STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST
Obor: 10. elektrotechnika, elektronika a telekomunikace

Autonomní dvouprocesorové vozidlo s naváděcím systémem a analýzou obrazu

Autoři projektu

Adam Ligocki
Chotěbuz K oboře 378
a.ligocki@seznam.cz

Zbigniew Opioł
Stonava 1081
opiol.zbigniew@gmail.com

Škola

SPŠE Havířov

Odborný konzultant

Ing. Ladislav Opioł

Prohlášení

Prohlašujeme, že jsme tuto práci vypracovali samostatně, použili jsme pouze podklady (literaturu, SW, atd.) citované v práci a uvedené v přiloženém seznamu a postup při zpracování práce je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorských zákonů) v platném znění.

V Havířově dne ……

Podpis: …….

Podpis: …….

Poděkování

Děkujeme panu Ing. Ladislavu Opiołovi, za pomoc při vedení projektu a za jeho cenné rady při aplikaci naších znalostí na tento projekt.

Děkujeme paní Bc. Libuši Hrejsemnou, za pomoc při vytváření desek plošných spojů.

Anotace

Projekt se zabývá vytvořením autonomního vozidla řízeného dvěma kooperujícími procesory, z nichž každý má své předem dané úlohy. Vedle dvou procesorů (ARM a AVR), je vozidlo vybaveno ultrazvukovými senzory vzdálenosti, elektronickým kompasem, GPS příjmačem a kamerou. Data získaná těmito zařízeními jsou dále zpracována a zaslána do počítače skrze sériový kanál tvořený bluetooth modulem, nebo po TCP protokolu pomocí wi-fi modulu. V počítači je vytvořeno grafické prostředí pro snadnou vizualizaci a čtení dat. Nejdůležitější přidanou hodnotou je rozpoznání objektů nasnímaného obrazu.

Klíčová slova

Autonomní robot, dvouprocesorový systém, ARM, AVR, navigační senzory, GPS, kamera, zpracování obrazu, bluetooth, wi-fi, GUI aplikace.

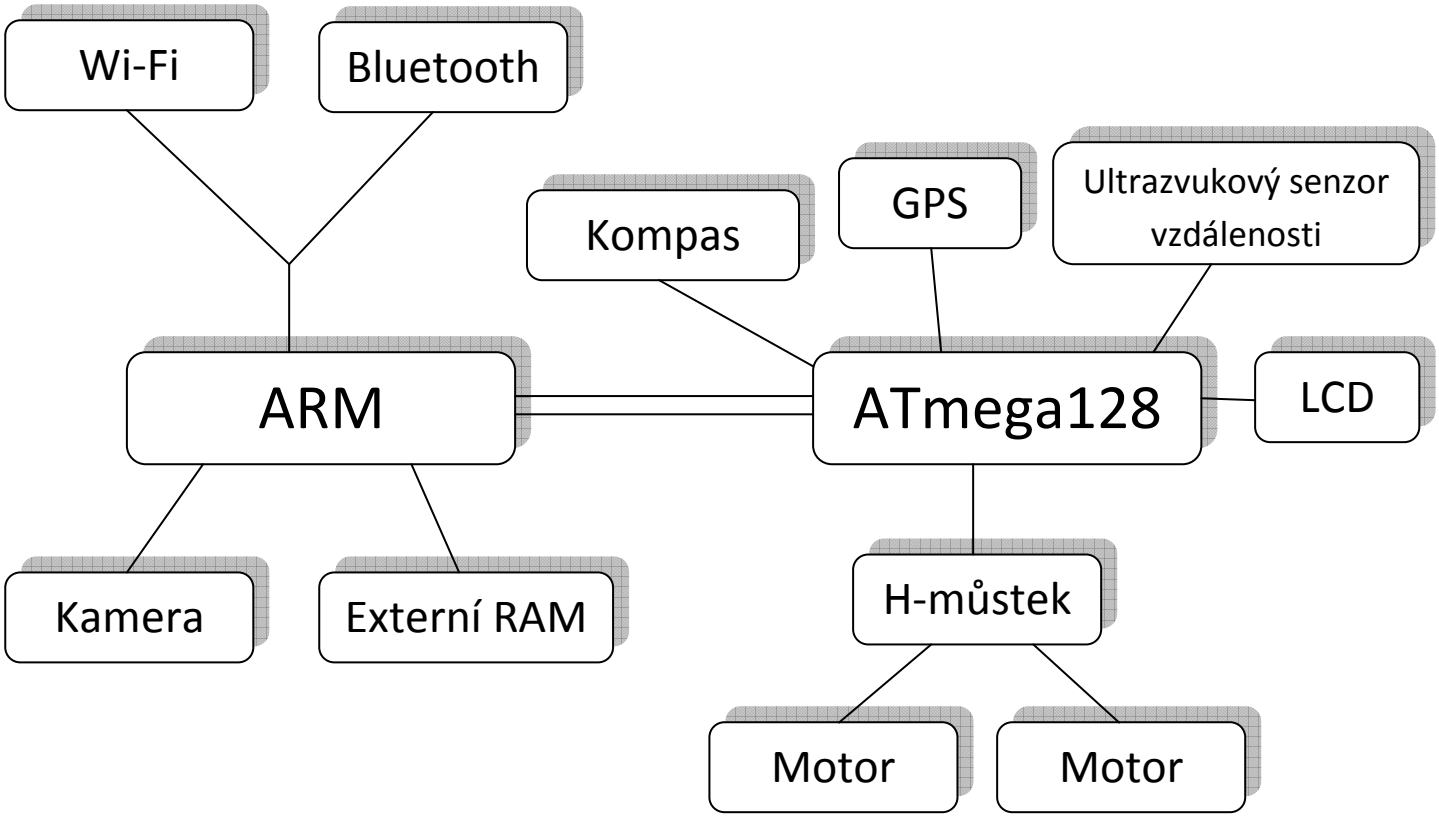
OBSAH

Úvod do problematiky

Základní myšlenkou celého projektu je vytvoření robota schopného nezávislého pohybu v prostoru. Pro tento účel je potřeba vytvořit schéma obsahující několik druhů senzoru. Následně je nutno data generována těmito senzory spolehlivě zpracovat a na základě výsledků měření jednoznačně určit další chování robota.

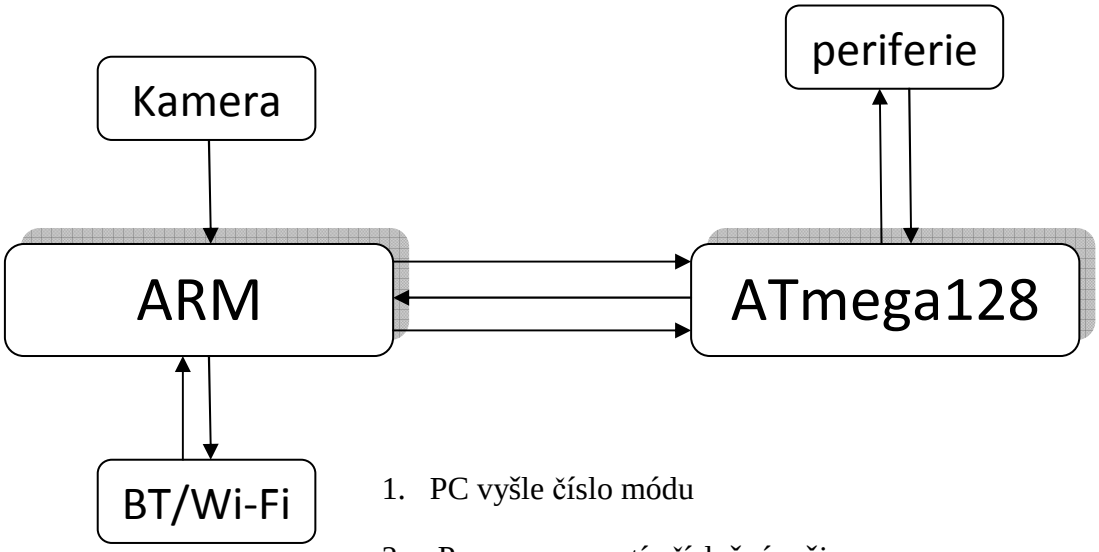
Hlavním prvkem celé práce je analýza obrazu prováděná na procesoru ARM. Základním stavebním kamenem práce s obrazem je detekce hran, od které lze dále odvozovat další postupy analýzy a zpracování obrazu. Samotná detekce hran je matematicky náročná operace, která si vyžaduje většinu strojového času i RAM paměti procesoru.

Všechny prvky řízení robota jsou podrobně popsány v kapitolách Hardware a Software.



Řízení robota

Mozkem celého systému je procesor ARM. Program nahraný v tomto mikrokontrolélu obsahuje několik níže popsaných algoritmů. Vedle této činnosti, procesor ARM řídí kameru, a zpracovává její výstup pomocí Cannyho algoritmu , a detekuje jemu známé značky a jejich vzdálenost.Procesor AVR pracuje v systému jako řadič, který obsluhuje všechny senzory a další periférie. Získáva data z ultrazvukovýxh senzorů vzdálenosti, z elektronického kompasu a GPS příjmače. Ovládá h-můstky, a LCD display.



1. PC vyšle číslo módu
2. Procesory spustí příslušné režimy
3. AVR provede měření
4. AVR zašle naměřená data do ARMu
5. ARM načte obraz a provede analýzu přijatých dat
6. Atmega provede příslušný pohyb vozidla
7. ARM odešle data do PC, která se zobrazí v GUI

Celkem lze hovořit o 3 algoritmech pro pohyb vozidla v prostoru, nepočítáme-li manuální řízení robota.

Algoritmus pro autonomní pohyb v prostoru

V tomto režimu robot využívá především ultrazvukových senzorů vzdálenosti, které mu poskytují relativně přesné informace o výskytu překážek v jeho okolí. Přesto však každé měření probíhá 3x, z důvodu existence možnosti vzniku chyby při měření (viz kapitola ultrazvuková čidla). Po fázi získání dat následuje fáze určení dalšího směru pohybu. V rámci této činnosti robot rozhoduje, zda je dráha volná, a může pokračovat po své aktuální trajektorií, nebo zda má provést odklon od své současné dráhy pohybu, či případně vycouvat z velmi úzké chodby, kde otočka by mohla působit potíže v podobě kolize zadní strany vozu která nedisponuje žádnou ochranou proti této situaci. Při vyhodnocování dalších pohybových úkonů, se běh programu pohybuje ve 4 logických blocích. Díky těmto blokům, algoritmus robota bere v úvahu také svůj poslední pohyb. Můžeme tedy hovořit o prvcích Mealyho automatu (stroj, jehož výstup (chování robota) je odvozeno od okamžitého vstupu (data ze senzorů), a posledního vnitřního stavu (poslední směr pohybu)), což zabraňuje stroji dostat se smyčky neustále se opakujícího sledu chaotických činností.Při vyhýbání se překážkám do stran robot vždy bere v úvahu data také z bočních senzorů, a také do vyhodnocení vkládá vygenerované pseudonáhodné číslo. Vozidlo se tak pohybuje v prostoru „inteligentně“, a vždy volí perspektivnější cestu. Tuto část algoritmu jsem zařadil do programu, protože se domnívám, že neustálé vyhýbání se překážkám pouze do leva, nebo pouze do prava evokuje pocit, „jednoduchosti provedení“, a nedostatečného řešení problematiky.

Příklad: pokud robot právě zatáčel doprava nemůže vyhodnotit další pohyb doleva. Pokud stroj couval, nemůže být dalším pohybem jízda to předu.

Algoritmus pro pohyb pomocí GPS a udržování azimutu

V tomto režimu jsou pochopitelně nejdůležitějšími senzory GPS příjmač a elektronický kompas . Nezbytné pro pohyb a správnou funkci jsou také ultrazvuková čidla, a grafické uživatelské rozhraní v počítači připojeném pomocí bluetooth modulu.Nejprve stroj načte data z GPS přijmače tak, že přečte příchozí datový řetězec a vybere z něj informace o GTM čase,nadmořské výšce, počtu satelitů se kterými je ve spojení, a pochopitelně údaje o zeměpisné výšce a šířce. Tento řetězec je přijat v procesoru ATmega128. Data o zeměpisné poloze přicházejí ve formátu stupňů, minut a desetinných míst minut. Tyto data je nutno převést pouze na stupně, aby bylo možno provádět další výpočty. Tuto operaci 8bitobý procesor ATmega128 není schopen provést. Proto vygeneruje nový řetězec obsahující pouze potřebná data, a přepošle je do počítače, respektivě do procesoru ARM. Tam jsou data poddána potřebným úpravám, a dále jsou načteny údaje z uživatelského rozhraní, která obsahují souřadnice cílového bodu.Nyní je vypočítán vektor mezi dvěma body, a následně je tento vektor vztažen na nulový azimut a provede se vypočtení azimutu pro další pohyb stroje.Nyní jsou tyto data přeposlána spět do procesoru ATmega128. Následuje fáze udržení

vypočítaného azimutu. Pro tento účel je vypracován pohybový algoritmus, který udrží vozidlo pohybovat se v určeném směru +/- 5°. Robot neustále snímá data z elektronického kompasu, a dle nich koriguje svůj směr pohybu tak aby co nejlépe odpovídal vypočtenému směru cílového bodu.Vyhodnotí-li stroj že se odklonil od stanoveného kurzu, zvýši rychlost 1 pásu, a docíli tak návratu na požadovaný směr. Dojde li k výraznému odklonu, v důsledku najetí na nerovnost, stroj se zastaví a provede korekci azimutu na místě, a dále opět pokračuje pohybem s aplikací mírné korekce za běhu. Při tomto režimu je také ošetřen případ čelního kontaktu s překážkou. Robot při svém pohybu neustále vyhodnocuje prostor před sebou, a v případě nalezení překážky se zastaví. Do budoucna je možné napsat algoritmus pro obětí překážky, avšak v současné době rozmístění senzorů této variantě nevyhovuje. Bylo by možné překážku objet zpusobem, že by robot provedl odklon od azimutu o 90°, popojel o 30cm, a opět pokračoval dále na stanovený kurz, ale v případě že by zde narazil na zábranu, bylo by nutné vytvořit sofistikovanější chování, aby nedošlo k nežádoucímu chování storje .

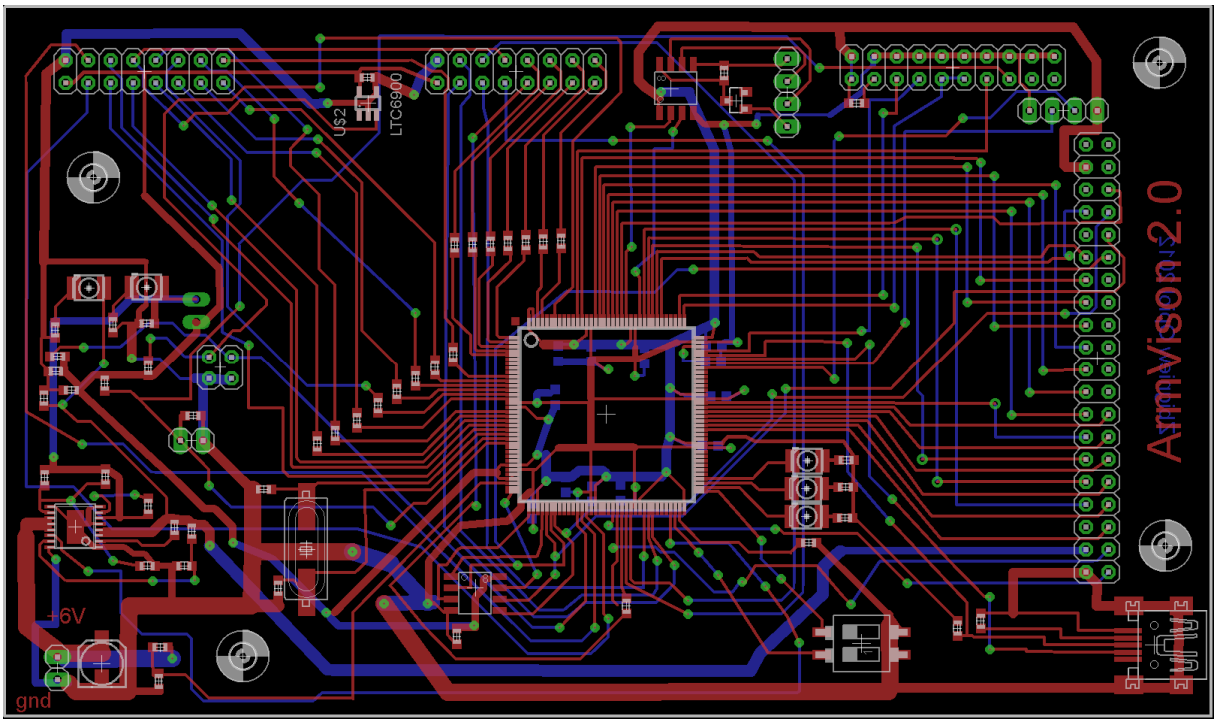
Algoritmus pro pohyb pomocí značek

ARM načte data z kamery a převede je do stupňů šedi. Následně spustí modifikovaný Cannyho hranový detektor, tím získá hrany a zbytek potlačí. Na hrany následně aplikujeme algoritmus na rozpoznání rohů. Dále vybereme 4 rohy, které splňují podmínky námi určeného dvojitého čtverce. Tento čtverec označuje pozici značky. Samotné vyhodnocení obsahu značky se provádí pomocí bitové masky vytvořené podle skutečné značky. V tomto módu autíčko jede pořád rovně, když detekuje značku provede činnost přiřazenou značce.

1. Hardware

1.1 ARM

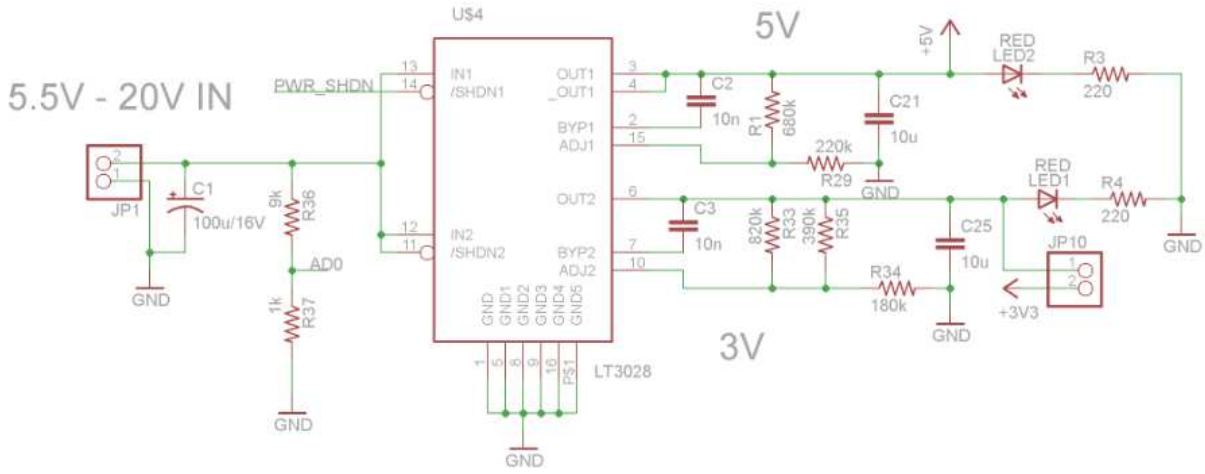
Jako řídicí jednotka byla navržena a sestrojena deska plošného spoje z mikroprocesorem ARM a sériovou paměti FRAM. Procesor Zpracováva data z kamery a sekundarního procesoru,který funguje jako řadič. Komunikace s PC probíhá skrz WiFi nebo bluetooth. Deska byla zhotovena firmou MATRON.



ob 1.1.1 – DPS procesoru arm

Napájení

Pro napájení desky je použít dvojitý nastavitelný měníč firmy Linear Technology. Měnič disponuje teplotní i proudovou ochranou a také ochranou proti přepólování . Výstupní napětí je nastavitelné v rozmezí 1.8 – 20V. Měnič má velmi nízký úbytek napětí, jmenovitě 300 mV.



ob 1.1.2 - napájení

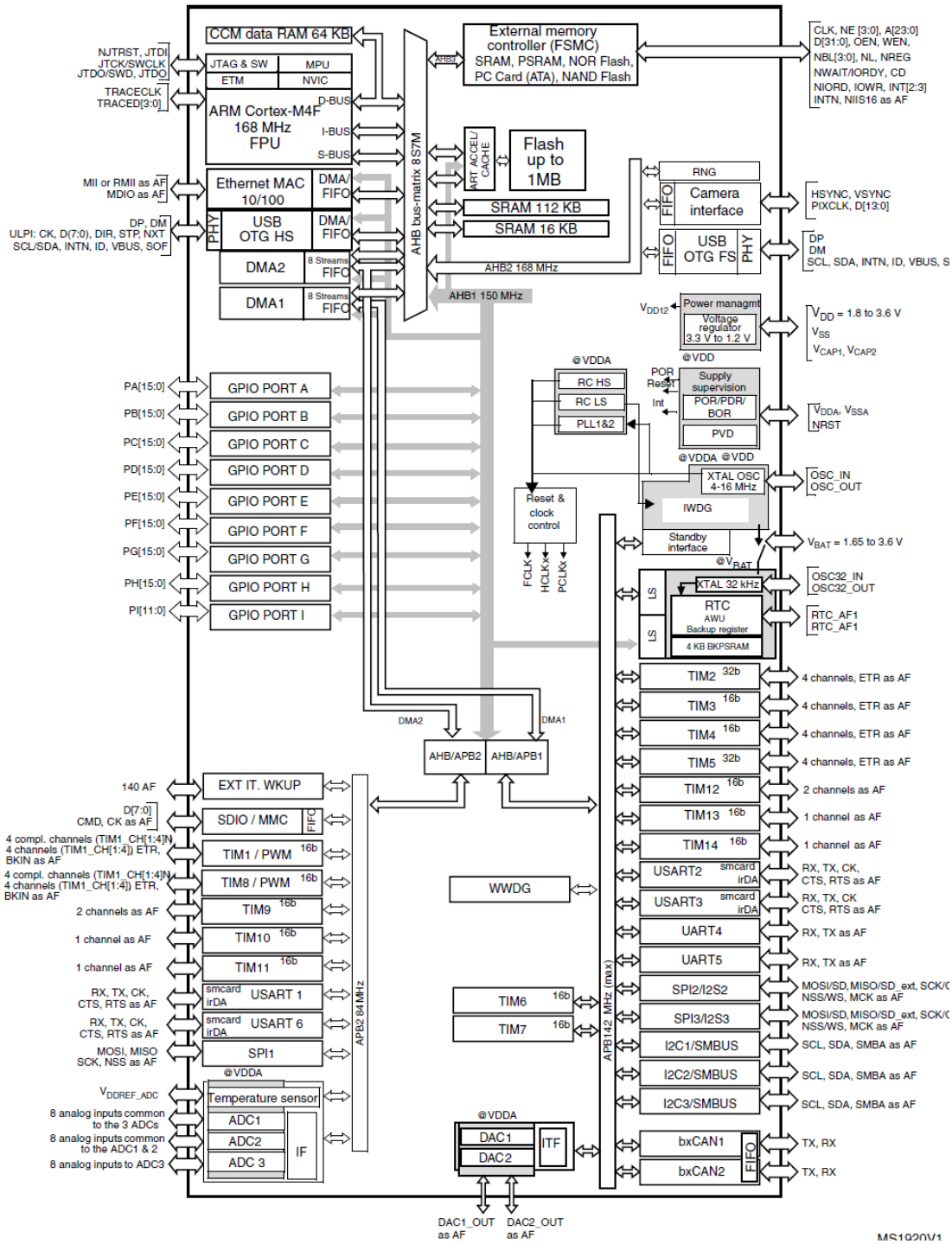
Na vstup je přivedeno hlavní napájeni. Pro sledevání jeho úrovně je přes odporový dělič připojen vstup na AD převodník. Každá větev stabilizátoru je jinak proudově zatížitelná. První větev má maximální výstupní proud 500 mA a je určená pro napájení periferii. Druhá větev disponuje maximálním výstupním proudem 100mA a je použita pro napájení všech obvodů na desce. Pro testovací účely byla vybavena odpojovacím konektorem, aby bylo možno provést měření výstupního napětí bez potencialního ohrožení komponent na desce. Větve jsou dále vybaveny signalizačními LED a větev 2 je možno zapínat/vypínat pomocí mikroprocesoru.

Mikroprocesor

Jelikož zpracování obrazu je matematicky náročné bylo nutné použit výkonný procesor, s velkou RAM ale zároveň v pouzdru LQFP. Jako nejvhodnější volbu se ukázalo použít ARM s integrovanou FPU. Tyto podmínky nejlépe splňuje nová řada procesorů Cortex M4F firmy ST, STM32F4. Řada byla uvedena na trh koncem roku 2011 a výkonem převyšuje konkurenci.

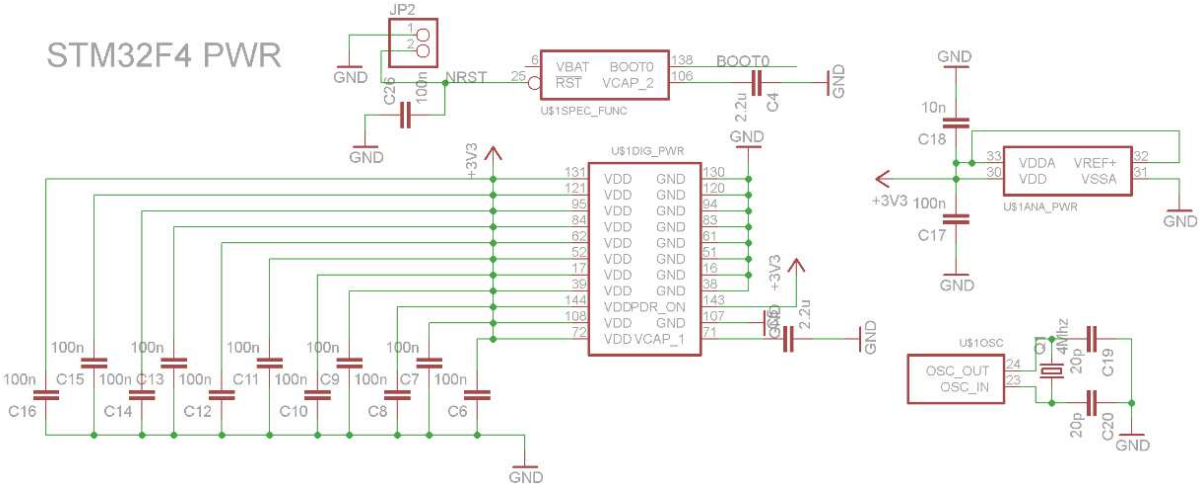
Nejdůležitější vlastnosti:

- 168 Mhz
- 210 DMIPS
- 192 Kbyte RAM
- 1 Mbyte Flash
- FPU
- DSP
- SIMD
- 6x UART
- 3x SPI



ob 1.1.3 – vnitřní schéma procesoru

Kvůli velkému výkonu si procesesor žádá kvalitní napájení. Proto je nutno použit 12 napájecích větví. Procesor také umožňuje při výpadku proudu automaticky přejít na záložní zdroj energie, tato funkce ale není použita. Procesor má integrovaný 16 MHz interní krystal, který je možné vyřadit a použit místo něho externí. V projektu je použit externí krystal 8MHz. Pomocí BOOT určujeme odkud proběhne bootování.Vref slouží jako referenční napětí pro AD převodník.



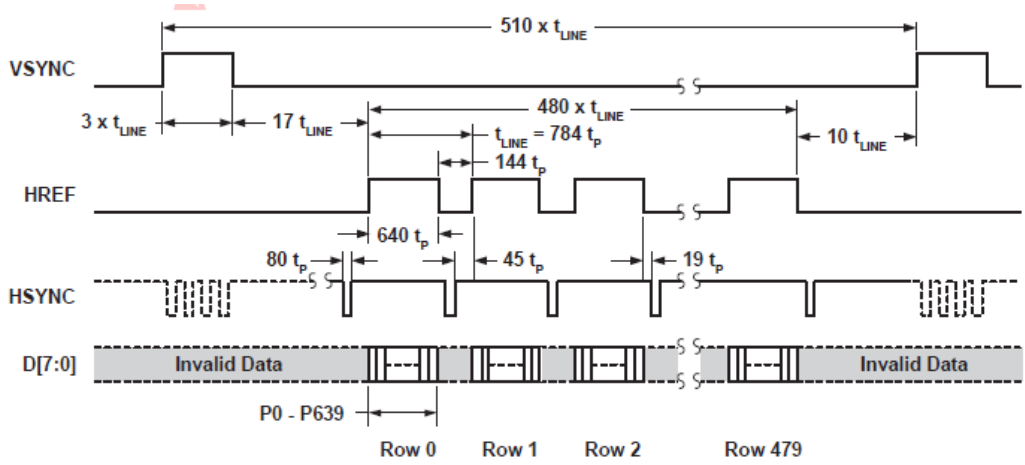
Ob 1.1.3 – napájení procesoru

RAM

Procesor disponuje poměrně velkou pamětí RAM, ale pro zpracování obrazu požadované velikosti to stále nestačí. Bylo proto zapotřebí použit externí RAM. Jelikož byl kladen důraz na malé rozměry desky byla zvolena sériová paměť. Pro komunikaci je použito SPI a jako RAM je použita FRAM firmy RAMTRON. Princip zápisu na FRAM a na FLASH je stejný, jediný rozdíl je v tom, že FRAM pracuje na ferroelektrickém principu, tudiž mezi dílčima zápisy není třeba prodlevy, což eliminuje potřebu bufferu a umožňuje zapisovat s maximální frekvence SPI, tedy 40 MHz. Další výhodou je obrovský počet zápisů (10^{14}) což při frekvenci 40 MHz odpovídá délce 20 let.

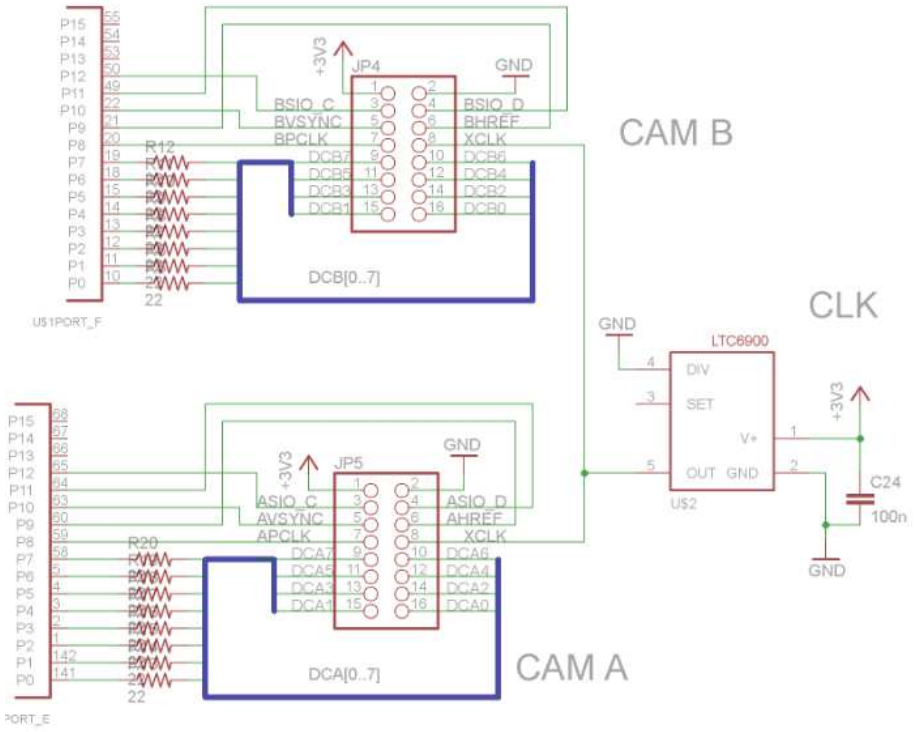
Kamera

Kamera byla zvolena od čínské firmy OmniVision zvláště pro svou nízkou cenu a dostupnost. Konkrétně model ov7670 s podporou VGA, QVGA, CIF, QCIF. Pro naše potřeby je použito mírně upravené QVGA, v podobě 320x200. Kamera se řídí po rozhraní SCCB které je HW kompatibilní s I2C. Data se přenášejí po 8-bitové sběrnici. Modul potřebuje externí hodinový signál 10-50MHz. Kamera umožňuje použit RGB nebo YUV. Jelikož se obraz dále zpracováva je použito YUV a uchovává se pouze jasová složka. Jako zdroj signálu pro modul je použit oscilátor LTC6905, který je pospsán níže. V RGB módu lzte volit mezi různými formáty dat 4:4:4, 5:5:5 a 5:6:5. Data jsou posílany ve dvou bytech. V případě YUV jsou data posílany ve formátu 4:2:2.



ob 1.1.4 – časový diagram čtení dat z kamery

Data jsou doprovázena standardními pomocnými signálý, jako VSYNC který označuje začátek snímku, HREF označuje začátek řádku(HSYNC má stejný vodič jako HREF, volí se v registru) a PCK označující taktovací sígnál jednotlivých bytů.



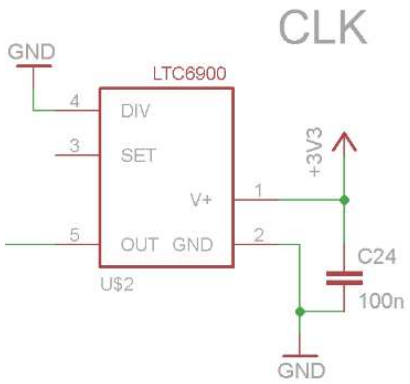
ob 1.1.5 – konektor pro kameru

Deska je vybavená dvěma pozicemi pro kamery a to z důvodu možného přidaní druhé kamery a provádění prostorového snímaní.

Konfigurace kamery je popsáná v sekci Software.

Oscilátor

Jako hodinový signál pro kameru slouží oscilátor firmy Linear technology LTC6905.



Jedná se o oscilátor vyráběný ve více variantách, každý s jinou frekvencí. Oscilátor má integrovanou děličku x1, x2 a x4. Dělicí poměr se určuje pomocí pinu DIV. Pro dělicí poměr 1, DIV je připojen do ‘1’, pro dělící poměr 4, DIV je připojen do ‘0’. Pro dělící poměr 2, je DIV ponechán plovoucí. Pro naši aplikaci je použit oscillator 80Mhz s děličkou 4x na výsledných 20 Mhz.

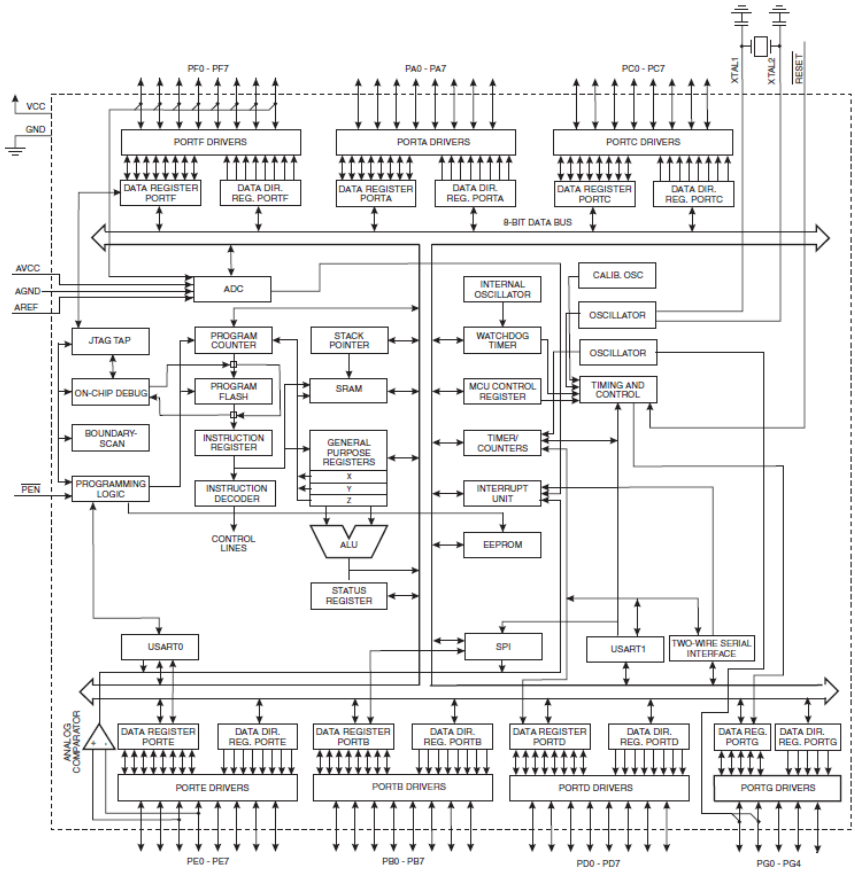
ob 1.1.6 - oscilátor

1.2 ATmega128

ATmega128 je MBD procesor of firmy Atmel. Je založen na 8 bitové architektuře AVR.

Základní vlastnosti :

- Jádro AVR, 16MIPS
- 128 kB flash
- 4 kB EEPROM
- 4 kB SRAM
- až 64 kB externí paměti
- SPI a JTAG
- 4 časovače
- 8 ADC převodníků
- TWI sběrnice
- 2 sériové kanály
- globální Pull-up

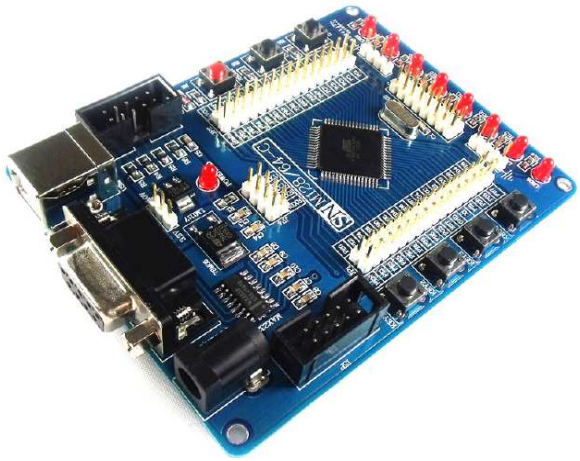


ob 1.2.1 – blokové schéma ATmega128

Jedná se o nejvýkonnější model architektury AVR, avšak přesto procesor má nesrovnatelně nižší výpočetní výkon, oproti procesoru ARM. V tomto projektu byl zvolen právě AVR, z důvodu efektivnosti jeho aplikace Všechny využité funkce procesoru jsou popsány v dalších kapitolách přímo při konkrétní aplikaci.

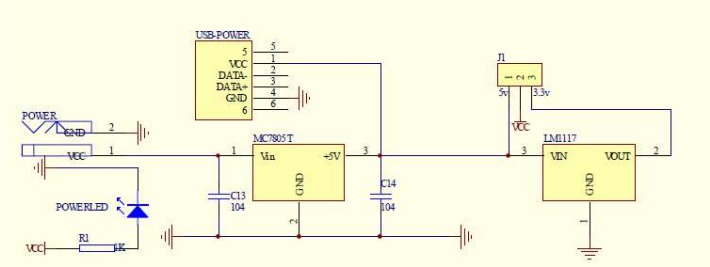
Na robotu je upevněn vývojový kit od firmy Sintech studio. Jedná se o výborně zpracovanou desku ,na které jsou realizovány vývody všech pinů, respektivě všech funkcí procesoru. Původně osazený krystal 7.3728MHz, by nahrazen krystalem 16MHz.

Deska obsahuje procesor, konektor pro sériový kanál, USB konektor, napájecí zdířku, obvody 78M05, LM1117, MAX232, vývod ISP, vývod JTAG 4 tlačítka spínaná proti zemi, 2 interrupt tlačítka, 8 led, a resetovací tlačítko.

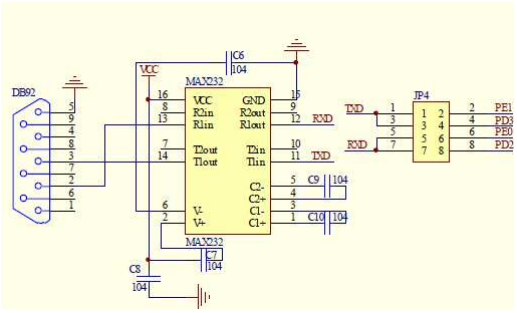


ob 1.2.2 – vývojový kit AVR mega128/64 mini

Napájení procesoru je realizováno obvodem 78M05, který vytváří stabilizovaných 5V. Toto napětí je také použito pro napájení většiny periferií. Napětí 3V3 je stabilizováno pomocí LM1117.



ob 1.2.3 – napájení procesoru

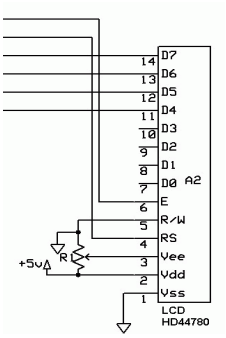


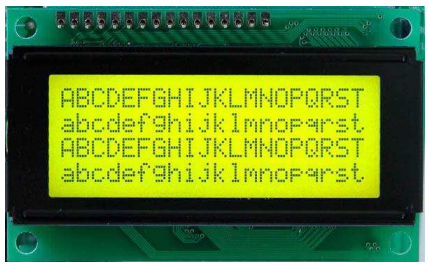
ob 1.2.4 – zapojení MAX232

Sériový kanál procesoru je přiveden na obvod MAX232, a díky nábojovým pumpám je logická úroveň převedena z TTL na RS232. Díky jumperum je možné na konektor přivést libovolný kanál procesoru.

LCD

Pro vizualizaci činnosti procesoru byl použit alfanumerický display s řadičem HD 44780. Je použit model s rozměry 20 * 4 znaků. Není plně kompatibilní s ASCII tabulkou, ale podporuje všechna písmenka bez diakritiky.



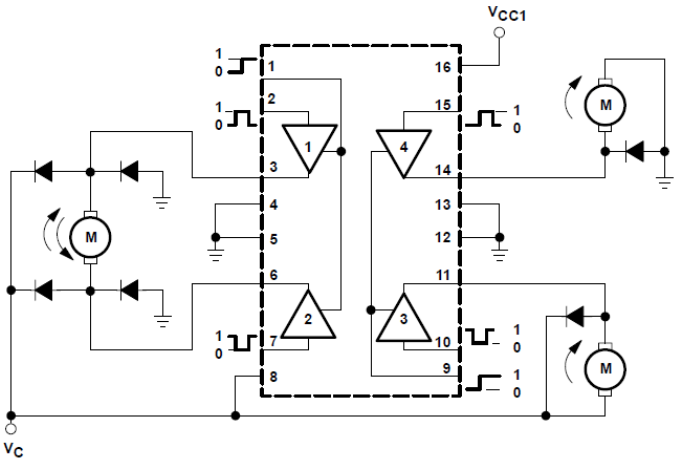


ob 1.2.5 – LCD display 20*4 znaků

Display je připojen k procesoru 4 datovými a 3 řidicími vodiči.Dále je display vybaven možností externího podsvícení, avšak ta nebyla využita z důvodu přetížení obvodu 78M05.

H-můstek

Pro řízení motorů poháněných robota byly použity 2 integrované obvody L293 od firmy Texas Instruments. Jedná se o klasický H-můstek sestavený ze výkonových zesilovačů.



ob 1.2.6 – integrovaný obvod L293

Motory byly zapojeny na H-můstek v obousměrném módu. Každý motor připojen na 2 paralelně spínané kanály, aby se výkon rozložil na celý obvod, ačkoliv výrobce udává, že každý kanál může vést proud až 1A, a při překročení kritické teploty dojde k jeho automatickému odpojení.

Ultrazvukový senzor vzdálenosti

Orientace robota v prostoru je naprosto klíčová schopnost v této aplikaci. Robot díky této schopnosti se dokáže velmi dobře vyhýbat překážce jakýchkoliv rozměrů. Ultrazvukový senzor emituje mechanické vlnění o kmitočtu 40 kHz, a počítá dobu, za jakou se signál vrátí zpět.

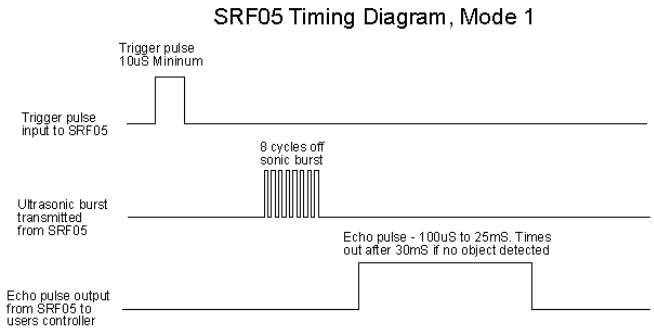
ob 1.2.7 – ultrazvukový senzor SRF-05

.

Byly použity 2 modely ultrazvukových čidel. Jako čelní senzor byl použit velmi kvalitní model SRF-05, který dokáže detekovat překážku o velikosti propisovací tužky na vzdálenost více jak 1 metru. Jako postranní senzory byly použity modely HY-SFR05, které nedisponují možností jednovodičové komunikace s procesorem, a také schopnost detekce překážky je nižší.

Mód 1 – SRF-04 kompatibilní

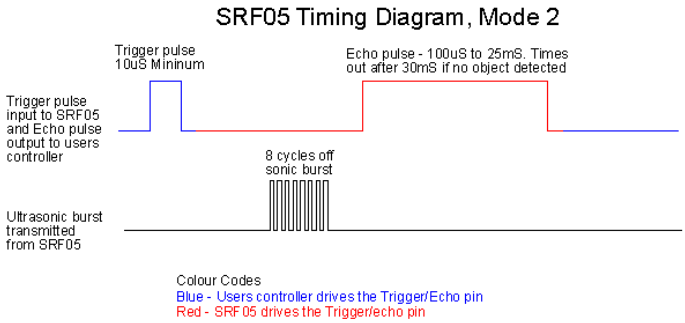
V tomto režimu je modul připojen na procesor 2 vodiči, přičemž po 1 vodiči (trigger/zpouštěč) je zaslán startovací impuls, při kterém senzor vyšle celkem 8 pulzů mechanického vlnění s frekvencí 40 kHz a vyčká návratu signálu zpět do příjmací části. Následuje analýza časové prodlevy, a zaslání informace o detekované vzdálenosti v podobě impulzu o délce přímo úměrné vzdálenosti po 2. vodiči (echo/odezva).



ob 1.2.8 – časový diagram SRF-05 v módu 1

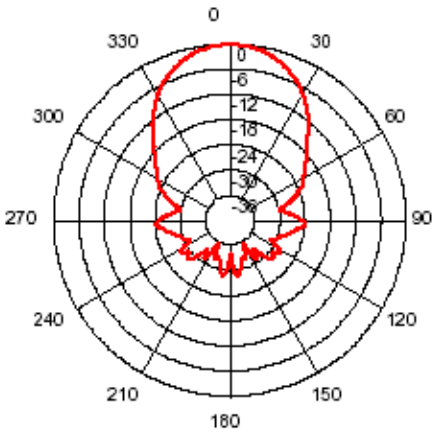
Mód 2 – komunikace po 1 vodiči.

Tento mód a shodný sled událostí jako mód 1, avšak veškerá komunikace probíhá po 1 vodiči.



ob 1.2.9 – časový diagram SRF-05 v módu 2

K přepnutí operačního režimu slouží pin MODE ,který pro aktivaci módu 2 je nutno vodivě propojit se zemí. Levnější kopie senzoru SRF-05 v podobě HY-SRF-05 nedisponuje možností módu 2.



Model SRF-05 je velmi citlivý a vzdálenost dokonce malých překážek určuje velmi přesně. Úhel operačního rozsahu je pohybuje okolo 70°. Maximální vzdálenost detekce překážky (4 mtery) není v praxi použitá, protože senzory jsou umístěny nízko nad zemí, a asi po 2 metrech zachytí odraz od země.

ob 1.2.10 – vyzařovací charakteristika SRF-05

Úhel operačního rozsahu je v případě modelu HY-SRF-05 je poloviční.Oba modely jsou při měření velmi náchylné na úhel stěny, na kterou je vysíláno echo. Při překročení mezního úhlu 50°, dochází k zkreslení dat, a detekovaná vzdálenost je často dvojnásobná .

Elektronický kompas

Pokud chceme, aby robot byl schopen samostatné orientace v prostoru pomocí GPS, a aby také byl schopen se samostatně dopravit na požadované souřadnice, je nezbytné jej vybavit kompasem, díky kterému bude možno robota nasměrovat na požadovaný azimut dle předem spočítaného vektoru.

Pro tento účel byl vybrán přístroj GY-26 od firmy Elechouse.

ob 1.2.11 – elektronický kompas GY-26

Modul se vyznačuje chybou menší jak 5°. Obvod je náchylný na horizontální polohu a při opuštění vodorovného položení dochází ke zvětšování chyby měření. Přístroj nabízí uživateli komunikaci dvěmi způsoby. Jednak po seriovém kanálu, a jednak pomocí TWI (two wire interface) sběrnice. Kompas umožňuje nastavení deklinačního úhlu. Jedná se o odklon magnetického severního pólu země od geografického severního pólu. V současné době je deklinační úhel pro oblast Ostravska asi 4° 9’ východně. Výrobce doporučuje při změně magnetického pole, respektive při výrazné změně geografických souřadnic, na kterých je kompas používán, kompas vždy kalibrovat.

Z pohledu komunikace je možno spojit procesor s kompasem 2 způsoby.

Sériová komunikace 9600/N/8/1

Existují předem dané 8bitové příkazy, které po doručení kompasu vyvolají adekvátní chování přístroje. Po provedení akce je zpětně do procesoru zaslán řetězec s naměřenými informacemi. Data jsou v ASCII kódu.

Například zasláním příkazu 0x31 (změř uhel) vyvoláme zpětný řetězec <0x0D-0x0A-0x33-0x35-0x39-0x2E-0x36-0x1C> = 359.6°

TWI sběrnice

Taktéž, jako u předchozí metody, pracujeme s 8bitovými příkazy. Princip fungování TWI však neumožňuje, aby slave (kompas) samostatně zaslal informace zpět, a proto všechna data uloží do paměti, ze které jsou data následně přečtená procesorem.

0x31: measure the angle (return the value of the angle)

0xC0: start calibration

0xC1: end calibration

0xA0-0xAA-0xA5-0xC5: return to the settings of the factory

0xA0-0xAA-0xA5-IIC_ADDR: change the IIC bus address

0x03-DECL_high: set the high 8 bits of declination angle

0x04-DECL_low: set the low 8 bits of declination angle

ob 1.2.12 – příkazy pro práci s el. kompasem

GPS modul

Pro komunikaci s GPS satelity je použit GPS příjmač UP501 od firmy Fastrex. Každou sekundu generuje až 10 řetězců obsahujících informaci o současné poloze příjmače. Výrobce udává chybu menší než 2,5 metru.

Ob 1.2.13 – GPS příjmač

Vedle hlavního napájecího pinu pro 3V, je možnost připojení záložního zdroje, který udržuje zařízení aktivní při proudovém odběru 10uA. Pro komunikaci s procesorem slouží sériový kanál s defaultním baudrate 9600 na napěťové úrovni CMOS. Zařízení disponuje integrovanou anténou, což umožňuje výbornou integraci při aplikaci.

2. SOFTWARE

2.1 **STM32F4**

V této kapitole jsou popsány rutiny pro ovladání periferii a nástin fungování hlavního programu. Celý program je možno najít v příloze. V hlavní pamětí se nachází 3 pole, každé pro snímek velikosti 320x200 px. PoleA[] slouží jako výstupní framebuffer a změny v něm budou známy koncovému uživateli. PoleB[] a poleC[] jsou interní pomocná pole pro mezivýpočty při zpracování obrazu. V paměti FLASH se nachází program a masky značek.

Načítaní dat z kamery

Před prvním použitím je nutno konfigurovat kameru. To se provede poslaním konfiguračních konstant (tzv.”Magic numbers”) do registru kamery. Jedná se o seznam doporučených hodnot pro konfiguraci.

```
void CamInit(void)
{           int i = 0;
CAMio();
while (config[i] != 0xFF)
{

WriteReg(config[i],config[i+1]);
  delay();
 i+=2;
}
}
```

V poli config[] se nachází konfigurační konstanty např. Dělička frekvence dat z kamery, formát dat, rozlišení atd. a adresy registrů ukončené koncovou značkou 0xFF. Kamera posílá data neustále, program přečte jeden snímek, další ignoruje. Snímek je uložen do RAM pro další zpracování.Jelikož kamera vysílá jeden pixel ve dvou bytech, data jsou nejprve uložena do dvou polí a následně přepočítana na jednotlivé pixely ve stupních šedi. Podprogram nejdřív čeká na dokončení přechozího snímku, což je určuje signál VSYNC. Poté se data

postupně načitají po řádcích se signálem HREF. Jednotlivé pixely se načítají synchronně se signálem PCK s frekvencí 2 MHz.

```
void CAM_Read(void)
{            int i,j;

 GPIO_WriteBit(GPIOB, LED2,(BitAction) 0);
 while(0==(((GPIOE->IDR >> (VSYNC))) &1))  {}                  // cekani na snimek
 while(1==(((GPIOE->IDR >> (VSYNC))) &1))  {}

             for (i = 0; i < Ymax;i++)
            {
                        while(0==(((GPIOE->IDR >> (HREF))) &1))          {}          //radek

                         GPIO_WriteBit(GPIOB, LED2,(BitAction) 0);
                                    for (j = 0; j < Xmax;j++)
                                    {
                                     while(0==(((GPIOE->IDR >> (PCK))) &1)){ };
                                     poleA[j][i]  = (unsigned char)GPIOE->IDR;
                                     while(1==(((GPIOE->IDR >> (PCK))) &1)){ }

                                    while(0==(((GPIOE->IDR >> (PCK))) &1)){ };
                                     poleB [j][i]  = (GPIOE->IDR);
                                    while(1==(((GPIOE->IDR >> (PCK))) &1)){ }
                                    }
                                     GPIO_WriteBit(GPIOB, LED2, (BitAction)1);
            }
}
```

Podprogram pro načtení dat z kamery

Zpracování obrazu

Detekce Hran

Pro detekování hran je použit modofikovaný Cannyho hranový detektor. Jendná se o algoritmus navržený Johnem Cannym z Berkeley Institute of Design pro počítačové zpracování obrazu. Algoritmus pracuje v 5 fázích.

1. Převedení barevného snímku na černobílý. Tato operace byla provedena při načítaní dat z kamery.

2. Redukce šumu. Byla realizováná pomocí gaussového filtru. Je použita matice 5x5.

```
const uint8_t blur[5][5]=
{       {2,4,5,4,2},
        {4,9,12,9,4},
        {5,12,15,12,5},
        {4,9,12,9,4},
        {2,4,5,4,2}  };
```

3. Výpočet amplitudy gradientu a úhlu .

   Je vypočtena derivace v horizontálním i vertikalním směru. Toho docilíme aplikovaním dvou kernelů.

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

   Poté vypočteme úroveň gradientu a zapíšeme do poleA[]. Jednotlivé hrany nejsou přesně dány jednou křivkou, ale více křvkami vedle sebe. Tento jev nám zaručuje potřebnou toleranci pro rozpoznání obrysu značky za stížených okolností.

$$D = \sqrt{D_x^2(x,y) + D_y^2(x,y)}$$

   Úhel gradientu vypočteme dle vzorce

$$\theta = \arctan\left(\frac{D_x(x,y)}{D_y(x,y)}\right)$$

   Výsledek zapíšeme do poleB[]. Úhel může mít pouze hodnoty 0,45,90,135 stupňů. Proto je nutno výsledek na tyto hodnoty zaokrouhlit. Výsledný úhel je zapsán do poleC[].

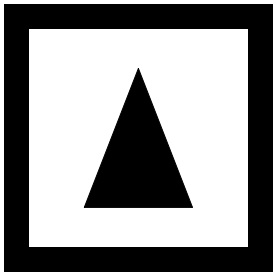
```
poleB[pixel][radek] = sqrt((tmpDx*tmpDx) +
(tmpDy*tmpDy));


tmpAngle = atan(fabs(tmpDx)/fabs(tmpDy));

thisAngle = (180*(double)tmpAngle)/3.14;
```

4. Potlačení méně výrazných hran. Tento krok byl nejdřív použit, ale pro další výpočty byla potřebná tolerance, kterou tento krok odstraňuje.
5. Hystereze. Poslední krok který definitivně určí které pixely jsou hrany .
   Pro tento účel byla sestavena skupina podmínek:
   - Je-li jas pixelu > Thigh, bod je vyhodnocen jako hrana.
   - Je-li jas pixelu < Tlow, bod je zavrhnut.
   - Je li jas pixelu >Tlow a současně jas pixelu < Thigh, je prohledáno okolí pixelu 3x3, je-li nalezena hrana, bod je taktéž vyhodnocen jako hrana.

ob 2.1.1 – výstup cannyho hranového detektoru

Scéna po aplikaci hranového detektoru. Jelikož značky jsou dobře rozpoznatelné, je možne použit velké hodnoty Tlow, což ma za následek eliminaci slabších hran a výrayné zjednoduššení scény.

Detekce Obvodu značky



Po získaní všech údajů o hranách je potřeba lokalizovat značku. Značka je ve tvaru čtverce a její hrubý obvod je rozpoznán jako 2 hrany. Toto je klíčové pro přesné identifikovaní obrysu značky a prevenci falešných detekcí.

Klíčovou funkcí v tomto procesu má detekce rohů. Základní princip detekce obvodu spočivá v nalezení 8 vzájemně odpovidajicích rohů. V celé scéně jsou detekovany všechny rohy,a zaznamenána do struktury pozice a orientace rohu. Po

lokalizovaní všech rohů následuje testování jednotlivých bodů zda-li tvoří čtverec. Nedřív je testováno jestli levý horní roh má v toleranci 7stupňů a rozmezí 30 až 150px odpovidající pravý horni roh. Všechny možné rohy jsou zaznamenány pro budoucí upřesnění. Není li podmínka splněna, automaticky je bod zavrhnut a algoritmus přechází na další levý horní.Je-li detekce úspěšná algoritmus hledá levé dolní rohy se stejnou podínkou jako pro pravý horní.

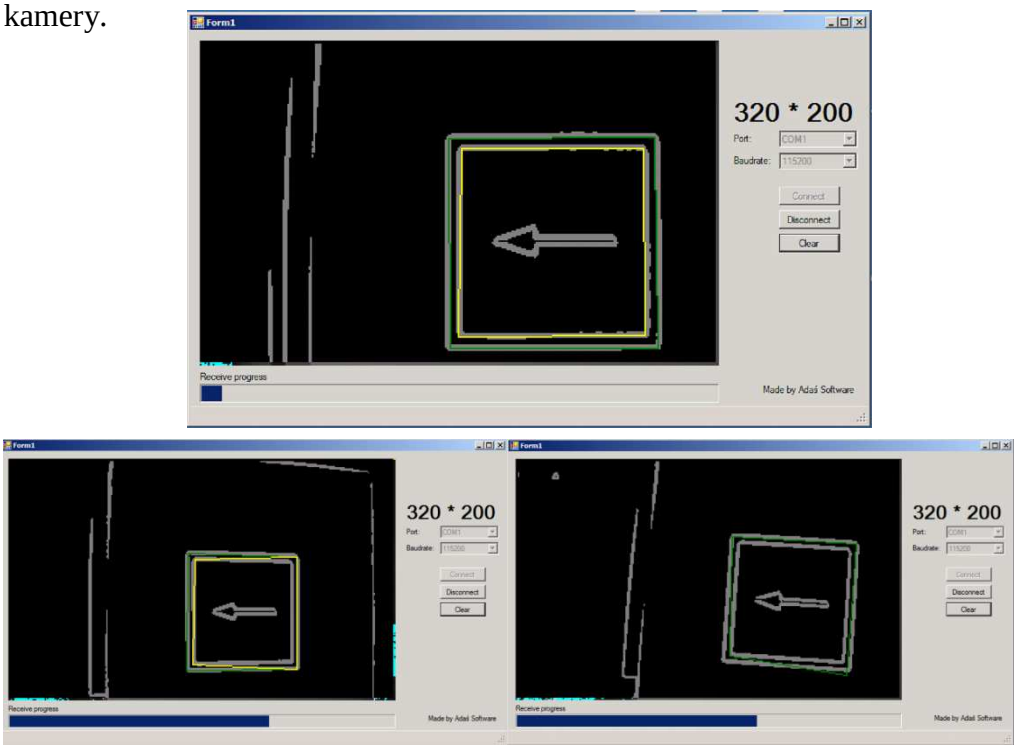
Po zpracování celé scény jsou možné výsledky dále upřesněny. Pro každý potencionální levý horní roh, porovnána vzdálenost od levého horního do pravého horního rohu, a rovná li-se se vzdálenosti levého horního a levého dolního rohu, je velmi pravděpodobné že se jedná o čtverec.Jelikož obvod značky je detekován jako dvě hrany a obě jsou uvnitř 7 stupňové tolerance je vypočítana odchylka jednotlivých bodů. Rohy s nejmenší odchylkou jsou přiřazeny do jednoho objektu. Pravý dolní roh plní pouze doplňkovou funkci kontroly 100% rozpoznání čtverce.

```
struct objekt{     u16 LHx;
                   u16 LDx;
                   u16 PHx;
                   u16 PDx;
                   u8 LHy;
                   u8 LDy;
                   u8 PHy;
                   u8 PDy;
                   u8 neuplny ;
                   u8 delka;

                   };
```

V této fázi je značka značena jako 2 čtverce. Pro lokalizováná značky je třeba najít pár čtverců vzdálených od sebe maximálně 10% (poměr šířky hrany k šířce značky).Pokud je test potvrzen jsou tyto dva čtverce definitivně označeny za značku a zapsány do struktury rozpoznano.

```
struct rozpoznano{
                   u16 LHx;
                   u16 LDx;
                   u16 PHx;
                   u16 PDx;
                   u8 LHy;
                   u8 LDy;
                   u8 PHy;
                   u8 PDy;

                   u8 velikost;
                   u8 vzdalenost;
                   u8 vyznam;
                        };
```

Na následujicím obrázku vidíme detekované dva čtverce, vizualizované pomocí implementace funkce line().Podle velikosti strany značky je vypočtena vzdálenost zmačky od kamery.

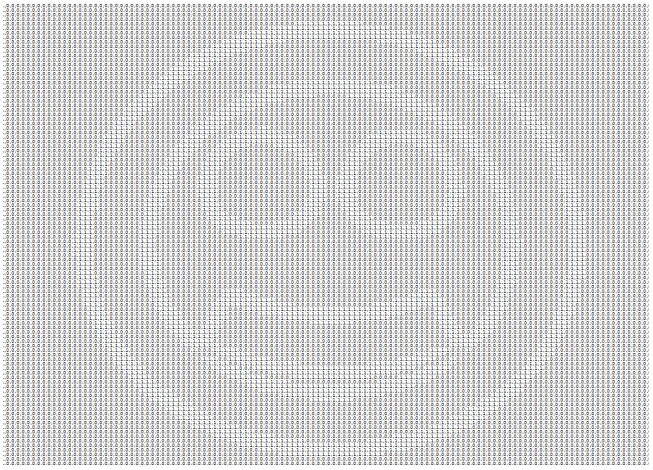


ob 2.1.2 – detekované značky

Obrázky zachycuji nedokonalou detekci vlivem otočením doleva a nakloněním. I přes nedokonalou detekci všech bodů, značky zůstavají použitelné.

Rozpoznání značky

Pro navigování autíčka byla vytvořena sada značek stejné velikosti,každá jiného významu. Princip rozpoznání značky spočíva v aplikování masky na plochu značky.Pro každou masku je počítana procentuální shoda, maska s největší schodou určuje význam značky. Maska je nasnímaná v největším možném rozměru.Pro správnou funkčnost při různých vzdálenostech je nutno masku zmenšit.



ob 2.1.3 – bitová maska

```
        pomer = (float)MASKA_VELIKOST / (float)Znacka[_znacka].velikost;

for (w=0;w<MASKA_POCET;w++)                    // test vsech masek
 {

for (radek = Znacka[_znacka].LHy,k=0; k < Znacka[_znacka].velikost; radek++,k++)
 // maska
                {

                        for (pixel = Znacka[_znacka].LHx,l=0; l < Znacka[_znacka].velikost ; pixel++,l++)
                                {

                                  if(poleA[pixel][radek]==triag[w][(int)(pomer*(float)k)][(int)(pomer*(float)l)])
                                        if(poleA[pixel][radek]==1)
                                         {
                                        poleA[pixel][radek]=3;
                                        uspesnost++;
                                        }
                                }
}       }
```

2.2 **Atmega128**

ATmega128 se programuje pomocí sběrnice SPI, po které je program nahrán do integrované flash paměti ze které je posléze spouštěn. Vlastní paměť pro program je velká 128kB, od čehož je odvozeno číslo procesoru. Protože počítač nedisponuje výstupem SPI sběrnice, je pro nahrávání programu do paměti procesoru použitý převodník USB/SPI. Procesory řady ATmega jsou postaveny na architektuře AVR pro kterou firma Atmel poskytuje podporu v podobě volně dostupného prostředí AVRstudio 5. Jedná se o software běžící na enginu VisualStudia 2010.

Samotný program pro řízení robota a práci se všemi senzory je napsán v jazyce C.

LCD a Klávesnice

Pro ovládání zobrazovače byl použit include soubor vytvořen panem Peterem Fleurym, který je publikovaný na jeho webových stránkách. Jedná se o profesionální software podporující vedle HD44780 také řadič KS0073. Je možné definovat prakticky jakékoliv nastavení displaye. Include soubor má vytvořené standardní funkce jako put_c( ), put_s ( ), gotoxy( ), nebo clrscr ( ).Pro naše potřeby byl ovladač mírně upraven a byl doplněn o některé užitečné funkce.

H-můstek

Ovládání H-můstků probíha skrze port C. Jedná o piny s otevřeným kolektorem, a proto potřebují stálou zátěž na vstupu. Procesor v době běhu programu vyhodnotí nutnost změny aktuálního směru pohybu robota, a zavede do pomocné proměnné příslušnou konstantu.

```
#define FORWARD 0x1B    // 00011011b - X X DB2 DA2 E2 DB1 DA1 E1
#define BACK 0x2D       // 00101101b
#define LEFT 0x1D       // 00101011b
#define RIGHT 0x2B      // 00011101b
#define STOP 0x00       // zastavit
```

ob 2.2.1 – konstanty pro výstupní stavy portu C

Následuje sekvence rutin, v níž procesor přečte pomocnou proměnnou, a tu přenese na výstup.Stavebním kamenem ovládání rychlosti motorů pomocí H-můstků, je použití 8bitového integrovaného časovače. Pomocí indikátoru přetečení vnitřního registru časovače, je možné pracovat s registrem DDRC, který přepíná port mezi vstupním a výstupním nastavením, a vytvářet tak pulně šířkovou modulaci (PWM) na výstupu portu C.

```
void engines_init()
{
    TCCR0 = (1 << CS02) | (0 << CS01) | (1 << CS00);    // nastav F_CPU/1024
    TIMSK = (1 << TOIE0);   //povolit interrupt přetečení
    TCNT0 = 0x00;   //výchozí hodnota čítacího registru
    gearbox = 'A';  // rychlost motorů
}
```

Ob 2.2.2 – inicializace časovače TIMER0

Pro jemné řízení pohybu vozidla, kdy každý pás se může pohybovat nejen jiným směrem, ale také jinou rychlostí, bylo použito dodatečné maskování registru DDRC, díky čemuž je možné vypínat s určitou frekvencí danou část portu, a snížit tak střídu výstupu.

Ultrazvukový senzor vzdálenosti

Ovládání ultrazvukového čidla vzdálenosti probíhá pomocí předávání impulzů o různé délce. Jsou ovládány celkem 3 zařízení, přičemž 1 pracuje v módu komunikace po 1 vodiči, a 2 využívají ke komunikaci 2 vodiče. Na začátku je na trimovacím vodiči vytvořena náběžná hrana impulzu trvajícího 100us, a následně je zpuštěna sekvence vyčkávání na náběžnou hranu na vodiči ECHO. Při detekci počátku příchozího impulzu je zpuštěn 16bitový časovač TIMER1, a zároveň je zahájená sekce vyčkávající na konec impulzu. V okamžiku zjištění sestupné hrany je časovač zastaven, a jeho hodnota je vrácena jako výsledek měření. Následně je tato hodnota převedena na hodnotu, která odpovídá vzdálenosti v centimetrech. Měření je prováděno 3 krát, přičemž konečný výsledek měření je aritmetickým průměrem všech tří hodnot. Tímto postupem eliminujeme chyby měření vzniklé přeslechem mezi senzory.

Elektronický kompas

Komunikaci se zařízením probíhá po sběrnici TWI. Práce s kompasem spočívá na principu, že se otevře spojení mezi masterem (procesorem), a slavem (kompasem), kdy master je v režimu zápisu. V tomto okamžiku dojde k zaslání příkazového řetězce, ze kterého slave rozpozná příkaz, a provede příslušnou operaci.Následuje uzavření komunikačního kanálu.Následně je kanál znovu otevřen, je zaadresován slave v režimu pro zápis, a je zaslaná adresa EEPROM paměti slavu kterou procesor chce přečíst. Následuje zaadresování slavu v režimu pro čtení, a slave vyšle na sběrnici byte obsažený v předem naadresované buňce paměti EEPROM. Komunikační kanál je poté uzavřen.

```
i2c_start_wait(Dev24C04+I2C_WRITE);
i2c_write(0x00);
i2c_write(0x31);

i2c_stop();

i2c_start_wait(Dev24C04+I2C_WRITE);
i2c_write(0x01);
i2c_start_wait(Dev24C04+I2C_READ);
cmps_angle_high = i2c_readNak();
i2c_stop();

i2c_start_wait(Dev24C04+I2C_WRITE);
i2c_write(0x02);
i2c_start_wait(Dev24C04+I2C_READ);
cmps_angle_low = i2c_readNak();
i2c_stop();
angle[i] = 0;
angle[i] = cmps_angle_high*255 + cmp
```

ob 2.2.3 – ukázka prvotního navázání spojení s kompasem

GPS

Ze strany procesoru, komunikace probíhá na sériovém kanálu UART0 s rychlostí přenosu 9600 baudů. Pro výměnu informací se používá protokol NMEA. Jedná se o protokol využívaný pro námořní přístroje a pro práci s GPS přijímači. NMEA protokol pracuje na principu posílání dat ve znakových řetězcích, přičemž jednotlivá data jsou od sebe oddělena čárkami. Tento způsob oddělovaní jednotlivých datových složek umožňuje velmi efektní čteni a interpretaci vlastních informací.

Obecně lze NMEA komunikaci rozdělit do 2 kategorií. NMEA příkazy a NMEA výstupy. Příkazy spouží k samotnému ovládání zařízení a s jejich pomocí je možné vyžádat informace o stavu zařízení, či stavu firmwaru. NMEA výstupy obsahují řetězce s daty. GPS modul začíná vysílat tyto řetězce automaticky, okamžitě po připojení zařízení na napájení.

GPS modul podporuje 7 výstupních řetězců, z nichž každý je kompilátem různých dat. Pro jednoduché určení pozice a času je nejvhodnější řetězec GGA – Global Positioning System Fix Data. Tento řetězec obsahuje následujicí data

GGA - Global Positioning System Fix Data

Time, position and fix related data for a GPS receiver.

Example:

$GPGGA,114353.000,6016.3245,N,02458.3270,E,1,10,0.81,35.2,M,19.5,M,,*50

ob – 2.2.4 – popis GGA řetězce z „NMEA manual for Fastrax IT500“

Čtení dat procesorem probíhá u podprogramu přerušení. Detekuje li procesor řetězec znaků “$GPGAA”, věnuje náldeujicím datům zvláštní pozornost, a z jejich obsahu vypočítá potřebná data o aktuálních souřadnicích, eventuálně času, počtu satelitů, atd.

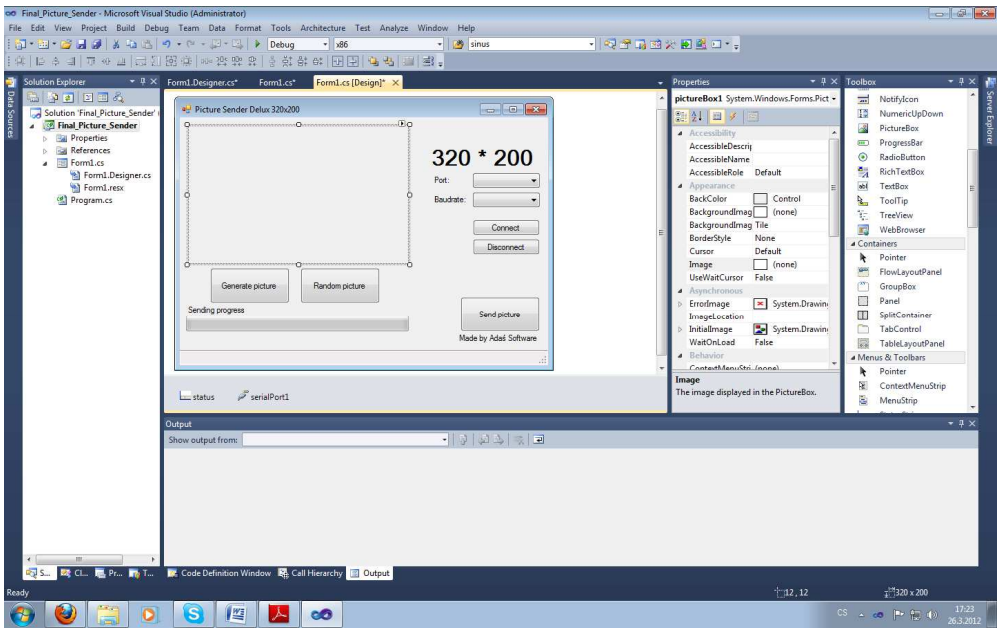
Zná-li procesor cílové souřadnice, dokáže vypočítat úhel mezi danými vektory, a určí tak azimut pro další trasu. Následně probíhá udržování získaného azimutu pomocí el-kompasu.

$$\cos\varphi = \frac{u_1 v_1 + u_2 v_2}{\sqrt{u_1^2 + u_2^2} \cdot \sqrt{v_1^2 + v_2^2}}$$

3. Grafické uživatelské prostředí (GUI)

Software byl navržen jako zobrazovací jednotka pro všechna získaná data ze senzorů instalovaných na robotu. Výstupním produktem má být uživatelsky přívětivé okno se snadným a intuitivním ovládáním řízeného robota. V okně budou zobrazovány údaje o rychlosti, azimutu, GPS souřadnicích, bude zde obraz vidění kamery instalované na stroji, a další prvky ovládání a vizualizace. Komunikace mezi autonomní jednotkou musí probíhat velmi rychle, a spolehlivě, protože ztráta spojení by vedla k havárii a možnému poškození stroje. Celý software je napsán v jazyce C# s použitím platformy .NET Windows Forms. Původní verze programu komunikuje s robotem pomocí bluetooth virtuálního sériového kanálu generovaného operačním systémem. Fyzicky počítač a robot komunikují pomocí integrovaných bluetooth a wifi modulů.

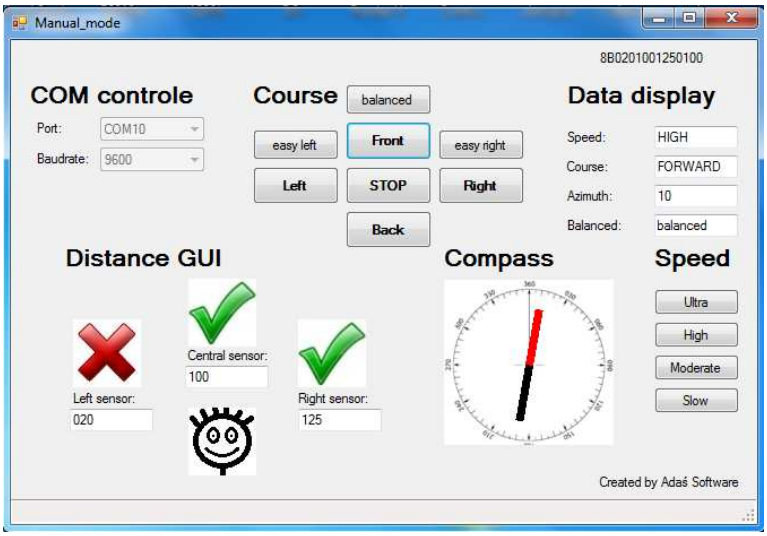
Celá aplikace byla vytvořená ve vývojovém prostředí Microsoft Visual Studio 2010 od firmy Microsoft. Jakožto produkt jedné firmy, má Visual Studio silnou podporu tvorby aplikací na .NET platformě pod jazykem podobě C# v podobě vytváření Win Form aplikací s podporou grafické konstrukce okna.



ob 3.1 – tvorba Win Form aplikace ve Visual Studiu 2010

Programátor má k dispozici celou paletu ovládacích prvků, které rozmisťuje po pracovním oknu a okamžitě jim může přiřazovat funkčnost psaním vlastního zdrojového kódu. Následně má programátor možnost přiřazování eventů k jednotlivým ovládacím prvkům, které pak reagují na různé vnější podněty, jakými jsou příchozí data, stisknutí klávesy, kliknutí myší.

Program je zkompilován do spustitelného formátu .exe pro operační systém Windows. Vedle samotného .exe souboru je nutno v kořenové složce mít umístěny také .bmp obrázky nutné pro správné vykreslování všech funkcí programu. V okamžiku stlačení tlačítka „Connect“ dojde k ověření navázání spojení s bluetooth jednotkou, a provede se připojení na komunikační interface s robotem.



ob 3.2 – zobrazovací formulář

V případě zdárného navázání kontaktu s robotem dojde k otevření zobrazovacího formuláře, ve kterém se zobrazují data ze všech čidel instalovaných na robotovi. Kontrolní panel pro manuální mód demonstruje všechny zobrazovací prvky programu, protože se předpokládá, že v tomto operačním módu uživatel vyžaduje maximum údajů o voze. V jiných módech je množství zobrazovaných údajů redukováno na nezbytné množství. Na kontrolním panelu je možné vidět data ze senzorů vzdálenosti, kompasu, stav motorů a softwarového převodu (PWM), a dále zde naznáme prvky pro ovládání pohybu robota. Pomocí eventů je možné v programu vytvořit kód, který je proveden vždy v okamžiku stisknutí libovolné klávesnice. V případě manuálního módu je možné pomocí klávesnice vykonávat s robotem všechny pohybové úkony, a libovolně nastavovat rychlostní stupně.

Závěr

Zhotovené vozidlo je schpno samostatného provozu, reaguje na všechna čidla, komunikuje s PC. Pokud do paměti procesoru nahrajeme více šablon značek, řízení vozidla bude sofistikovanější. Algoritmy hledání cesty lze neustále zlepšovat.

Použité zdroje a literatura

Literatura k procesoru ATmega128

ATmega128.pdf - technická dokumentace procesoru ATmega128

AVR mega128 64 mini development board.pdf - technická dokumentace vývojové desky osazené procesorem ATmega128

http://www.avrfreaks.net/ - internetové fórum uživatelů procesoru AVR

Zdroje k GPS příjmači

http://www.fastraxgps.com/products/gpsantennamodules/500series/up501/ - technická dokumentace GPS příjmače

NMEA manual for Fastrax IT500 Series GPS receivers_V1.7.pdf - dokumentace NMEA protokolu

http://www.engineersgarage.com/embedded/avr-microcontroller-projects/gps-interface-circuit#tabset-tab-5 - ukázka užití GPS příjmače na procesoru AVR

http://www.adafruit.com/blog/2012/01/20/new-product-up501-breadboard-friendly-66-channel-gps-module-w10-hz-updates/ - ukázka užití GPS příjmače na procesoru AVR

Literatura k ultrazvukovému detektoru vzdálenosti

http://www.robot-electronics.co.uk/htm/srf05tech.htm - technická dokumentace čidla SRF-05

http://home.roboticlab.eu/en/examples/sensor/ultrasonic_distance/srf05 - ukázka práce s čidlem SRF-05

Literatura k elektronickému kompasu

http://home.roboticlab.eu/en/examples/sensor/ultrasonic_distance/srf05 - technická dokumentace elektronického kompasu

Zdroje k LCD display a TWI sběrnici

http://jump.to/fleury - internetová strana pana Petera Fleury, na kterých publikoval zdrojové kódy pro ovládání LCD a TWI sběrnice

Literatura k jazyku C#

http://www.dreamincode.net/forums/forum/84-c / - fórum programátorů v jazyce C#

http://www.codeguru.com/forum/ - fórum programátorů v jazyce C#

http://msdn.microsoft.com/en-us/library/ - reference jazyka C#

"Programivání Microsoft Windows Forms v jazyce C#" Charles Petzold

http://stackoverflow.com/ - fórum programátorů v jazyce C#

http://www.techotopia.com/index.php/C_Sharp_Essentials - reference jazyka C# a ukázky kódu

http://www.daniweb.com - fórum programátorů v jazyce C#

Literatura k rozpoznání obrazu

http://www.divms.uiowa.edu/~cwyman/classes/spring08-22C251/homework/canny.pdf

http://www.cse.unr.edu/~bebis/CS791E/Notes/EdgeDetection.pdf

http://perso.limsi.fr/Individu/vezien/PAPIERS_ACS/canny1986.pdf

Literatura k mikroprocesoru ARM:

http://www.st.com/internet/mcu/product/252135.jsp

http://www.linear.com/product/LT3028 - datasheet měniče

Všechny odkazy jsou platné k 4.4 2012.