UNIVERZA V LJUBLJANI
Fakulteta za elektrotehniko

MERILNIK POSPEŠKOV V AVTOMOBILU

Rok Vinder

Predmet: Seminar
Nosilec predmeta: doc.dr. Marko Jankovec

1. Časovni in finančni plan projekta

Za razvoj novega produkta vse od ideje do fizičnega izdelka je običajno predvidena časovna doba treh mesecev. Seveda je ta doba odvisna tudi od zahtevnost projekta in ekipe inženirjev, ki sodelujejo na projektu. Pomembno je, da projekt ne zastara in, da pustimu projektu odprte možnosti za razširitve in to prevsem programske.

Za razvoj in izdelavo projekta predvidevam časovno dobo štirih mesecev. V privi fazi se bom lotiv ideje, nato izbire elementov in risanja tiskanega vezja. Za to fazo predvidevam časovno dobo dveh mesecev. Za izdelavo tiskanine in postavitev elementov bom potreboval en mesec. Zadnjo fazo, ki je programiranje in testiranje modula pa bom opravil v enem mesecu.

Razvoj projekta finančno ocenjujem na 220€. V to ceno je vštet en modul katerega bom testiral (120€), razlika pa bo porabljena za nabavo orodja, ki ga bom potreboval pri spajanju elementov.

2. Uvod

Modul za detekcijo pospeška in pojemka je razširjen na različnih področjih. Najbolj tam, kjer nas zanima pozicija oziroma orientacija nekega predmeta glede na g silo ali pa g sila, ki deluje na to telo. Gravitacijska sila je sila, ki kaže vedno v središče zemlje oziroma središče planeta na katerem se trenutno nahajamo. Da si bi gravitacijsko silo lažje predstavljali, lahko vzamemo za primer vodno tehtnico. Senzor, ki ga uporabljam lahko primerjamo s tremi vodnimi tehtnicami, ki so obrnjene tako kot, da bi tvorili tri osni koordinatni sistem. Če si definiramo neko referenčno točko, lahko predmet obračamo v vse smeri in vedno bomo vedeli za pozicijo tega predmeta. Modul torej lahko uporabimo kot vizualizacija tridimenzionalnega predmeta, za katerega lahko preko električnih signalov vemo njegovo orientacijo glede na g silo. Moramo pa se zavedati, da senzor ne deluje kot žiroskop in ne zaznava rotacije predmeta okoli svoje navpične osi, vendar zaznava smer oziroma orientacijo predmeta glede na razpored g sile senzorjev v sistemu. Klasična aplikacija za takšen nadzor bi bila vizualizacija odprtost zračne lopute. Še ena klasična aplikacija je tudi proženje zračnih blazin. V tem primeru bi bil modul primeren predvsem zato, ker je v enem senzorju integrirana detekcija g sile tako v smeri vožnje kot tudi v bočni smeri (vklop bočnih zračnih blazin).

3. Izbira senzorja

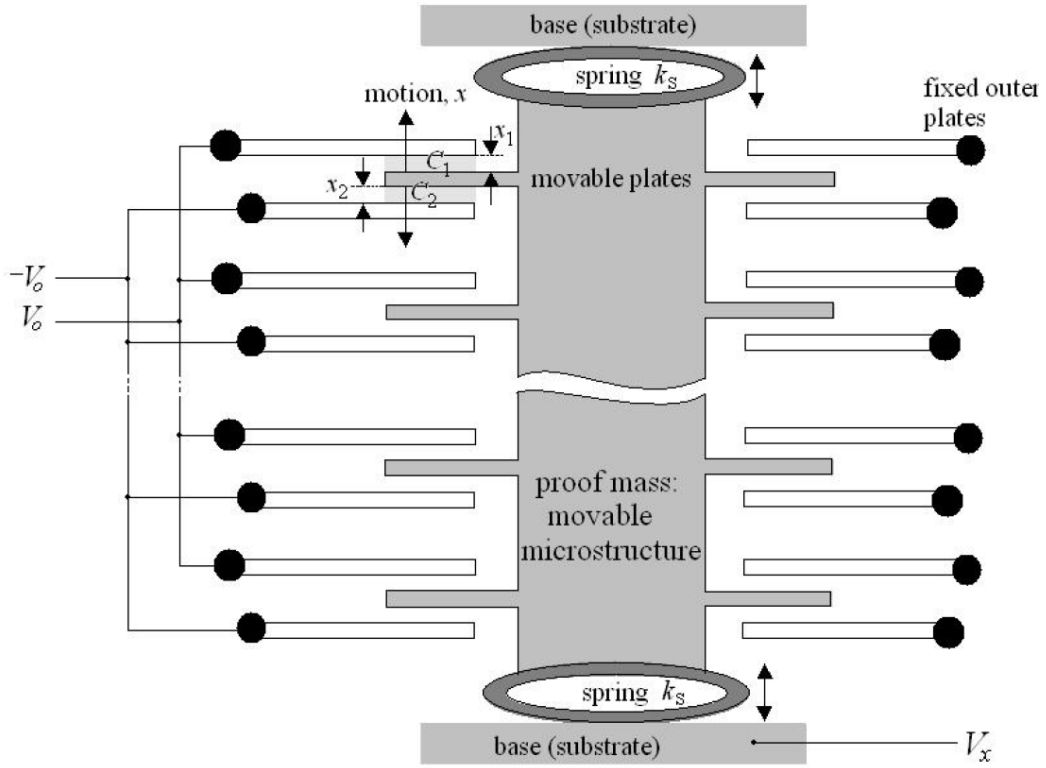
Pri meritvah električnih signalov oziroma fizikalnih veličin si moramo najprej izbrati senzor, s katerim bomo zajemali veličine, sistem za obdelavo podatkov (v večini primerov gre za mikrokrmilnik) in prikaz oziroma shranjevanje podatkov v bazo. V nalogi sem se odločil za senzor MMA374L, mikrokrmilnik AVR ATmega16L in LabView programsko opremo. Senzor MMA374L je primeren za merjenje pospeška od -3g do +3g oziroma -11g do +11g. Testne meritve so pokazale, da imamo opravka s pospeški oziroma pojemki od -1.5g do +1,5g zato je senzor MMA374L primeren za želeno aplikacijo. Mikrokrmilnik ATMega16L zajema analogne podatke senzorja in jih pošilja programski opremi LabView. Krmilnik ima potrebno periferijo za zajemanje podatkov in komuniciranje z osebnim računalnikom preko RS232 komunikacije. Modul priključimo na osebni računalnik z USB vodilom, zato je potreben tudi vmesni člen med mikrokrmilnikom in računalnikom. Za pretvorbo je potrebno integrirano vezje FT232, ki je virtualni serijski komunikacijski port. Operacijski sitem ob prvi priključitvi zazna napravo, ki ji moramo dodeliti voznike ("driver"), da lahko operira z njo. Če povzamemo bo torej senzor zaznal pospešek, ki ga analogno predstavi mikrokrmilniku. Nato ga Mikrokrmilnik pošlje preko rs232 komunikacije integriranemu vezju FT232. Ta pretvori signale v USB obliko komunikacije, LabView pa prebere podatke, ki jih nato obdela in prikazuje. Ob vpisu želenih parametrov v programu LabView, lahko modul deluje tudi kot samostojna enota v avtomobilu ali kakšnemu drugemu sistemu.

4. Delovanje senzorja

Senzorje pospeška lahko delimo na analogne (v tem primeru imamo kot izhodni podatek senzorja napetost, ki je linearno odvisna z g silo) in digitalne (tu imamo običajno protokol, s katerim komuniciramo s senzorjem, ki nam pošilja podatek o g sili). Prednosti digitalnih senzorjev so predvsem, da jih lahko povezujemo v mreže, podatke, ki jih dobimo so odvisni od komunikacije in ne od napetosti (analogni senzorji), seveda pa so digitalni senzorji zato bolj sofisticirani, saj morajo imeti integriran tudi mikrokrmilnik, ki krmili komunikacijo. Pri analognih senzorjih lahko rešimo določena odstopanja s programsko kompenzacijo, važno je le, da je senzor

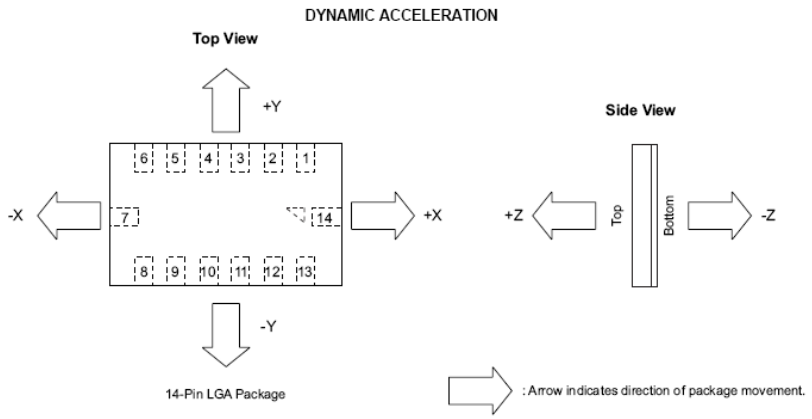
čimbolj linearen. Senzorje lahko razdelimo tudi po območju oziroma velikosti in sicer za nizke, srednje in visoke g sile. Razpon g sile je lahko od 1G pa tudi do 100G.

Kljub temu, da gre za tako imenovane ‘’solid state’’ elemente, kar pomeni, da element nima nobenih mehanskih delov, je na samem substratu le neka vzmet, ki je linearno povezana z g silo. Takšne sisteme, ki imajo mehanske komponente vgrajene v integrirana vezjia imenujemo MEMS (micro electromechanical systems). Senzor ima na substratu dve mikro mehanske vzmeti, ki sta povezani v blok z odcepi, ta odcep pa je del kondenzatorja. Omenjeni kondenzator ima tri odcepe. Dva sta mirujoča glede na senzor (to sta zunanja odcepa), notranji odcep pa je gibljiv (glede na g silo), ker je preko bloka, ki je na mikro vzmeti povezan na substrat. Senzor ADXL05 ima 46 parov kondenzatorjev. Cilj je torej linearno razmerje med g silo, razliko kapacitivnosti in zunanjo napetostjo. Zunanja fiksna kontakta kondenzatorja sta napajana s pravokotnimi impulzi, izhod pa je na sredinskem odcepu kondenzatorja. Napajalni napetosti zgornjega in spodnjega kontakta se razlikujeta po faznem zamiku, ta je 180°. V stacionarnem stanju oziroma, ko na kondenzator ne vpliva G sila, je razdalja med srednjim odcepom in zgornjim odcepom enaka kot med srednjim odcepom in spodnjim odcepom kondenzatorja. V tem primeru je izhodna napetost V_x enaka 0V. V trenutku, ko senzor zazna g silo, se na V_x pojavi neka pravokotna napetost zaradi diference kapacitivnosti. V primeru pospeška dobimo na V_x pravokotne impulze (višina oziroma maksimalna napetost impulzov določa pospešek), če na senzor vpliva pojemek V_x dobimo pravokotne impulze, ki so zamaknjeni za 180°. Signal je ojačan in moduliran, saj želimo na izhodu senzorja čisto enosmerno napetost.



MEMS izvedba pospeškomera - MEMS pospeškomeri

Senzor MMA7341L proizvajalca Freescale je triosni analogni senzor. Senzor deluje na 3,3 V logiki in kar pomeni da je 0G definirana s polovično napajalno napetostjo. To pomeni, da bo pri 0G izhodna napetost senzorja 1,65V. Senzor ima tudi vhode. Ob priklopu "selftest" na logično '1' nam senzor avtomatsko pokaže g sile na vse tri osi in sicer na X in Y os -0.1G in Z os +1G.



Smeri pospeškov glede na ohišje senzorja MMA7341 - Dokumentacija MMA7341L pospeškomera

Tako lahko testiramo odstopanja. "g-Select" vhod nam služi za nastavitev občutljivosti senzorja. Če ima "g-Select" logično '1' potem merimo silo do 11G, če je ta vhod vezan na logično '0', potem merimo silo do 3G. "Sleep" je namenjen za preklop na način z minimalno porabo (3μA) oziroma tako imenovan način v pripravljenosti ("stand by mode").

Občutljivost pospeškomera

g-Select	g-Range	Sensitivity
0	3g	440 mV/g
1	11g	117.5 mV/g

5. Mikrokrmilnik:

Mikrokrmilnik AVR proizvajalca Atmel je zasnovan na Harvardski RISC ("Reduced instruction set computer") tehnologiji. Prednosti pred CIS ("Complex Instruction Set Computer") tehnologijo je predvsem v tem, da imajo manjši nabor ukazov, katere dolžine so enako dolgi in se izvršijo v enem urinem ciklu, zato so tudi hitrejši. AVR družina mikrokrmilnikov je ena izmed prvih, ki je imela FLASH spomin. Periferije se razlikujejo po družinah AVR mikrokrmilnikov. Najmanjša serija je tinyAVR, največja pa XMEGA. AVR, ki sem ga uporabil v specialistični nalogi je ATMega16L. Poleg 16k Bajtov FLASH spomina ima ATMega16L še 512 Bajtov EEPROM spomina, 1k Bajt SDRAM, JTAG vmesnik, 8 kanalov analogno digitalnih pretvornikov, UART komunikacijo, itd. Zadnja črka oznake mikrokrmilnika "L" označuje, da je napajanje oziroma celotna logika sistema od 2,7V do 5,5V. V primeru, da gre za oznako mikrokrmilnika brez črke "L", je logika sistema od 4,5V do 5,5V. Prednost mikrokrmilnika je tudi vgrajeni oscilator, ki ga nekateri mikrokrmilniki podobnih družin nimajo.

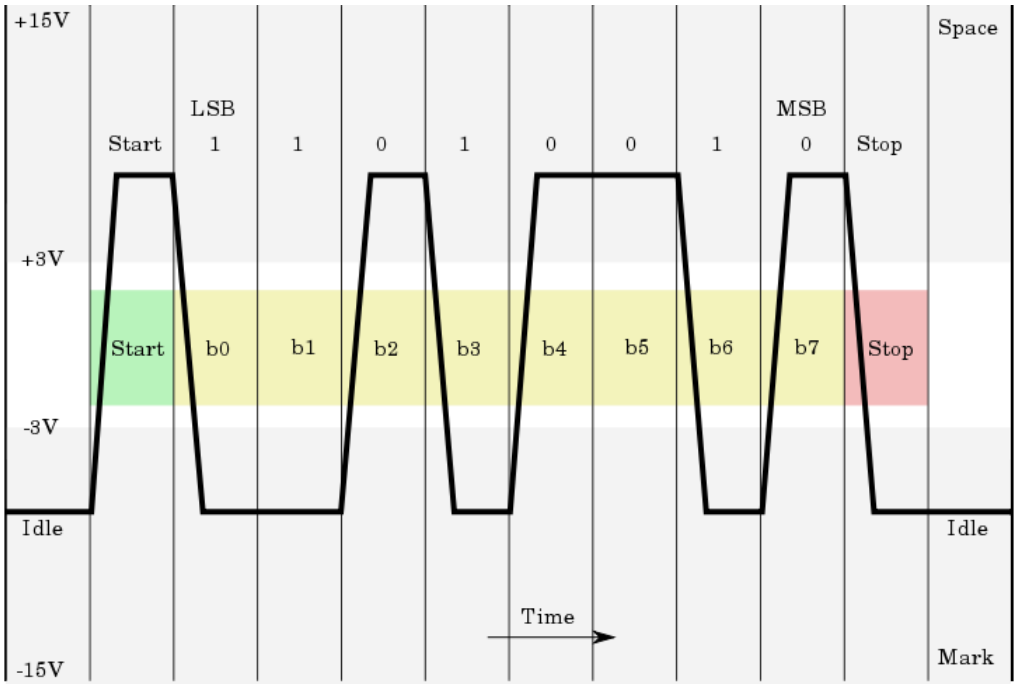
A/D pretvorba:

Med pomembno periferijo, ki je danes v skoraj vsakem mikrokrmilniku je tudi analogno digitalni pretvornik. Poznamo več načinov oziroma izvedb analognih pretvornikov. ADC ("analog to digital converter") bit za bitom pretvarja neko analogno

napetost v digitalno tako, da primerja vse digitalne kombinacije pri tem pa uporablja digitalni števec. ADC z zaporednimi približki deluje tako, vzame polovico referenčne vrednosti in jo nato primerja z iskano vrednostjo. V primeru, da je ta vrednost višja zapiše na LSB ("Least significant bit") enico, v nasprotnem primeru pa ničlo. Biti se vpisujejo v SAR ("Successive approximation register"). Krog se ponovi vendar, da nato razpolavljamo zgornji oziroma spodnji del referenčne napetosti in jo zopet primerjamo z iskano vrednostjo. Nato se ničla ali enica postavi na drugi bit. Tolikokrat kot se krog ponovi toliko je biten ADC.

6. Pretvornik komunikacije FT232:

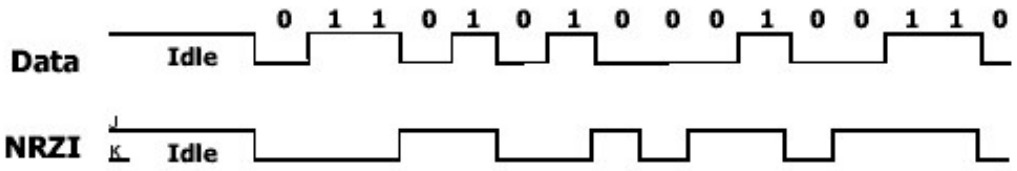
RS232 komunikacija je prva in ena izmed osnovnih serijskih komunikacij, katero so prvič predstavili že leta 1962. Vse nadalje serijske komunikacije so izhajale iz omenjene komunikacije. Razlikujejo se le po strojnih (hardware) parametrih, kot so časovni poteki, napetostni nivoji, itd. in po protokolu, ki zajema logiko komuniciranja med dvema ali več napravami. RS232 komunikacija ima logični nivo ničle od -15V do -3V in nivo logične enice med +3V do +15V. Protokol serijske komunikacije v osnovi poteka tako, da nam prvi bit predstavlja začetek komunikacije, potem si sledijo podatkovni biti in na koncu komunikacije se postavi stop bit.



Časovni potek RS232 komunikacijskega protokola

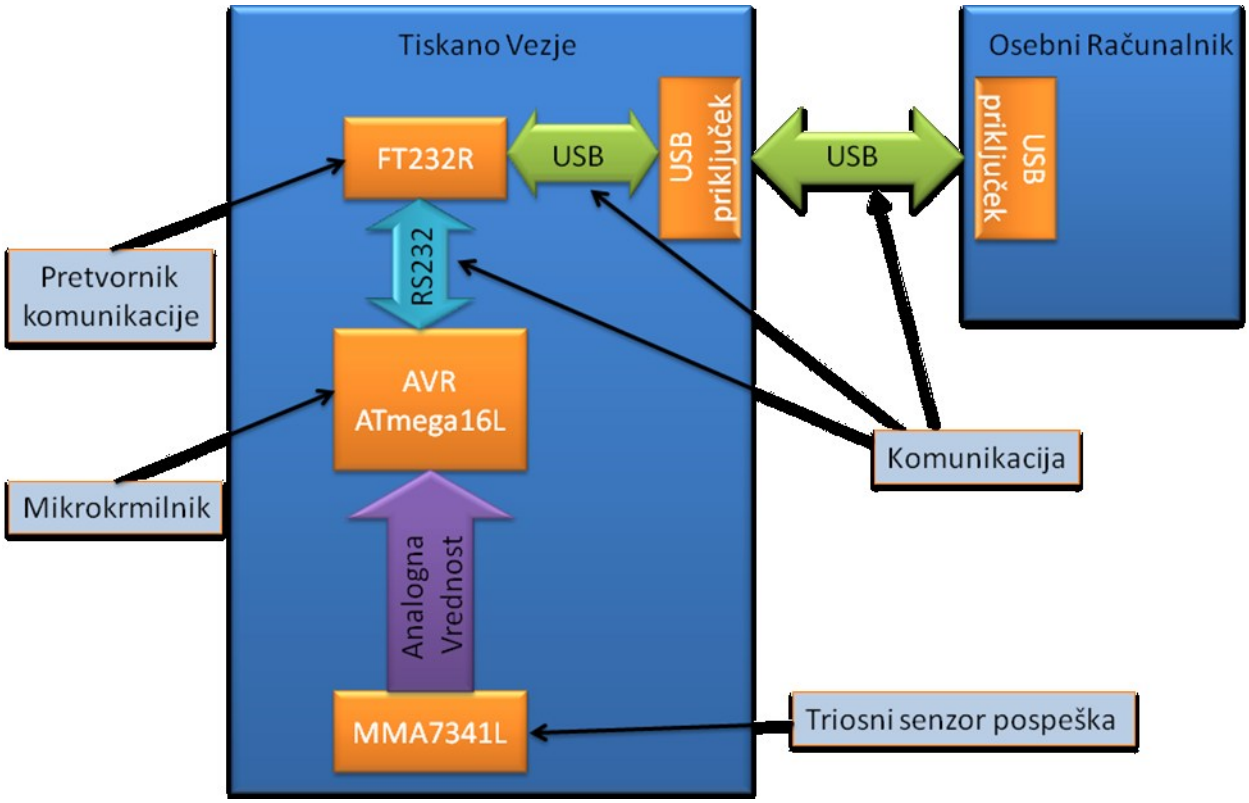
USB komunikacija je vedno uporabljena v kombinaciji z neko glavno ("master") enoto. V večini primerov je to računalnik, ta komunikacija se imenuje tudi "device to hoast or hoast to device", kar pomeni povezava z računalnikom in napravo oziroma naprava z računalnikom. Povezave med USB napravami ("pear to pear") ne obstajajo. Pretok podatkov v USB komunikaciji je po FIFO pomnilniku, kar pomeni prvi vhodni podatek bo tudi prvi izhodni podatek ("first in first out"). V osnovi komunikacija poteka tako, da računalnik pošilja SOF ("start od frame") – začetek prenosa na časovni interval ene milisekunde. Računalnik po določenem urniku pošilja in prejema zahtevane pakete, ki jih nato pomni. V enem časovnem intervalu paketa lahko preverimo podatke o video signalu spletne kamere (takšen prenos podatkov je prosto tekoč), pozicijo računalniške miške (pošiljanje podatkov le ob zunanji prekinitvi), kontrola oziroma manipulacija z neko napravo (prenos podatkov takrat, ko dobimo zahtevo od računalnika, pri katerem je čas za prenos pomemben), printerji (prenos podatkov z 100% zanesljivostjo, vendar čas za prenos ni tako pomemben).

Dekodiranje in enkodiranje podatkov pri USB komunikaciji je NZRI (Non Return to Zero Invert), kar pomeni kodiranje brez povratka na ničlo ta signal pa je tudi invertiran. To pomeni, da kodiranja linija spremeni stanje oziroma logični nivo vsakič, ko se na nekodirani liniji pojavi logična ničla. Pri NZR (Non Return to Zero) je kodiranje enako le da, kodirana linija spremeni stanje le takrat, ko je na nekodirani liniji logična enica.



Primer NRTZ kodiranja podatkov - Specifikacije USB komunikacije

7. Blok shema modula:



Blok shema modula

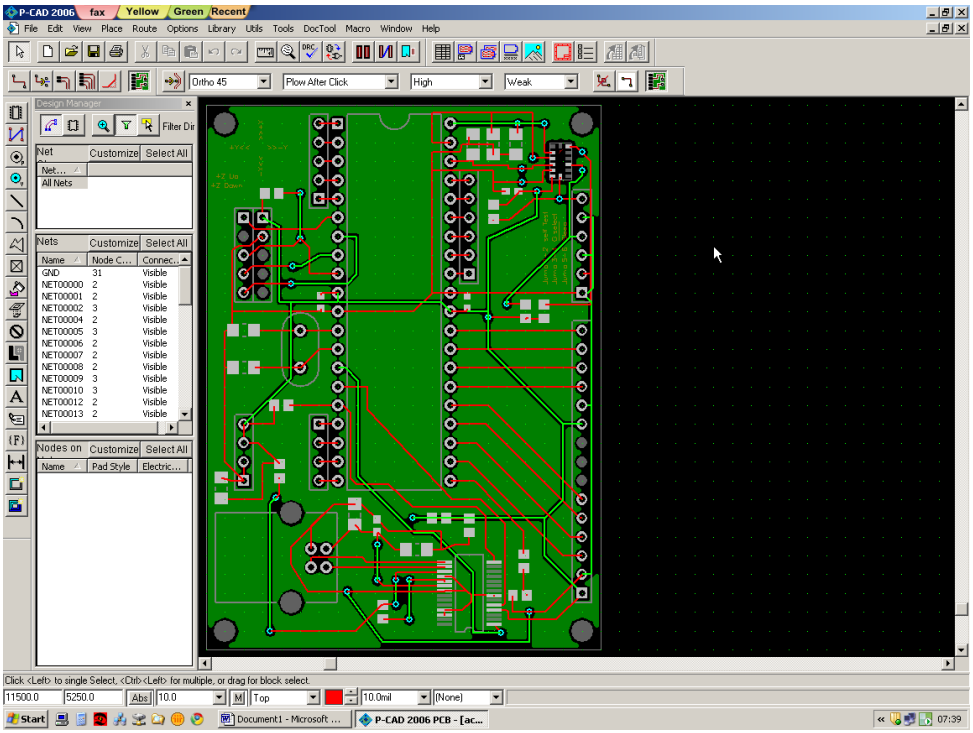
Kosovnica

Element	Vrednost	Ohišje
Pin1	5x	header
Pin2	5x	header
Pin3	4x	header
Pin4	4x	header
Pin5	5x	header
Pin6	6x	header
Pin7	15x	header
Pin8	6x	header
C1	10[µF]	SMD 1206
C2	0.1[µF]	SMD 1206

C3	0.1[μF]	SMD 1206
C4	4.7[μF]	SMD 1206
C5	100[nF]	SMD 0603
C6	3,3[nF]	SMD 1203
C7	3,3[nF]	SMD 1206
C8	3,3[nF]	SMD 1206
C9	0,1[μF]	SMD 0603
C10	0,1[μF]	SMD 0603
C11	27[pF]	SMD 1206
C12	27[pF]	SMD 1206
R1	10[kΩ]	SMD 0805
R2	10[kΩ]	SMD 0805
R3	10[kΩ]	SMD 0805
R4	270[Ω]	SMD 0805
R5	10[kΩ]	SMD 0805
R6	10[kΩ]	SMD 0805
R7	10[kΩ]	SMD 0805
R8	270[Ω]	SMD 0805
Kvartz	8[MHz]	HC-49/US
D1 (Schottky diode)	/	SMD 0805
D2 (LED)	/	SMD 0805
D3 (LED)	/	SMD 0805
MMA374L	/	LGA14
ATMega16L	/	PDIP40
FT232R	/	SSOP 28
KON1 (USB)	/	Type B

7. PCAD

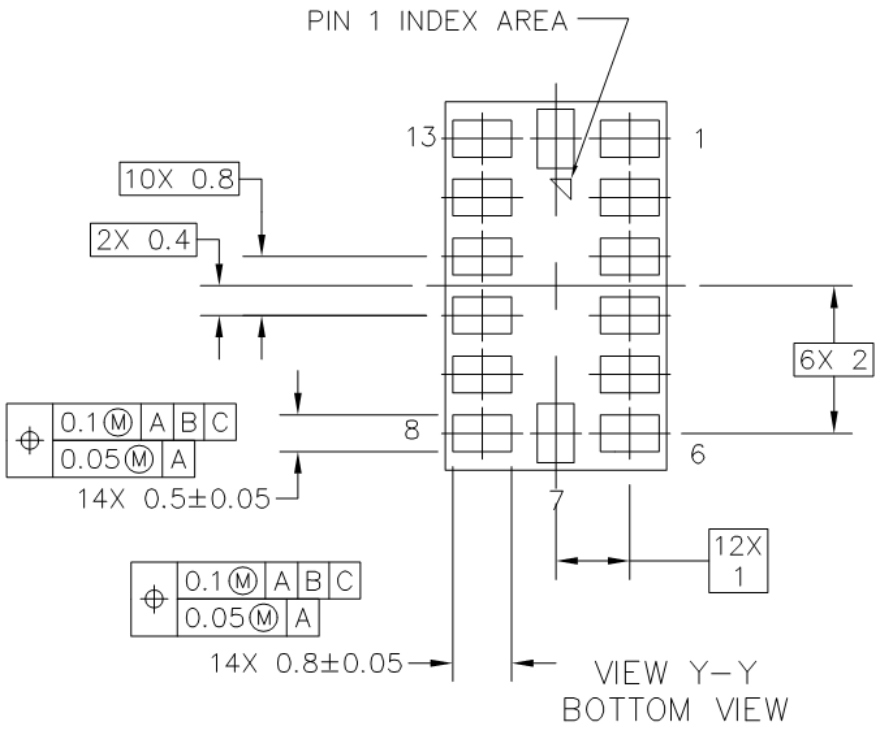
PCAD je eno izmed programskih paketov za dimenzioniranje elektronskih tiskanih vezij. Programski paket je sestavljen iz programov za povezavo simbolnih povezav (SCH), izdelave tiskanega vezja (PCB), knjižnico za urejanje elementov in drugih programov. Podatke o ohišju nekega elementa najdemo v originalnih dokumentacijah, vendar večinoma standardnih knjižnic že vsebuje standardne elektronske elemente.



Programsko orodje PCAD

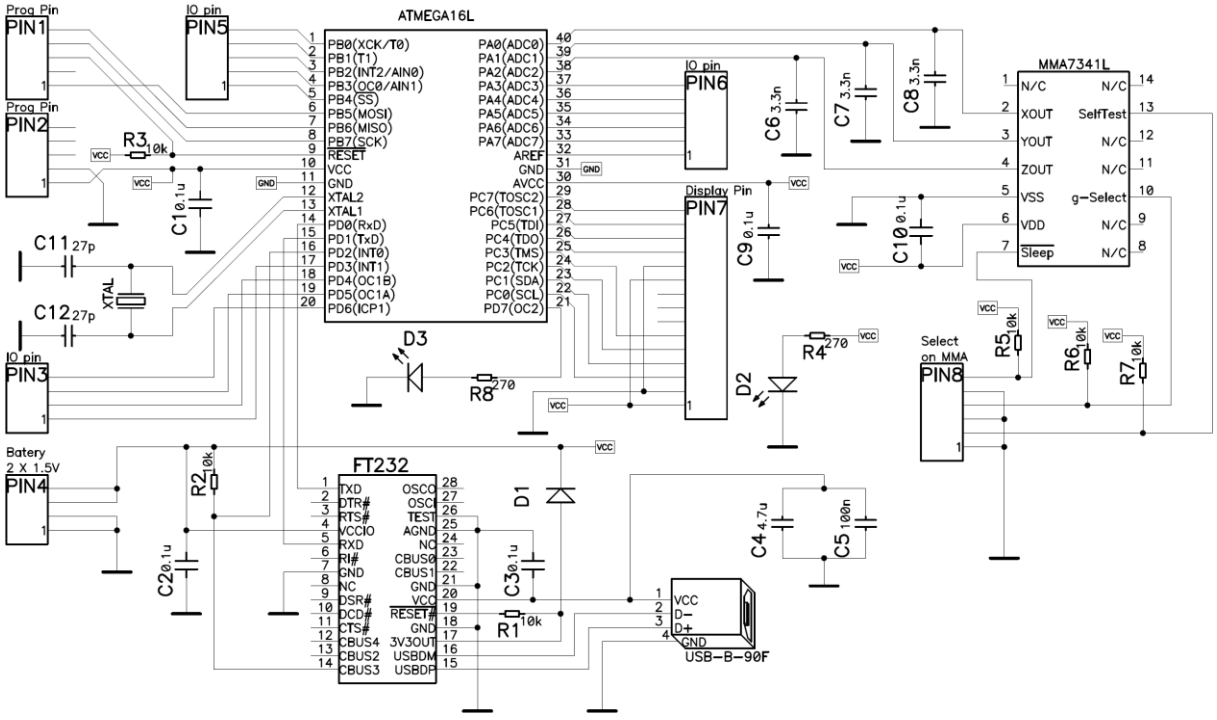
Shematske povezave kreiramo v programskem orodju Schematics (SCH). Povezujemo lahko elemente, ki jih najdemo v knjižnicah ali pa jih kreiramo sami z Library Executive. Posamezne elemente postavimo tako, da kliknemo na Place>part in nato izberemo simbol oziroma element, ki ga hočemo postaviti. Za vklop povezav pa izberemo Place>wire. Rotacijo ("rotation") elementov izvedemo s klikom na določen element, nato pritisnemo črko R. Zasuk ("flip") izvedemo podobno le, da pritisnemo črko L. Razlika med zasukom in rotacijo je ta, da pri rotaciji obračamo element za 90°, medtem, ko zasuk pomeni zrcaljenje elementa. Funkciji veljata enako pri simbolih (SCH) in vzorcih elementov (PCB). Raster oziroma mreža, v kateri

povezujemo elemente, je običajno colska, kar je enako sto milsov. Mils je enota za razdaljo, kar predstavlja 0,00254 cm. V specialistični nalogi sem uporabljal senzor pospeška MMA7341L, ki ima ohišje SSOP 28, površino 4 x 5 mm in 14 priključkov. Ko rišemo vzorec oziroma ohišje nekega elementa je zelo pomembno, da izberemo pravi raster mreže. Običajno si izberemo tak raster, ki najmanjši ali enak od najmanjše razdalje v vzorcu. Za SSOP 28 sem izbral raster 0.01 milsa, saj so v dokumentaciji najmanjši razdelki 0.04 milsa.

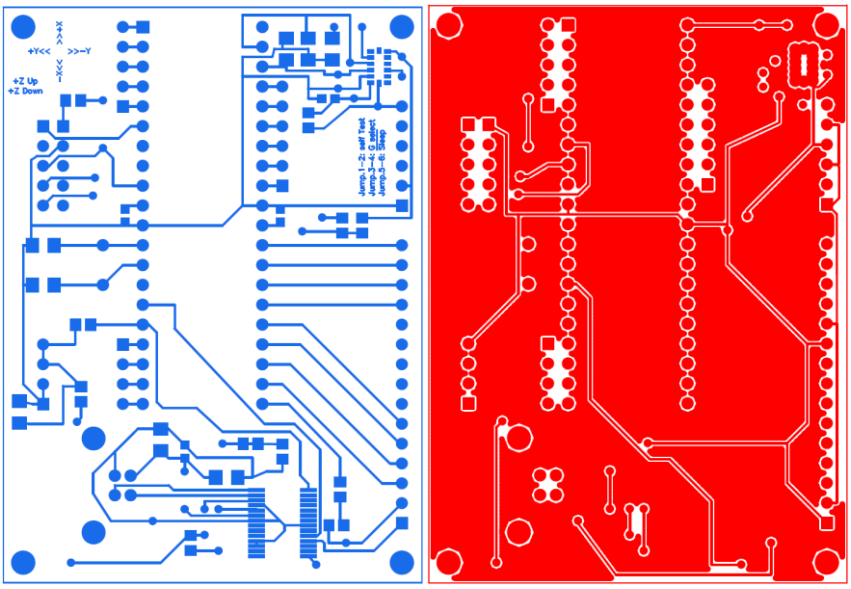


Dimenzije ohišja SSOP 28 - Dokumentacija MMA7341L pospeškomera

8. Shema vezja in tiskanega vezja



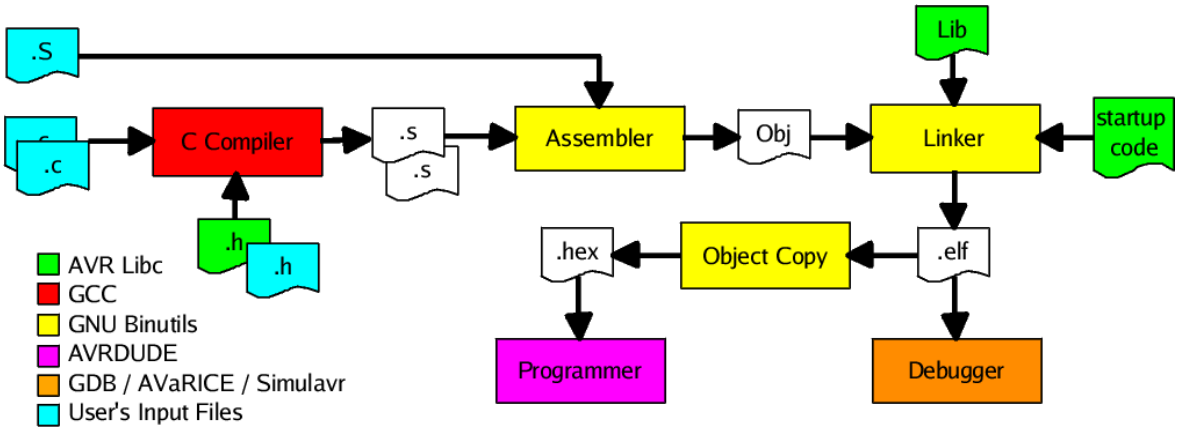
Shema vezja



Zgornja in spodnja plast tiskanega vezja

9. CodeVision:

CodeVision je prevajalnik, ki zna prevesti standarden C jezik v heksa kodo, ki jo pošljemo mikrokrmilniku. Med prevajanjem se C koda najprej pretvori v assemblersko kodo. Nato povezovalec ("linker") združi še ostale dele kode kot so razne knjižnice in začetne kode. Na koncu objekt za kopiranje ("object copy") kopira v heksa datoteko, ki jo nato pošljemo mikrokrmilniku. Prednost prevajalnika je tudi čarovnik in vse knjižnice, ki zajemajo veliko družino AVR mikrokrmilnikov. S čarovnikom lahko kreiramo neko osnovo za naš program oziroma čarovnik sam ustvari inicializacijsko kodo mikrokrmilnika.



Blokovna shema prevajalnika CodeVision

Večopravilnost:

Večopravilnost ("multitasking") je način programiranja mikrokrmilnikov v katerem pregledujemo zunanje signale na točno določeno časovno konstanto. Notranje prekinitve povzroča časovnik ("timer"), ki ima nalogo skočiti v rutino vsako časovno konstanto, ki mu jo določimo. V rutini časovnika določimo urnik ("Schedule"). Naloga urnika je skakati v rutine, ki so po vrsti določene in se ciklično izvajajo od prve do zadnje. Urnik lahko v C-kodi napišemo na več različnih načinov. V mojem primeru sem za urnik uporabil "case" strukturo in števec. Vsakič, ko se izvede prekinitev, se števec poveča za ena in tako kliče rutine, ki so definirani v "case" strukturi.

Večopravilnost pa zahteva tudi svoja pravila, ki se jih moramo držati. Najbolj pomembno pravilo je, da se morajo vse rutine končati pred iztekom do naslednje prekinitve. Razlog za to pravilo je preprost. V primeru, da se bi prekinitev izvedla pred iztekom rutine, se zaradi skoka ne bi izvedla vsa koda, ki je napisana v tej rutini. Drugo pravilo je, da morajo biti vse prekinitve enako dolge. To nam predvsem omogoči lažje računanje časovnih intervalov. In tretje pravilo je, da zunanje prekinitve niso možne. Če bi bile zunanje prekinitve možne, bi lahko podrli časovne konstante, ki smo jih definirali s časovnikom.

Če upoštevamo pravila in programiramo sistem kot večopravilnost nam ta nudi velike prednosti. Pregled nad posameznimi operacijami je veliko bolj pregleden, kot pa če bi spisali program le v enem kosu. Različne časovne zakasnitve lahko definiramo na enostaven način brez, da bi s tem podirali druge časovne zakasnitve v programu. In kar je najbolj pomembno, mikrokrmilnik se zaradi svoje hitrosti preleta programov obnaša kot, da bi podprograme izvajal paralelno čeprav jih izvaja serijsko.

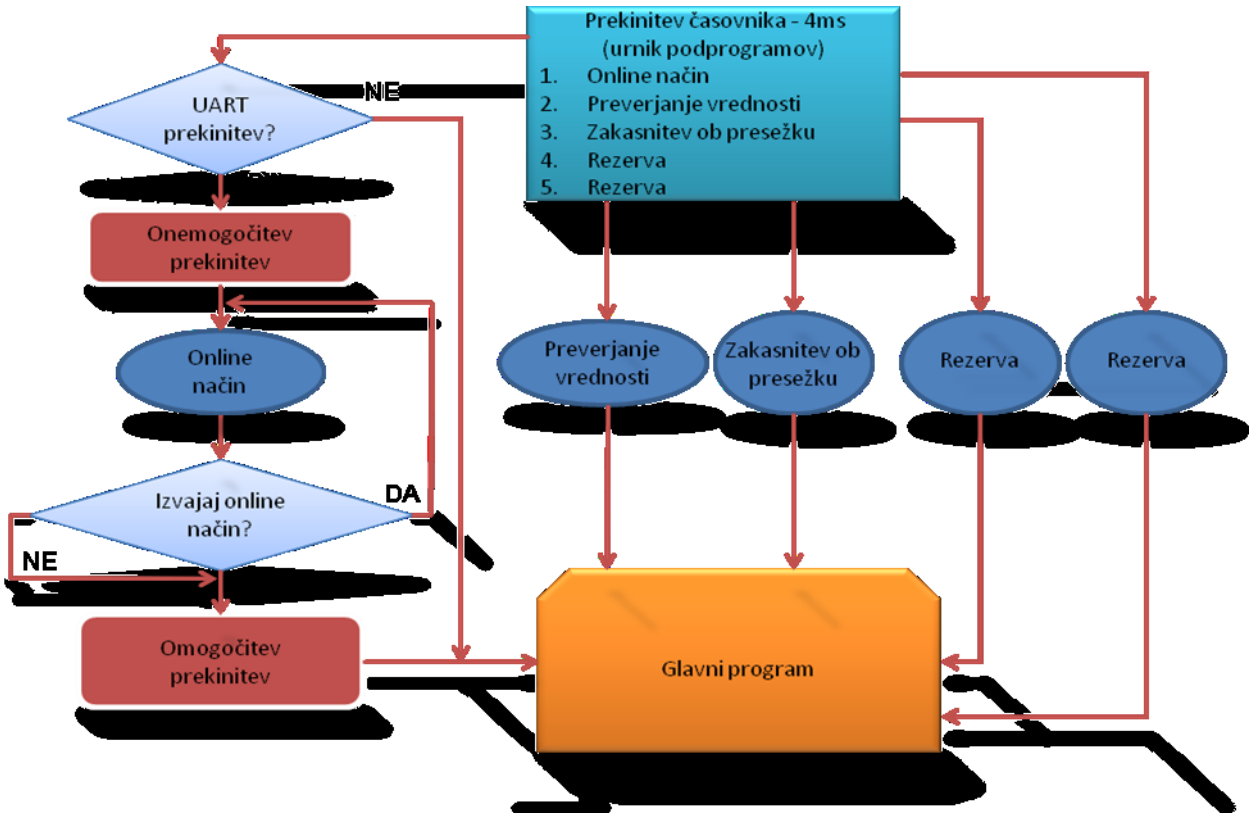
Glavni program:

V glavnem programu se običajno izvajajo tisti deli programa, ki ne vplivajo močno na delovanje oziroma odzivnost sistema, kot so na primer zapletene računske operacije ali pa masovna obdelovanje podatkov. V glavnem programu sem dodal analogno digitalno pretvorbo, saj se ta izvaja večinoma časa glede na to, da so podprogrami zelo kratki.

Potek in prekinitve programa:

Potek programa poteka tako, da program najprej preleti deklaracijo globalnih spremenljivk, nato preleti inicializacijo mikrokrmilnika. Začne se izvajati glavni program, ki se vrti v neskončni zanki. Po nekem določenem času se sproži notranja prekinitev v našem primeru časovnik, ki skoči na prvi podprogram. V primeru, da na UART pomnilniku čaka neka vrednost se začne izvajati "online" podprogram. To pomeni, da smo preko USB vodila povezani na modul in spremljamo trenutne podatke, ki jih nam mikrokrmilnik pošilja. V tem podprogramu tudi onemogočimo prekinitve, saj želimo le spremljati vrednosti ali pa nastavljati parametre modula. Ko končamo z "online" načinom, prekinitve spet omogočimo. V primeru, da UART prekinitev ne zazna nobenega poslanega znaka pomeni, da preskočimo "online"

način in se vrnemo v glavni program. Glavni program se zopet vrti v neskončni zanki. Ko pride druga prekinitev časovnika urnik kliče podprogram za preverjanje vrednosti. V primeru, da so te večje od nastavljenih, se aktivira določen izhod na mikrokrmilniku. Ko je rutina končana skočimo zopet v glavni program, ki se vrti v neskončni zanki. Nato se ob prekinitvi zgodi še zadnja rutina, ki nam zakasni pulz, če je bil ta prožen v drugi rutini. V zadnji rutini se števec za urnik podprogramov postavi na nič tako, da se ob naslednji prekinitvi zopet začne izvajati prvi podprogram. Časovnik je nastavljen na 125kHz, kar pomeni, da je en pulz časovnika enak 1/125000kHz = 0,000008s = 8us. Časovnik skoči v prekinitveno rutino ob vrednosti FFh, kar pomeni, da mora prišteti do 255. Iz tega sledi, da se vsaka rutina izvede 255 * 8us = 0,00204s = 2,04ms. Prelet celotnega urnika se izvede v časovni dobi 10,2ms, saj imamo pet podprogramov.



Potek prekinitev v programski kodi CodeVision

10. LabView programsko orodje

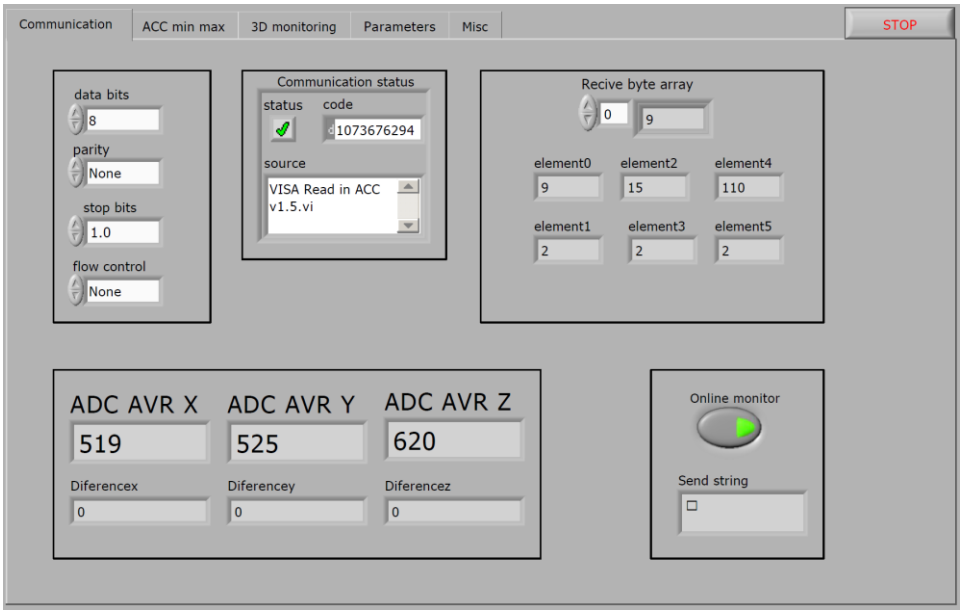
Splošno o LabView in osnovni koncept delovanja

LabView je zmogljivo programsko orodje za zajemanje in obdelavo električnih signalov proizvajalca National Instruments. Ker gre za okolje z grafičnim programiranjem, je potek programa pregleden in ne zahteva posebnega znanja o programiranju. Osnovna značilnost programa je ta, da izhod neke funkcije dobimo šele takrat, ko so na njo prišli vsi vhodi. Programsko orodje LabView je sestavljeno iz kontrolne plošče in blok diagrama. V kontrolni plošči se nam pojavljajo indikatorji in kontrole. S kontrolami lahko simuliramo signale, numerične vrednosti, digitalna stikala in druge kontrole. Indikatorji pa so namenjeni za opazovanje trenutnih (grafov, numeričnih vrednosti, logičnih indikatorjev) in tudi izmerjenih vrednosti (charts). Blok diagram nam predstavlja stikalno shemo programa. V njem povezujemo različne sklope, ki jih lahko po želji dodajamo in odstranjujemo. Na voljo so nam simulacije signalov, matematične operacije, operacije, ki se neposredno vežejo na operacijski sistem Windows (delo z datotekami), zajemanje podatkov (DAQ), merjenje in statistika signalov, FTT (hitra Fouriere-va transformacija), "for", "while" in časovne zanke, možnost uporabe osnov C jezika in še veliko drugih. Zelo uporaben pripomoček v LabView je osvetljevanje ali "highlight", kjer lahko spremljamo grafičen potek programa [4]. Ekspres funkcije so funkcije, ki ne potrebujejo dodatnih parametrov za delovanje in so namenjene hitremu in nezahtevnemu sestavljanju programa. Slabost teh funkcij je, da smo omejeni z njihovimi opcijami. Ostale funkcije zahtevajo več vpisovanja in definiranja vhodnih parametrov, vendar imamo zato veliko več opcij pri oblikovanju izhodnih signalov. Express funkcije uporabimo tam, kjer ne vidimo potrebe po uporabi eksaktnih funkcij, primerne pa so predvsem za hitro in enostavno grajenje programa [4].

Opis kontrolne plošče

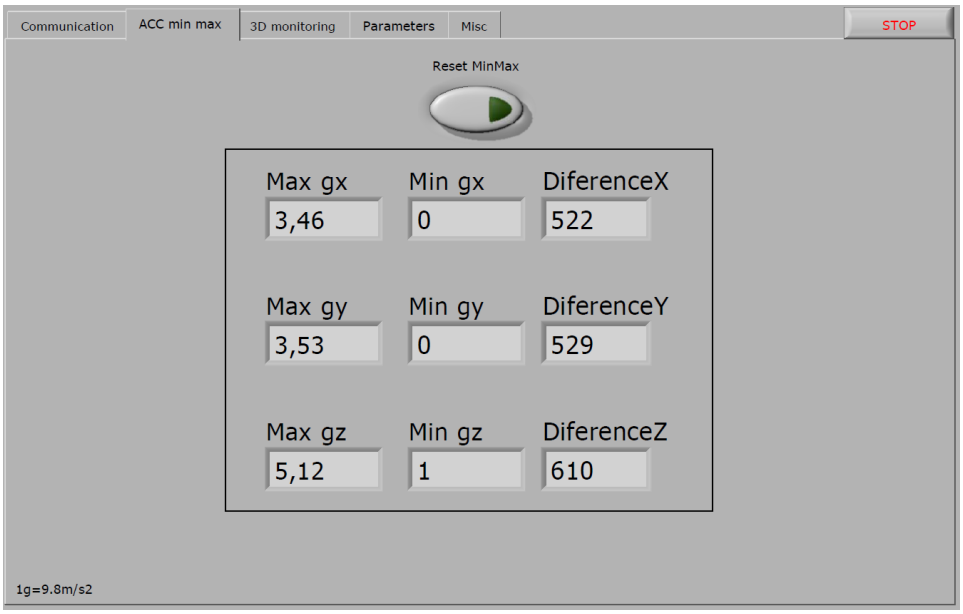
Kontrolno ploščo sestavljajo grafi, charti, kontrole, indikatorji, 3D objekti in drugi grafični pripomočki. Kontrolna plošča je neposredno povezana z blok diagramom. To lahko opazimo tako, da ko dodamo nek element v kontrolni plošči, se ta avtomatsko pojavi tudi v blok diagramu in obratno. Bistvo kontrolne plošče je torej pregled podatkov in kontrola programa. Če imamo v kontrolni plošči veliko podatkov, ki jih moramo spremljati, lahko v njo dodamo "Tab control". Ta omogoča preskoke med različnimi zavihki, ki jih lahko zapolnimo z različnimi kontrolami in indikatorji.

V prvem zavihku ("Communication" – slika 3.16) nastavljamo parametre serijske komunikacije in spremljamo pretok podatkov, ki jih serijska komunikacija posreduje. S tipko na "Online monitor" omogočimo pretok podatkov s serijsko komunikacijo rs232.



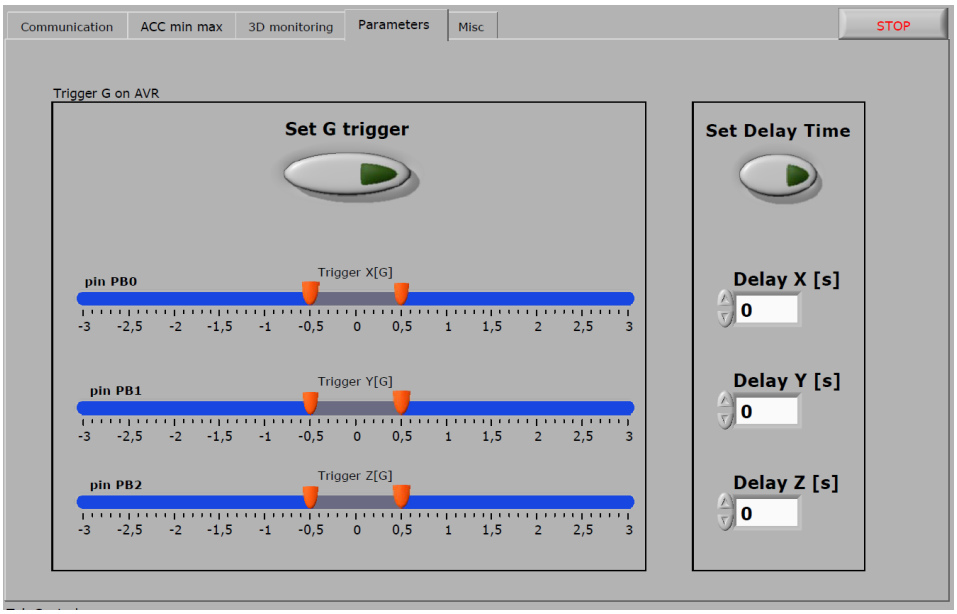
Kontrolna plošča LabVIEW - "Communication" zavihek

Drugi zavihek ("ACC min max" – slika 3.17) je spremljanje minimalnega in maksimalnega pospeška, ki je bil kadarkoli dosežen od trenutka, ko se je začela izvajati serijska komunikacija. Te vrednosti lahko tudi pobrišemo s tipko na "Reset MinMax".



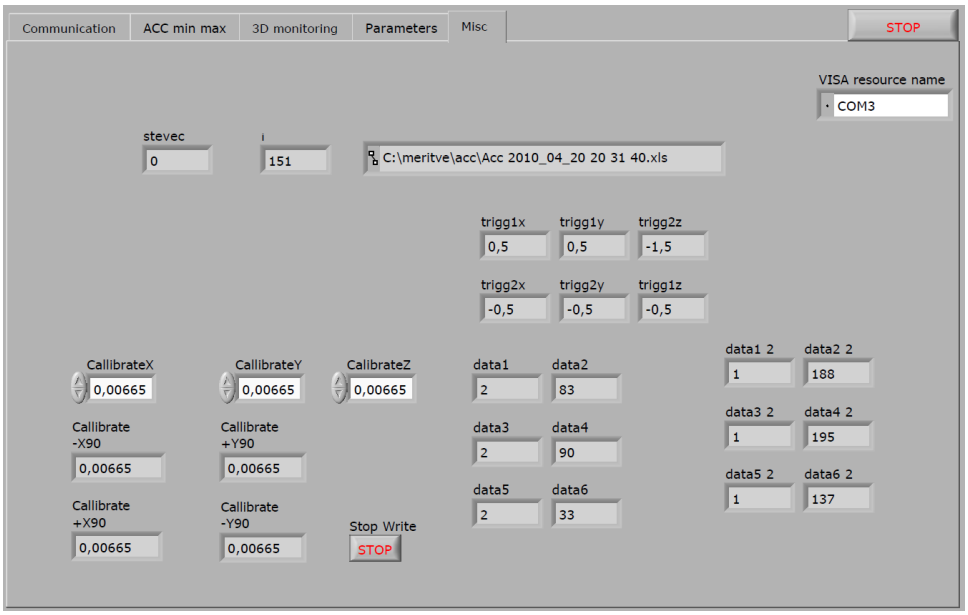
Kontrolna plošča LabVIEW - "ACC min max" zavihek

Tretji zavihek ("3D monitoring" – slika 3.20) je najbolj pomemben, saj vsebuje kalibracijo sistema, vrednosti pospeškov v vseh treh smereh in 3D pregled objekta. V naslednjem zavihku ("Parameters" – slika 3.18) določamo parametre mejnih vrednosti in zakasnitve po doseženih mejnih vrednostih. Parametre enostavno vpišemo tako, da jih z drstnikom nastavimo na želene vrednosti (pri tem si lahko pomagamo z "3D monitoring" zavihkom) in pritisnemo na kontrolo "Set g trigger". Enako naredimo s parametrom zakasnitve po doseženi mejni vrednosti. Nastavimo parameter na želeno vrednost zakasnitve in kliknemo na kontrolo "Set Delay Time".



Kontrolna plošča LabVIEW – "Parameters" zavihek

Zadnji parameter ("Misc" – slika 3.19) je namenjen kontrolam in indikatorjem, ki za uporabnika niso pomembni, pomembno pa je, da so urejeni v nekem skupnem zavihku.



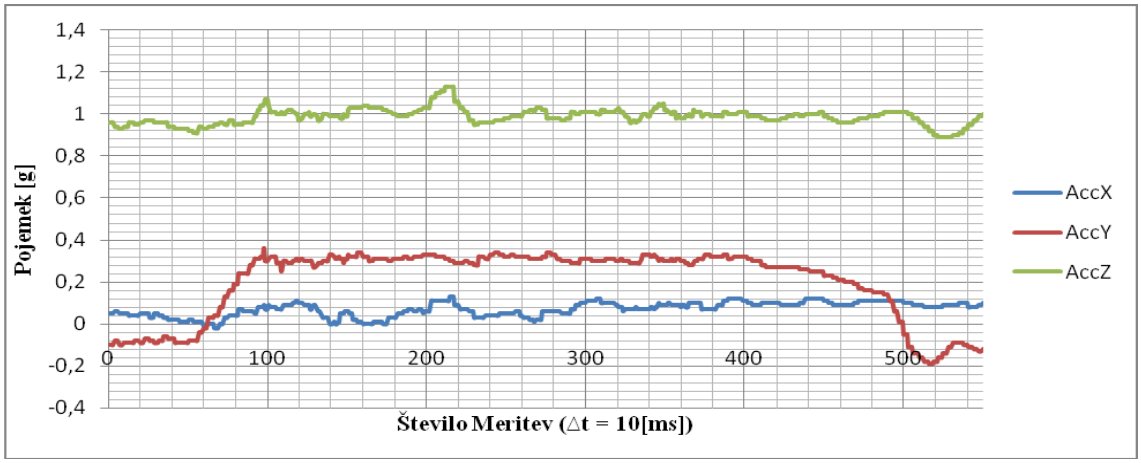
Kontrolna plošča LabVIEW – "Misc" zavihek

11.Merjenje pojemka pri intenzivnem zaviranju

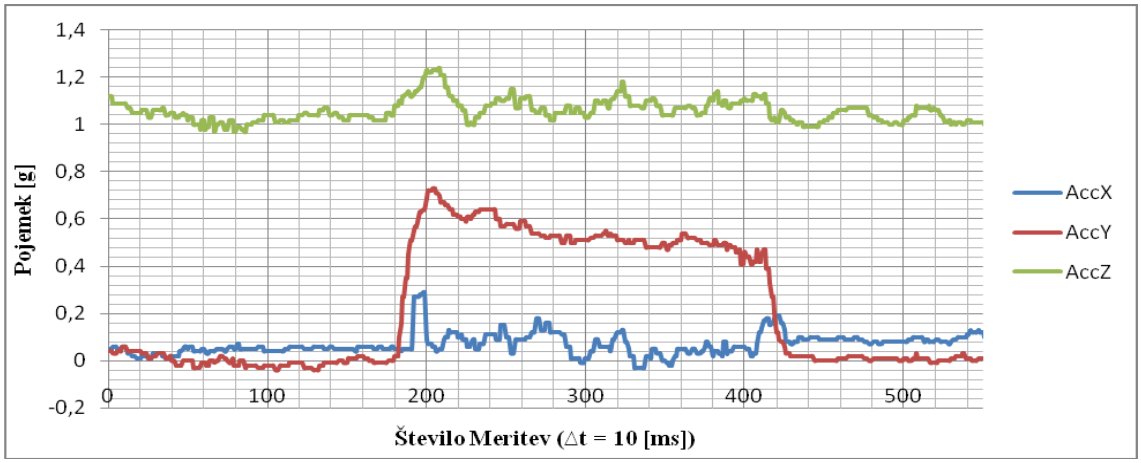
Pri merjenju pospeška, oziroma pojemka, je pomembno, da elektronski modul trdno pritrdimo na podlago predmeta, nad katerem bomo izvajali meritev. V primeru zdrsa elektronskega modula merjeni pospešek ne bo ekvivalenten realnemu pospešku merjenega predmeta, saj bo ob trenutku zdrsa prišlo do različnih premikov med površino predmeta in elektronskim modulom.

Rezultati meritev in postavitev meje za proženje

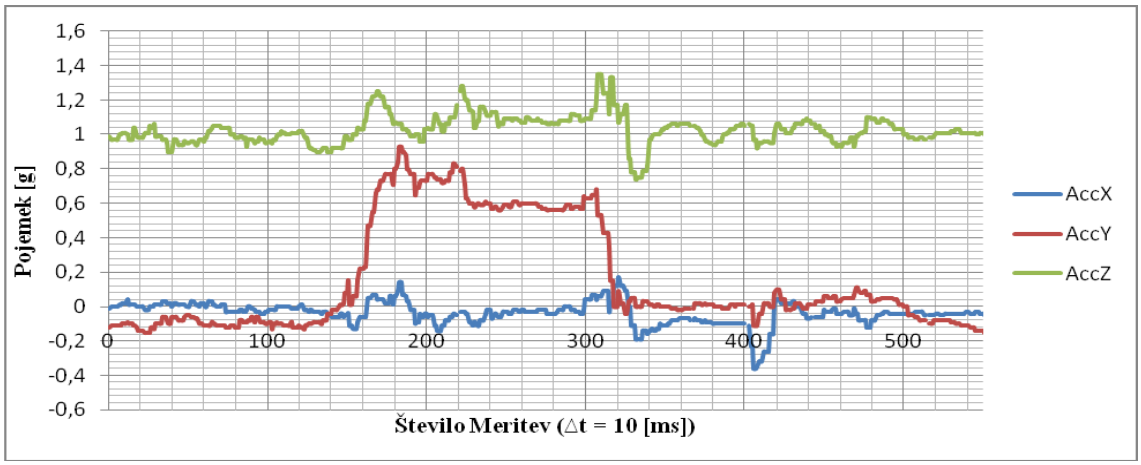
Meritve sem izvajal pri zaviranju iz konstantne hitrosti 70 km/h na 0 km/h. Modul sem imel orientiran tako, da je bila smer Y osi enaka smeri vožnje. Več meritev je pokazalo, da imamo opravka z maksimalnim pojemkom do enega g-ja. Modul, s katerim sem meril, je bil trdno pritrjen na armaturno ploščo avtomobila, saj bi v primeru zdrsov prišlo do nepravilnih meritev. Vrednost merjenja je bila tudi odvisna od vremenskih razmer. V deževnih pogojih lahko pride do zdrsa gume in s tem do manjših pojemkov kot pri stiku gume s suho podlago. Iz grafov na slikah 4.1, 4.2 in 4.3 je razvidno, da večji kot je pojemek pri zaviranju hitreje se bomo ustavili. Meritve so tudi pokazale, da je smiselno nastaviti mejo proženja na približno 0,8g.



Slika 4.1 Meritve pojemka pri počasnem zaviranju



Slika 4.2 Meritve pojemka pri srednje hitrem zaviranju



Slika 4.3 Meritve pojemka pri hitrem zaviranju

13. Časovna in finančna reakpitulacija:

Časovno sem za projekt potreboval 4 mesece. V prvi fazi sem se najprej osredotočil na idejo projekta, nato sem začel iskati primerne elemente, ki bojo služile kot celota oziroma samostojni modul. Za to fazo sem potreboval mesec dni. Po izbiri elementov sem začel z izdelavo PCB-ja oziroma tiskanega vezja za katerega sem približno poteboval dodaten mesec dni. Po izdelavi tiskanega vezja in dobavo elektronskih elementov sem začel z spajkanjem elementov na tiskano vezje. Ta faza me je časovno ni obremenila, finančno pa najbolj. Za spajanje elementov sem investiral v SMD spajkalnik na vroči zrak, ki me je finančno stal 150€, spajkalne paste, cin, žica za odspajkavo 60€, ostali elementi 20€ in tiskano vezje, ki ga je financerala fakluteta za elektrotehniko 100€ (deset obojestranskih

tiskanih vezji). V zadnji fazi je sledilo programiranje mikrokrmilnika in LabVIEW programa, kar mi je vzelo največ časa (dva meseca).

Modul deluje v skladu z pričakovanji vendar se ga bo v prihodnosti dalo tudi razširiti. Po končanem projektu sem zajel veliko programskega znanja (programiranju v realnem času, LabView grafičnim programiranju) kot tudi praktičnega znanja (spajkanja SMD komponent in spajkanja integriranih vezji, ki imajo relativno majhno razdaljo med priključnimi "nogicami").

Literatura

[1] *Brian Green: Tkanina Vesloja, New York, 2004 (Gravitacija in večno vprašanje, Enakost gravitacije in pospeševanja)*

[2] *http://photos.autoexpress.co.uk/images/front_picture_library_UK/dir_624/car_photo_312310_25.jpg (Slika avtomobila) [20.4.2010]*

[3] *http://dankuchma.com/cee498/presentations/ECE%20S07%20Accelerometer%20Types.ppt (Izvedbe pospeškomerov) [20.4.2010]*

[4] *http://mafija.fmf.uni-lj.si/seminar/files/2007_2008/MEMS_accelerometers-koncna.pdf (MEMS pospeškomeri) [20.4.2010]*

[5] *http://www.freescale.com/files/sensors/doc/data_sheet/MMA7341L.pdf*

(Dokumentacija MMA7341L pospeškomera) [20.4.2010]

[6] *http://upload.wikimedia.org/wikipedia/commons/b/b0/Rs232_oscilloscope_trace.svg (Časovni potek RS232 komunikacijskega protokola) [20.4.2010]*

[7] *http://www.faculty.iu-bremen.de/birk/lectures/PC101-2003/14usb/FINAL%20VERSION/usb_protocol.html (Specifikacije USB komunikacije) [20.4.2010]*

[8] *http://www.soc-machines.com/pdfs/Atmega16.pdf*

(Dokumentacija mikokontrolerja ATmega16L) [20.4.2010]

[9] *http://www.elektroda.pl/rtvforum/files-rtvforum/stk200_551.jpg (Shema programatorja STK 200) [20.4.2010]*

[10] *http://www.avrfreaks.net/wiki/images/Compilerchain.png (Blokovna shema prevajalnika CodeVision) [20.4.2010]*

[11] *Brian W. Kernighan, Dennis M. Ritchie: Programski jezik C, Ljubljana, 1991 (Switch)*

[12] *Tadej Tuma, Andrej Nussdorfer: Mikrokrmilniški sistemi, Ljubljana, 2004 (Časovno rezinjenje)*

[13] *Richard Barnett, Larry O'cull, Sarah Cox: Embedded C Proggraming and the Atmel AVR, Canada, 2003 (Codwizardavr code generator)*

[14] *Rick Bitter, Taqi Mohiuddin, Matt Nawrocki: LabVIEW advanced programming techniques, Florida, 2007 (Virtual Instruments)*