TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

ČÍSLICOVÝ REGULÁTOR PRŮTOKU NA DESCE SILABS

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **David Taller**
Vedoucí práce: Ing. Petr Školník, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

FLOW CONTROLLER REALIZED VIA SILABS DEVELOPMENT KIT

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **David Taller**
Supervisor: Ing. Petr Školník, Ph.D.

Liberec 2015

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce panu Ing. Petru Školníkovi, Ph.D. a konzultantovi panu Ing. Tomáši Náhlovskému, Ph.D. za pomoc a rady při zhotovení této bakalářské práce.

Abstrakt

Práce se v teoretické části zabývá pochopením problematiky automatického řízení konkrétně regulace. Následný rozbor spolu s popisem všech složek PID regulátoru, ošetření wind-up efektu a některých možných postupů pro volbu ideální velikosti jednotlivých složek regulátoru. Druhá polovina teoretické části se zaměřuje na popis a použití komponent v praktické části bakalářské práce a popsání způsobu komunikace se zvoleným vývojovým kitem. V praktické části jsou poté rozepsány postupy při realizaci regulátoru na reálném procesu spolu s popisem logiky programu a v neposlední řadě shrnutí naměřených hodnoty a zhodnocení reakce a funkčnost navrženého regulátoru.

Klíčová slova:

číslicový regulátor, vývojový kit BIG8051, řízení průtoku, PID regulátor

Abstract

Working with the theoretical part deals with an understanding of the problems of automatic control of specific regulation and subsequent analysis together with a description of all components of the PID controller, treatment of wind-up effects and some possible methods for selecting the ideal size of the individual constants of the controller. The second half of the theoretical part focuses on the description and use of components in practical part of the thesis, and describe how to communicate with the selected development kit. In the practical part they are then broken down procedures for the implementation of the regulator on the real process, along with a description of the program logic, and finally to the summary measured values and evaluation of response and functionality of the proposed controller.

Key words:

discrete regulator, development kit BIG8051, flow control, PID regulator

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

IDE	Integrated Development Environment, vývojové prostředí
USB	Universal Serial Bus, univerzální sériová sběrnice
LCD	Liquid Crystal Display, displej z tekutých krystalů
CAN	Controller Area Network, sběrnice využívaná pro komunikaci
A/D	Analogově digitální převodník
D/A	Digitálně analogový převodník
USART	Universal Synchronous / Asynchronous Receiver and Transmitter, zařízení pro sériovou komunikaci.
RAM	Random-Access Memory, paměť s libovolným výběrem
SFR	Special Function Register, speciální funkční registry
MIPS	Million Instruction Per Second, jednotka výkonnosti procesoru
EMIF	External Memory Interface, rozhraní pro externí paměť
SPI	Serial Peripheral Interface, sériové periferní rozhraní
SMBus	System Management Bus, dvouvodičová komunikační sběrnice
PWM	Pulse Width Modulation, pulzně šířková modulace je způsob přenosu dat
PCA	Programmable Counter Array, pole programovatelných čítačů
PID	Proporcionálně integračně derivační regulátor
PSD	Proporcionálně sumační derivační regulátor

1 Úvodní část

Cílem této bakalářské práce je vytvoření diskrétního regulátoru na vývojové platformě BIG8051. S tím související seznámení s touto platformou, potřebnými prostředky potřebné pro zhotovení regulátoru a prostudování všech dostupných možností využití platformy jako regulátoru průtoku vody. Dalším cílem práce je seznámení s problematikou regulace průtoku řízené mikroprocesorem řady ´51, popsat a aplikovat zvolenou metodu regulace spolu s uživatelským rozhraním pro nastavení a obsluhu regulátoru. V neposlední řadě zhotovit praktickou ukázku zvolené implementace regulátoru na vývojovém kitu spolu se všemi potřebnými komponenty a neměření chování regulátoru při jeho různé konfiguraci. Posledním cílem této práce je pokusit se zrealizovat vzdálenou komunikaci s vývojovým kitem pomocí ethernetového rozhraní, který umožní alternativní možnost ovládání regulátoru.

Jako regulační zařízení jsem zvolil nízkonapěťové ponorné čerpadlo. Zpětnou vazbu ze soustavy s informací o aktuálním průtoku mi poskytne průtokoměr s dostatečným rozsahem snímání průtoku pro účely této práce. Tyto dvě komponenty byly zvoleny z důvodu dostupnosti v laboratoři a dostačujícími vlastnostmi pro demonstraci využití vývojového kitu jako regulátoru.

Téma regulace a regulační systémy nás obklopuje čím dál více s rostoucím počtem integrování řídicích systémů, výpočetní techniky a automatizace do všech možných odvětví běžného života. Regulační systémy ulehčují a automatizují výrobu na výrobních linkách, umožňují nám používat a řídit dopravní prostředky jako jsou automobily, lodě, nebo dopravní letadla a jejich využití se najde například i ve zdravotnictví u přístrojů, které zachraňují životy. Regulace a regulační soustavy najdou své využití všude tam, kde je zapotřebí udržovat nějaké zařízení na žádané hodnotě. Například udržování motoru na žádaných otáčkách, tavící pec na požadované teplotě, nebo v mém případě udržet požadovaný průtok ponorného čerpadla.

2 Vývojový kit BIG8051

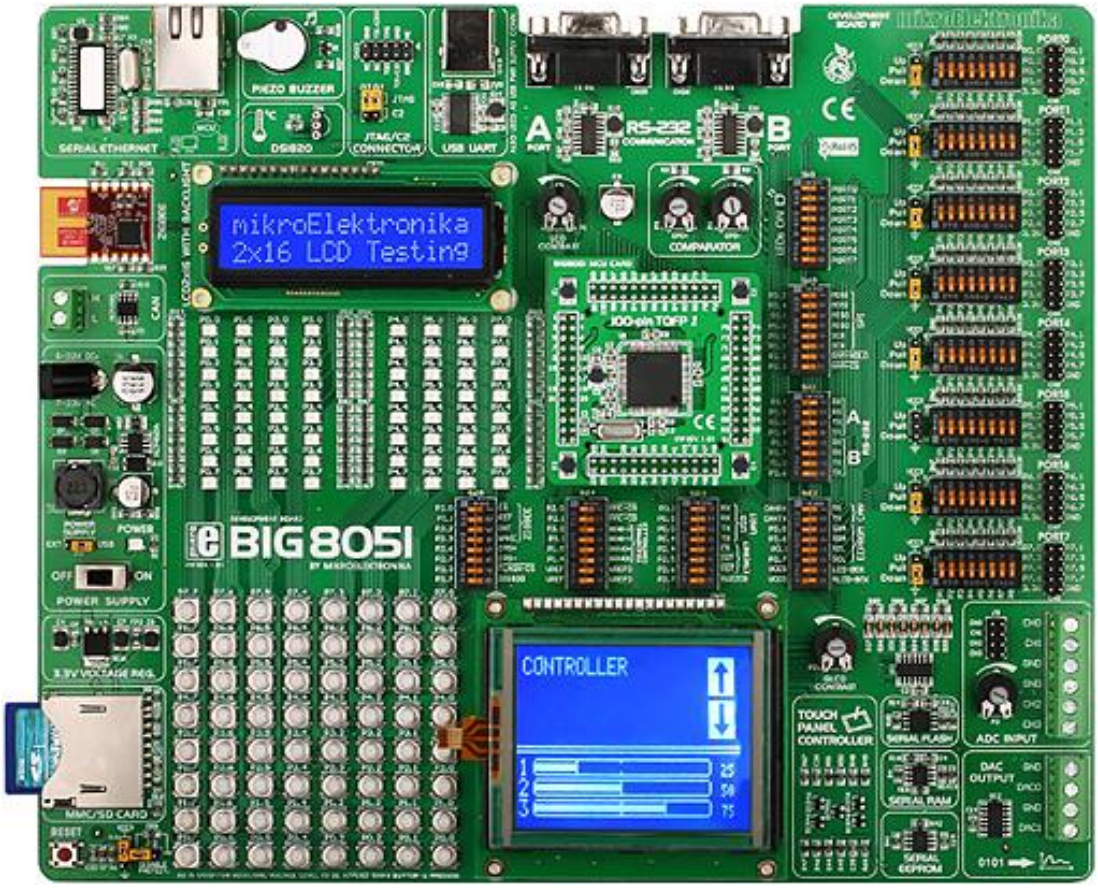
2.1 Popis desky

Vývojový kit BIG8051 od firmy MikroElektronika byl zvolen z důvodu dostupnosti zapůjčení tohoto kitu ve škole a zároveň s dostačující výpočetní výkonností procesoru a podpůrných prostředků pro realizaci řízení regulace v reálném čase společně s výhodou integrovaných displejů a dotykové vrstvy přímo na desce, které efektivně poslouží k interakci s uživatelem při nastavování konstant regulátoru spolu s žádaným průtokem v regulační soustavě. Jeho další výhoda je umístění integrovaného komunikačního síťového rozhraní pomocí konektoru RJ-42, přes které zrealizuji vzdálenou komunikaci s procesorem pomocí webového uživatelského rozhraní. Vývojový kit lze naprogramovat pomocí jazyka C a kompilátoru MikroC od výrobce desky, který usnadňuje programování platformy díky jeho předpřipraveným knihovním funkcím, které ulehčují práci s periferiemi a dalšími prostředky procesoru. Výrobce vývojového kitu také nabízí přehledné IDE vytvořené přímo pro knihovnu MikroC, pomocí kterého generuji zkompilovaný program do souboru typu Intel HEX, který následně nahrávám do mikroprocesoru pomocí USB adaptéru, který soubor nahraje skrze rozhraní JTAG. Desku lze napájet buďto střídavým napětím 7V až 23V nebo stejnosměrným 9V až 32V. Další možností napájení desky je pomocí rozhraní USB. V mé práci nebylo zapotřebí využívat větší napájecí napětí, nežli je 5V, které poskytuje napájení pomocí USB portu. Proto jsem zvolil tuto možnost jako zdroj napájení. [1]

Vývojový kit BIG8051 nabízí velké množství již integrovaných obvodů, periferií a komunikačních rozhraní, jejichž hardwarové aktivování nebo naopak zakázání, je možné provádět pomocí 15 přepínačů SW1 až SW15 rozdělených po osmi přepínačích, které propojují piny procesoru s ostatními komponenty desky. Deska nabízí využití LCD displeje o možnosti zobrazení dvou řádků po 16 znacích každý znak o velikost 7x5 bodů. Jako druhou zobrazovací periferii nabízí deska grafický LCD displej o velikosti 128x64 bodů. Mezi hlavní výhody tohoto displeje patří možnost vykreslování grafů, obrazců, nebo uživatelské rozhraní pomocí bitmapy. Pro manuální interakci s procesorem je na desce umístěno 64 tlačítek, rozmístěných do čtverce o velikosti 8x8 tlačítek. Každý řádek tlačítek představuje jeden port procesoru a každé tlačítko v řádku jeden pin procesoru. Každé z tlačítek může být připojeno buďto k napětí 3,3V, nebo k zemi. Tohoto se docílí správném

nastavení polohy jumperu J10. Ke každému pinu procesoru jsou spolu s tlačítky také připojeny led diody. Tyto led diody lze zapínat, nebo vypínat pomocí přepínače SW9.

Mezi dostupné integrované komunikační prostředky s externími zařízeními lze zahrnout sběrnici CAN, dvě sériové rozhraní RS-232, ethernetový modul, dva D/A převodníky, A/D převodník se čtyřmi kanály a přídavný ZigBee modul pro bezdrátovou komunikaci s procesorem. Jako poslední možností komunikace s okolím lze využít vyvedení každého pinu procesoru na pravé straně desky, který lze nakonfigurovat buďto jako logický vstup, nebo výstup. Každý pin procesoru nastavený jako vstup je tolerantní přivedením maximálního napětí 5V. Pro každý tento vstupně/výstupní pin je určen jeden přepínač, kterým lze nastavit manuálně dva typy chování pinu. První typ je pull-up, který bude na pinu držet napětí 3,3V. Druhým typem je opak a to typ pull-down, který bude naopak na pinu držet napětí 0V. [1]



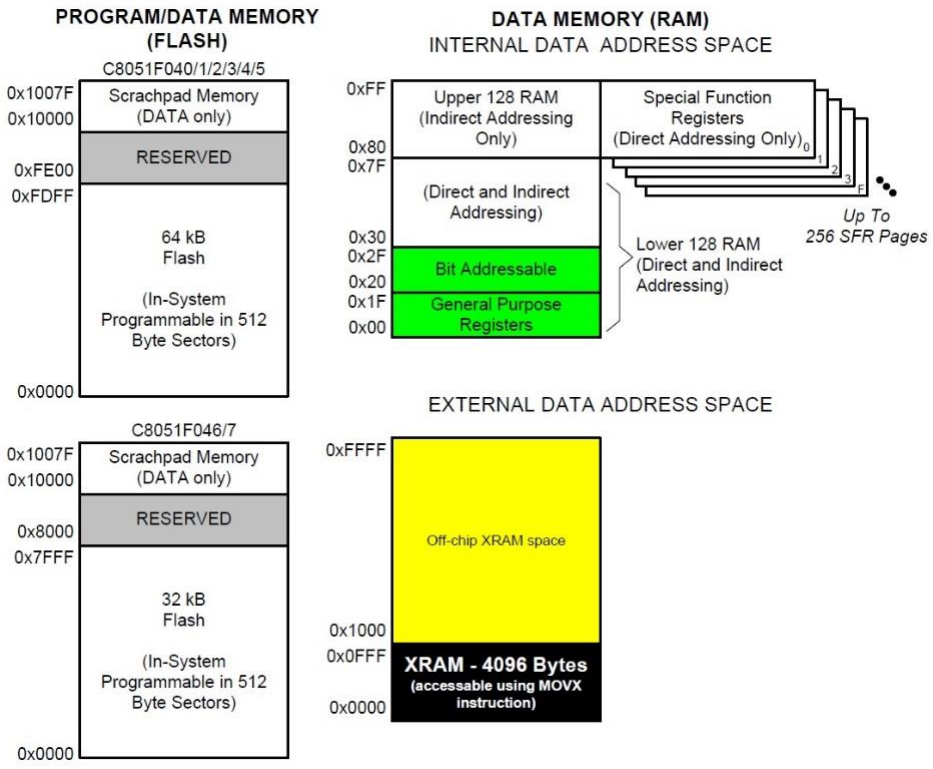
Obrázek 1 - Vývojový kit BIG8051. [1]

2.2 Procesor

Platforma BIG8051 je osazena 8 bitovým mikroprocesorem řady ´51 od firmy Silicon Laboratories, který je taktován krystalem o frekvenci 24, 5 MHz a nese označení C8051F040. Procesor je založen na jádře CIP-51 a je plně kompatibilní s instrukční sadou MCS-51 standardu 803x/805x. Procesor zahrnuje využití periferií obvyklých pro řadu procesorů ´51, jako je pět 16 bitových čítačů/časovačů, dvě plně duplexní USART sběrnice, A/D a D/A převodníky apod. Procesor nabízí paměťový prostor o velikosti 256 bytů interní RAM paměti spolu s 128 bitů adresního prostoru pro SFR registry. Jádro CIP-51 obsahuje celkem 109 instrukcí. Při maximální frekvenci 24,5 MHz dosahuje procesor podle dokumentace nejvyšší propustnost 25MIPS. [1]

2.3 Paměťový prostor

Jádro CIP-51 obsahuje dva adresní prostory. Jeden adresní prostor je určený pro program a druhý pro data. Tyto adresní prostory zabírají celkem 256 bytů interní RAM paměti, z kterých vyšších 128 bytů RAM paměti je duálně mapováno. To znamená, že k horním 128 bytům paměti se přistupuje nepřímým adresováním a ke spodním 128 bytům se přistupuje přímo v adresním prostoru SFR.

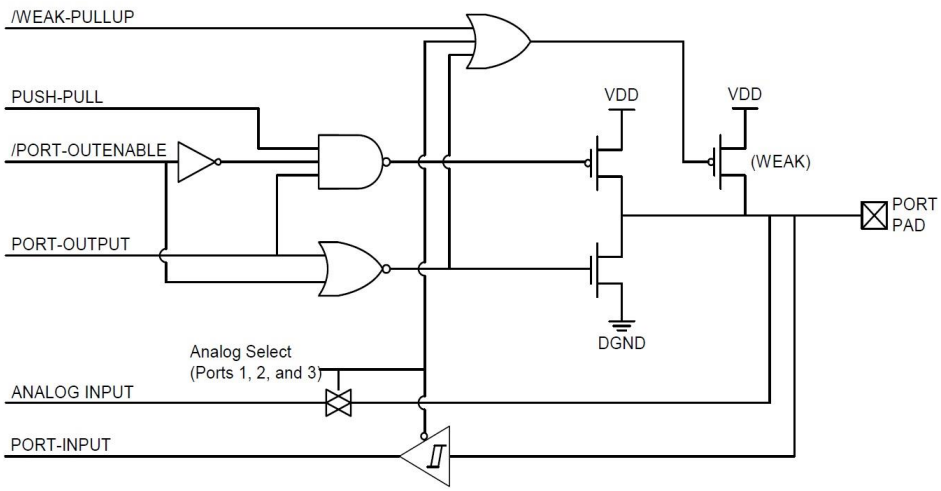


Obrázek 2 - Schéma paměťového prostoru procesoru. [1]

V jádře CIP-51 obsahuje adresní prostor pro SFR registry až 256 SFR stránek dostupné přímým adresováním, které umožňuje umístění dostatečného množství registrů pro nastavení a ovládání všech periferií. Jádro má implementováno v operační paměti také doplňujících blok o velikosti 4KB a rozhraní pro práci s externí pamětí EMIF. Toto rozhraní umožňuje přístup k datům mimo čip, nebo k pamětí mapovaným periferiím. Čip poté může pomocí 4 bytů adresovat skrze celou 64KB externí pamětí. Mapování externí paměti lze mapovat buďto na paměť na čipu, paměť mimo čip, nebo v kombinaci s obojím například adresy do 4KB budou data adresovány na čip a adresy přesahující 4KB budou adresovány do EMIF paměti. [1]

2.4 Vstupně/výstupní rozhraní procesoru

Procesor je osazen osmi porty po osmi digitálně vstupně/výstupních pinech. Každý pin lze nastavit jako open-drain, push-pull, nebo weak pullup. Ke každému portu existuje příslušný SFR registr, který definuje výše zvolený typ chování pinu.

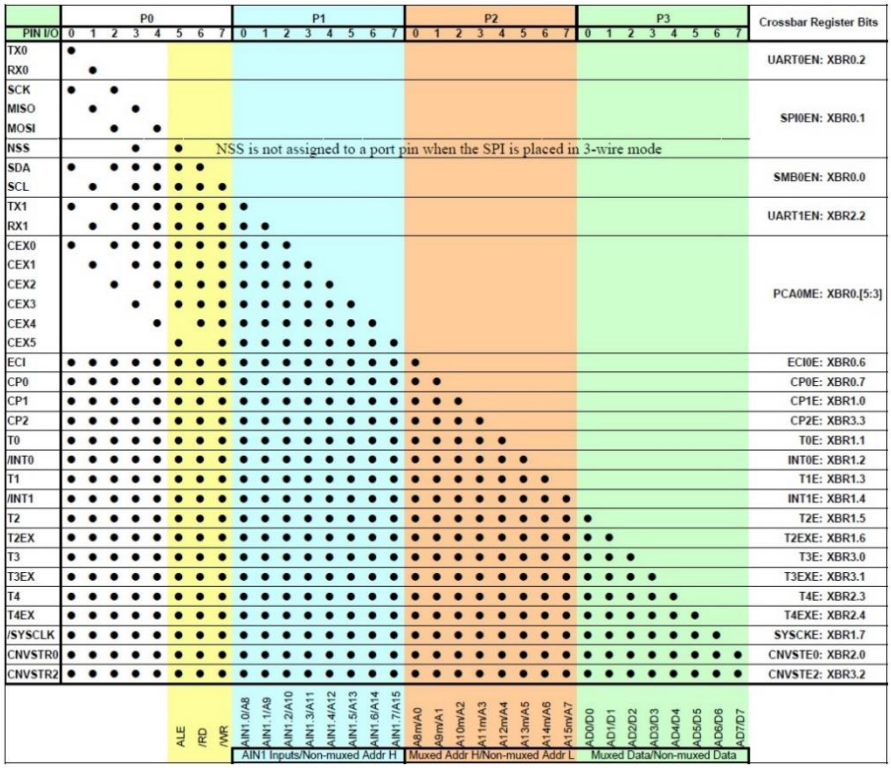


Obrázek 3 - Blokové schéma zapojení vstupně/výstupního pinu. [1]

Procesory od firmy Silabs řady C8051F04x mají široké možnosti pro komunikaci s periferiemi, pro kterou jsou určeny první čtyři porty procesoru P0 až P3. Každý pin těchto portů může být definován jako univerzální vstupně/výstupní pin, nebo jako vstup do procesoru obsluhující například rozhraní UART. Další možností je vyvolání na pinu procesoru žádost o přerušení a následnou změnu kontextu. Abychom věděli, jaká služba bude přiřazena na

určitý pin z portů P0 až P4, je zde umístěn prioritní dekodér, který se stará o priority jednotlivých služeb při přiřazování k vstupně/výstupním pinům.

Nastavení prioritního dekodéru, určujícího přiřazení služby na pin procesoru se definuje registry XBR0 až XBR3 popsané v obrázku výše. Například pokud nastavíme bit XBR0.2 do logické 1, bude signál TX0 a RX0 nastaven na pin procesoru P0.0 a P0.1 vždy, protože UART má nejvyšší prioritu. Ještě je potřeba uvést, že pokud by byla služba UART zakázána a my bychom přiřadily bit XBR0.2 do logické 1, tak pokud by žádala služba s nižší prioritou o piny P0.0 a P0.1 budou službě přiřazeny, protože služba UART není aktivní. Na obrázku níže můžeme vidět rozložení pinů a služeb od nejvyšší priority (UART0EN), až po nejnižší prioritu (CNVSTR2).



Obrázek 4 - Prioritní dekodér. [1]

Jako vstupní rozhraní pro komunikaci s procesorem se kromě využití externího přerušení a komunikačních sběrnic UART, CAN, SMBus, nebo SPI, nabízí možnost použití třech kanálů 12, nebo 8 bitového A/D převodníku. Jako zdroj signálu pro převodník se zde volí nastavením multiplexoru, umístěného před převodníkem, mezi dvěma typy zdrojů. První je pomocí čtyř vstupních kanálů vyvedených na desce pro možnost připojení čtyř dvojic vstupních vodičů (uzemnění a signál), každý určený pro jeden kanál A/D převodníku.

Další varianta přepnutí multiplexoru je na zdroj signálu pro A/D převodník z portu P3. Tuto možnost využívám při získání souřadnic dotyku na dotykové vrstvě, která je připojena právě na port P3.

Mezi výstupní rozhraní můžeme zahrnout dva integrované 12 bitové D/A převodníky (DAC1 a DAC2). Každý z nich umožňuje rozsah napětí 0V až 2,47V a nabízí možnost pracovat v rozlišení 8, nebo 12 bitů. Při použití 12 bitového rozlišení se u každého modulu nastavuje výstupní napětí pomocí dvou 8 bitových registrů v rozsahu 0 až 4096. První z registrů (DACxL) reprezentuje spodních nejméně významných osm bitů (LSB) a druhý registr (DACxH) zbylé nejvíce významně čtyři bity (MSB) výstupní hodnoty. Pokud zvolíme 8 bitový režim tak výstupní hodnota se využívá pouze z registru DACxL v rozsahu 0 až 256. Zvolení impulzu pro začátek interpretování hodnoty z registrů DACxL, nebo DACxH na výstup napětí D/A převodníku jsou k volbě čtyři možnosti. První tři možnosti obnovy výstupního napětí na převodníku je jeden z trojice časovačů (Timer 2 až Timer 4) po jehož přetečení se přečte 8, nebo 12 bitová hodnota z registrů DACxL a DACxH a převede se na výstupní napětí převodníku. Poslední možnost zdroje impulzu pro obnovu výstupního napětí na výstupu převodníku je po zapsání hodnoty do registru DACxH.

Další výstupní rozhraní, které vývojový kit nabízí je využití generátoru PWM signálu na výstupním pinu procesoru v rozlišení 8, nebo 16 bitů. Na platformě je k dispozici možnost využití modulu procesoru PCA (Programmable Counter Array) s možností volby až šesti zvolených výstupních pinů generující signál PWM. PWM výstup se generuje principem porovnání hodnoty v příslušném čítači, určeného pro tyto výstupní PCA moduly, s hodnotou v registru náležejícímu k příslušnému modulu viz kapitola PWM ve spojení s BIG8051. Velikostí periody generovaného PWM signálu lze manipulovat pomocí zdrojů čítače ovlivňující jeho rychlost inkrementace. Čím bude větší frekvence čítání, tím bude perioda signálu kratší a naopak.[1]

2.5 Test výpočetního výkonu procesoru

Pro realizaci regulátoru je pro mne důležité jakou rychlostí je schopen procesor vypočítat algoritmus PID regulátoru. Bylo tedy zapotřebí zhotovit následující test. Bohužel knihovna MikroC nenabízí funkce, které by vraceli absolutní čas od spuštění procesoru například v milisekundách, jako jsme zvyklí v jazyce C pomocí funkce `clock()`. Touto funkcí bych mohl jednoduše zjistit uplynulý čas po a před výpočtem PID algoritmu a jejich rozdílem bych vypočítal jakou rychlostí je schopen procesor PID algoritmus vypočítat. Tento nedostatek jsem vyřešil využitím systémového časovače. Výpočet PID algoritmu jsem umístil do nekonečného cyklu v hlavním vlákně programu a po každém průchodu algoritmem PID jsem inkrementoval celočíselnou pomocnou proměnnou. Tuto hodnotu poté vyhodnocuji každou sekundu pomocí systémového časovače a vypisuji ji na displej. Test jsem poprvé prováděl na použití pouze samotného PID algoritmu bez zatížení procesoru generováním PWM signálu a vyhodnocováním externího přerušení, které poté využívám v řízení regulačního soustavy. Výsledek byl mezi 186 až 187 výpočty PID algoritmu za sekundu. Poté jsem test prováděl i se zatížením procesoru generováním PWM signálu a obsluhou externího přerušení. Výsledek byl skoro ekvivalentní bez zatížení procesoru generováním PWM a obsluhy přerušení. Výsledek byl pouze s rozdílem jednoho nebo dvou výpočtů PID algoritmu, tedy 184 až 185 výpočty za sekundu.

Z výsledku usuzuji, že mi procesor může zaručit zhruba 184 výpočtů PID algoritmu za sekundu, spolu s vyhodnocováním informace ze senzoru o aktuálním průtoku v soustavě pomocí externího přerušení a generování PWM signálu pro řízení soustavy. Minimální perioda, s kterou mohu vyhodnocovat PID algoritmus je tedy zhruba 6 milisekund, což je pro mé účely více než dostačující. Ve svém programu poté volím periodu výpočtu PID algoritmu po jedné sekundě. Důvodem je vyhodnocování aktuálního průtoku z průtokoměru po jedné sekundě, kvůli jeho nízké frekvenci generovaného výstupního signálu.

Tabulka obsahuje níže přehled doporučené periody výpočtů PID algoritmu pro rozdílné nasazení regulátoru.

Tabulka 1 - Doporučené vzorkovací periody PID regulátoru. [11]

Vzorkovací perioda T	Proces
0,5 – 20 µs	Stabilizace výkonových systémů, letové simulátory, trenažéry
10 – 500 µs	Přesné řízení a modelování, elektrické systémy, energetické systémy, přesné řídící roboty
10 – 100 ms	Zpracování obrazů, virtuální realita, umělé vidění
0,5 – 1 s	Monitorování a řízení objektů, chemické procesy, elektrárny
1 – 3 s	Regulace průtoků
1 – 5 s	Regulace tlaku
5 – 10 s	Regulace hladiny
10 – 20 s	Regulace teploty

3 Využité prostředky pro komunikaci s komponenty soustavy

3.1 Přerušení procesoru

Přerušení procesoru je významné hlavně při obsluze asynchronních událostí. Asynchronní událostí můžeme označit událost, jejíž vznik nemůžeme přesně stanovit, ale potřebujeme na ni zareagovat a zpracovat ji. Princip přerušení bych popsal na příkladu. Mějme procesor, který mimo jiné operace jako mohou být například časově kritické sekce řízení regulační soustavy, obsluhuje také neplánované akce od uživatele. Neplánovanou akcí od uživatele může být například vstup z klávesnice. Bylo by nepřínosné umisťovat logiku obsluhy akce od uživatele do hlavního vlákna a tázat se například v nekonečném cyklu pokaždé, jestli uživatel nestiskl nějakou klávesu. Proto elegantnějším řešením je, pokud by stisk klávesy z klávesnice vyvolal přerušení na procesoru. Procesor by přerušil aktuální výpočet, odskočilo by se z adresy, na které se procesor právě nachází na adresu, která ukazuje na obslužnou rutinu externího přerušení. Tato rutina by pak dále obsloužila již konkrétní stisknutou klávesu a provedla příslušnou akci.

Přerušení můžeme rozdělit na tři způsoby žádání zařízení o přerušení procesoru. První způsob se nazývá dotazovací (pooling). Všechny zařízení, které vyžadují přerušení, generují stejný požadavek o přerušení a je tedy obslouženo i stejnou obslužní rutinou. Obslužná rutina nejprve zjistí, z které zařízení žádá o přerušení a poté se provede příslušná akce. Tato metoda je neefektivní jak z pohledu přehlednosti kódu, tak i v neefektivním opakovaném vyhodnocování podmínek zjišťujících, jaké zařízení žádá o přerušení. Druhou metodou je přerušení, v kterém se zařízení identifikuje pomocí kódu přerušení. Zařízení vyšle společně s požadavkem na přerušení také tzv. kód přerušení. Podle tohoto kódu procesor pozná, kterou obslužní rutinu má vykonat. Tato metoda má nevýhodu v tom, že může dojít ke kolizím stejným typům kódů přerušení. Třetí a současně nejpoužívanější metodou je obsluha přerušení pomocí řadiče přerušení. Řadič přerušení má k sobě přiřazený určitý počet vstupů, ke kterým jsou připojeny zařízení žádající na jednotlivých vstupech o přerušení procesoru. Aktivuje-li nějaké zařízení požadavek na přerušení, řadič požadavek příjme a předá ho procesoru. Pokud je požadavek procesorem akceptován, proběhne například kontrola procesorem, jestli je vůbec dané přerušení aktivováno. Pokud ano, řadič předá procesoru kód typu zařízení žádajícího o přerušení. Nevýhoda

z mého pohledu je nutnost prvotního naprogramování přiřazení kódů zařízení k příslušným vstupům řadiče. Tato nevýhoda obvykle odpadá, pokud používáme již sestavený kit od výrobce, který má řadič již naprogramován. Výhodou této metody žádosti o přerušení je následná možnost modifikace řadiče, pomocí které můžeme určovat jednotlivé priority přerušení, blokovat některé vstupy apod.[2]

3.2 Obsluha přerušení ve spojení s BIG8051

Procesor C8051F040 zahrnuje i s rozšířenou sadou možnost 20 druhů zdrojů přerušení s možností volby mezi dvěma prioritními úrovněmi. Každé přerušení má k sobě přiřazen jeden, nebo více příslušných bitů v SFR registrech. Pokud je příslušící bit nastaven řadičem přerušení do logické 1, je vyvoláno přerušení na daném zdroji, ke kterému je registr určen. Po nastavení příslušného bitu se po dokončení právě prováděné instrukce přepne do rutiny obsluhující konkrétní přerušení. Každé přerušení by mělo končit příkazem return pro navrácení procesoru na adresu, kde se procesor nalézal před přerušením. Každé přerušení se dá povolit, nebo zakázat pomocí příslušných registrů IE až IE2. Pokud chceme jakékoliv přerušení aktivovat je zapotřebí nastavit EA bit do logické 1, který globálně povoluje, nebo zakazuje všechna přerušení. Každé přerušení má pevně určený adresní vektor, díky kterému procesor ví, kde se nachází rutina pro obsloužení přerušení a jakou má přerušení prioritu.

Externí přerušení (/INT0 a /INT1), které využívám v regulované soustavě pro zpracování informace o aktuálním průtoku z průtokoměru, lze nastavit aby se obslužní rutina vykonala buďto při napěťové úrovni 0 na vstupním pinu, nebo na sestupnou hranu signálu. Tohoto nastavení se dosáhne nakonfigurováním příslušných registrů IT0 a IT1. V mém případě využívám hranově orientované externí přerušení, v kterém si čítám čtvercové impulzy z průtokoměru. Při této konfiguraci není zapotřebí softwarově nulovat příslušný bit vyvolávající přerušení. Každé přerušení lze individuálně naprogramovat na dvě prioritní úrovně: low a high. Přerušení s nízkou prioritou může být přerušeno přerušením s vyšší prioritou. Naopak přerušení s prioritou high nemůže být přerušeno žádným jiným požadavkem. Každé přerušení má pro nastavení priority svůj konkrétní SFR registr (IP – EIP2). Ve výchozím nastavením jsou všechny priority nastaveny na hodnotu low.

Latence odezvy na přerušení záleží na stavu, ve kterém se nachází procesor v čase, kdy je vyvoláno přerušení. Nejkratší časová odezva na přerušení je 5 taktů systémových hodin.

První takt indikuje žádost o přerušení a během zbylých 4 taktů se dokončí příkaz LCALL, který zavolá a připraví obslužnou rutinu. Oproti tomu nejdelší reakce může trvat až 18 taktů systémových hodin. Nastává tomu tehdy, když se procesor zrovna nalézá ve stavu, kdy je zavolána funkce RETI, která slouží pro návrat na adresu programu a v tom momentě přijde požadavek na přerušení. První takt systémových hodin je detekováno přerušení, poté 5 taktů trvá vykonání RETI instrukce, dále 8 taktů je věnováno pro dokončení instrukce DIV a 4 takty pro vykonání instrukce LCALL a zavolání obslužné rutiny.[1]

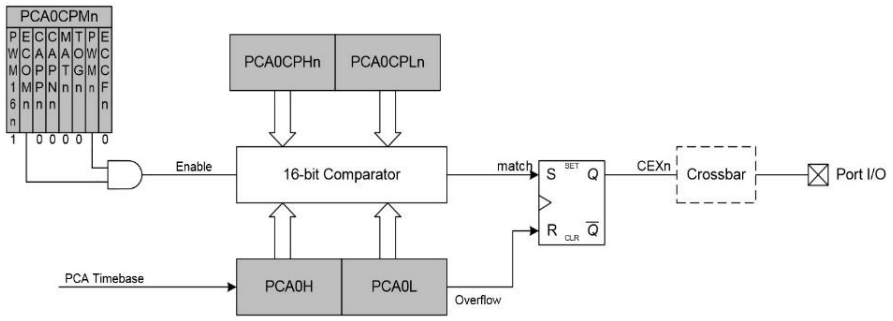
3.3 Pulzně šířková modulace

Pulzně šířková modulace, neboli PWM je způsob kódování přenosu dat mezi vysílacím a přijímacím zařízením. Pulzně šířková modulace je signál vysílaný s konstantní časovou periodou T, u kterého se určuje velikost střídy napětí. Střída napětí určuje poměr délky impulzu, vzhledem k délce mezery uvažované v jedné periodě T. Střída se uvádí buďto jako poměr (1:1, 2:1, 1:5, ...), kdy je nutné uvést, které číslo označuje část periody s napětím a které část bez napětí. Další možností je střídu uvést procentuálně například 100%, 50%, 0.5%, kde procento vyjadřuje zastoupení napětí na výstupu vzhledem k jedné periodě T. V odborné anglické literatuře, se můžeme setkat, se synonymem pro střídu napětí s názvem duty cycle. Příklad použití PWM může být například ovládání stejnosměrného motorku, kde procesor generuje PWM signál a přijímací část představuje budič motoru. Z opačného pohledu, kdy procesor bude PWM signál zpracovávat, tato situace může nastat například při získávání informací z inteligentního snímače teploty, z kterého nám rozkódována informace například po dosazení do příslušného vzorečku může sdělit aktuální teplotu snímaného objektu.[3]

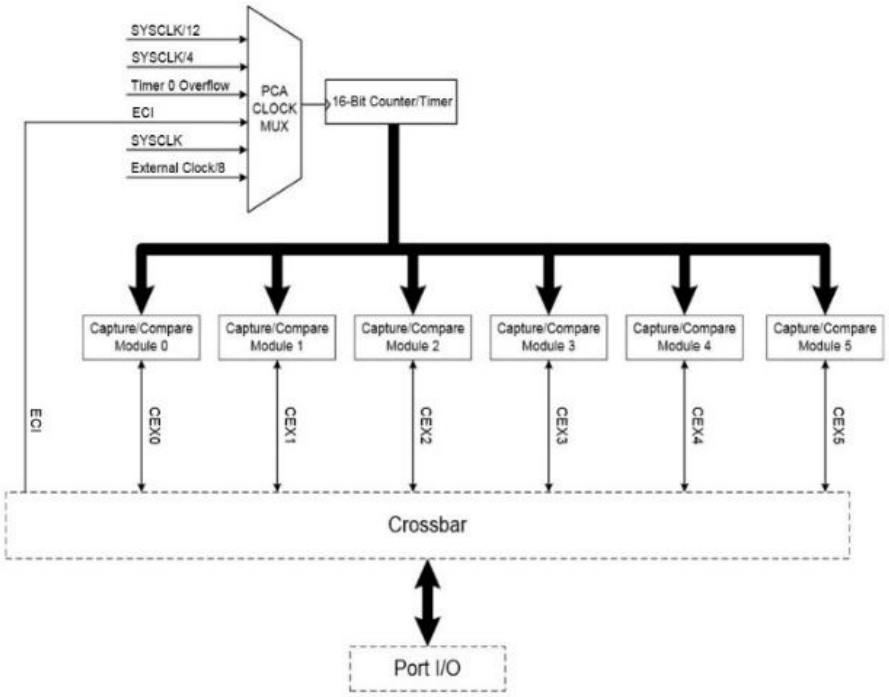
3.4 PWM ve spojení s BIG8051

Pro generování PWM signálu se na platformě BIG8051 nabízí možnost využití modulu procesoru PCA (Programmable count array). Jedná se o 16 bitový čítač/časovač, jehož výstup je připojen na šest modulů, které každý zvlášť může pracovat v šesti operačních režimech. Hranově orientovaný režim pro zachytávání signálu, softwarový časovač, vysokorychlostní výstup, frekvenční výstup, nebo 8 a 16 bitový generátor PWM signálu, který bude generován na přiřazeném výstupním pinu procesoru. Pro nastavení režimu každého z modulů slouží jim přiřazené SFR registry (PCA0CPMn). Pro mé účely genero-

vání PWM signálu jsem vybral modul0 s nastavením 16 bitového pulzně šířkového modulátoru. Jeho princip spočívá v nastavení dvou 8 bitových registrů, které jsou pomocí 16 bitového komparátoru porovnávány s nastavenou hodnotou v registru PCA čítače. Pokud čítač dosáhne hodnoty nastavené v registrech PCA0CPH0 a PCA0CPL0, které značí horních a spodních 8 bitů nastavené hodnoty, přepne logickou hodnotu na přiřazeném výstupním pinu modulu do logické 1 (3,3V) a naopak po přetečení čítače nastaví výstupní pin do logické 0 (0V). Díky této vlastnosti nastavuji pomocí registrů PCA0CPH0 a PCA0CPL0 střídu PWM signálu v rozmezí od 0 – 65535. V mé implementaci programu nastavuji hodnotu registrů v rutině přerušení od PCA čítače, které je třeba povolit bitem ECF v PCA0MD registru, který slouží pro konfiguraci PCA. PCA čítač má na výběr mezi šesti zdroji hodinového signálu. Systémové hodiny bez dělení, nebo systémové hodiny dělené konstantou 4, nebo 12. Další možnost je přivést jako zdroj signálu pro čítač přetečení od systémového časovače *T0*, nebo připojení externího hodinového signálu. Pro účely této práce jsem zvolil systémové hodiny bez dělení pro co nejpřesnější interpretaci napětí na výstupním pinu.[1]



Obrázek 5 - Blokové schéma generování 16 bitového PWM signálu. [1]



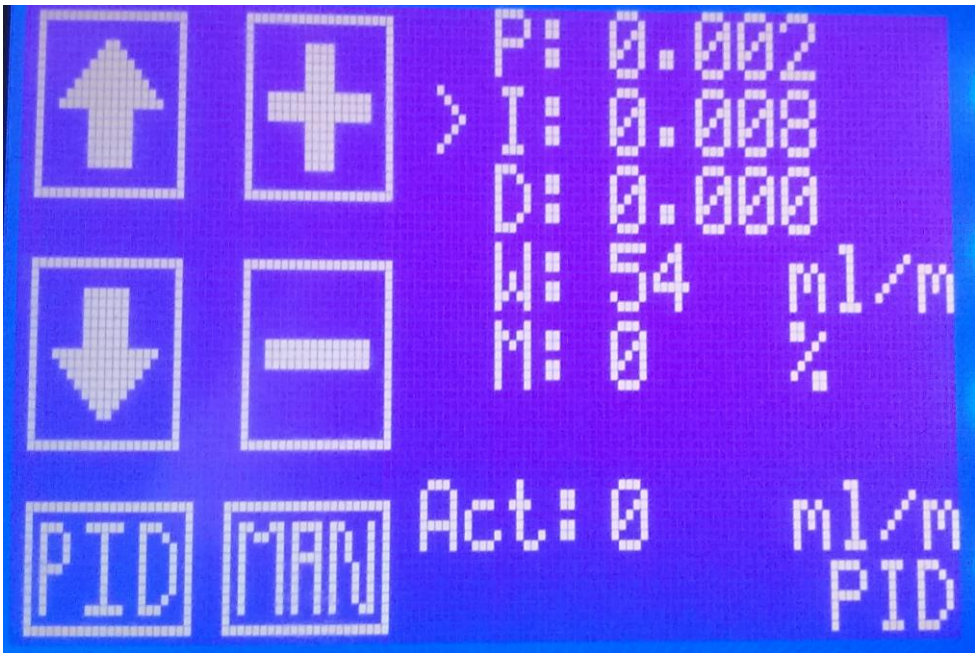
Obrázek 6 - Blokové schéma PCA. [1]

3.5 Rozhraní pro nastavení parametrů regulátoru

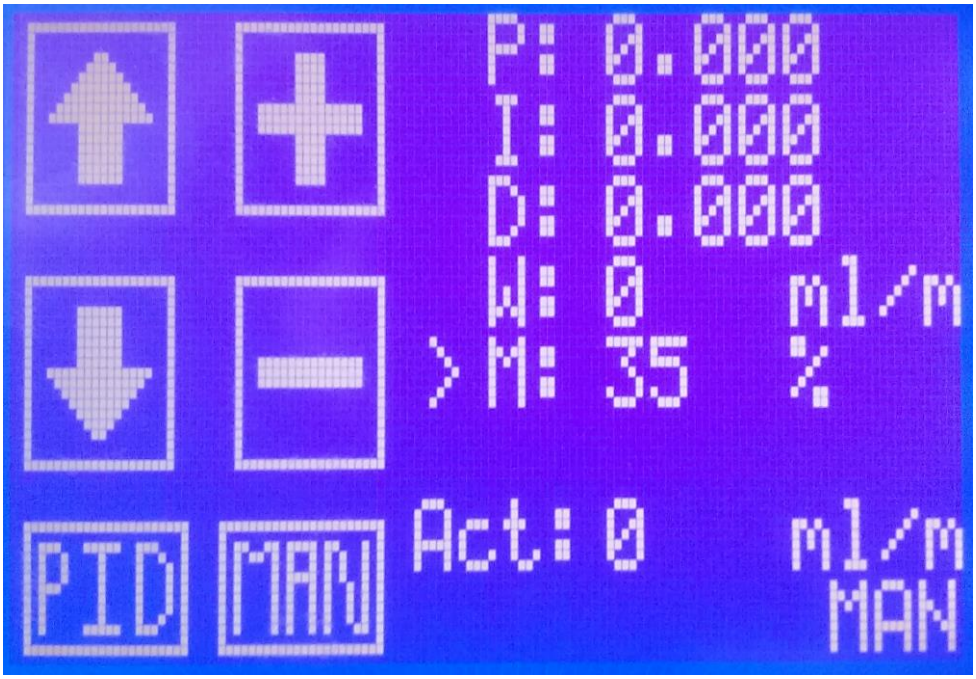
Pro komunikaci s procesorem se nabízí několik možností při použití vývojového kitu BIG8051. První možností je využití na desce dostupného komunikačního rozhraní jako je sběrnice CAN, SMBus, UART, nebo SPI, pomocí kterých je možné komunikovat s externím zařízením, jako je například klávesnice, nebo jakékoliv jiné zařízení umožňující interakci s uživatelem a zároveň podporující některé z těchto komunikačních rozhraní. Další externí komunikace s deskou může být pomocí ethernetového rozhraní, které bude sloužit jako alternativní možnost zadávání parametrů regulátoru. Další možností je využití již implementovaných periferií na desce. Mezi tyto periferie lze zahrnou 64 tlačítek připojených na každý pin procesoru. Další podle mne efektivnější a nejvíce uživatelský přívětivá z nabízených možností je využití grafického displeje spolu s dotykovou vrstvou pro přehlednou a intuitivní obsluhu regulátoru uživatelem. Pro výpis aktuálních hodnot regulátoru se mi nabízela možnost využití buďto LCD displeje s 2 řádky po 16 znacích, nebo výpis hodnot přímo na grafický displej s dotykovou vrstvou. Zvolil jsem druhou variantu s využitím grafického displeje, protože se mi výpis všech potřebných hodnot spolu s vykreslenými tlačítky pro ovládání regulátoru na velikost displeje 128x64 bodů vešel a nebylo potřeba využití LCD displeje. Bohužel omezení v kombinaci LCD a dotykové vrstvy bylo i ze strany

desky, protože využití dotykové vrstvy spolu s LCD displejem není možné. Piny procesoru se v tomto použití překrývají a je třeba si zvolit pouze mezi jednou z těchto periferií.

Dotyková vrstva pracuje na principu odporová folie. Odporová folie je tenká vrstva citlivá na tlak, který je vyvíjen na její plochu. Princip spočívá v umístění dvou folií přes sebe, kde každá folie má na jednom konci umístěný vodič připojený k zemi a na druhém konci připojený vodič k napájecímu napětí 3,3V. Vrstvy jsou na sobě položeny tak, že první vrstva má vodiče umístěné vlevo k zemi a vpravo k napětí 3,3V. Tato vrstva bude snímat polohu dotyku na ose x a druhá vrstva je o 90° pootočena, takže má jeden vodič dole připojený k zemi a druhý vodič nahoře připojený k napájecímu napětí. Tato vrstva bude určovat polohu dotyku v ose y. Všechny tyto čtyři vodiče jsou vyvedeny pomocí plochého kabelu nalevo touchpadu a připojeny do desky k A/D převodníku, který bude vyhodnocovat napětí na každé z vrstev. Například pokud budeme chtít vyhodnotit pozici dotyku na ose x, je zapotřebí připojit levý vodič na vrstvě snímané osu x k nule a pravý vodič k napájecímu napětí a stisk na určitém místě vyvolá odpor mezi oběma vodiči a tato změna se projeví napětí na kabelu, který je přiveden do A/D převodníku. Čím více stiskneme na dotykové vrstvě vlevo, tím se bude napětí blížit k nule a naopak při stisku více vpravo se bude napětí blížit k napájecímu napětí. Při snímání osy y je zapotřebí připojit spodní vodič k zemi a vrchní k napájecímu napětí. Pro získání pozice dotyku poté pouze stačí aktivovat požadovanou vrstvu a přečíst hodnotu z analogově digitálního převodníku.



Obrázek 8 - Komunikační rozhraní v PID režimu



Obrázek 7 - Komunikační rozhraní v manuálním režimu

4 Komponenty regulační soustavy

4.1 Ponorné čerpadlo

Jako regulované zařízení jsem zvolil nízkonapěťové ponorné čerpadlo, které je určeno pro čerpání vody, což pro mé laboratorní účely bez problému postačí. Čerpadlo je možno napájet buďto z autobaterie, solárního modulu, nebo ze zdroje stejnosměrného napětí až 12V, při kterém je schopno přečerpat až 10 l/min.

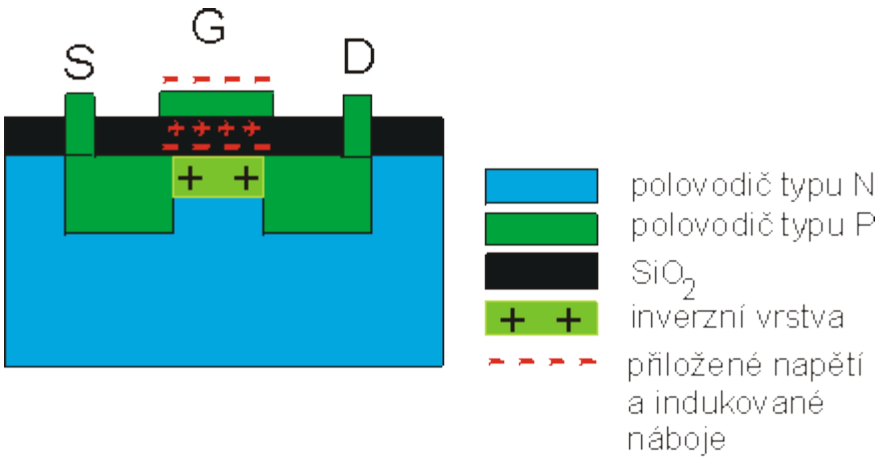
Obrázek 9 - Ponorné čerpadlo. [4]

Je třeba dát pozor na dlouhodobější používání, protože při tomto maximálním výkonu je doporučena doba provozu maximálně 30 minut, poté by hrozilo přehřátí a následná porucha čerpadla. Doporučené provozní napětí se tedy pohybuje od 6V do 9 V, při kterém je schopno pracovat v trvalém provozu, pokud je zároveň ponořeno ve vodě, která slouží jako chlazení. Čerpací tlak při maximálním výkonu dosahuje 0,6 bar a maximální čerpací výška je 6 metrů.

Čerpadlo je poháněno stejnosměrným motorkem řízeného střední hodnotou napětí na budiči motorku. Nabízeli se zde dvě možnosti jak čerpadlo řídit pomocí velikosti napětí. První možnost byla pomocí D/A převodníku, nebo generování signálu PWM na pinu procesoru. Jelikož u regulační soustavy je kladen důraz na rychlost výpočtu, tak jsem zvolil řízení pomocí PWM signálu, který je méně výpočetně náročný, než výstup D/A převodníku a deska nabízí již předpřipravené registry po jednoduchou obsluhu generování PWM signálu. [4]

4.2 Výkonový člen

Z mikroprocesoru lze na výstupní pin přivézt maximální napětí pouze 3,3V a budič stejnosměrného motorku pracuje s napětím až 12V. Je tedy potřeba mezi regulátor a čerpadlo umístit výkonový člen, který napětí úměrně zesílí. Pro tento účel je vhodné využít vlastnosti tranzistoru, který může sloužit jako zesilovač a dokáže z malého napětí, nebo proudu na vstupu vyvolat velké změny na výstupu. Protože pracujeme s napětím, je vhodné použít tranzistor typu MOSFET, který se otevírá, nebo zavírá na základě přivedeného napětí na hradlo G. Tranzistor typu MOSFET obsahuje tři hradla. První z nich se nazývá zdrojové (Source) značeno ve schématu písmenem S. Dalším hradlem je odtokové (Drain) značeno písmenem D. Třetím hradlem tranzistoru je řídící hradlo (Gate) značeno písmenem G. Tranzistor je tvořen dvěma polovodiči typu N a P a tenkou vrstvou izolantu například z $Si0_2$.



Obrázek 10 - Schéma principu vodivosti MOSFET tranzistoru. [5]

Princip spočívá v umístění dvou polovodičů typu P (source a drain) na polovodič typu N, které slouží jako elektrody. Řídící elektroda G je umístěna mezi S a D a je od nich izolována izolační vrstvou SiO_2. Pokud přivedeme na řídící elektrodu G dostatečně vysoké napětí, avšak ne tak vysoké, aby došlo k poškození izolační vrstvičky, vytvoří se na povrchu polovodiče typu N v blízkosti řídícího hradla G tzv. inverzní vrstva. Inverzní vrstva je vrstva opačné vodivosti, v tomto případě vodivosti polovodiče typu P. Tím dojde k propojení elektrod S a D a může mezi nimi procházet proud. [6]

V mé práci mám provedenou stejnou realizaci akorát s opačně přivedenými póly na jednotlivá hradla S, G a D, nežli je znázorněno na obrázku. Na hradlo G je přivedeno napětí

PWM signálu, které spojuje svými kladnými elektrodami hradla S a D. Na hradlo S je jako zdroj přivedeno uzemnění napájecího napětí čerpadla a na hradlo D je přivedeno uzemnění čerpadla. Pokud na hradlo G není přivedeno dostatečně vysoké napětí tak se polovodiče S a D nespojí a čerpadlo není spojeno se zemí svého napájecího napětí a neproudí tedy do něj žádný proud (je vypnuté). Po přivedení napětí na hradlo G se spojí země napájecí napětí 12V z napájecího zdroje se zemí čerpadla a tím začne do čerpadla procházet proud (čerpadlo sepne). Viz obrázek - Blokové schéma zapojení výkonového členu

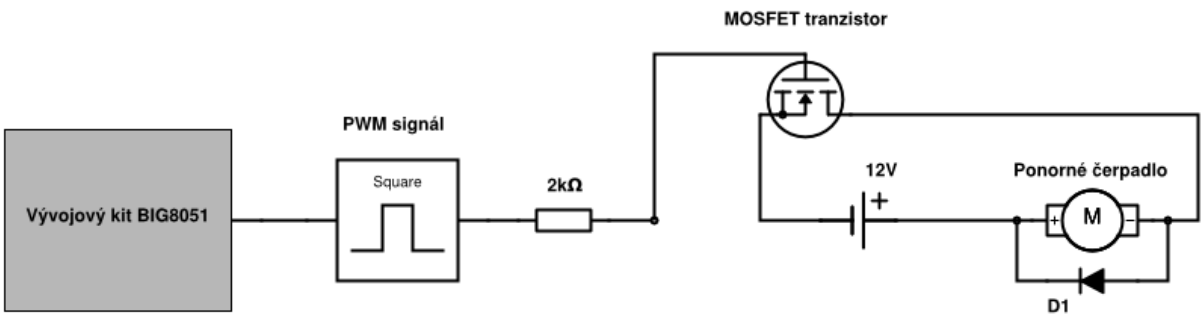
Při výběru tranzistoru byl kladen důraz na co nejkratší dobu úplného otevření a úplného uzavření spoje mezi S a D. Důvodem je generování signálu PWM, které pracuje v celkem vysoké frekvenci. Pokud by přepínání mezi napětím 0V a 3,3V přivedeného na hradlo G signálem PWM probíhalo moc rychle, mohlo by se stát, že by tranzistor nestíhal rozpojovat kontakt mezi S a D. To by způsobovalo jeho trvalé otevření, protékal by ním pořád proud, který by způsobil přehřátí tranzistoru a jeho následné zničení.

Doba přetečení časovače pro obsluhu PWM signálu je 32ms. Z toho vyplývá, jestliže je vstupem na hradle G PWM signál o střídě 99% (3,27V), má tranzistor maximálně 340ns na rozpojení hradla S a D.

Jako výkonový člen jsem zvolil tranzistor typu MOSFET s názvem IRL540N, který v parametrech uvádí dobu úplného otevření 81ns a dobu úplného uzavření 62ns. Tranzistor po dosáhnutí úplného uzavření, nebo otevření zahrnuje časovou prodlevu, která slouží k ochraně proti rychlému přepínání mezi stavy sepnuto a nesepnuto. Po úplném otevření má tranzistor prodlevu 11ns a po úplném uzavření má prodlevu 39ns. Prodleva po uzavření je výrazně delší oproti úplnému otevření a to podle mého názoru z důvodu ochlazení tranzistoru. Z toho vyplývá, že tranzistor poskytuje nejrychlejší úplné otevření za 120ns i se započítanou prodlevou po uzavření, Úplné uzavření tranzistoru je tedy možné očekávat po 73ns i se započítanou prodlevou po úplném otevření. Při výše zmíněné střídě 99% se tranzistor stihne úplně uzavřít. Pokud by nastala opačná krajní situace přivedení střídy 1% (0,03V), tak se tranzistor stihne úplně otevřít, proto je pro mé účely dostačující.

Mezi zem a napájecí napětí ponorného čerpadla jsem umístil diodu, která zabrání po rozepnutí přivedeného proudu do ponorného čerpadla vygenerování tzv. napěťové špičky, která by mohla spínací tranzistor prorazit. Napěťová špička je krátkodobé přepětí i o de-

sítky procent, které může být vygenerováno rozepnutím přívodu napájecího napětí čerpadla. Dioda způsobí, že se po rozepnutí proudu obvod uzavře a nedojde tedy k vygenerování nežádoucí napěťové špičky. Dioda je zde tedy pouze z důvodu ochrany tranzistoru.



Obrázek 11 - Blokové schéma zapojení výkonového členu

4.3 Průtokoměr

Abychom mohli ponorné čerpadlo regulovat, je zapotřebí mít informaci o tom, jaký aktuální průtok právě prochází soustavou. K tomu nám poslouží průtokoměr od společnosti B.I.O-TECH, s rozsahem měření od 0,015l/min až 0,8 l/min, který je sice znatelně nižší než maximální možný průtok čerpadla, ale pro mé laboratorní účely dostačující. Kvůli této vlastnosti jsem zvolil celou soustavu měřit a řídit v jednotkách mililitr za minutu. Snímatelný rozsah výkonu čerpadla a zároveň nastavení žádané hodnoty je tedy od 15 – 800 ml/min.

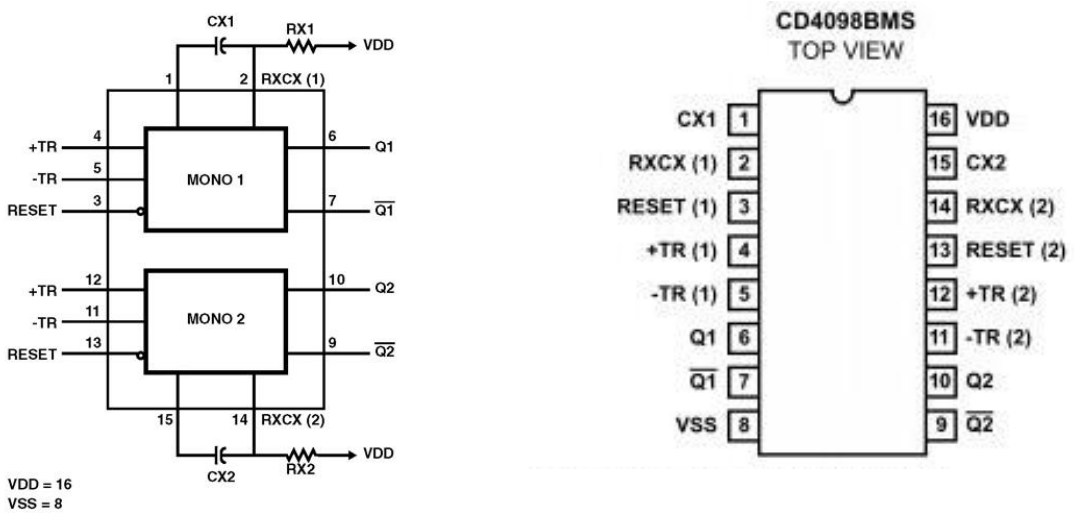
Obrázek 12 – Průtokoměr. [7]

Snímací systém průtokoměru zaručuje téměř neomezenou životnost a vysílá konečně upravené obdélníkové signály, které jsou přivedeny na vstupní pin procesoru. S každou sestupnou hranou signálu se vyvolá rutina externího přerušení na procesoru, která podle počtu obdélníkových signálů vypočítá aktuální průtok v soustavě. Průtokoměr obsahuje tři vodiče. První z nich generuje výše zmíněný obdélníkový signál, druhý je uzemnění napájecího napětí a třetí je napájecí napětí v rozsahu 5V až 24 V stejnosměrného napětí. Mezi vstup provozního napětí průtokoměru a výstupního pinu vysílající obdélníkový signál je zapotřebí umístit odpor 1,6kΩ až 2,2kΩ, kvůli ustálení signálu. Bez použití tohoto odporu jsou obdélníkové tvary signálu nepřesné a dochází ke generování nežádoucího šumu. Pro mé účely jsem zvolil provozní napětí 5V.

Nevýhodu, kterou má vývojový kit BIG8051 je málo způsobů reakce na vyvolání externího přerušení. Procesor umožňuje reagovat pouze na hladinu přivedeného impulzu pouze na úroveň low, nebo na sestupnou hranu signálu viz kapitola Obsluha přerušení ve spojení s BIG8051. Externí přerušení zde využívám na sčítání počtu impulzů z průtokoměru, který generuje čtvercový signál. Pro tento signál se hodí využít reakci externího přerušení na sestupné hrany signálu. V externím přerušení si poté čítám pulzy, které po přetečení časovače nastaveného na periodu po jedné sekundě počet načítaných pulzů vyhodnocuji a nuluji pro další čítání. Problém nastává v nízké výstupní frekvenci signálu z průtokoměru, který je zhruba 10Hz až 20 Hz. To mi způsobuje, že mám poměrně málo informací z průtokoměru za jednu sekundu, které jsou navíc omezeny čítáním pouze sestupných hran. Pokud v prvním cyklu čítání pulzů napočítám například 12 sestupných hran a v druhém pouze jen 11 je to rozdíl zhruba 9%, což se na informaci o aktuálním průtoku a na celé regulační soustavě projeví její nepřesností. Motivace byla tedy upravit výstupní signál z průtokoměru tak, abych mohl čítat pulzy i při náběžné hraně tohoto signálu a měl tedy dvojnásobné množství informací z průtokoměru a minimalizoval možnou odchylku ve výpočtu aktuální průtoku.[7]

4.4 Obvod CD 4098

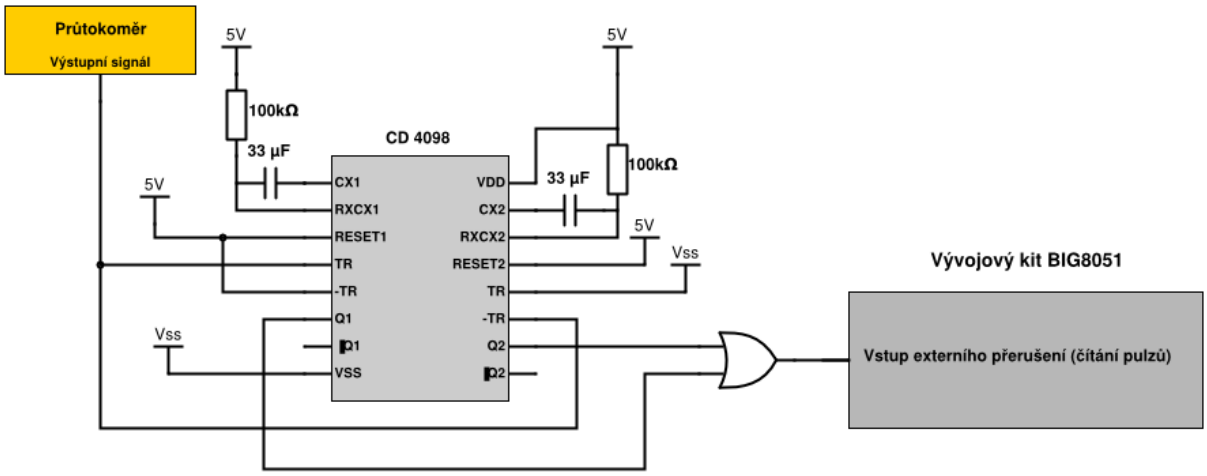
Nabízela se tedy možnost využití obvodu, umístěného mezi výstupní signál z průtokoměru a pinem externího přerušení na desce, který bude na vstupu reagovat na náběžnou i sestupnou hranu výstupního signálu z průtokoměru a na výstupu obvodu generoval impulz, který bude obsahovat sestupnou hranu, na kterou již deska zareaguje. Tomuto chování vyhovuje obvod CD 4098.



Obrázek 13 - Schéma obvodu CD 4098. [8]

Tento obvod obsahuje dva identické na sobě nezávislé monostabilní obvody, které podle zapojení mohou reagovat na vstupu na náběžnou (vstup +TR) nebo na sestupnou hranu (vstup -TR) přivedeného signálu. Na výstupu Q je poté vygenerován čtvercový impulz, který dosahuje hodnoty provozního napětí obvodu přivedeného na pin VDD. Negovaný výstup je přiveden na pin $\bar{Q}$. Délka výstupního pulzu se nastavuje pomocí velikosti kondenzátoru CX a odporu RX. Tato délka impulzu se dá vypočítat vztahem $T_x = \frac{CX*RX}{2}$ s velikostí kondenzátoru od 0,01µF do 100µF a externího odporu od 5kΩ. Velikost pulzu jsem řešil zapojením 10nF kondenzátoru a odporu o velikosti 5kΩ, který budu podle potřeby zvětšovat, nebo zmenšovat odporovým trimerem. Je třeba dát pozor, aby signál nebyl moc dlouhý a nepřekročil dobu kdy je vygenerována další hrana signálu z průtokoměru. Zároveň je zapotřebí ho generovat dostatečně dlouhý, aby vyvolal externí přerušení na pinu procesoru. [8]

Na obrázku níže je zobrazeno blokové schéma zapojení obvodu CD4098. Pro délku generovaného pulzu jsem zvolil kondenzátor o velikosti 33µF a odpor 100kΩ. Při této konfiguraci se mi budou generovat pulzy dlouhé 1,65ms. Tato délka stačí k tomu, aby mikroprocesor zareagoval na sestupnou hranu a vyvolal rutinu externího přerušení a zároveň pulz není tak dlouhý, aby délkou přesáhl další pulz z průtokoměru.



Obrázek 14 - Blokové schéma zapojení obvodu CD4098

Na obrázku níže můžete vidět chování obvodu po přivedení signálu z průtokoměru. Žlutý signál reprezentuje výstup z obvodu CD4098 a modrý signál pulzy z průtokoměru.

Obrázek 15 - Snímek z osciloskopu, zobrazující výstup obvodu CD4098

5 Regulace

5.1 Druhy a popis regulace

Regulací rozumíme udržování určité výstupní veličiny, může se jednat například o teplotu, tlak, nebo průtok v regulované soustavě na zadané požadované hodnotě. Regulátor je technické zařízení, které tuto funkci realizuje. Regulátor zpracovává vstupní informace, kterými jsou požadované a aktuální hodnoty v regulované soustavě a pomocí akčních veličin se snaží zasáhnout do soustavy tak, aby mezi výsledky žádané a aktuální hodnoty byl minimální rozdíl (minimální regulační odchylka). Základní rozdělení regulátorů z pohledu jejich chování v čase lze rozdělit regulátory na spojité a diskrétní. Spojité regulátory dokáží reagovat na změny v čase, to znamená, že dokáží měnit svoji výstupní veličinu spojitou funkcí v každém časovém okamžiku. Diskrétní regulátory mění svoji výstupní veličinu s časem nespojitě. Nespojitě v tomto případě znamená, že ke změně hodnot dochází v určitém časovém okamžiku, obvykle ve zvolené časové periodě skokově z jedné hodnoty na jinou, která se pak až do další časové periody udržuje v soustavě konstantní. Většina moderních regulátorů sice pracuje diskrétně, ale jejich změny na vstupech a výstupech jsou prováděny po velice krátkých časových intervalech, takže se navenek chovají jako spojité.

Základním prvkem diskrétních regulátorů jsou číslicové počítače, obvykle jednočipové procesory. Nezbytnou součástí těchto procesorů pro zhotovení číslicového regulátoru jsou časovací obvody, popřípadě i další pomocné hlídací obvody. Pro řízení regulované soustavy je zapotřebí, aby procesor měl buďto integrované, nebo jinak externě připojené vstupně/výstupní obvody, jako jsou například A/D, D/A převodníky, generátor PWM signálu pro potřebnou komunikaci s komponentami soustavy, kterými mohou být čidla, senzory, motorky apod.

Další rozdělení regulátorů může být z pohledu jejich konstrukce. První typem je kompaktní regulátor. Jde o kompletní přístroj umístěný v pouzdru obsahující vlastní mikropočítač, vstupy, výstupy, obvody pro přizpůsobení chování regulátoru, komunikační rozhraní, jako například tlačítka, nebo klávesnice a v neposlední řadě zobrazovací rozhraní, pro informování uživatele o aktuálních hodnotách. U tohoto typu regulátoru výrobci nabízí řadu možností provedení vstupů, výstupů, komunikačních rozhraní apod. Nevýhodou je zde omezení možnosti změnit konfiguraci regulátoru uživatelem. Uživatel je schopný měnit parametry regulátoru, ale pokud by chtěl například regulátor rozšířit o vstupní část je tento typ regulátoru poněkud nepružný. Jako druhý typ regulátorů jsou modulární regulátory, které řeší problém rozšiřitelnosti vstupů a výstupů soustavy. U tohoto typu je možnost využití modulárních vstupně/výstupních modulů a jejich výměnou, nebo doplněním lze upravovat počet vstupů a výstupů regulátoru oproti kompaktnímu typu.

Obrázek 17 - Modulární regulátor. [9]

Obrázek 16 - Kompaktní regulátor. [10]

Výhodou modulárních regulátorů je možnost rozšíření o počet vstupů, ba i dokonce regulačních jednotek, které mohou pracovat buďto každá samostatně, nebo lze regulace provázat a ovlivňovat se navzájem. Příklad použití modulárního regulátoru může být pro regulaci a řízení stanic tepelného hospodářství. Jeden regulátor pak umí zabezpečit ekvivalentní regulaci třeba i desítek topných větví, ohřev užitkové vody, automatické střídání čerpadel apod.

Jako poslední možnost rozdělení regulátorů bych zde uvedl na základě možnosti uživatele ovlivnit chování regulátoru na programové rovině. Většinu regulátorů lze konfigurovat

pouze pomocí nastavení konstant regulátoru, nebo volbou mezi několika typy regulačních algoritmů jako jsou například PID, nebo dvoupolohový regulátor apod. Proto existují i volně programovatelné regulátory, které umožňují například uživateli naprogramovat vlastní ovládací algoritmus, nebo možnost návrhu ovládacích obrazovek pomocí využití připraveného vývojového prostředí od výrobce regulátoru. V tomto vývojovém prostředí lze ovládací algoritmus regulátoru většinou jednoduše sestavovat pomocí připraveného rozhraní. Například grafické rozhraní umožňující skládat algoritmus pomocí časových a logických funkčních bloků.

5.2 PID regulátor

Proporcionálně integrační derivační regulátory jsou bezkonkurenčně nejčastěji používané regulátory v průmyslu. Podle statistik je až 95% všech regulačních algoritmů typu PID a většina z nich využívá pouze proporcionální a integrační složku. Níže je popsána obecná rovnice PID regulátoru.

$$u(t) = K_p * e(t) + K_i * \int_0^t e(t)dt + K_d * \frac{de(t)}{dt} \qquad (1)$$

PID regulátor je složen ze tří složek, každá mající svojí roli na výsledek výpočtu akčního zásahu regulace. Regulátor se řadí před řízenou soustavu a jeho vstupem je regulační odchylka značená $e(t)$, která je vypočítána rozdílem žádané hodnoty regulované veličiny od aktuální hodnoty regulované veličiny v regulační soustavě získanou například ze senzoru, čidla, otáčkoměru apod. PID regulátor potom dále pracuje pouze s touto regulační odchylkou a snaží se, aby byla její hodnota co nejnižší, v ideálním případě rovna nule. Jeho výstupem je poté akční veličina značena $u(t)$, která je převedena na konkrétní akční zásah odpovídající fyzikální veličině určené pro manipulování s regulovanou soustavou například stejnosměrné napětí. Akční zásah je přiveden buďto do řídící soustavy přímo, pomocí zvoleného vodícího média, nebo skrze výkonový zesilovač, který například hodnotu napětí na výstupu D/A převodníku z regulátoru zesílí na požadovanou odpovídající hodnotu vzhledem k pracovnímu rozsahu budiče motoru. Výpočet akční veličiny se provádí, jak již bylo řečeno, třemi složkami regulátoru PID. Složky regulátoru lze zahrnout do výpočtu akční veličiny všechny, nebo jen některé z nich. Jejich výsledky se poté sečtou a vytvoří výstupní akční veličinu, která je následně zpracována a převedena na akční zásah.

Každý z výsledků tří složek má svůj účinek na změnu akční veličiny. Všechny složky mají k sobě přiřazenou konstantu, která určuje, jaký vliv bude mít daná složka na výpočet akční veličiny. Pokud je tato konstanta nulová, nebude složka zahrnuta do výpočtu vůbec. Každá složka má svůj účel a každá z nich reaguje jinak na rozdíl, nebo změnu regulační odchylky v soustavě. Konstanty slouží k nastavení chování a reakce regulátoru na změnu, nebo velikost regulační odchylky. [11]

5.2.1 Složky PID regulátoru

První složkou regulátoru je proporcionální složka značena písmenem `P`, která má na svém výstupu vynásobenou regulační odchylku `e (t)` zvolenou proporcionální konstantou značenou `Kp`. Proporcionální složka určuje celkovou dynamiku a reakci celého systému na vzniklé velikosti regulační odchylky. Zesílení a účinnost `P` složky lze ovlivnit velikostí právě této konstanty. S rostoucí velikostí konstanty K_p roste citlivost a přesnost regulace, na úkor zmenšující se stability regulátoru a větší náchylnosti k nestálosti a rozkmitání celé soustavy. Ve vzorci proměnná `t` značí aktuální čas.

$$P_{term}(t) = K_p * e(t) \tag{2}$$

Druhou složkou regulátoru je integrační složka značená písmenem `I`. Výstupem integrační složky je hodnota o velikosti součtu integrálů vypočítaných z regulačních odchylek od času 0 do aktuální času `t`. Rozdíl oproti proporcionálnímu regulátoru je hlavně ten, že integrační složka stále přičítá integrál regulační odchylky v každém cyklu výpočtu. Z toho vyplývá, že čím je regulační odchylka větší a čím je déle v systému, tím více integrační složka ovlivňuje akční veličinu. Integrační složka je vhodná s kombinací s ostatními složkami regulátoru pro doladění akční veličiny. Její hlavní účinek je odstranění trvalé regulační odchylky regulátoru `P`. Účinnost integrační složky na výpočtu výsledné akční veličiny se nastavuje pomocí konstanty K_i. Vzoreček popisuje integrační složku spojitého PID regulátoru.

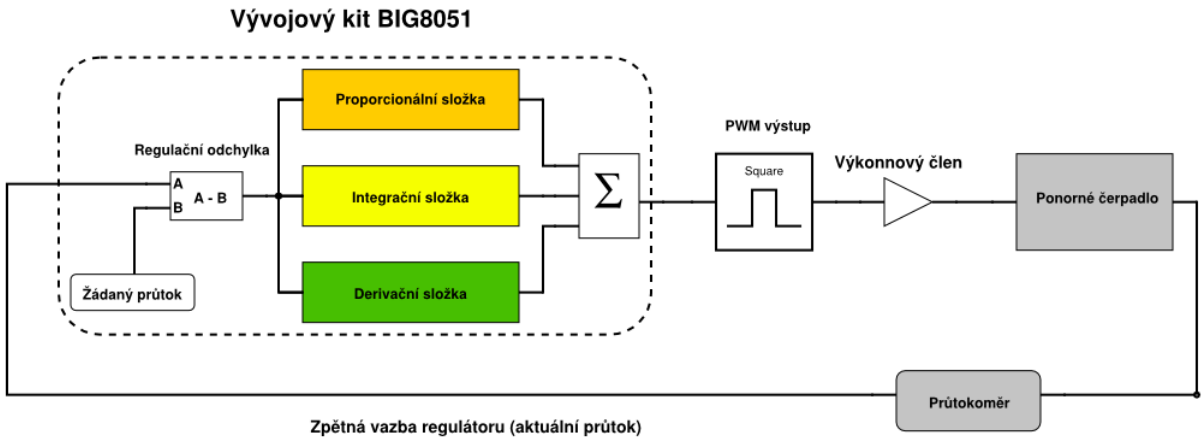
$$I_{term} = K_i * \int_0^t e(t)dt \tag{3}$$

Ze vzorečku lze vyčíst, jak již bylo napsáno, že akční veličina je přímo úměrná integrálu regulační odchylky a tento integrál je zesílen konstantou `Ki`. Příliš vysoké nastavení konstanty `Ki` může způsob rozkmitání celé soustavy.

Třetí složkou PID regulátoru je složka derivační značena písmenem *D*. Tuto složku jako jedinou není možné použít samostatně. Důvodem je, že výstupní veličina reaguje pouze na změny regulační odchylky. Pro konstantní regulační odchylku je derivace této veličiny nulová, protože derivace konstanty je nula. Pokud regulační odchylka klesá rychleji, derivační složka zvyšuje svůj výstup a naopak. Regulátor typu *D* lze použít pouze s proporcionálním složkou *P*, nebo integrační složkou *I*. V tomto spojení slouží jako urychlovač reakce regulátoru na změnu regulační odchylky. Je třeba dát pozor při použití této derivační složky, aby nepřinesla spíše potíže místo užitku a to hlavně u soustav, které jsou zahlceny vysokým šumem. Tento šum působí na velikost regulační odchylky, a tudíž stále ovlivňuje derivační složku, která může celou soustavu rozkmitat. Derivační složka, stejně jako všechny ostatní složky regulátoru, má svojí konstantu *Kd*, která určuje, jakou účinnost bude mít derivační složka na regulaci.

$$D_{term} = K_d * \frac{de(t)}{dt} \qquad (4)$$

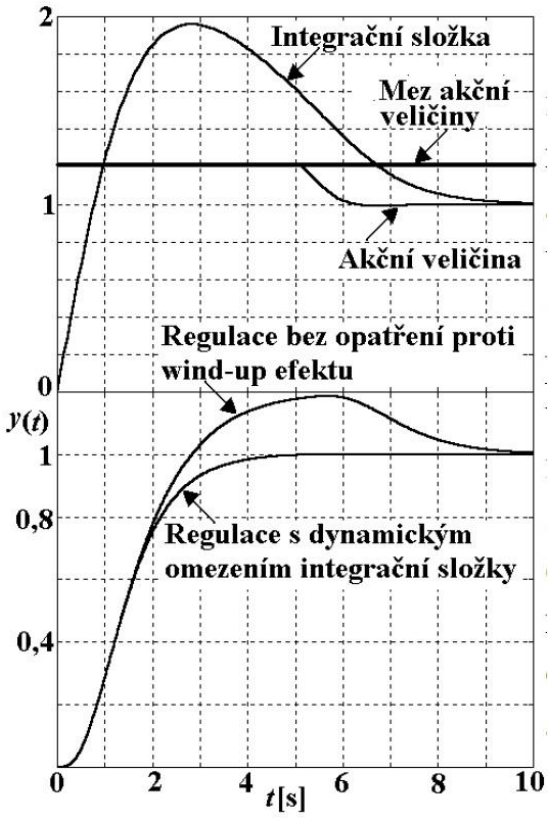
Výše zmíněné složky *P* a *I* se dají používat samostatně, nebo spolu paralelně zapojeny i s kombinací složky *D*. Tyto regulátory se nazývají sdružené regulátory, neboli kombinované. Jsou součtem vlastností jednotlivých složek PID regulátoru. V praxi se ale v drtivé většině setkáme se zapojením proporcionálně integračního regulátoru, protože zkušenostmi se ukázalo, že tato kombinace je v oblasti regulace účinná i bez derivační složky, která může mít nežádoucí vliv na zesílení šumu. Výhodou derivační složky je potlačení vlivu případné poruchy, nebo rychlou změnu akční veličiny. [11]



Obrázek 18 - Blokové schéma zapojení PID regulátoru

5.3 Wind-up efekt

Wind-up efekt je termín převzatý z angličtiny související s výpočtem akční veličiny regulátoru konkrétně výpočtu veličiny integračního regulátoru `I`. Wind-up efekt by se dalo přeložit jako zbytečné přesáhnutí, nebo naopak poklesnutí v našem případě integrační složky, která v každém kroku výpočtu algoritmu buďto narůstá, pokud je žádaná hodnota větší jak aktuální hodnota akční veličiny, nebo klesá, pokud je tomu naopak.



Obrázek 19 - Wind-up efekt s a bez omezení integrační složky. [12]

Nežádoucí efekt se objevuje většinou u soustav s pomalejší dynamikou, které trvá déle, než se dostane na žádanou hodnotu v regulované soustavě. Uveďme příklad na řešení praktické úlohy v této práci. Mějme čerpadlo v klidovém stavu (aktuální průtok 0 ml/min), po kterém začneme žádat průtok například 30 ml/min. V první řadě se vypočte regulační odchylka, na kterou zareaguje integrační regulátor tak, že bude integrovat regulační odchylky v čase, v kterém jsou soustavou snímány. Soustava má určitou dynamiku a trvá ji, než čerpadlo přečerpá první mililitry kapaliny skrze průtokoměr, abychom zaznamenali změnu aktuálního průtoku a zareagovali na zmírnění růstu akční veličiny.

Integrační složka může v tuto chvíli již překonat mez akční veličiny, na kterou je čerpadlo schopné reagovat. Tento jev nám v soustavě moc nevadí až do té doby, dokud aktuální hodnota soustavy nepřesáhne žádanou hodnotu a je zapotřebí, aby čerpadlo čerpalo méně, aby se rozdíl vyrovnal. V tomto momentě může být integrační složka příliš vysoká a bude jí trvat déle, než se znovu odečte do hodnoty, kdy se její změna promítne na výstupu. Toto platí i pro opačný případ kdy se integrační složka odčítá pod hranici, pro kterou je integrační složka užitečná. Toto se dá omezit jednoduchou podmínkou, kdy se testuje maximální a minimální hranice integrační složky, která je pro regulovanou soustavu efektivní.[12]

5.4 Odvození PSD regulátoru

PSD (proporcionálně sumační derivační) regulátor je diskrétní forma spojitého PID regulátoru. Od PSD regulátoru očekáváme stejnou funkci jako od spojitého PID regulátoru. To znamená, aby vstupní regulační odchylku zesiloval, integroval a derivoval podle námi zadaných parametrů. Proto při odvození PSD regulátoru se odvozuje a vychází z rovnice spojitého PID regulátoru. Níže můžete vidět obecnou rovnici PID regulátoru, z které jsem vycházel. Jednotlivé konstanty složek regulátoru jsou značeny Kp pro proporcionální, Ki pro integrační a Kd pro derivační složku. Proměnná t značí aktuální čas výpočtu, $e\ (t)$ značí regulační odchylku a výstup regulátoru $u\ (t)$ označuje akční veličinu regulátoru v čase t. Ve své práci jsem vycházel z rovnice spojitého regulátoru popsané níže. Viz rovnice (5). Tato rovnice oproti rovnici popsané výše (viz rovnice (1)) obsahuje zastoupení konstanty K_p ve všech složkách regulátoru. Výhoda je, že nastavení velikosti konstanty K_p urychlí, nebo naopak zpomalí reakci celého regulátoru.

$$u(t) = K_p[\, e(t) + \frac{1}{K_i} * \int_0^t e(t)dt + K_d * \frac{de(t)}{dt}] \qquad (5)$$

Proporcionální složka PID regulátoru se v diskrétním tvaru vyjádří jednoduše jako konstanta K_p vynásobena regulační odchylku $e\ (t)$.

$$P(t) = K_p * e(t) \qquad (6)$$

Při přepočtu integrační složky do diskrétní (sumační) formy je zapotřebí se vypořádat s výpočtem integrálu. Přepočet integrálu do diskrétní formy jsem provedl pomocí metody lichoběžníkové metody. Princip lichoběžníkové metody spočívá v součtu vypočítaných obsahů lichoběžníků do času t, jejichž velikost je tvořena z hodnot regulační odchylky.

Šířka všech lichoběžníků je dána periodou vzorkování T (v milisekundách). Vzoreček níže popisuje výpočet sumační složky v čase t.

$$I(t) = \frac{K_p}{K_i}\int_0^t e(t)dT \approx I(t-1) + \frac{K_p * T * (e(i) + e(i-1))}{2 * K_i} \qquad (7)$$

Ze vzorečku lze vyčíst, že je zapotřebí si pamatovat předchozí výsledek sumační složky a regulační odchylky. Konstantu T mám zde zvolenou na hodnotu 1 000, protože vzorkování provádím každou sekundu.

Základní provedení diskrétní derivační složky je popsáno vzorečkem níže pomocí zpětné diference.

$$D(t) = K_p * K_d * \frac{de(t)}{dt} \approx \frac{K_p * K_d * (e(t) - e(t-1))}{T} \qquad (8)$$

Tento způsob odvození diskrétní derivační složky je nejjednodušší, ale bohužel jeho forma je velice náchylná na případný vzniklý ruch v soustavě. Z tohoto důvodu ve své práci používám formu diskrétní derivační složky odvození pomocí tustinovi transformace, kde parametr α může nabývat hodnoty od 0.05 až 0,2. [13]

$$D(t) = \frac{2 * K_p * K_d}{T + 2 * K_d * \alpha} * \big(e(t) - e(t-1)\big) + \frac{2 * K_p * K_d * \alpha - T}{T + 2 * K_d * \alpha} * D(t-1) \qquad (9)$$

6 Realizace regulátoru

6.1 Struktura programu

Všechny parametry regulátoru, které budou v této kapitole uvedeny, je možné zadávat a měnit pomocí dotykové vrstvy, která poskytuje jednoduchou obsluhu regulátoru pomocí šesti tlačítek. Hodnoty se zobrazují na grafickém displeji pod dotykovou vrstvou. Celá soustava zobrazuje velikost průtoku vody průtokoměrem v jednotkách ml/min. Pracovní rozsah s ohledem na průtokoměr, který má výrazně nižší snímací rozsah než rozsah výkonu čerpadla. Pracovní rozsah soustavy je tedy od 15 ml/min do 800 ml/min.

Regulační soustava je řízena dvěma 16 bitovými časovači. První je určen pro synchronizaci výpočtu PID algoritmu a druhý pro vyhodnocování impulzů z průtokoměru. Dále je zde použit 16 bitový generátor PWM signálu pro řízení výkonu čerpadla. Jako posledním nutným prostředkem pro regulování soustavy je zpětná vazba o aktuálním průtoku v soustavě. Tato zpětná vazba je zajištěna průtokoměrem, z kterého čítám pulzy v rutině externího přerušení. Program je navrhnut tak, aby umožňoval řízení průtoku ve dvou režimech ovládání soustavy. První režim řízení je manuální, ve kterém je zapotřebí zadat parametr `M` v rozsahu 0 – 100%. Tato procentuální velikost značí střídu výstupního PWM signálu z desky, který je dále zesílen výkonovým členem. Pokud zadáme v manuálním režimu parametr `M` na 100%, do čerpadla proudí skrze výkonový člen napětí 12V, které představuje maximální snesitelné napětí pro čerpadlo a čerpadlo tedy pracuje na plný výkon a naopak. Druhým režimem je PID režim, kde jak sám název napovídá, jedná se o řízení průtoku pomocí PID regulace. Zde je zapotřebí nastavit velikosti parametrů jednotlivých složek `P`, `I`, `D` podle žádaného chování soustavy. Další parametr potřebný v PID režimu je žádaný průtok, který značí požadovaný průtok v soustavě pomocí parametru `W`. Do programu je také nutné zavést obsluhu grafického displeje spolu s dotykovou vrstvou. Toto jsem řešil vytvořením několika pomocných funkcí, které jsou poté volány podle potřeby z hlavního vlákna programu.

První mnou implementovanou funkci, kterou bych zde zmínil je funkce `printValues(char Value)`. Tato funkce přebírá jako parametr znak, který značí proměnnou, jež se má na svém místě na displeji vypsat. Parametrem mohou být znaky `P`, `I` a `D`, které

intuitivně vypíší příslušící velikost konstanty regulátoru. Dále znak *W*, který vypíše proměnnou obsahující velikost žádaného průtoku a znak *A*, který vypíše proměnnou obsahující informaci o aktuálním průtoku. Posledním parametrem je znak *M*, který vypíše procentuální žádanou velikost výkonu v manuálním režimu v rozmezí 0% až 100%. Tato funkce je takto koncipována hlavně z důvodu, abych po změně kteréhokoli parametru mohl vypsat pouze změněný parametr na displej, protože obsluha displeje je časově náročná operace a bylo by značně nepřínosné vypisovat redundantní data.

Další tři pomocné funkce jsou pro obsluhu dotykové vrstvy. První funkcí je *int pressDetect()* vracející celé číslo, které udává aktuální hodnotu 12 bitového A/D převodníku připojeného, ke kterému je připojena dotyková vrstva. Tato hodnota mi poslouží po správné konfiguraci obou vrstev pro snímání polohy dotyku na dotykové vrstvé jako informaci o tom, jestli je na dotykovou vrstvu vyvíjen tlak. Pokud je hodnota na A/D převodníku 4096, tak vím, že tlak je vyvíjen a může se přejit k vyhodnocování polohy dotyku. K tomuto účelu jsem vytvořil dvě funkce vracející velikost A/D převodníku, která určuje polohu dotyku na ose x *unsigned getX()* a na ose y *unsigned getY()*. Obsluha dotykové vrstvy se provádí v nekonečném cyklu hlavního vlákna, v kterém se nachází podmínka tázající se výše zmiňované funkce *pressDetect()* jestli bylo stisknuto na dotykové vrstvě. Pokud ano zavolají se obslužné metody pro vyhodnocení polohy dotyku. Pokud souřadnice náleží oblasti některého z tlačítek na displeji tak se vykoná příslušná akce.

```
void main() {
    ... //inicializace a nulování proměnných
  while(1){ //nekonečný cyklus
    printValues('A'); //výpis aktuálního průtoku

    if (pressDetect()){ //test, jestli byl stisknut dotykový displej
       touchPanelControl(); /*metoda vyhodnocující souřadnice stisku
                            a popřípadné provedení příslušné akce */
    }
    if (run){ //pokud je spuštěn PID režim
      TR0_bit = 1; //spuštění časovače pro výpočet PID algoritmu
    }else if (run == 0){ //pokud je vypnut PID režim
      TR0_bit = 0; //vypnutí časovače pro výpočet PID algoritmu
        /*přepočet procentuální hodnoty výkonu čerpadla
        z velikosti parametru 'M' v manuálním režimu do výstupu PWM */
      CEX0_Compare_Value = 65535-((((double)(*pManualValue))*65535)/100);
    }
    Delay_ms(300);
  }
}
```

Obrázek 20 - Zdrojový kód hlavní metody

Zbytek programu patří samotnému řízení soustavy. Synchronizace výpočtů je zajištěna pomocí dvou 16 bitových časovačů *T0* a *T1*.

První časovač *T0* s periodou 1 sekunda spouští ve své obslužní rutině výpočet diskrétního PID algoritmu. V první části se nalézá výpočet regulační odchylky s následujícím výpočtem PID algoritmu. Regulační odchylka se vypočítává z aktuálního a žádaného průtoku v jednotkách ml/min. Výstup integrační složky je ošetřen proti wind-up efektu (viz kapitola Wind-up efekt) podmínkou, která ověřuje, jestli je integrační složka v mezích, pokud ne tak je nastavena na nejbližší testovanou hranici. Ošetřen je i výstupní výsledek z PID algoritmu, jehož hodnota musí být mezi 0 až 10 000. Pokud hodnota nevyhovuje dané podmínce, tak je upravena na nejbližší testovanou hranici. Vrchní hranice zde omezuje překročení maximálního výkonu čerpadla, které je 10 000 ml/min. Akční veličinu je potřeba převést na akční zásah, kterým je v mém případě napětí na výstupu PWM pinu. Viz kapitola PWM ve spojení s BIG8051.

```
void Timer0InterruptHandler() org IVT_ADDR_ET0{ //s periodou 1s probíhá výpočet PID algoritmu
     ... //nulování registrů časovače a inicializace proměnných
     //pomocné výpočty pro derivační složku
     ad1 = (double)(Kp * Kd * 2) / (TSAMPLING + Kd * ALPHA *2);
     ad2 = (double)(Kd * ALPHA * 2 - TSAMPLING)/ (TSAMPLING + Kd * ALPHA * 2);

     *pError = (*pWFlow) - (*pActualFlow); //výpočet regulační odchylky
     *pPropCompute = (*pError) * Kp; //výpočet proporcionální složky
     //ošetření integrační složky proti dělení nulou a wind-up efektu
     if (Ki != 0 && ((*pIpast)<10000) && ((*pIpast)> -10000)){
         *pIntCompute = (*pIpast) + ((double)(Kp * ((*pError) + (*pEpast)) * TSAMPLING  * 0.5 ) / (Ki));
     }else{
         *pIntCompute = 0;
     }
     *pDerCompute = (*pDpast) * ad2 + ((*pError) - (*pEpast)) * ad1; //výpočet derivační složky
     pid = (*pPropCompute) + (*pIntCompute) + (*pDerCompute); //součet všech tří složek výpočtu
     if(pid > 10000){ //omezení akční veličiny maximálnímu, nebo minimálnímu průtoku (max 10 000 ml/min)
        pid = 10000;
     }
     if(pid < 0){
        pid = 0;
     }
     //přepočet výsledku na 16 bitovou hodnotu pro časovač PWM
     CEX0_Compare_Value =  65535-((((double)(100*(pid))/10000)*65535)/100);
    //uložení hodnot pro příští výpočet
    *pDpast = *pDerCompute;
    *pIpast = *pIntCompute;
    *pEpast = *pError;
  }
```

Obrázek 21 - Zdrojový kód výpočtu PID algoritmu

Nyní nastává otázka převedení akční veličiny na střídu PWM, která se udává v rozmezí 0 (3,3V) až 65535 (0V). Toto jsem řešil pomocí použití dvou trojčlenek. První mi pomohla vyjádřit procentuální velikost výsledku akční veličiny vzhledem k možnému maximálnímu výsledku a to 10 000. Toto procentuální vyjádření poté dosadím do druhé trojčlenky, která mi vypočte toto procentuální zastoupení z hodnoty 65535. Nyní následuje ještě menší úprava výsledku, protože velikost střídy PWM určuje, v jaké hodnotě čítače se signál přepne do logické 1 a je tedy potřeba, ještě výsledek odečíst od hodnoty 65535. Tento přepočet lze vidět v předchozím obrázku - Zdrojový kód výpočtu PID algoritmu.

$$střída\ pwm = 65535 - \frac{\left(\frac{100 * výsledek\ pid}{10000}\right) * 65535}{100} \qquad (10)$$

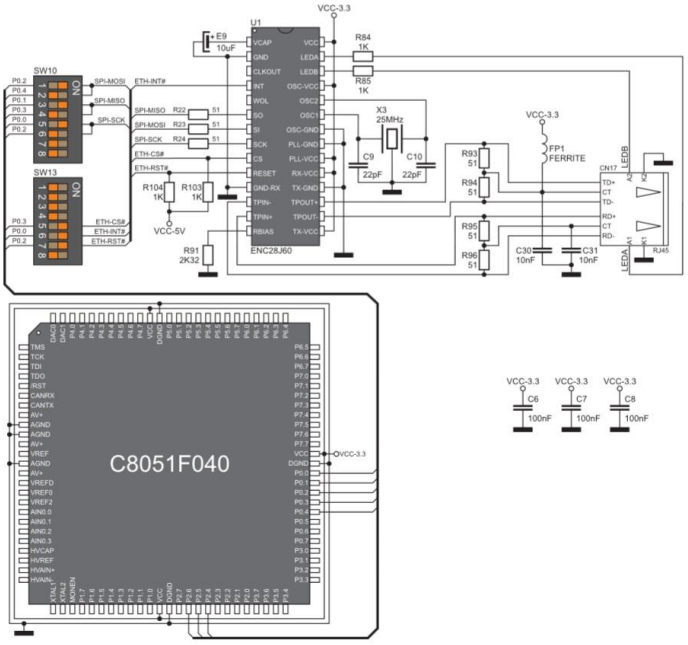
Druhý časovač *T1* slouží pro výpočet aktuálního průtoku v soustavě z informací z průtokoměru s periodou vyhodnocení 1 sekunda. Obslužná rutina po přetečení časovače využívá informaci o nasčítaných náběžných i sestupných hran signálu z průtokoměru, který byl čítán v obslužné rutině externího přerušení. Externí přerušení je vyvoláváno výstupem z obvodu CD 4098. Základní informace, z které algoritmus výpočtu aktuálního průtoku vychází je z vlastnosti průtokoměru, která je, že po vygenerování 10 500 impulzů na výstupu průtokoměru protekl průtokoměrem 1 litr. V rutině přerušení poté každou sekundu vyhodnocuji, kolik přišlo náběžných i sestupných hran. Tento údaj poté vydělím hodnotou 21, čímž získám počet proteklých mililitrů průtokoměrem za sekundu. Soustava pracuje s jednotkou ml/min, proto musím tento výsledek ještě vynásobit 60.

```
void Timer1InterruptHandler() org IVT_ADDR_ET1{ //s periodou 1s probíhá výpočet aktuálního průtoku
      ... //nulování registrů časovače a inicializace proměnných
      *pActualFlow = ((double)(*pSensorTicks) / 21)*60; /*přepočet počtu nasčítaných hran
                                                          ze signálu průtokoměru na aktuální
                                                          průtok v ml/min */
      *pSensorTicks = 0; //nulování počtu nasčítaných hran ze signálu průtokoměru
  }
}
```

Obrázek 22 – Zdrojový kód výpočtu aktuálního průtoku

6.2 Komunikace pomocí ethernetového rozhraní

Posledním bodem mého zadání bakalářské práce je pokusit se zhotovit vzdálenou komunikaci s mikroprocesorem pomocí ethernetového rozhraní, který je integrováno na desce. Bohužel již při realizaci regulátoru spolu s obsluhou dotykového a grafického displeje jsem využil skoro celou paměť procesoru a na implementaci vzdálené komunikace nezbylo paměťové místo. Z tohoto důvodu jsem implementaci realizoval v jiném programu, pouze jako ukázku komunikace použití tohoto rozhraní. Pro používání komunikace skrze ethernet je zapotřebí přepnout přepínače 1, 3 a 5 na přepínači SW10 do polohy ON. Dále je zapotřebí na přepínači SW13 zapnout do polohy ON přepínače 5, 6 a 7 viz obrázek níže.



Obrázek 23 - Schéma zapojení ethernetového modulu. [1]

Komunikace je zprostředkována pomocí sériového rozhraní SPI (Serial Peripheral Interface). Rozhraní poskytuje plně duplexní přenos, tudíž je možno zároveň přijímat a vysílat data. Knihovna mikroC nabízí řadu funkcí pro ulehčení komunikace pomocí ethernetového rozhraní. V první řadě je to funkce `SPI_Ethernet_Init(unsigned char *mac, unsigned char *ip, unsigned char fullDuplex)`, která inicializuje modul SPI pro komunikaci pomocí ethernetu. Jako parametry přijímá MAC adresu, IP adresu a určení způsobu komunikace (full-duplex, nebo half-duplex) na kterém bude komunikace pracovat.

Další důležitou funkcí je funkce `SPI_Ethernet_doPacket()`, která vyhodnocuje příchozí pakety. Její princip spočívá ve zpracování jednotlivých paketů ze zásobníku, v kterém jsou uloženy příchozí pakety. U každého paketu zjistí, jakým typem komunikace byl paket přijat (TCP, nebo UDP). Podle toho se zavolá obslužná metoda `SPI_Ethernet_UserTCP(…)` pro zpracování TCP paketu, nebo `SPI_Ethernet_UserUDP(…)` pro zpracování UDP datagramu. Obě tyto metody mají vstupní parametry IP adresu odesílatele, číslo portu, přes který byl paket (datagram) odeslán a délku zprávy v bytech, které jsou předány z právě zpracovávaného paketu, nebo datagramu. Funkce `SPI_Ethernet_doPacket()` má být volána co nejčastěji, proto je umístěna v hlavním programu v nekonečném cyklu.

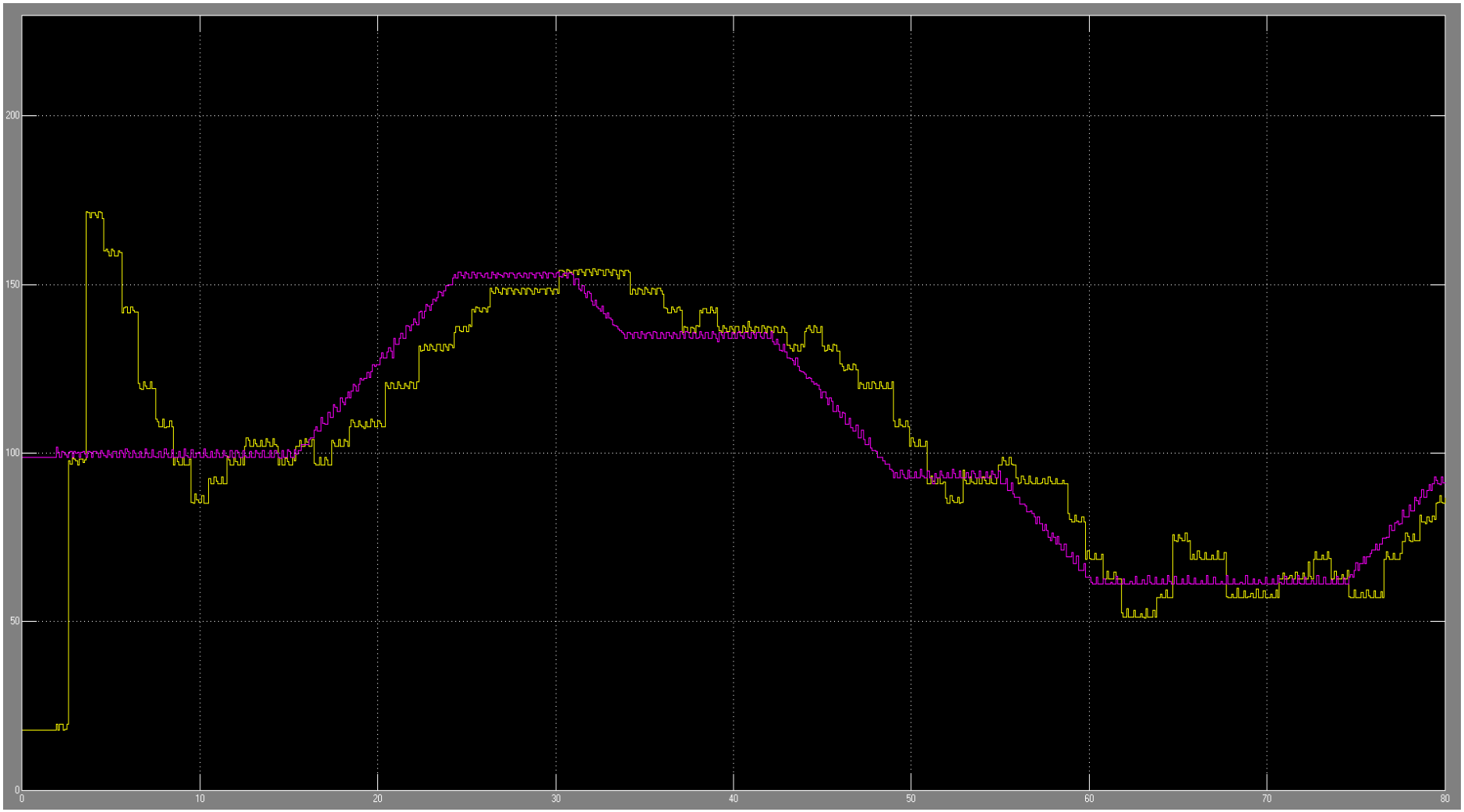
V mé práci využívám komunikaci pomocí TCP a parametry předávám skrze metodu GET v url adrese. Na základě těchto parametrů vyhodnocuji příslušnou akci. Pomocí webového rozhraní lze nastavit všechny parametry, stejně jako za využití dotykového displeje. Žádost o přidání jakékoli hodnoty začínají v GET požadavku písmenem `p` (plus), poté následuje bez mezer, kterého parametru se to týká (P, I, D, M, W). Zmenšení jakékoli hodnoty pracuje na stejném principu akorát místo prvního znaku `p` (plus) se nachází znak `m` (mínus). Poslední možnou obsluhou je přepnutí režimu regulátoru mezi řízení průtoku pomocí PID regulace, nebo pomocí manuálního řízení. Toto přepnutí se provede po odeslání GET požadavku s parametrem `r` (režim). Příklad požadavku o přidání žádaného průtoku může být ze strany klienta realizován odkazem `http://192.168.20.60/?pW`.

Obrázek 24 - Webové rozhraní pro komunikaci s regulátorem

6.3 Měření regulační soustavy

Měření soustavy jsem prováděl za pomoci programu Matlab a nástroje Simulink. V laboratoři je k dispozici měřící deska, která je připojena k počítači a umožňuje přivést hodnotu napětí pomocí analogových vstupů do prostředí Simulink. Bylo tedy zapotřebí z vývojového kitu vyvést napěťovou úroveň, která by reprezentovala aktuální a žádaný průtok regulátoru. Toto jsem vyřešil pomocí dvou D/A převodníků, jejichž rozsah je od 0V do 2,48V. Pro účely měření jsem převedl tuto hodnotu napětí v prostředí Simulink na hodnotu průtoku 0 ml/min až 250 ml/s vynásobením hodnoty napětí vhodnou konstantou.

Následně jsem provedl měření chování soustavy s různými velikosti konstant proporcionální a integrační složky. Zastoupení derivační složky regulátoru jsem vypustil, protože zapojení všech potřebných obvodů jsem realizoval skrze nepájivé kontaktní pole, které bohužel po zapojeních všech potřebných obvodů vykazovalo silný nežádoucí šum, na který reagovala derivační složka příliš mnoho a způsobila nestálost soustavy. Při měření jsem postupoval nejdříve nastavením nejnižší možné hodnoty proporcionální složky a k ní postupně přidával zastoupení integrační složky. Je zapotřebí zmínit, že výpočet integrační složky je realizován tak, že čím je vyšší hodnota integrační konstanty K_i, tím je pomalejší reakce integrační složky. V grafu níže je žádaný průtok reprezentován fialovou křivkou a aktuální průtok reprezentován křivkou žlutou. Měření začíná s minimálním aktuálním průtokem a žádaným průtokem 100 ml/min. Následně je žádaný průtok upravován tak, aby vystihl chování soustavy. Osa x značí uplynulý čas v sekundách a osa y průtok v rozmezí od 0 ml/min až po 250 ml/min. Na obrázku níže je vybráno nejideálnější nastavení konstant regulátoru. Toto nastavení bylo zvoleno po experimentech s různými velikosti konstant proporcionální a integrační složky.



Obrázek 25 - Průběh chování regulátoru (K_p = 0.002, K_i = 0.009)

7 Závěr

Všechny stanovené cíle v úvodní části byly splněny a realizovány. Jediný problém nastal s realizací vzdálené komunikace s procesorem pomocí ethernetového rozhraní, na kterou nezbyl paměťový prostor v procesoru. Z tohoto důvodu byla vzdálená komunikace realizována jako ukázka v jiném samostatném programu, ale se všemi parametry a požadavky, které jsou potřeba pro řízení této regulační soustavy.

V průběhu realizace jsem se setkal s nedostatkem vývojového kitu BIG8051, kterým bylo reakce externího přerušení pouze na sestupné hrany signálu. Tato vlastnost byla nežádoucí při vyhodnocování počtu pulzů z průtokoměru. Průtokoměr generuje signál o nízké frekvenci a čítání pouze sestupných hran by zmenšilo počet informaci o aktuálním průtoku na polovinu a tím zvýšilo nepřesnost měření. Tato nežádoucí vlastnost byla úspěšně eliminována obvodem CD 4098, který je popsán v této dokumentaci.

Jelikož můj obor informační technologie není zaměřen na elektrotechniku, nejsou v něm tedy tolik probrány základy a principy realizace elektronických obvodů, neměl jsem tedy žádnou zkušenost se zapojováním elektronických součástek a ostatních využitých obvodu v této práci. Práce byla tedy pro mne přínosná mimo jiné i ve zdokonalení se v této oblasti elektroniky.

Jako další záměr práce bych se pokusil vývojový kit rozšířit o paměťový prostor pomocí připojení SD karty a implementoval bych ethernetové komunikační rozhraní v programu regulátoru. Poté se pokusil aplikovat regulátor na konkrétní aplikaci v praxi.

Použitá literatura

[1] MikroElektronika - BIG8051 Development System - Silabs C8051 Board (programmer, Ethernet, DAC, CAN) [online]. 2014 [cit. 2015-05-17]. Dostupné z: http://www.mikroe.com/big8051/

[2] *Přerušení procesoru* [online]. [cit. 2015-05-17]. Dostupné z: http://www.umel.feec.vutbr.cz/~adamek/komp/data/3.htm

[3] *Pulzně šířková modulace* [online]. [cit. 2015-05-17]. Dostupné z: http://www.dhservis.cz/psm.htm

[4] *Datový list čerpadla* [online]. [cit. 2015-05-17]. Dostupné z: http://www.produktinfo.conrad.com/datenblaetter/525000-549999/539090-an-01-cs-Tauchpumpe_Typ04.pdf

[5] *MOSFET tranzistory* [online]. [cit. 2015-05-17]. Dostupné z: http://lucy.troja.mff.cuni.cz/~tichy/elektross/soucastky/dva_prechody/uni_tranzistor.html

[6] STENGL, Jens Peer a Jenö TIHANYI. *Výkonové tranzistory MOSFET*. 1. čes. vyd. Praha: BEN - technická literatura, 1999, 191 s. ISBN 80-86056-54-6.

[7] *Datový list průtokoměru* [online]. [cit. 2015-05-17]. Dostupné z: http://www.produktinfo.conrad.com/datenblaetter/150000-174999/155374-da-01-ml-Durchflussmesser_FCH_M_de_en.pdf

[8] *Datový list obvodu CD 4098* [online]. [cit. 2015-05-17]. Dostupné z: http://www.classiccmp.org/rtellason/chipdata/cd4098.pdf

[9] *Obrázek modulárního regulátoru* [online]. [cit. 2015-05-17]. Dostupné z: http://static.abc.cz/generate-images/firma_nabidky/velka/31/18281.jpg

[10] *Obrázek kompaktního regulátoru* [online]. [cit. 2015-05-17]. Dostupné z: http://www.autonics.se/prod/basi/autonics_products_sv.nsf/0/582E6F855DCF76A7C125763A004916CA/$file/1269245826_admin_0.jpg

[11] BALÁTĚ, Jaroslav. *Automatické řízení*. 2., přeprac. vyd. Praha: BEN - technická literatura, 2004, 663 s. ISBN 80-7300-148-9.

[12] DOC. DR. MGR. ING. HLAVA, Jaroslav. *Computer control* [online]. In: . [cit. 2015-05-17]. Dostupné z: https://elearning.tul.cz/mod/resource/view.php?id=38331

[13] *Navrhování a realizace regulátorů* [online]. [cit. 2015-05-17]. Dostupné z: http://www.person.vsb.cz/archivcd/FEI/NRR/Navrhovani%20a%20realizace%20regulatoru.pdf

Přílohy

A Obsah přiloženého CD

Přiložené CD obsahuje v kořenovém adresáři tuto práci v elektronické podobě ve formátu pdf. Dále CD obsahuje adresář *výsledky měření*, v kterém jsou uloženy všechny naměřené grafy chování soustavy při různých nastavení regulátoru. Jednotlivé obrázky jsou pojmenovány ve formátu *PxIx.png*, kde písmeno *x* značí zastoupení jednotlivých složek regulátoru při měření. Například obrázek *P2I8.png* obsahuje graf měření chování regulátoru při nastavení proporcionální složky na hodnotu 2 a integrační složky na hodnotu 8.

Druhým adresářem je adresář *zdrojový kód*, který obsahuje zdrojový kód programu napsaný v jazyce C a využití knihovny MikroC.

Posledním adresářem je adresář videozáznam. Tento adresář obsahuje video záznam ve formátu mp4, který dokumentuje chování regulátoru, natočený v laboratoři TK4.