Факултет техничких наука
Департман за индустријско инжењерство и менаџмент
Студијски програм – Мехатроника, роботика и аутоматизација
Школска година: 2014/2015

Дигитална управљачка електроника

Пројекат реализовали: Маја Хаџиселимовић X1 15/2014

Лазар Живковић X1 14/2014

Марко Тиквић X1 10/2014

Ментори: проф. др Милош Живанов

доц. Владимир Рајс

Нови Сад 2015.

Садржај

1 Увод

Током основних и мастер академских студија студенти са смера мехатроника се сусрећу са проблемима где је понекад неопходна употреба микроконтролера, а понекад у многоме може олакшати решавање истог. Један од облика са којима су студенти упознати са микроконтролерима јесу тзв. развојна окружења, која се састоје од микроконтролера и пратеће електронике која је неопходна за његов исправан рад и омогућава брз и олакшан развој *embedded* апликација (уређаји који привидно обављају рачунарску логику, али не садрже рачунар у себи, нити њихов рад зависи од једног). Студенти су до сада већ имали прилику да се сретну са развојним окружењима произвођача Микроелектроника [1]. Поред Микроелектронике на располагању су развојна окружења произвођача Ардуино (енг. *Arduino*) [2], Тексас Инструментс (енг. *Texas Instruments*) [3] и других. Овај пројекат је настао са намером да се студентима приближи основни принцип рада ових окружења и дизајн и употребу софтвера везаног за њих, како би се створила основа за развој сопствених апликација, па чак и сопствених развојних окружења. Компоненте употребљене у пројекту су релативно лако доступне, тако да студенти не би требали да имају већих проблема уколико се одлуче да реплицирају пројекат описан у наставку документа.

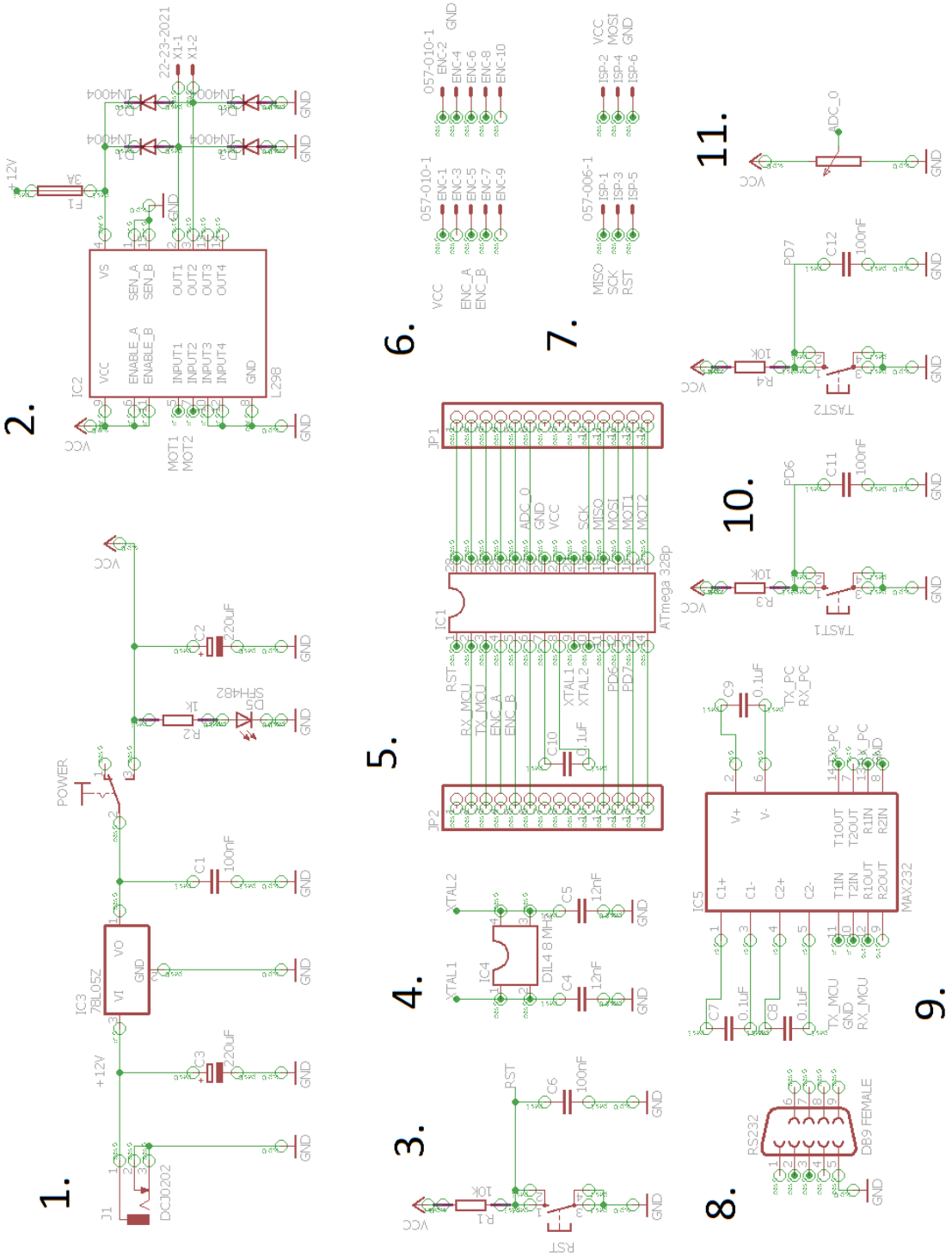
За реализацију пројекта употребљени су:

- Atmel Studio 6.2 – интегрисано развојно окружење (енг. *IDE – Integrated Development Enviroment*) и AVR-GCC компајлер за реализацију софтверског кода, написаног у C (/*це/*) програмском језику. [4] [5]
- CadSoft EAGLE - графичко окружење за пројектовање схеме система [6]
- ZeptoProgII – ISP (енг. *In-circuit Serial Programming*) програматор за ажурирање софтвера на микроконтролеру. [7]

Напомена: сви примери описани у даљем тексту се налазе на пратећем меморијском медијуму у тренутку предаје пројекта и налазе се на следећем гитхаб (енг. *github*) репозиторијуму одакле могу бити преузети у било ком тренутку: www.github.com/markotikvic/avr-dev-board/.

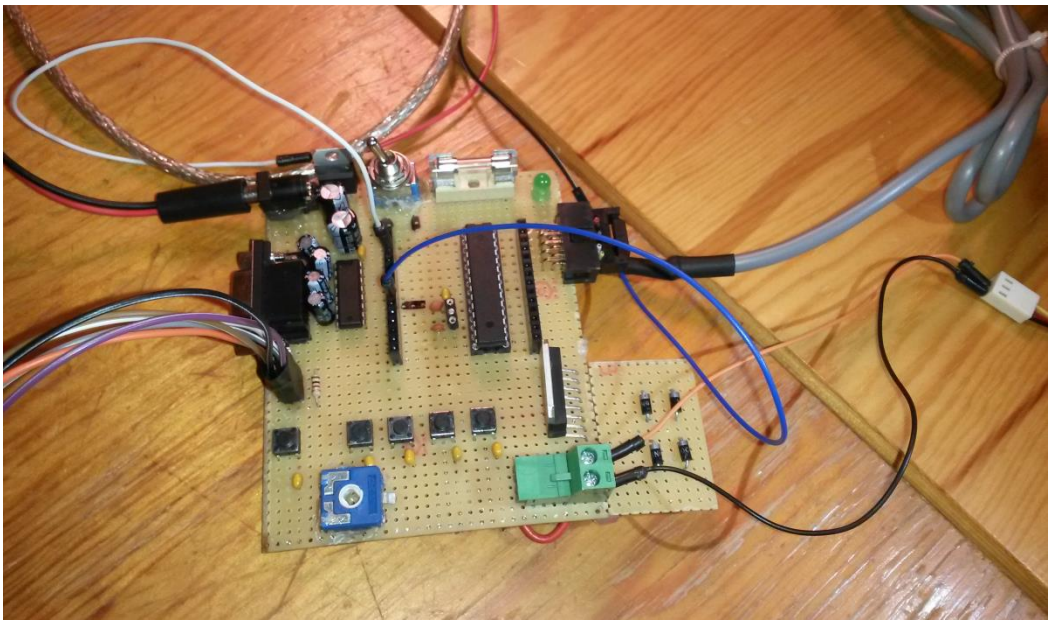
2 Структура система

На слици 2.1 приказана је схема система пројектована у CadSoft EAGLE софтверу.



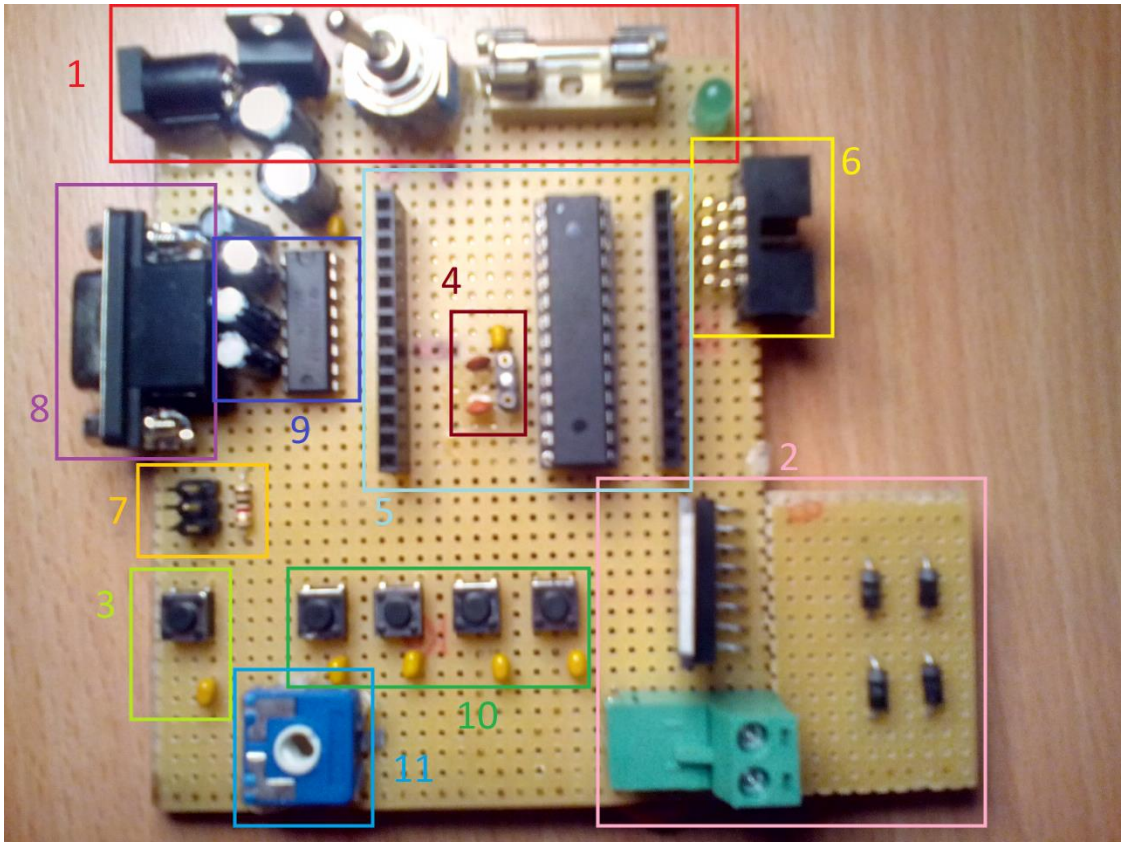
Слика 2.3.1.1 Схема система

Слика 2.2 представља физичку реализацију система описаног схемом 2.1.



Слика 2.2 Изглед система

На слици 2.3 је приказана модуларна подела система која одговара подели система на слици 2.1.



Слика 2.3 Распоред компоненти на реалном систему

3 Опис појединачних модула

Ова глава документа служи за опис појединачних модула/компоненти означених на слици 2.3. Модул означен бројем *n* на слици је описан поглављем *3.n.*

3.1 Напајачко коло

Напајачко коло састоји се од следећих компоненти:

- L7805 стабилизатора напона
- два кондензатора капацитивности 220 uF
- кондензатора капацитивности 100nF
- двоположајног прекидача
- осигурача
- отпорника од 1kΩ
- LED
- женски DC конектор

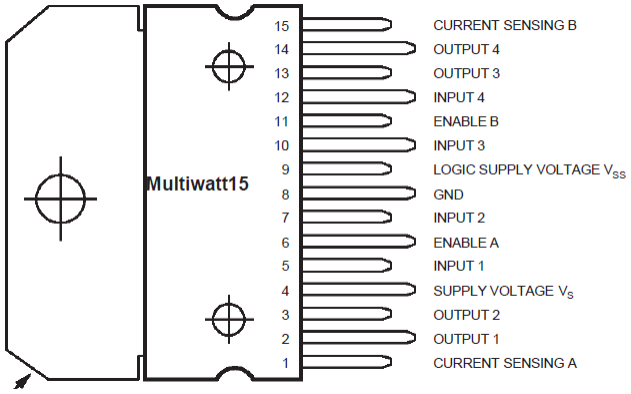
Основа напајачког кола је L7805 стабилизатор напона [8] чија је улога да пружи стабилан напонски ниво од 5 V за напајање микроконтролера, оптичког енкодера, MAX232 интегрисаног кола [9] и ISP програматора. Стабилизатор ради на принципу снижавања напонског нивоа доведеног на улаз и пружа стабилан напонски ниво од 5 V на излазу. Што је већа разлика напонских нивоа између улаза и злаза то су топлотни губици на стабилизатору већи. У нашем случају на улаз стаилизатора је доведен напон од 12 V. Један кондензатор од 220uF служи за отклањање шума на улазном сигналу, док кондензатор од 100nF служи за отклањање шума на излазном сигналу. Када је прекидач у затвореном положају, филтрацију шума на излазном сигналу стабилизатора врше кондензатори од 100nF и 220uF у паралелној спрези. Када је прекидач у отвореном положају, напон са излаза стабилизатора се не спроводи до пина контролера означеног са V_{cc}. Улога осигурача је да заштити коло од прекомерне струје тј. уколико потрошња кола пређе границу од 500 mA осигурач ће одреаговати и прекинути довод електричне енергије до остатка кола. Улога отпорника и светлосне диоде је да кориснику пружи информацију о томе да ли је коло под напоном. Преко DC конектора се на улаз стабилизатора доводи напон од 5 – 35 V.

3.2 H-мост

Улога H-моста је да омогући довољну струју за покретање већих потрошача као што су на примјер DC мотори, степ мотори и серво актуатори. H-мост употребљен у пројекту је L298N [10]. За исправан рад H-моста портебне су још 4 заштитне диоде и кондензатор од 100nF. L298N је интергисано коло које се састоји из два H-моста, од којих се сваки може користи засебно. Употреба H-моста се обавља на следећи начин (референце у наредном пасусу се односе на слику 3.2.1):

На пин број 4 интегрисаног кола са слике се доводи напонски сигнал за напајање потрошача које коло треба да напаја. Овај напонски ниво не мора бити исти

као напон којим се напаја остатак кола. У даљем тексту ће се овај напонски ниво ословљавати као ''напон потрошача''. Заправо, ни напонски извор не мора бити заједничи што је погодно у ситуацијама које захтевају напајање потрошача чија је потрошња већа од максималне дозвољене струје која сме да протиче кроз остатак кола. Да би смо могли да користимо један од мостова унутар кола (назовимо их А и В) прво морамо да доведемо позитиван напонски ниво на пин 6, уколико желимо да користимо мост А, односно на пин 11 уколико желимо да користимо мост В. Овај напонски ниво мора бити исти (или већи) од напонског нивоа доведеног на пин 9. Пин 9 кола служи за довођење референтног напона ''логике'' кола и у даљем тексту ће се ословљавати као ''логички напон''. Након укључења једног од мостова (у нашем случају је то мост А) управљањем напонским нивоима на пиновима 5 и 7 заправо управљамо напонским нивоима на пиновима 2 и 3 респективно. На пример, уколико микроконтролер постави пин 5 на ниво логичког напона до пина 2 ће се спровести напон потрошача. Наравно, исто важи и за пинове 7 и 3. Улога диода је заштите коло и потошач од пробојног напона, док кондезатор има улогу филтритрања шума на напонском сигналу који доводимо до пина 4.



Слика 3.2.1 Скица погледа одозго L298N интегрисаног кола

3.3 Ресет коло

Ресет коло се састоји од:

- отпорника од 10kΩ
- кондензатора капацитивности 100nF
- тастера

Служи за ресетовање рада микроконтролера. Треба напоменти да је добро пројектовано и реализовано ресет коло неопходно за исправан рад контролера. Притиском тастера, струја која иначе напаја контролер се преко отпорника одводи на масу чиме се онеспособљава рад микроконтролера. Након отпуштања тастера, контролер почиње да извршава код од почетка.

3.4 Спољашњи осцилатор

Уколико апликација изискује већу прецизност основног такта рада микроконтролера корисник може употребити кристални (или керамички) осцилатор. Неопходно је повезати изводе осцилатора са кондензаторима капацитивности 12nF да би осцилатор почео да осцилује. У свим приложеним примерима у овом пројекту употребљен је унутрашњи RC осцилатор учестаности 8 MHz.

3.5 Микроконтролер

Интегрисано коло које чини основу развојног окружења. У пројекту је употребљен миктроконтролер ATmega328P [11] компаније Атмел. На окружењу се налази и пар женских лествица које омогућавају лакши приступ пиновима контролера. Микроконтролер је изузетно погодан за овакве пројекте јер га је могуће ''програмирати''. То значи да се кориснику пружа могућност да рачунарски код (алгоритам апликације), написан у жељеном програмском језику (у нашем случају је то C програмски језик), изврши на контролеру. На високом нивоу апстракције, све што контролер ради је да доводи своје пинове у једно од два могућа стања, а то су стање логичке јединице тј. висок напонски ниво и стране логичке нуле тј. низак напонски ниво и врши очитавање вредности са појединачних пинова. Дакле, све што алгоритам апликације ради је подизање и спуштање напонских нивоа на пиновима контролера у одређеној секвенци. Управо та секвенца, у комбинацији са физичком повезаношћу контролера и осталих компоненти на окреужењу, одређује рад тј. понашање тих компоненти. Тако управљамо L298N колом (а индиректно и радом потрошача који је повезан са колом), вршимо комуникацију са рачунаром (преко MAX232 интегрисаног кола), одређујемо позицију енкодера и сл.

3.6 Конектор за енкодер

Дворедни десетопински оријентисани конектор. Конектор служи као интерфејс за ротациони енкодер ISC3806 [12].

3.7 ISP програматор

Конектор за ISP (In-circuit Serial Programming) програматор за ажурирање софтвера унутар микроконтролера. Приликом повезивања програматора на контролер од велике је важности да се сви пинови програматора и контролера поклапају. То значи да MISO (енг. *Master Input Slave Output*), MOSI (енг. *Master Output Slave Input*), RST (енг. *reset*), VCC (напон напајања) и SCK (енг. *Slave Clock*) пинови на програматори морају да се повежу са истим тим пиновима на контролеру. Уколико претходна ставка није задовољена није могуће успешно програмирати контролер, а у неким ситуацијама га је чак и могуће оштети.

3.8 Модул за RS232 комуникацију

Основа модула је MAX232 интегрисано коло (означено вројем 9 на слици 2.3) које врши „превођење“ TTL напонских нивоа на RS232 напонски ниво. Модул је

потребан за остваривање серијске комуникације између развојне плоче и рачунара. Уколико рачунар не садржи серијски порт (већина модерних лап-топова) неопходно је користити кабл који у себи садржи FTDI интегрисано коло, како би се комуникација остварила преко USB порта. За исправан рад кола неопходно је додати четири кондензатора чије су спецификације дате схемом на слици 2.1. Поред наведених компоненти модул се састоји још и из женског DB9 конктора.

3.9 Тастери

Од четири тастера који се налазе на плочи, 2 су повезана на пинове контролерра у могу се употребити за развој даљих апликација док су 2 остављена неповезана за будући развој плоче. Између тастера и пинова контролера додати су кондезатори од 100nF у циљу хардверског *debouncing-a* како би се компензовала мана физичке реализације тастера (понекад се може десити да на један притисак тастера дође до појаве више сигнала, што може проузорковати нежељено понашање апликације). Треба напоменути да је поред хардверске реализације пожељно урадити и софвтверску имплементацију *debouncing-a.*

3.10 Потенциометар

Улога потенциометра је да пружи могућност тестирања АД конверзије микроконтролера. Могуће је измерити напонске нивое у опсегу од 0 – AVcc V, где је вредност AVcc једнака напону напајања контролера подељеном са 2.

4 Примери рада система

Ова галва служи за опис употребе појединачних модула система као и да пружи увид у алгоритам рада апликација имплементираних у приложеним примерима.

4.1 АД конверзија

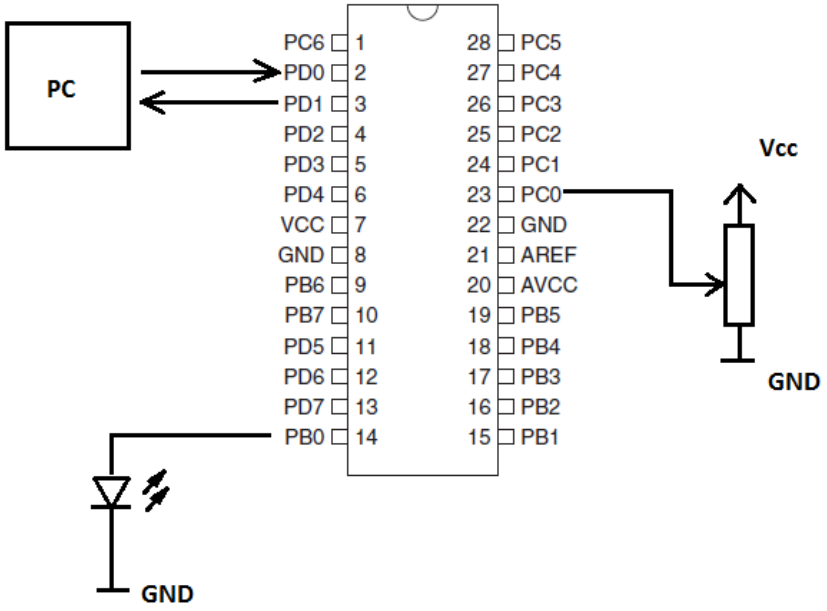
Као извор аналогног сигнала је искоришћен потенциометар (модул 10 са слике 2.3), који је повезан на одређени пин контролера. Код је реализован тако да се сваких 50 ms очитава вредност из АД конвертора и шаље преко серијске комуникацјие на рачунар, у људски читљивом формату. Контролер упоређује вредност сигнала на излазу потенциометра са унутрашњом референцом максималне вредности аналогног сигнала. Резултат конверзије се увек налази у опсегу 0 – 2^{10} (1024).

Псеудо код:

```
//glavna petlja
inicijalizuj tajmer
inicijalizuj uart
inicijalizuj AD konveriziju
_adc_prag = 300 /*predefinisana vrednost od interesa; ~1/3 napona napajanja*/
while(true)       /*uslov beskonačne petlje*/
```

započni AD konverziju
pošalji rezultat konverzije na serijski port
if*(rezultat konverzije > adc_prag)* */*ukoliko trenutna vrednost napona*/*
upali diodu */*na potenciometru iznosi ~1/3 napona*/*
else */*napona napajanja, upali diodu.*/*
ugasi diodu */*u suptrotnom je ugasi*/*
čekaj 50 ms

Скица повезивања дата је сликом 4.1.2.



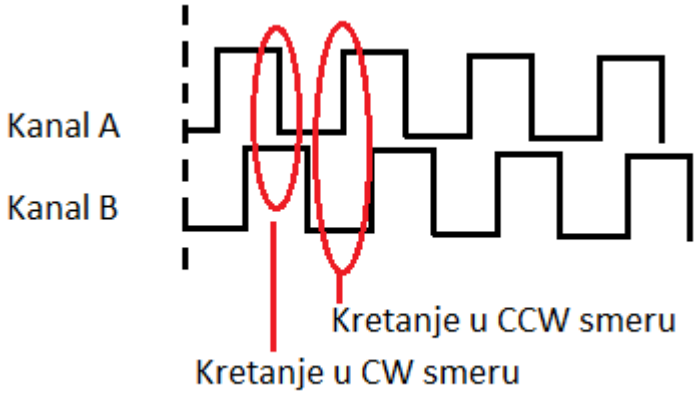
Слика 4.1.1 Скица схеме употребљене у примеру АД конверзије

Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-adc-example

4.2 Рад са енкодером

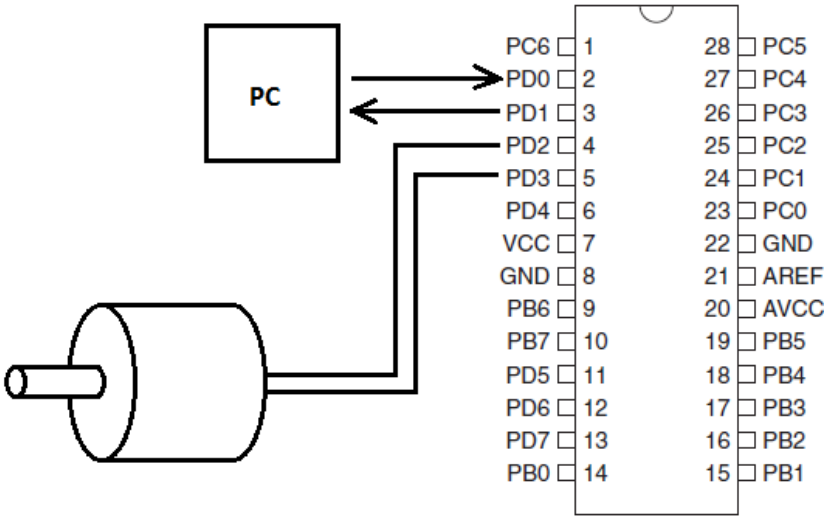
У овом примеру је илустрован рад са контролом спољашњих прекида употребом оптичког енкодера. Излазни канали енкодера су повезани на пинове контролера, који се посматрају у циљу детековања појаве ниског напонског нивоа. На опадајућу ивицу сигнала једног од канала енкодера улази се у прекидну рутину микроконтролера. Прекидна рутина миктроконтролера је рутина која се изврши током рада контролера уколико се испуни услов за њено извршавање. У овом случају то је појава ниског напоског нивоа на предодређеном пину контролера. Излазни сигнали из енкодера су фазно помери за половину периоде што нам омогућава детекцију смера обртања вратила енкодера. Приликом уласка у прекидну рутину знамо једну чињеницу – дошло је појаве ниског напонског нивоа канала 1 енкодера. Пошто су канали фазно померени, приликом обратања вратила у једном смеру доћи ће до

појаве ниског напонског нивоа на каналу 2, тј. до појаве високог напонског нивоа уколико се ради о обртању у супротном смеру. Дакле, да бисмо одредили смер обртања вратила довољно је проверити вредност напонског нивоа на једном каналу енкодера и то на оном који не прави услов за улазак у прекидну рутину. Приликом очитавања смера обртања долази до инкреметовања односно декрементовања вредности која представља апсолутну позицију енкодера. У главној петљи се сваке секунде очитава тренутна вредност позиције и шаље преко серијске комуникације на рачунар у људски читљивом формату. Сигнал који се добија са енкодера је приказан сликом 4.2.1.



Слика 4.2.1 Илустрација сигнала на излазу енкодера

На слици 4.2.2 се налази скица начина повезивања енкодера и микроконтролера, као и микроконтролера и рачунара.



Слика 4.2.2 Скица схеме употребљене у примеру рада са енкодером

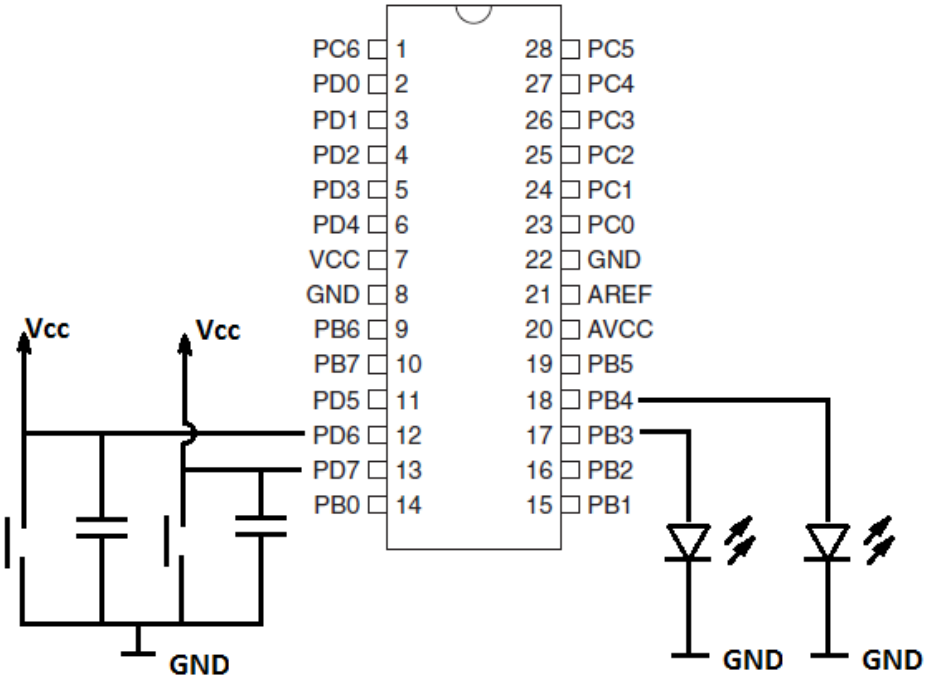
Псеудо код:

```
//glavna petlja
inicijalizuj tajmer
inicijalizuj uart
inicijalizuj brojač
_br = 0          /*inicjalijzuj promenljivu u kojoj će se čuvati trenutno stanje brojača*/
while(true)      /*uslov beskonačne petlje*/
        _br = preuzmi trenutnu vrednost brojača
        pretvori _br u niz karaktera
        pošalji _br na serijski port
        čekaj 1000 ms

//prekidna rutina
/*ulazak u prekidnu rutinu uslovljava pojava logičke nule na kanalu A enkodera
tj. kanal A == 0*/
if(kanal B == 0)
        pozicija--
else
        pozicija++
```

Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-ext-counter-example

4.3 GPIO (General Purpose Input Output)

У овом примеру илустрован је рад са улазима и излазима контролера. У зависности од стања тастера на развојном окружењу појединачни излази контролера се активирају тј. деактивирају. У главној петљи се врши провера стања пинова на које су повезани тастери, уколико је притиснут један од тастера контролер ће поставити одређени пин у стање логичке 1, у супротном вредност пина се поставља у стање логичке 0. Притиском на други тастер на окружењу долази до инвертовања тренутног стана једног од пинова контролера. Односно, уколико је пин био постављн у стање логичке 1, након притиска тастер биће постављен у стање логичке 0 и обрнуто. Скица начина повезивања тастера и микроконтролера дата је сликом 4.3.1.



Слика 4.3.1 Скица схеме употребљене у примеру рада са тастерима

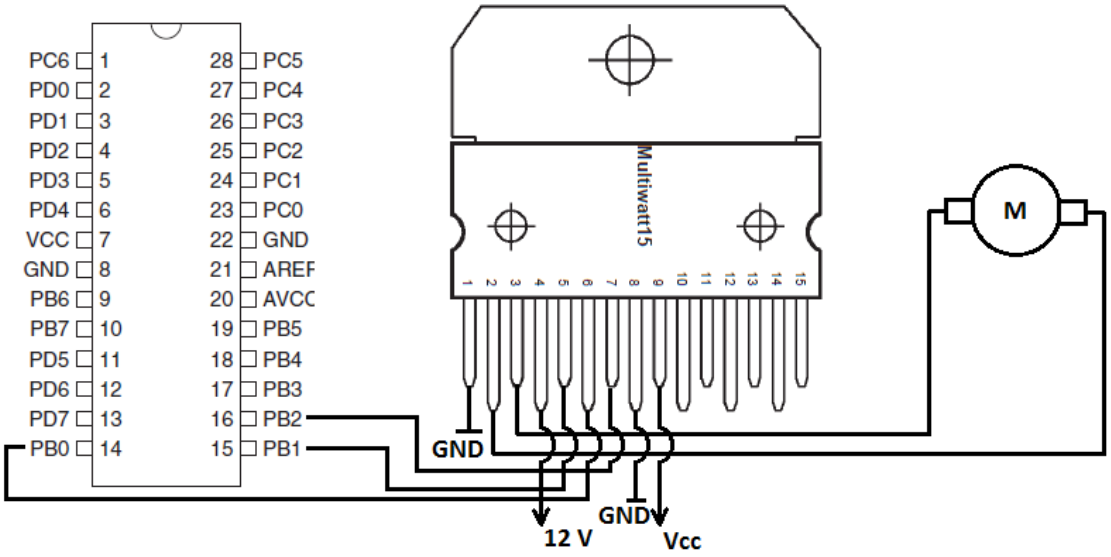
Псеудо код:

```
//glavna petlja
inicijalizuj ulazne pinove 1 i 2
inicijalizuj izlazne pinove 1 i 2
_ulaz2_prethodno_otpušten = false
while(true)      /*uslov beskonačne petlje*/
        if(ulazni pin 1 == 0)
                izlazni pin 1 = 1
        else
                izlazni pin 1 = 0

        if(ulazni pin 2 == 0 && _ulaz2_prethodno_otpušten == true)
                promeni stanje izlaznog pina 2
                _ulaz2_prethodno_otpušten = false
        else if(ulazni pin 2 == 1)
                _ulaz2_prethodno_otpušten = true
```

Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-gpio-example

4.4 Рад са H-мостом

Овај пример демонстрира наизменично покретање мотора једносмерне струје у смеру казаљке на сату и у смеру супротном од кретања казаљке на сату. За покретање мотора је употребљен H-мост L298N. Да би се мотор заштитио од оштећења која настају приликом наглих промена смера обртања између промена смера је унешено временско кашњење у трајању од пар секунди. Периодично се на сваке 3 секунде мењају стања пинова контролера који су повезани са пиновима H-моста. Ове промене условљавају промену смера протицања електричне струје која покреће мотор, а самим тим и промену смера обртања излазног вратила мотора. Графичка репрезентација начина повезивања H-моста, микроконтролера и мотора једносмерне струје приказана је на слици 4.4.1.



Слика 4.4.1 Скица схеме употребљене у примеру рада са H-мостом

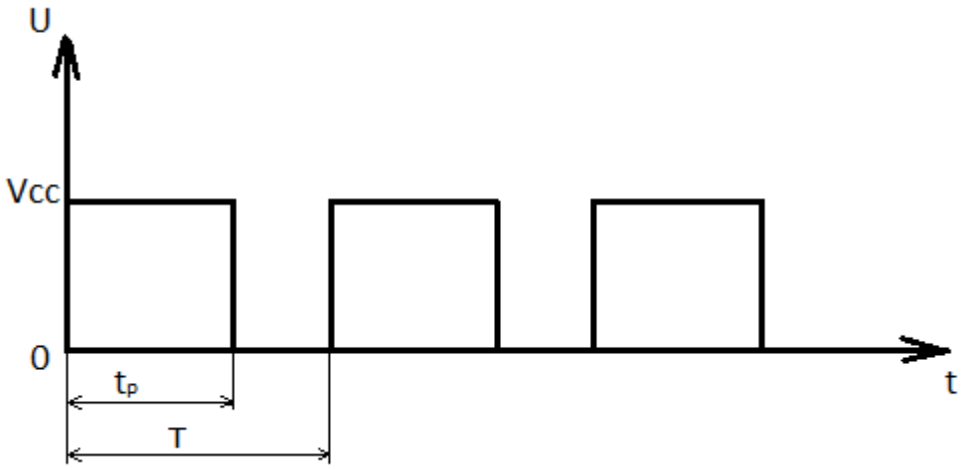
Псеудо код:

```
//glavna petlja
inicijalizuj H most /*ova f-ja obuhvata i inicijalizaciju pinova kontrolera*/
while(true)      /*uslov beskonačne petlje*/
        aktiviraj input 1 H mosta      /*pokretanje motora u jednom smeru*/
        čekaj 3 sekunde
        deaktiviraj input 1 H mosta   /*zaustavljanje motora*/
        čekaj 2 sekunde
        aktiviraj input 2 H mosta      /*pokretanje motora u drugom smeru*/
        čekaj 3 sekunde
        deaktiviraj input 2 H mosta    /*zaustavljanje motora*/
        čekaj 2 sekunde
```

Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-motor-example

4.5 PWM (Pulse Width Modulation)

PWM је у основи техника контроле количине енергије која се преноси. Уколико неки дигитални сигнал посматрамо као сигнал константне учестаности и уколико његова период траје *T* времена, тада можемо управљати количином енергије која се пеноси тим сигналом тако што ћемо сигнал поставити на висок напонски ниво у сваком *nT* тренутку и поставити га на низак напонски ниво у сваком *nT+t* тренутку, где је $0 \leq t \leq T$. Количина енергије која се пренесе сигналом је одређена односом *T/t* који се још назива и фактор испуне сигнала. Овај алгоритам је имплементиран на контролеру употребом тајмера контролера. Фактором испуне сигнала се управља брзином обртања излазног вратила мотора (или интезитетом којим LED светли). У овом примеру су употребљени тајмери за реализацију PWM сигнала. Уколико је потребно управљати брзином мотора начин повезивања је идентичан ономе са слике 4.4.1, у случају контроле количине светлости коју диода емитује довољно је повезати LED на један од излазних пинова контролера и извршити иницијализацију PWM сигнала на том пину. На слици 4.5.1 се налази пример сигнала константе учестаности, дужине трајања периоде *T* и фактором испуне $\frac{t_p}{T}$.



Слика 4.5.1 Пример PWM сигнала

Алгоритам апликације и генерисања PWM сигнала дат је следећим псеудо кодом:

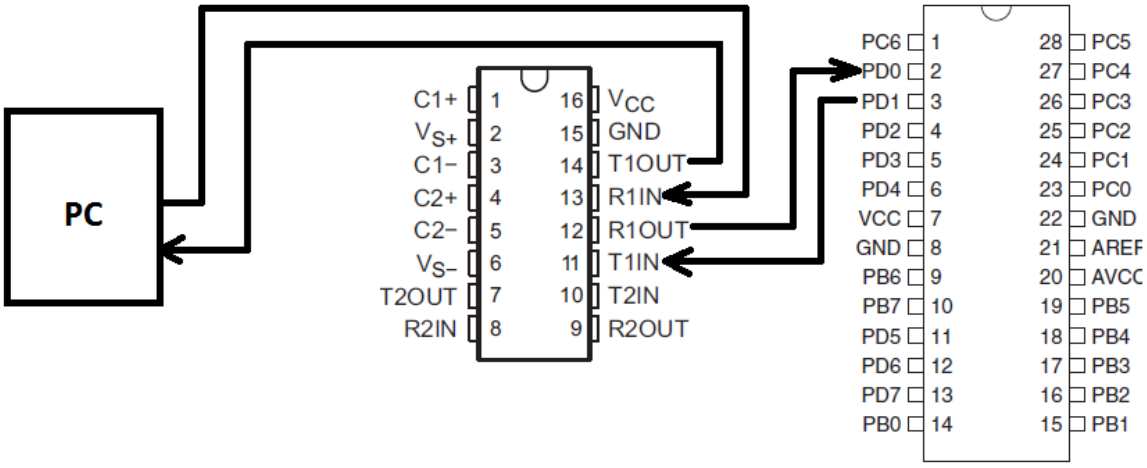
```
//glavna petlja
inicijalizuj tajmer
inicijalizuj uart
inicijalizuj H most      /*ova f-ja obuhvata i inicijalizaciju pinova kontrolera*/
inicijalizuj PWM         /*na jednom od ulaznih pinova H mosta*/
while(true)      /*uslov beskonacne petlje*/
```

```
for( i from 0 to vreme trajanja periode PWM signala )
        podesi PWM referencu na i
        čekaj 5 ms
for( i from vreme trajanja periode PWM signala to 0 )
        podesi PWM referencu na i
        čekaj 5 ms


//PWM petlja
brojač++
if(brojac == perioda PWM signala)
        predefinisani pin kontrolera = 1
        brojač = 0
else if(brojac == referenca PWM signala)
        predefinisani pin kontrolera = 0
```

Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-pwm-example

4.6 UART (Universal Asynchronus Receiver Transmiter)

Овај пример илуструје комуникацију између рачунара и развојне плоче. За исправан рад примера потребно је дефинисати поклапајуће брзине комуникација и на рачунару и на контролеру. Другим речима, и рачунар и контролер морају слати/примати податке истим брзинама. Пинови контролера предвиђени за слање и примање података (бајтова информација) су повезани на MAX232 интегрисано коло, на које је са друге стране повезан DB9 конектрор преко којег се врши повезивање на рачунар. MAX232 служи за превођење напонских нивоа са 5 V и 3,3 V на нивое од +/- 5 V, +/- 3,3 V i +/- 12 V. Да би се постигла комуникација путем RS-232 протокола, неопходно је користити и FTDI коло (које се налази у сваком USB-на-RS-232 kablu). До сада се у примерима спомињала комуникација са рачунаром и на сликама је начин повезивања био представљан као директно спајање микроконтролера са рачунаром. Ово у пракси није могуће без додатних компоненти описаних у глави 3.8. Детаљнија скица начина повезивања рачунара и микроконтролера у циљу обављања комуникације по RS-232 протоколу дата је сликом 4.6.1. Псеудо код слања података путем серијске комуникације је приказ кроз претходне примере, те га није потребно додатно понављати.



Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-uart-example

4.7 Тајмери

Тајмери пружају могућност контролеру да периодично извршава инструкције, уводи временско кашење између извршавања инструкција и још много других ствари. Како је већина реалних апликација аутономна (зависи од времена) јасно је да тајмери представљају један од основних и најважнијих механизама микроконтролера. У примеру је представљен рад са тајмерима где се врши паљење и гашење три LED-овке на различитим учестаностима. Фокус примера је на реализацији псеудо-паралелног извршавања инструкција. Шта то значи? Да бисмо разумели концепт добре програмерске праксе (бар када су у питању микроконтролери) прво треба разумети како се то код извршава унутар једног контролера. Зарад једноставности претпоставићемо да се ради о контролеру са једним језгром и једном нити (што у нашем случају и јесте тако, али треба знати да постоје контролери много сложенији од оног употребљеном у овом пројекту). Уколико желимо да три диоде трепере на различитим учестаностима, логично је да је између паљења и гашења сваке диоде мора постојати одређени временски размак. Најједноставнија врста увођења временског размака у извршавање програма јесте увођење временског кашњења. Временско кашњење задржава програм, на месту где је кашњење уведено, и спречава извршавање свих даљих инструкција све док не прође време трајања уведеног кашњења. Наравно, овакав приступ решавању проблем је сасвим прихватљив у неким ситуацијама (заправо, ако обратимо пажњу, сви примери до сада су у себи имали неки вид временског кашњења), али постоје неретке ситуације где овај приступ решавању проблема уводи још више проблема. Како се све инструкције унутар нашег контролера извршавају редоследом којим су написане, врло лако се може закључити да увођењем било каквог временског кашњења између паљења и гашења прве диоде неминовно уносимо додатно кашњење између гашења и паљења друге диоде. Уколико замислимо да је потребно управљати радом 50 диода на различитим учестаностима, јасно се види да је једноставно увођење временског кашњења непрактичан приступ.

Из тог разлога је потребно разумети принцип примењен у програмирању алгоритма за решавање овог проблема. Следећи псеудо код би требао да помогне у томе:

*/*U prekidnoj rutini tajmera se vrši dekrementovanje 3 unapred definisane promenljive (vreme1/2/3) sve dok njihova vrednost ne dostigne 0. Postoje funkcije koje omogućavaju dodeljivanje vrednosti ovim promenljivama. U kodu su označene kao postavi_vremeX_na_N. Kao i funkcije za proveru vrednosti gorepomenutih promenljivih, ozančićemo ih sa proveri_vremeX.*/*

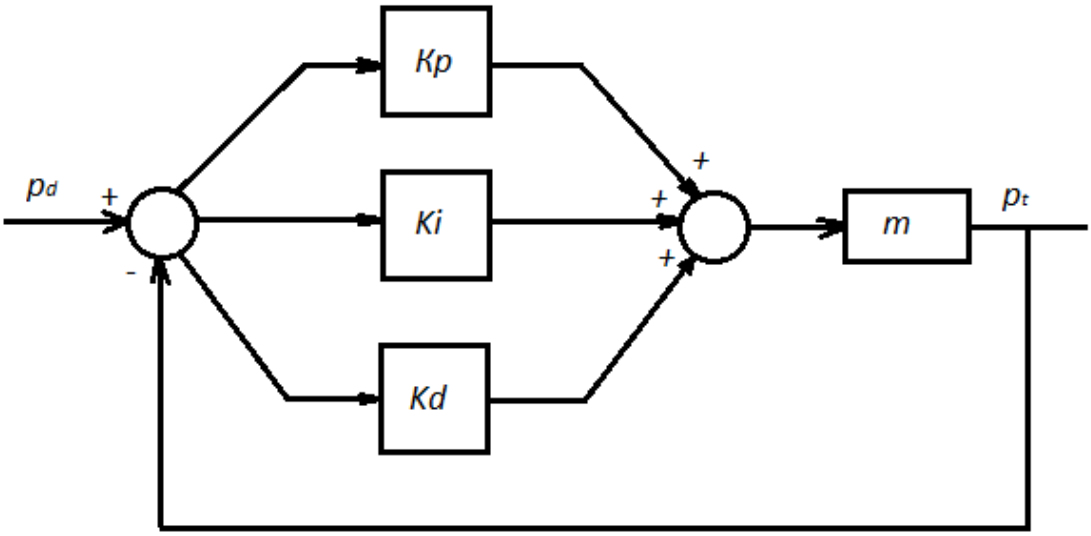
```
//glavna petlja
inicijalzuj pinove na kojima se nalaze diode
inicijalizuj tajmer
while(true)
    postavi_vreme1_na_100
    if(proveri_vreme1 == 0)
        promeni stanje diode 1
        postavi_vreme1_na_100
    postavi_vreme2_na_200
    if(proveri_vreme2 == 0)
        promeni stanje diode 2
        postavi_vreme2_na_200
    postavi_vreme3_na_150
    if(proveri_vreme3 == 0)
        promeni stanje diode3
        postavi_vreme3_na_150

//prekidna rutina tajmera
if(vreme1 > 0)
    vreme1--
if(vreme2 > 0)
    vreme2--
if(vreme3 > 0)
    vreme3--
```

Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-timer-example

4.8 ПИД регулација позиције мотора

За очитавање позиције мотора употребљен је ISC3806 оптички енкодер. Регулација се врши на основу предефинисане вредности позиције изражене у броју апсолутних импулса енкодера у једном смеру. Уколико је излазно вратило мотора направило мање обртаја од задате референце мотор ће се обртати у једном смеру, а у супротном случају у другом смеру и тако све док не дође у мирну радну тачку (која, у

зависности од параметара регулатора може имати различите вредности, али би увек требала да је у околини референтне тачке). Начин повезивања је идентичан ономе са слике 4.4.1. Блок схема ПИД регулатора је представљена на слици 4.8.1.



Kp - proporcionalno pojačanje
Kd - diferencijalno pojačanje; sadrži diferencijalni član
Ki - integralno pojačanje; sadrži integralni član

Слика 4.8.1 Блок схема ПИД регулатора

Пример представља интеграцију модула покривених до овог дела у тексту. Алгоритам рада је садржан у слици 4.8.1.

Пример се може преузети са: https://github.com/markotikvic/avr-dev-board/tree/master/code/avr-dev-board-pid-example

5 Закључак

Пројектом је покривен широк спектар апликација које студенту могу помоћи у бољем разумевању рада микроконтролера, њихове примене, интеграције у постојеће системе и развој интерфејса са другим уређајима. Може се приметити да приложени примери изостављају рад са I^2C комуникацијом и рад са EEPROM-ом. Надамо се да је градиво покривено овим пројектом довољно да студенте мотивише да изостављене библиотеке развију сами.

Треба нагласити и грешке које су начињене приликом дизајна/реализације пројекта:

- Како су модули додавани на окружење убрзо је постало јасно да количина потребног ожичавања увелико отежава проналажење и уклањање кварова. Штампана плоча би била неупоредиво бољи избор.
- У тренутку пројектовања модула за рад са оптичким енкодером на располагању није била информација о његовој струјној потрошњи. Испоставило се да је одабрани стабилизатоп напајња поддимезионисан за ту употребу.
- MAX232 коло и DB9 конектор треба заменити са FTDI колом и микро USB прикључком, како због велике уштеде на простору, тако и због олакшаног повезивања окружења и модерних рачунара. Овај проблем иде руку под руку са проблемом штампане плоче.

6 Референце

[1] Микроелектроника

[2] Arduino

[3] Texas Instruments

[4] AtmelStudio

[5] AVR-GCC

[6] CadSoft EAGLE

[7] ZeptoProg II

[8] L7805

[9] MAX232

[10] L298N

[11] ATmega328P

[12] ISC3806