. Kód

Je velice podobný jako u serva. Změní se pouze vysílané PWM hodnoty na výstupu. Do souboru main.c vložíme následující kód:

```
#include "project.h"
#include "stdio.h"

#define MAX_SAMPLE  8

int main(void)
{
       LCD_Start(); Start LCD display block
       PWM_Start(); //Start PWM block

   int32 Count = 0;// Variable to hold ADC count
   int32 mV = 0;  // Variable to hold the result in mV converted from ADC counts
   uint8 sampleCount = 0;  // Variable to count number of collected samples
   int32 Samples = 0;      // Variable to hold cumulative samples
   uint32 average = 0;     // Variable to hold the average volts for 8 samples
   long int amount=0;      // Variable to hold value for LCD to print
```

```
   char displayStr[6] = {'\0'};  //String for LCD
   char displayStr1[6] = {'\0'}; //String for LCD

ADC_Start(); //Start ADC and start conversion
ADC_StartConvert();
       LCD_ClearDisplay();        //Clear Display
       LCD_Position(0,0);         //Write static text on Display
       LCD_PrintString("POT [mV] ="); //Value on pin with potenciometr in mV
       LCD_Position(1,0);
       LCD_PrintString("Motor[%] =");   ////Value on pin with motor in %

while(1) //Loop fo ever
{
   ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);//Read ADC count and convert to mV
   Count = ADC_GetResult16();
   mV = ADC_CountsTo_mVolts(Count);            //Convert count to mV
   Samples = Samples + Count; // Add the current ADC to the cumulated samples
   sampleCount++;

/* If 8 samples have been collected then average the samples and update the
display*/
          if(sampleCount == MAX_SAMPLE)
          {
              average = Samples >> 3;
              Samples = 0;
              sampleCount = 0;

              PWM_WriteCompare(average); //Change Compare value for PWM
              amount=(average)*(0.393);
           }
       sprintf(displayStr,"%4ld",mV); //transform number to text line
       sprintf(displayStr1,"%4ld",amount);     //transform number to text line

       LCD_Position(0,12);        //Print Values on display
       LCD_PrintString(displayStr);
       LCD_Position(1,12);
       LCD_PrintString(displayStr1);
    }
}
```

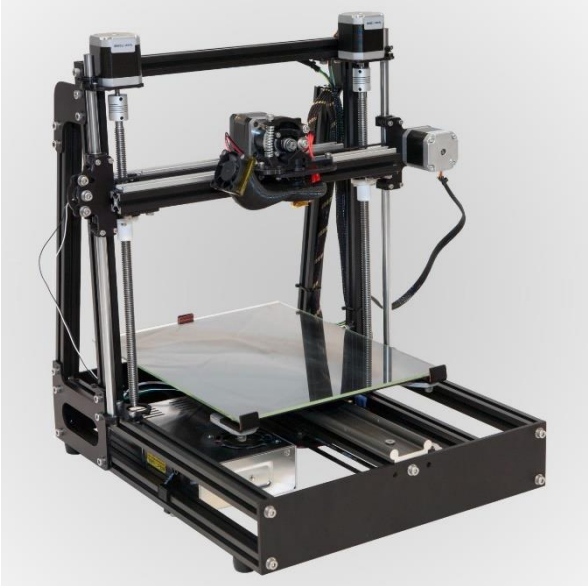
Při otáčení potenciometru se mění rychlost motoru, která je na displeji vypisována v procentech.

3. Mechatronický model – Experiment

Pro experimentální účely s PSoC byla zvolena dvourozměrná pojízdná plošina neboli rovinný manipulátor, na který se dá umístit jakékoli zařízení. Hlava 3D tiskárny, laser, gravírovací zařízení a podobně. V tomto experimentu byl zvolen ultrazvukový senzor, který odměřuje vzdálenost od podložky, čímž můžeme vytvořit 3D mapu pracovního povrchu neboli reliéf.

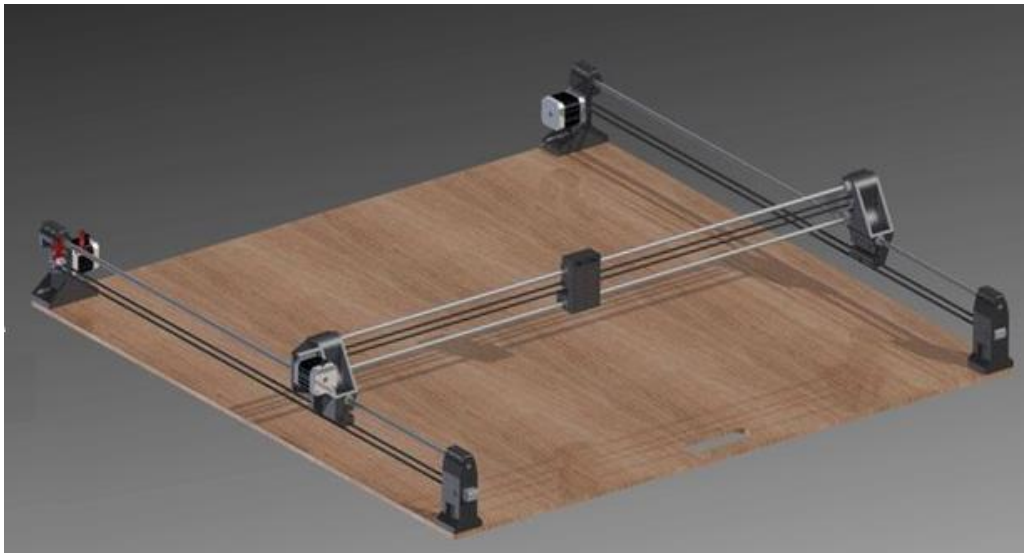
3.1. Popis konstrukce a fungování

Většina 3D tiskáren a malých CNC strojů na bázi Arduina je koncipována tak, že je základna, na které leží předmět, pohyblivá v jedné ose a hlava s nástrojem je připevněna na rám a pohybuje se v druhé ose (Obrázek 17). Toto řešení se stalo běžným hlavně díky své malé prostorové náročnosti a jednoduchosti. Má však svá úskalí. Například při řezání laserem se musí dbát na dobré ukotvení předmětu, a jemný chod motorů.



Obrázek 17 Ukázka možnosti pojezdu plošiny (http://josefprusa.cz/mendelmax2.jpg)

Pro tento experiment byla zvolena konstrukce odlišná. Platforma se skládá ze dvou na sebe kolmých pojezdů, které vykonávají pohyby v osách X a Y. Jeden z nich je pevně připevněn k podložce, zatímco druhý se pohybuje po vodicích tyčích prvního pojezdu (Obrázek 18). Předmět ležící na podložce tedy setrvává stále na stejném místě, a veškerý pohyb je realizován nástrojem.



Obrázek 18 Ukázka konstrukce

(https://thingiverse-production-new.s3.amazonaws.com/renders/cf/9b/2a/45/f5/6e72cd04bd092b68f6a270e53ae6c3c0_preview_featured.jpg)

V ose X jsou krokové motory dva, aby byl zajištěn rovnoměrný posuv, a nedocházelo ke vzpříčení kolmého pojezdu na vodicích tyčích. Samotná plošina je pak na pojezdu Y, kde pohyb obstarává již jen jeden krokový motor.

Celá tato sestava, její uspořádání a 3D modely, byla použita ze stránky [13]. Jedná se o volně šiřitelnou a modifikovatelnou platformu.

Pohyb je v jednotlivých osách realizován díky řemenům, které jsou poháněny krokovými motory. Pohyblivé základny se posouvají po svých vedeních v závislosti na otočení motoru. Pro fungování jsou zapotřebí krokové motory, řemenice, řemeny, vodící tyče a lineární ložiska pro jednotlivé pojezdy. Spojovací materiály jako šrouby, a kabeláž jsou samozřejmostí.

Díky 3D tisku se dala tato plošina z poměrně komplikovaných dílů vytvořit za krátkou dobu, navíc všechna uložení ložisek, děr pro šrouby a dalších konstrukčních prvků byla vcelku přesná, takže sestavit celý model nebylo složité.

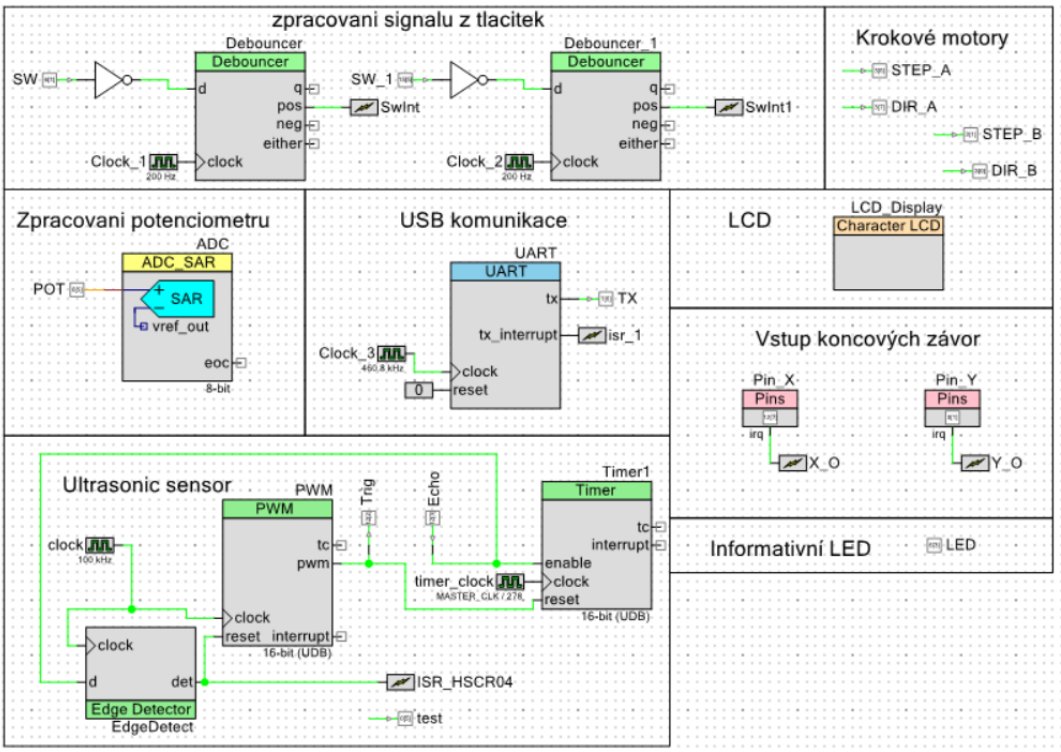
Pro řízení krokových motorů byly využity modulární drivery A4988, které jsou řízeny jen pomocí dvou vstupních signálů a velkou výhodou je snadné přepínání velikosti jednotlivých kroků.

Pro odměřování vzdálenosti byl použit pro mikrokontroléry standardní modul HC-SR04.

3.2. Ukázka blokových schémat a přiblížení struktury kódu pro PSoC

Významné prvky kódu jsou inspirovány video manuály od výrobce Cypress [8] a uživatelů zabývajícími se programováním této platformy na YouTube. Vzorem také byly příspěvky uživatelů na fóru na stránkách výrobce[9]. Odtud také pochází uspořádání blokových schémat a část kódu pro použití ultrazvukového senzoru s PSoC [11].

Pomocí blokových schémat byly vytvořeny základní ovládací prvky pro obsluhu plošiny (Obrázek 19).



Obrázek 19 Blokové schéma pro obsluhu plošiny

Po spuštění programu se obě osy přesunou do počátečního bodu, který je definován koncovými optickými spínači. V tomto bodě se souřadnice nastaví na hodnoty X=0 a Y=0. Když se proces dokončí, ukáže se na displeji počet vzorkovacích kroků, které můžeme měnit pomocí potenciometru. Počet vzorků v obou osách lze zvolit nezávisle na sobě. Výběr zvolené hodnoty se potvrdí integrovaným tlačítkem.

Po úspěšném nastavení těchto hodnot se odešle pomocí UART celkový počet bodů, který bude odměřen, aby Matlab věděl, kolik hodnot má přijmout. Po stisknutí tlačítka „START“, ke kterému vyzve nápis na displeji, se spustí čtecí algoritmus, který základu s ultrazvukovým senzorem přesouvá po trajektorii měření ve tvaru „hada“.

V každém měřeném bodě se plošina zastaví, proběhne proces měření vzdálenosti, a výsledná data se odešlou opět přes UART do počítače, kde je program v Matlabu zpracovává, a tvoří z nich 3D síť.

V průběhu celého procesu se na displeji vypisuje aktuální poloha plošiny, počet naměřených bodů, a poslední naměřená vzdálenost. Proces měření se dá kdykoliv zastavit tlačítkem „START“. Pokud se po v takto zastavené pozici stiskne druhé integrované tlačítko, měření se ukončí, na displeji se vypíše aktuálně naměřený počet bodů a plošina se vrátí do své výchozí polohy, tedy X,Y = 0. K tomu dojde i v případě, že měření proběhne až do konce.

3.3. Popis jednotlivých částí programu

Ke každé významné části budou uvedeny základní informace a vysvětlen princip fungování.

3.3.1. Ovládání krokových motorů

Díky modulárním driverům A4988 je ovládání velice snadné. Do modulu se posílají pouze informace o směru otáčení motoru (DIR) a požadavek na krok, tedy pulzní signál „HIGH LOW“ (STEP). Tato sekvence je zavedená jako funkce mimo části main a for, aby motory pracovaly plynule. Pro oba motory je stejná, změní se pouze název pinu, kam se signály odesílají. Kód vypadá následovně:

```
void step_A(bool dir, int16 steps)  //deklarace funkce
{
  DIR_A_Write(dir);                 //směr otáčení - DIR (HIGH / LOW)
  CyDelay(50);
  for(int i=0;i<steps;i++)          //počet kroků, o kolik se má motor otočit
  {
    STEP_A_Write(HIGH);
    CyDelayUs(800);                 //pulzy vysílané na pin STEP
    STEP_A_Write(LOW);
    CyDelayUs(800);
  }
}
```

Počet kroků pro tuto funkci, aby se motor posunul o správnou vzdálenost, je řešen tak, že se v každém bodě po měření přesune požadovaná pozice, a počet těchto kroků je prostý rozdíl požadované pozice a té současné. Směr otáčení závisí na znaménku rozdílů těchto hodnot. Celkový počet kroků, neboli absolutní pozice plošiny, je ukládán do proměnné amount, ze které se vypočítá současná pozice. Výsledný kód vypadá takto:

```
if((X-curX)>0)
{clockwise_A=true;}                 //rozhodne se o směru
else
{clockwise_A=false;}

step_A(clockwise_A,abs(X-curX));    // absolutní hodnota je počet kroků
amount_A=amount_A+(X-curX);         //celkový počet kroků
```

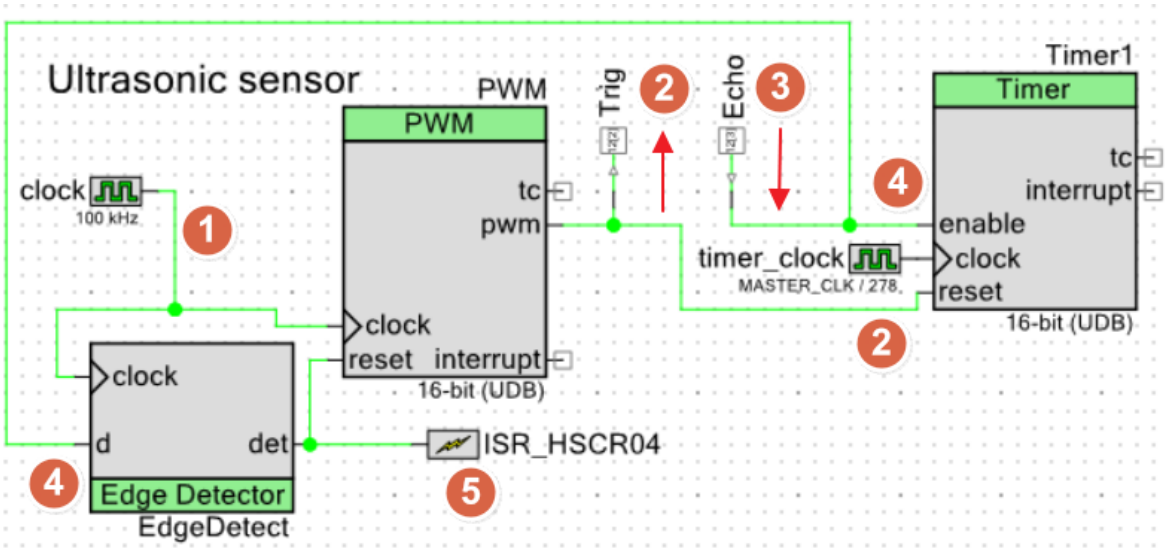
Blokové schéma není nějak zajímavé. Zapotřebí jsou jen 4 výstupní piny.

3.3.2. Odměřování vzdálenosti pomocí ultrazvuku

Obecně měření ultrazvukem funguje tak, že vysílač na krátký časový úsek vyšle ultrazvukový signál, a přijímačem se zjišťuje doba, za kterou se odrazil od předmětu - překážky. Z tohoto času se posléze určí, jak je daleko na základě znalosti šíření ultrazvukových vln vzduchem při daných podmínkách. Rychlost šíření závisí na teplotě vzduchu, ale pro tak malé vzdálenosti, které jsou měřeny v tomto experimentu, se mohou zanedbat.

PSoC musí obstarat obsluhu senzoru HC-SR04, což obnáší vyslat signál začátku měření, mezitím měřit čas a čekat na odpověď senzoru, že se ultrazvukový signál odrazil zpátky.

Všechny tyto kroky lze realizovat blokovým schématem, což je velice výhodné, protože celá funkce se realizuje na blocích UDB, takže bude odečet probíhat velice rychle a přesně, bez potřeby čekání na procesor. Uspořádání schémat bylo převzato z [11] (Obrázek 20).



Obrázek 20 Blokové schéma ultrazvukového senzoru

Celý „obvod“ funguje tak, že blok PWM generuje v pravidelných intervalech signály pro senzor, který v těchto intervalech vysílá ultrazvukové pulzy. Zároveň se tímto signálem resetuje časovač, který začne počítat čas znovu od nejvyšší hodnoty, tedy 50 ms. Když senzor příjme zvuk, vyšle signál, který přijde pinem Echo. Tím se uloží čas, který byl aktuálně v časovači, blok Edge Detector zaregistruje tento signál, a tím se aktivuje interrupt, díky čemuž se změní hodnota proměnné v kódu, a dojde ke čtení času.

Tento proces proběhne 80x, a výsledná hodnota je průměrem všem měřených hodnot. Kód je již velice jednoduchý:

```
CY_ISR(ISR_HSCR04) // Definice interruptu
{
  flag = 1;        //Změna hodnoty proměnné
}
CyDelayUs(900);
for(int i=0;i<80;i++) // počet opakování měření vzdálenosti
{
  if (flag == 1 )
  {
    flag = 0;
    count = TOPCOUNT - Timer1_ReadCounter(); //rozdíl časů
    if(i==0)
    {prumer = count;}
```

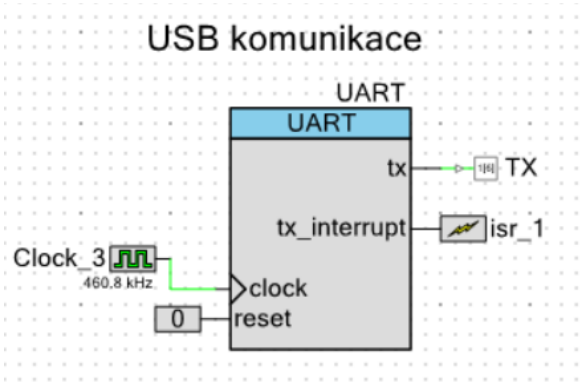
```
    if (count!=1)
    {
      prumer = (prumer + count)/2; //průměr hodnot měření
      CyDelay(1);
    }
  }
}
```

3.3.3. Komunikace pomocí UART

UART (universal asynchronous receiver/transmitter) je typ komunikace mezi zařízeními. Funguje tak, že každé ze zařízení má vyhrazeno 2 piny, a to vysílač a přijímač. (Transmitter – TX a receiver – RX). Piny jsou logicky připojeny tak, že vysílač vysílá na pin přijímače druhého zařízení a zase naopak. Celá „konverzace“ pak probíhá odesíláním bajtů (byte). Pokud se například odesílá text, každé písmeno reprezentuje jiný bajt, složený z osmi bitů, což není problém. Komplikace nastává při posílání čísel. Pomocí bitové hodnoty se dá odeslat jen číslo o velikosti 2^8, což je 256. Jelikož se přes UART odesílá hodnota aktuální pozice, která je vyšší 256, musí se hodnota rozdělit na „high“ a „low“ byte. Hodnoty se odešlou zvlášť a po přijetí se znovu spojí.

Například číslo 62 500 odpovídá bitové hodnotě 1111 0100 0010 0100. Ta se rozdělí na „high“ = 1111 0100 a „low“ = 0010 0100. Takto jsou odeslány v definovaném pořadí, a následně zase složeny.

Pro UART je v Creatoru stejnojmenný blok. Zapojení vypadá následovně (Obrázek 21):



Obrázek 21 Blok UART

Pro potřeby experimentu je postačující pouze vysílač, takže byl přijímač zcela zakázán.

Kód pro rozložení hodnot a odeslání:

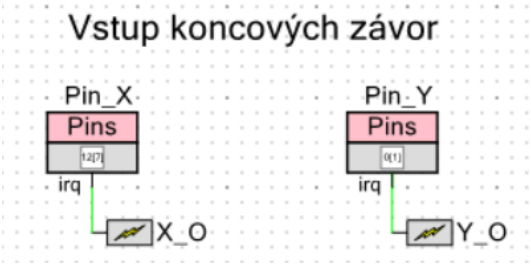
```
char Xlo;  char Xhi; //inicializace – číslo se odešle jako znak ASCII

Xlo = (X/stepX) & 0xFF;     // „ořezání“ hodnoty na low byte
Xhi = (X/stepX) >> 8;       // „ořezání“ hodnoty na high byte
UART_WriteTxData(Xlo);      //odeslání hodnoty LOW přes UART
CyDelay(20);
UART_WriteTxData(Xhi);      //odeslání hodnoty HIGH přes UART
CyDelay(20);
```

3.3.4. Koncové spínače

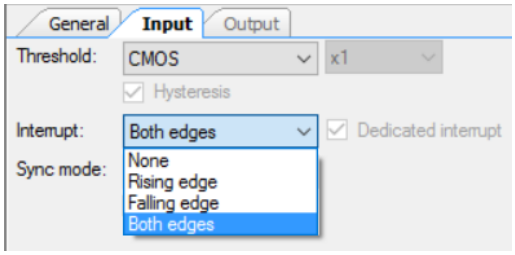
Nejsou funkčně moc zajímavé, ale jedná se o dobrou ukázku, že PSoC počítá s interrupty v širokém měřítku, proto se dá přepnout i vstupní pin na pin s interruptem. Na první pohled se zdá, že nedojde k žádné změně, ale v bloku vstupu můžeme definovat, kdy se má interrupt aktivovat, zda na skoku z „LOW“ na „HIGH“, opačně, nebo při obou variantách. To dává možnost například zapnout LED až při odlehčení tlačítka, takže při mačkání je možnost třeba zapnutí LED zrušit. Tento příklad je velice jednoduchý, ale využití může být důležité.

V případě tohoto experimentu dojde k aktivaci po obou skocích, protože se přepíná stav, kdy plošina je a kdy není v koncové pozici.

Blokové schéma (Obrázek 22) a nastavení interruptu (Obrázek 23):



Obrázek 22 Blokové schéma– koncový spínač



Obrázek 23 Nastavení interruptu

Zbytek je již podobný jako u ultrazvukového senzoru. Definují se funkce, které se vykonají po aktivaci interruptu. V tomto případě se mění hodnota proměnné podle toho, zda plošina je či není v počátku.

V kódu se dále vyskytuje ovládání potenciometru a výpis hodnot na displeji, což bylo ukázáno v kapitole 2.3.

3.3.5. Matlab a zpracování hodnot

Data, která se odesílají přes UART do počítače, se zpracovávají pomocí Matlabu. Jako první se spojí přijaté bity „low“ a „high“ zpátky na jedno číslo. Následně jsou tyto hodnoty uloženy do pole dat, které se vykreslí do grafu. Tento cyklus se opakuje stále dokola, dokud čítač nedosáhne přijaté hodnoty počtu měřených bodů a program se neukončí.

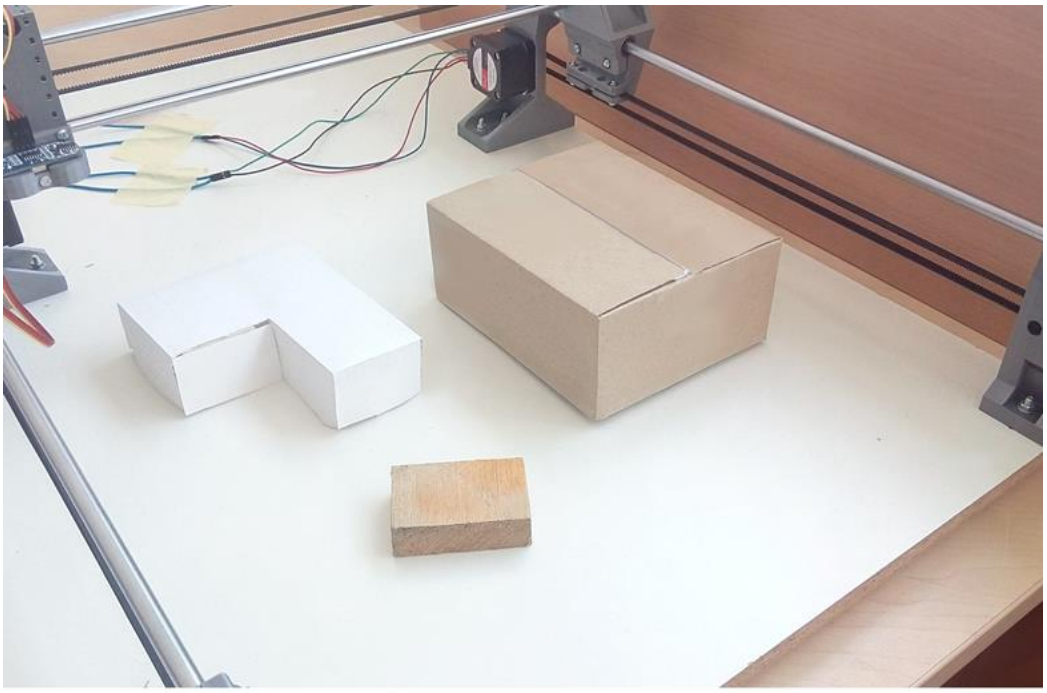
Veškeré přijaté informace se také zapisují do textového souboru, aby mohly být později využity.

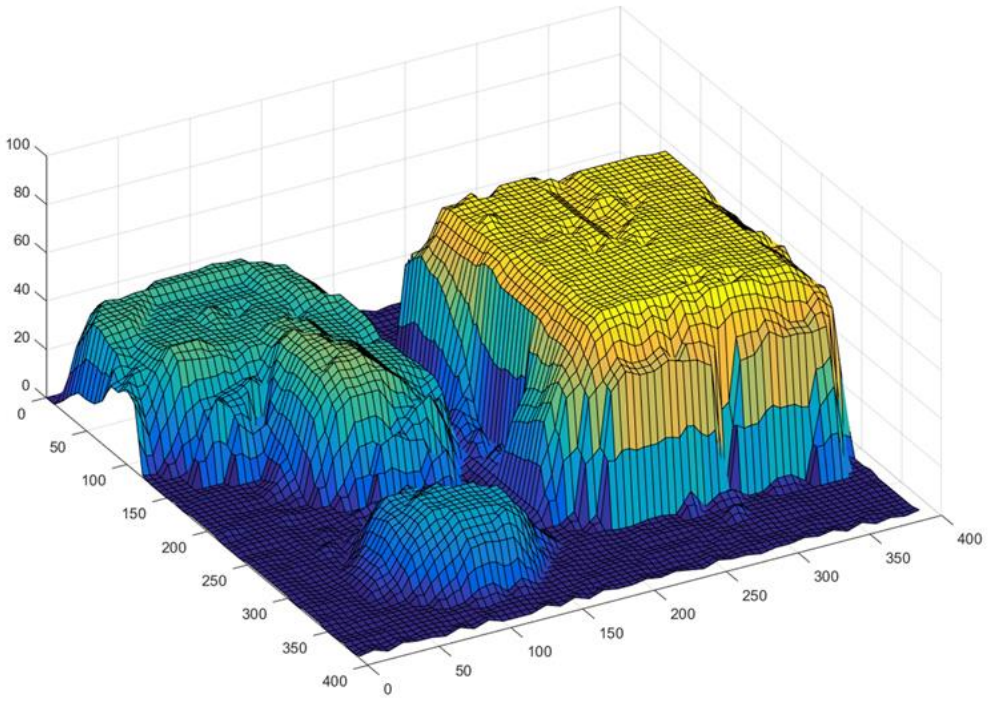
3.3.6. Celkový výstup

Po ukončení měření vznikne 3D síť naměřených bodů, která se může porovnat s originálním uspořádáním měřené plochy (Obrázek 24)Samotné měření proběhlo při rozlišení 50x50 vzorků, tedy sít vznikla z 2 500 bodů.

Na první pohled je zřejmé, že je ze získaných dat těžké určit hrany měřených předmětů. Důvodem bude nejspíš odražení zvukových vln od bočních ploch při měřeních prováděných v blízkosti těchto hran. Vznikne zde jakýsi plynulý náběh, který se nedal odstranit ani při pokusech s vyšším rozlišením. Dalším důvodem je interpolace, tedy nepřesnost při prokládání křivek jednotlivými body. Jde také vidět, že se při měření

ovlivňují předměty samotné, proto vznikl výrazný výstupek mezi malým a velkým kvádrem. Pro přesnější měření by byla vhodnější jiná konstrukce senzoru nebo zcela jiný způsob měření. Nynější rozměry senzoru nedovolují o moc zjemnit měření, protože rušivé odrazy zvukového signálu zůstávají stále stejné. Pro experimentální účely s PSoC je ovšem výsledek uspokojivý.





Obrázek 24 Porovnání naměřených dat s realitou

4. Závěr

V úvodní části jsem se seznámil s vývojovým kitem CY8CKIT-050 PSoC, zjistil, jaké má výhody oproti ostatním platformám a prozkoumal strukturu system on chip, která je tvořena bloky UDB, což přináší řadu výhod. Na stránkách Cypress jsem prostudoval fóra pro uživatele této platformy, kde sdílí vytvořené projekty, což značně usnadnilo práci s vývojem kódů.

Ve druhé části jsem provedl jednoduché experimenty, které jsem posléze slovně i graficky popsal a vysvětlil jejich fungování. Konkrétně dva způsoby blikání s LED, několik využití integrovaného potenciometru, který je použit k ovládání pohybu servomotoru a DC motoru. V této kapitole jsou vysvětlivky v C kódu uvedeny anglicky, aby mohly být zpětně uloženy na fóru výrobce, odkud pochází většina nápadů na řešení dané problematiky.

Ve třetí části jsem sestavil jednoduchý experiment využívající vývojový kit CY8CKIT-050 PSoC. Rovinný manipulátor, na kterém je umístěn ultrazvukový senzor, díky kterému se odměřuje vzdálenost k podložce. Většina částí byla vytištěna pomocí 3D tiskárny. Při stavbě manipulátoru byla předlohou volně šiřitelná a modifikovatelná platforma [13]. Je popsána konstrukce pojezdů jednotlivých os a práce ultrazvukového senzoru, který v měřeném bodě odesílá informace ke zpracování přes rozhraní UART do počítače. Tímto způsobem byl vytvořen 3D reliéf podložky v programu Matlab, a následně porovnán s reálným uspořádáním. Sestavením rovinného manipulátoru a zpracováním základní dokumentace pro práci s PSoC byly naplněny cíle práce definované v zadání.

U ultrazvukového senzoru se nedá přesně určit oblast měření, protože je senzor veliký a signál se ze senzoru šíří všemi směry, přičemž se může odrazit od nechtěné plochy dříve než od měřeného předmětu. Vhodnějším způsobem měření by byl laser či jiná metoda, u které lze jednoznačně určit, jaká oblast je právě měřena. Pokud by se konstrukce manipulátoru měla využít pro kontaktní účely, jako je gravírování, zaměřil bych se zkoumání rozdílné tuhosti a síly v osách, zapříčiněnou rozdílným počtem motorů v každé z nich.

Současná přesnost pohybu manipulátoru je s ohledem na kvality senzoru na základní úrovni, ale pro potřeby přesnějších aplikací se dá o hodně zjemnit krok motorů, čímž by se dalo dosáhnout poměrně vysoké přesnosti polohování.

Možná bych zvolil lehce odlišnou konstrukci, kdy by obě osy byly pevně připevněny k podložce [14].

Všechny potřebné kódy pro PSoC a Matlab jsou na přiloženém CD i s elektronickou verzí této práce.

Použitá literatura

[1] About Cypress [online]. [cit. 2017-5-15]. Dostupné z: http://www.cypress.com/about-cypress

[2] Microcontroller (MCU) and Programmable System-on-Chip (PSoC®) Families [online]. [cit. 2017-5-15]. Dostupné z: http://www.cypress.com/products/microcontroller-mcu-and-programmable-system-chip-psoc-families

[3] 32-bit ARM® Cortex®-M3 PSoC® 5LP [online]. [cit. 2017-5-16]. Dostupné z: http://www.cypress.com/products/32-bit-arm-cortex-m3-psoc-5lp

[4] PSoC® Creator™ Integrated Design Environment (IDE) [online]. [cit. 2017-5-15]. Dostupné z: http://www.cypress.com/products/psoc-creator-integrated-design-environment-ide

[5] Getting Started with PSoC®5LP [online]. [cit. 2017-5-15]. Dostupné z: http://www.cypress.com/file/41436/download

[6] PWM [online]. [cit. 2017-5-18]. Dostupné z: https://www.arduino.cc/en/Tutorial/PWM

[7] PSoC® 5LP DEVELOPMENT KIT QUICK START GUIDE [online]. [cit. 2017-5-15]. Dostupné z: http://www.cypress.com/file/45271/download

[8] PSoC® 101 Video Tutorial Series: How To Use the ARM® Cortex®-M0 Based PSoC 4 [online]. [cit. 2017-5-15]. Dostupné z: http://www.cypress.com/training/psoc-101-video-tutorial-series-how-use-arm-cortex-m0-based-psoc-4

[9] Cypress Developer CommunityTM [online]. [cit. 2017-6-10]. Dostupné z: http://www.cypress.com/forum

[10] Uživatel szatocs (16 Oct 2013 04:18 PM PDT) , ..Attachments: Download pwm.zip.. [online]. [cit. 2017-5-18]. Dostupné z: http://www.cypress.com/forum/psoc-5-device-programming/vary-led-brightness-potentiometer

[11] Пример подключения ультразвукового дальномера HCSR-04 к PSoC (PSoC5) [online]. [cit. 2017-6-10]. Dostupné z: http://mylab.wmsite.ru/moi-razrab/cypress-psoc/dalnomer-hcsr-04/

[12] PSOC 101 - WHAT IS PSOC? [online]. [cit. 2017-5-15]. Dostupné z: http://www.cypress.com/blog/psoc-creator-news-and-information/psoc-101-what-psoc

[13] XY 800 laser engraver table [online]. [cit. 2017-6-10]. Dostupné z: https://www.thingiverse.com/thing:1258128

[14] A simple to build 3D printer with a combined X/Y stage [online]. [cit. 2017-5-15]. Dostupné z: https://github.com/dherrendoerfer/The-Robin-3D-Printer-Project