AN EFFICIENT MICROCONTROLLER BASED ARCHITECTURE FOR LINEAR POWER SUPPLIES

A Thesis

Presented to the Faculty of Engineering

of Ankara University

In Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Engineering

by

Teoman Soygül

With the Sponsorship of

June 2009

With the Sponsorhip of

ÖZ

Modern elektronik cihazlarda boyutlarin olabildigince kucuk olmasinin oldukca onem kazandigi bir zamanda, ozellikle laboratuvarlarin en onemli parcasi sayilan ayarlanabilir kuc kaynaklari icin de daha etkin bir mimari gelistirilebilecegi aciktir. Ayarlanabilir gerilim ve akim cikisinin, dijital ekranli gostergelerin, kisa devre korumasinin, bilgisayar kontrolunun ve hatta gerekirse lineer degil fonksiyonel cikisin bir arada toplandigi bir guc kaynagi insaa etmek dijital elektronigin guncel olanaklari ile oldukca mumkun gorunurken butun bu ozelliklerin tek bir mikrokontrolur tarafindan saglanmasi ise onemli olcude arastirma ve gelistirme gerektirmektedir. Bu amacla gelistirilen bu projede, onceden sunulan bazi teorik ongorumlerin de test edilmesi ile analog parcalar ile yapilmasi muhtemel sistemlerin tek bir dijital kontrol unitesinde birlestirilmesi ile en az masraf ile en yuksek kazanc elde edilerek endustriyel kalitede bir calisma meydana getirilmistir. Mimarinin beklenenin dahi cok uzerinde bir etkinlige sahip olmasi, benzeri yaklasimlarin diger butun elektronik cihazlarin maliyetleri ve boyutlarinin dusurulurken basariminin artirilabilecegini kesin olarak gostermektedir.

For the greater good…

TEŞEKKÜRLER

En basindan beri bu projenin var olmasinin sagladigi icin tez danismanim Prof. Dr. Necmi Serin'e tesekkur ederim. Gelecekte bu ve benzeri projelerin gelismesine kolaylik sagladiklari icin Prof. Dr. Ali Ulvi Yılmazer, Prof. Dr. Çelik Tarımcı ve Doç. Dr. Hüseyin Sarı'ya ayrica tesekkur ederim. Bu projenin ana sponsoru olarak arastirma ve gelistirme asamasinda verdikleri destekten oturu Türkiye Odalar ve Borsalar Birliği'ne ozel olarak tesekkur ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

KISALTMALAR

uC: Microcontroller (Mikrokontrolur)

MCU: Microcontroller Unit (Mikrokontrolur Unitesi)

uP: Microprocessor (Mikroislemci)

ADC: Analog to Dijital Converter (Analog Dijital Cevirici)

DAC: Digital to Analog Converter (Dijital Analog Cevirici)

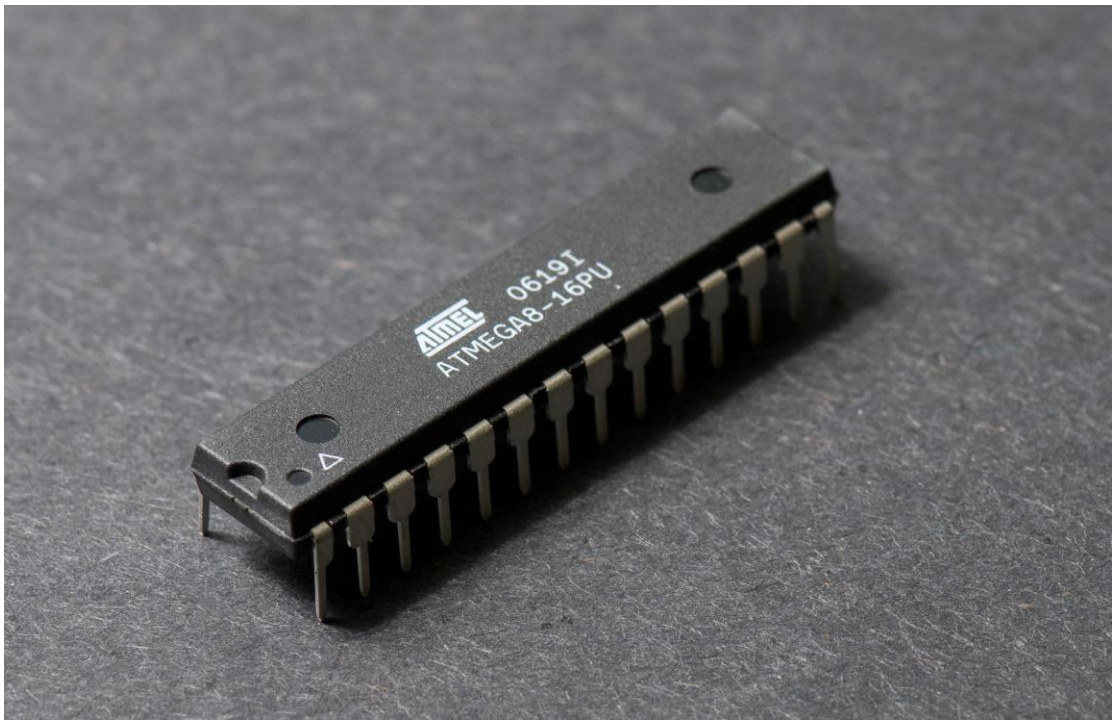
FRM: Firmware (Yerlesik Bellek)

ÖNSÖZ

Bu tezin devaminda, literaturde mikrokontrolur teriminin yerine daha sık kullanilan µC (microcontroller) veya MCU (microcontroller unit) terimleri kullanilacaktir. Mikrocontroller terimi yerine bazen mikroislemci denmesinin nedeni ile birlikte devre tasariminda kullanilan diger parcalar hakkinda genel bilgiler, ilerleyen bolumlerde sunulmaktadir. Ozellikle diyagramlarin ve grafiklerin bu tezin Ingilizce versiyonu da dusunulerek hazirlanmasindan dolayi, ayrica ceviri yapilmayarak orjinal halleri ile kullanilmislardir. Bu sayede ozellikle grafiksel olarak ifade edilen kaynak kodu ve dijital diyagramlar, yazildiklari dil ile uyumludurlar.

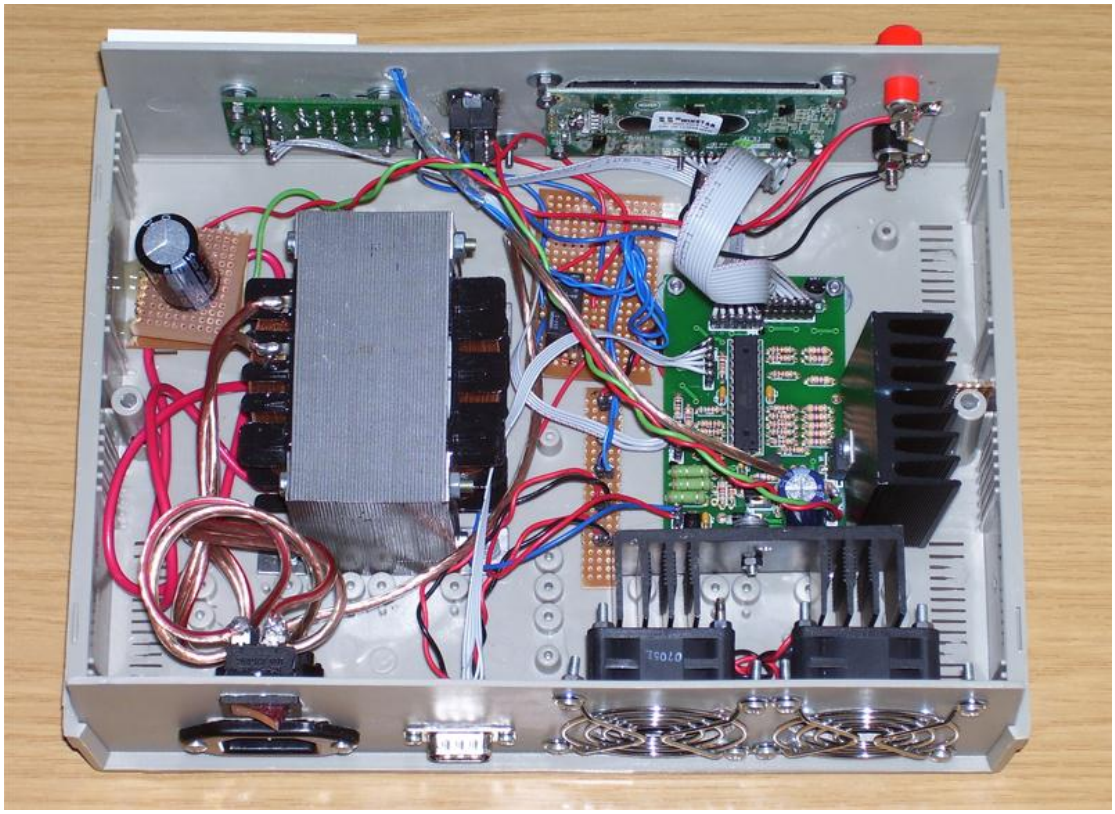
Bölüm 1 Giriş

Temelde en cok kullanilan laboratuvar ekipmanlarindan olan ayarlanabilir guc kaynaklari, eski tasarimlarin oldukca guvenilir olmalari nedeni ile genel olarak analog yapiya sahiptirler. Dijital ekranli instan arayuzu veya bilgisayar baglantilari olsa bile temel olarak degisken guc uretme isi analog devre elemanlari ile yapilir. Bu tezin geneli itibari ile beklenti, geleneksel olarak analog parcalara yuklenen isin oldukca buyuk bir kisminin dijital bilesenler tarafindan da yapilabilecegini gostermektir. Bu surecte, dijital bilesenlerin de tek bir paket icinde toplandigi bir microkontrolur kullanmak da toplam uretim maliyetini oldukca dusurucu bir etmen olacaktir. Bu tezin deneysel kismini olusturan elektronik devrenin tasarimindaki temel bilesen, Figur 1 uzerinde goruldugu gibi Atmel Atmega8 mikrokontrolür'dur.



Şekil 1.1: Atmel Atmega8 mikrokontrolür [1].

Elektronik devrenin tasariminda, ATmega8 disinda hicbir entegrenin kullanilmasina gerek kalmamasi, modern MCU'larin ne denli yetenekli oldugunu gostermektedir. Iyi programlanmalari halinde, sistemdeki diger butun analog parcalarla iletisim kurmak icin gereken butun donanima sahiptirler. MCU kullanilarak devrenin butunu ile tamamlanmis hali asagidaki gibidir.



Şekil 1.2: Tezin deneysel kisminin tamamlanmis hali.

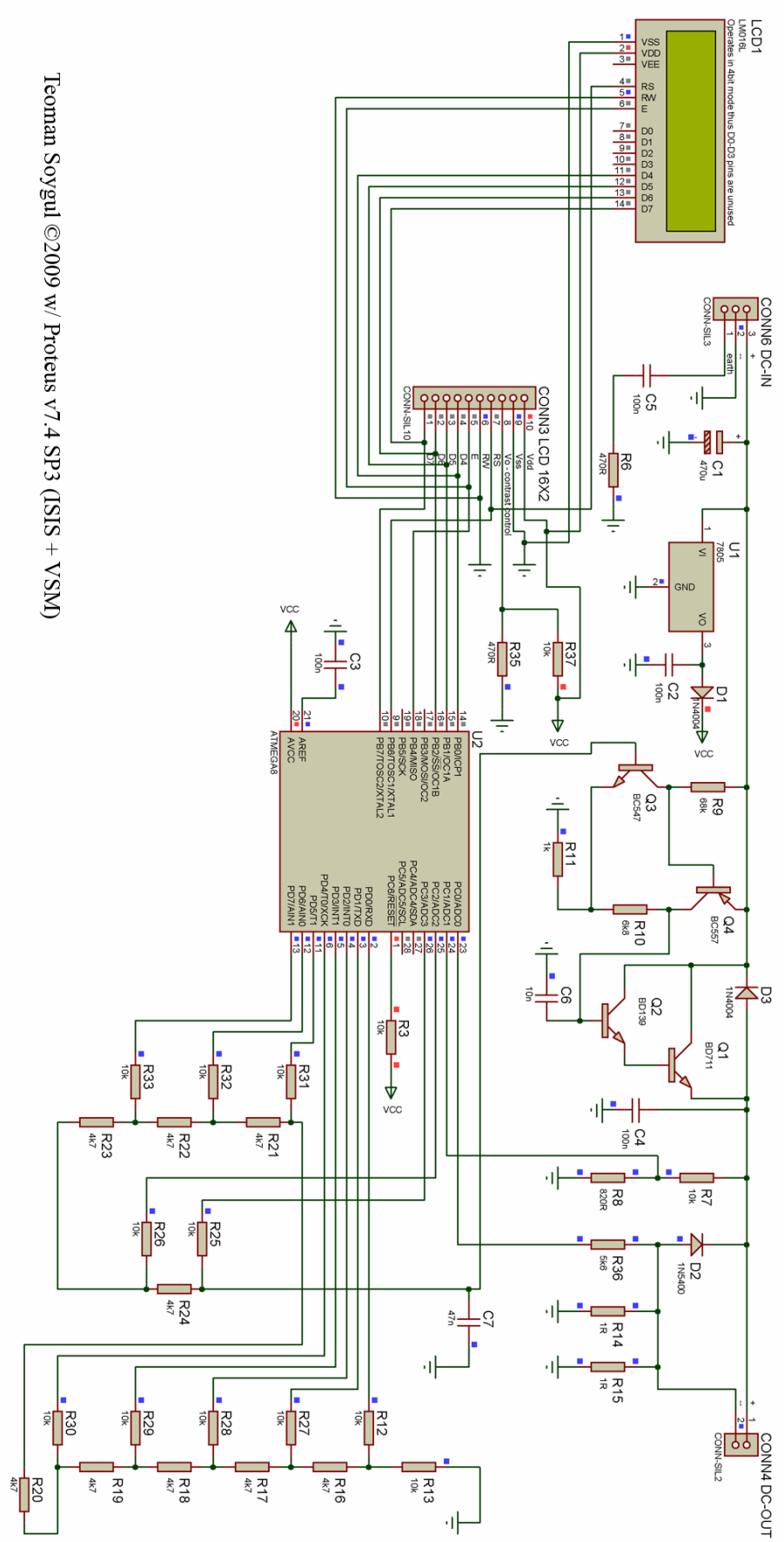
Geleneksel bir guc kaynagini dusunerek, yukaridaki figurde bircok analog parca eksik gorunmektedir. En basit haliyle bir crowbar circuit [2] (kisa devre korumasi) dahi goze carpmamaktadir. Bunun nedeni, kisa devre korumasinin MCU seviyesinde yazilimsal olarak yapilmasidir. Bu devre icin hazirlanilan yazilima sadece birkac satir kod eklenilerek, normal sartlar altinda bircok ek parca gerektirecek olan kisa devre koruma sistemini, cok daha etkin bir sekilde uygulamaya konulmustur. Bu haliyle MCU, 100 µs icinde kisa devreye tepki vererek akimi kesmektedir ki bu en iyi

analog korumdanan bile birkac kat daha hizlidir. Bu ornege benzer sekilde, bircok farkli durum icin ek analog devre elemanlarinin yapmasi gereken is yazilim seviyesine indirgenerek, lineer guc kaynaklari icin etkin bir mimari ornegi ortaya konmustur. Basariyla tamamlanan bu deneysel calismanin sonucu, asagida gorulmektedir.

Şekil 1.3: Deneysel calismasinin sonucu.

Bölüm 2 Donanım Tasarımı

Her bir basamagin ayri bir isleve sahip oldugu guc kaynagi devresinde, butun basamaklarin biraraya getirilmesi ile islevsel bir devre elde edilmis olunur. Sekil 4.1 uzerinde gorulecegi uzere, devrenin dijital kisminin disinda en buyuk alani analog yukseltec basamagi almaktadir. Tabi yukseltec ile MCU arasindaki baglantiyi saglayan R2 Ladder'da harici bir DAC kullanilmamasi nedeni ile yer ve enerji tuketimi acisindan onemli bir unsurdur. Bu devre gucunu 7805 ile anakolda saglamasi nedeniyle bu durum sorun yaratmazken, pilli uygulamalar icin kacinilmasi gereken bir cozumdur. Ote yandan maximum DAC hizina bu islevi en basit haliyle yapan R2 Ladder ile ulasilabilir. R2 ile guc yukselteci arasinda hicbir buffer etmeni bulunmadigi icin veya transistor staturation gecikmeleri olmadigi icin, MCU ile yukseltec arasindaki analog veri iletisimi oldukca seri bir sekilde gerceklesmektedir. Bu iletisimin oldukca hizli olmasinin gerekliligi, kisa devre korumasinin tehlike aninda olabilecek en kisa surede aktif hale gelmesinin gerekli olusundan ile gelmektedir. MCU'nun halihazirda gerekli cozunurluge sahip bir ADC'ye sahip olmasi nedeni ile, basit birer feedback line ile akim ve gerilim, hizli veri iletisiminin sagladigi rahatlik icinde surekli kontrol altinda tutulur. Sekil 4.1 uzerindeki semada, tus takimi, yerlesik bellek guncelleme ve bilgisayar baglantilari gosterilmemis olmakla birlikte, bu cizim aslen butunu ile bir devre simulasyonudur. Sonraki sayfada gorulen guc kaynagi devresi, fiziksel olarak insaa edilmeden once Proteus VSM ile SPICE simulator ortaminda test edilmesi [3], ayri devre basamklarinin butunu ile calisirligini gormek icin oldukca onemlidir. Bu sayede, henuz hicbir fiziksel altyapi hazir degilken bile simulator ortaminda butun devrenin calisir hali test edilebilir.

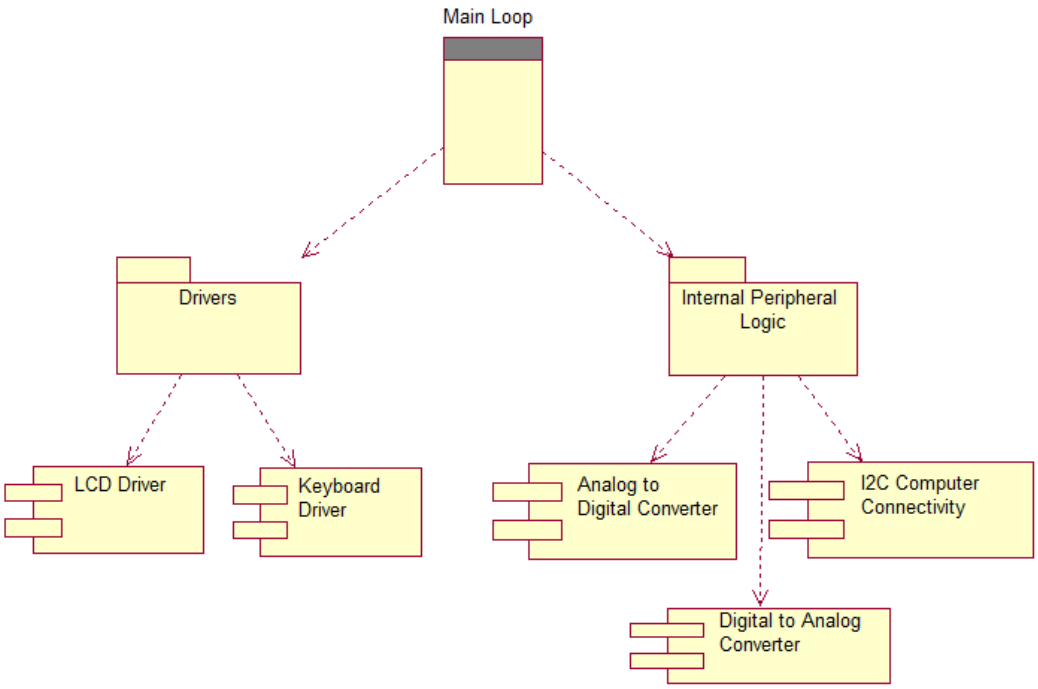


Şekil 2.1: Butunu ile lineer guc kaynagi devresi.

Bölüm 3 Microcontroller Yerleşik Bellek Tasarımı

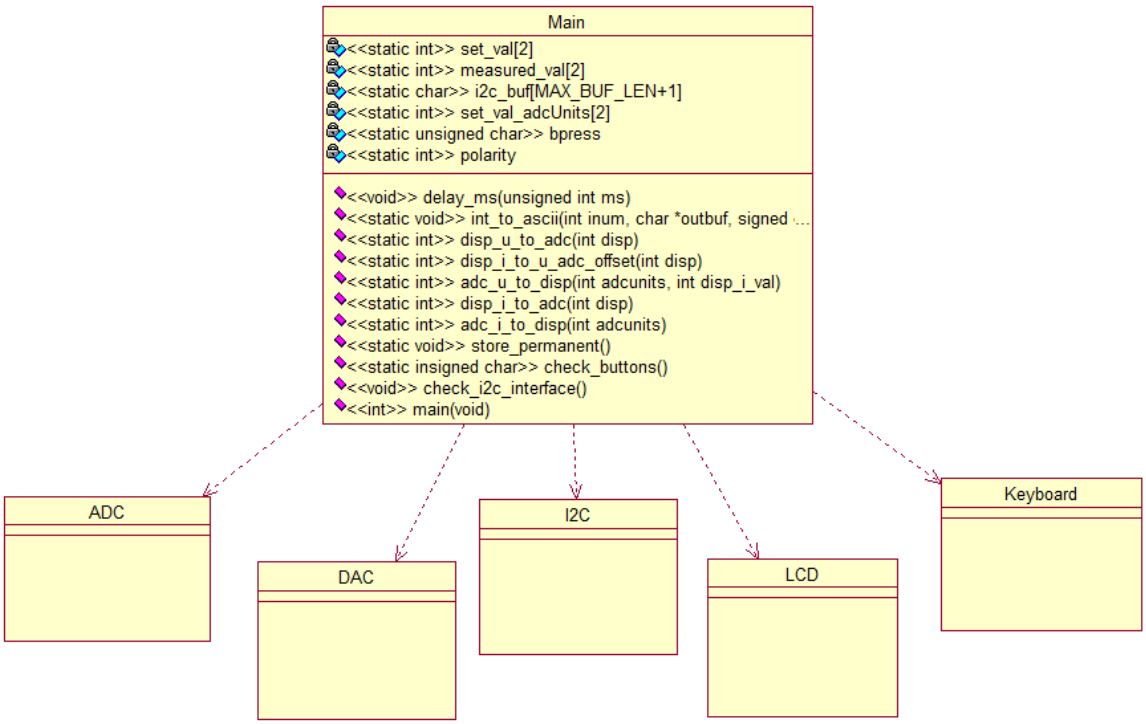
Bu tezin deneysel kisminin en onemli parcalarindan biri, fiziksel devrenin hemen hemen butun islerliginin tanimlandigi MCU yerlesik bellek (firmware) tasarimidir. EK 1'de bu devre icin tasarlanan yerlesik bellek yaziliminin en onemli parcasi olan ana dongunun kaynak kodu gorulmektedir. Yazilimin tamami yaklasik olarak 1700 satir olmasi nedeni ile EK 1 'de sadece ana modul eklenmis olup ve bu tez dosyasi ile sunulan CD icinde butun kaynak kodu bulunmaktadir. Yazilim butunu ile Atmel AVR Studio 4 ile WinAVR 2009.03 (C99) standardi icin tasarlanmistir [5].

Bu devre dahilinde MCU islevini tanimlayan yazilim, ana hatlari ile moduler olarak tasarlanmistir. Bunun amaci, programda bilgi akisini saglayan ana kod ile bu akisin devamliligini saglayan yardimci kodlari ayirmaktir. Ornegin, olculen akim ve gerilim degerlerini LCD ekrana yazdirmak icin kullanilan modul, ana yazilimdan butunu ile ayri tutulmustur. Bu sayede, daha ileri modeller icin daha genis bir grafik LCD kullanilmak istendiginde, butun yerlesik bellek yazilimi yerine sadece bu modulun degistirilmesi ile istenilen sonuca erisilebilecek olmasidir. Bu devre icin, MCU yazilimi esas olarak buyuk bir ana dongu (main loop) ve iki ana yazilim paketinden (package) olusmaktadir. Sekil 5.1 uzerinde gorulen bu moduler yapi, kontrol yaziliminin diger butun elemanlarini icine alir. MCU 'nun insan arayuzunu olusturan parcalar (LCD ekran ve klavye) ile iletisimi, surucu yazilim bilesenleri ile saglanir (diyagramlar tezin Ingilizce versiyonu dusunulerek butunu ile Ingilizce olarak hazirlanmistir). MCU icinde halihazirda var olan analog dijital cevirici, I2C iletisim yolu gibi icsel bilesenlerin kullanilmasi icin gereken yazilim bilesenleri de icsel cevre birimi mantigi paketinde toplanmistir.



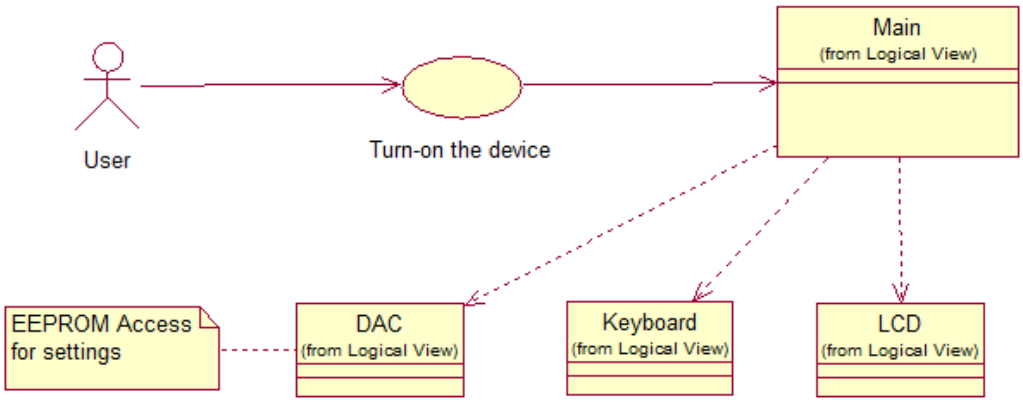
Şekil 3.1: MCU yerlesik bellegi bilesen (component) diyagrami.

Her bir yazilim paketi icinde sunulan bilesenler, bir veya birden cok alt sinifa ayrilmistir. Tabi C programlama dili icin sinif gibi bir kavram mevcut olmasa bile, nesne tabanli programlama mantigi ile her bir is icin ozellestirilmis fonksiyonlar ayri modul dosyalari icinde birlestirilmislerdir. Bu durumun en buyuk yarari, projenin gelistirilip 32 bitlik bir MCU ile guncel bir tasarim yapilmak istendiginde, yazilim uzerinde cok az degisiklik yaparak bu guncellemenin rahatlikla yapilabilecek olmasidir. Sekil 5.2 uzerinde, MCU yerlesik belleginin tasariminin butununu olusturan siniflarin tamami gosterilmektedir. Butun fonksiyon ve degiskenleri acikca diyagram uzerinde gosterilmis olan Main sinifi, programin en onemli parcasi durumundadir. EK 1 icinde butun kaynak kodu bulunan bu sinif, bir kere cihaz calismaya basladigi zaman butun kontrolu ele alip, olasi ve sira disi durumlarin butunu icin verilmesi gereken uyarilari dikte eder. Daha az detay ile gosterilmis diger siniflar ise, yardimcilar olarak adlandirilir. Bunlar, asil kodun islevselligi icin gereken alt bilesenlerdir.



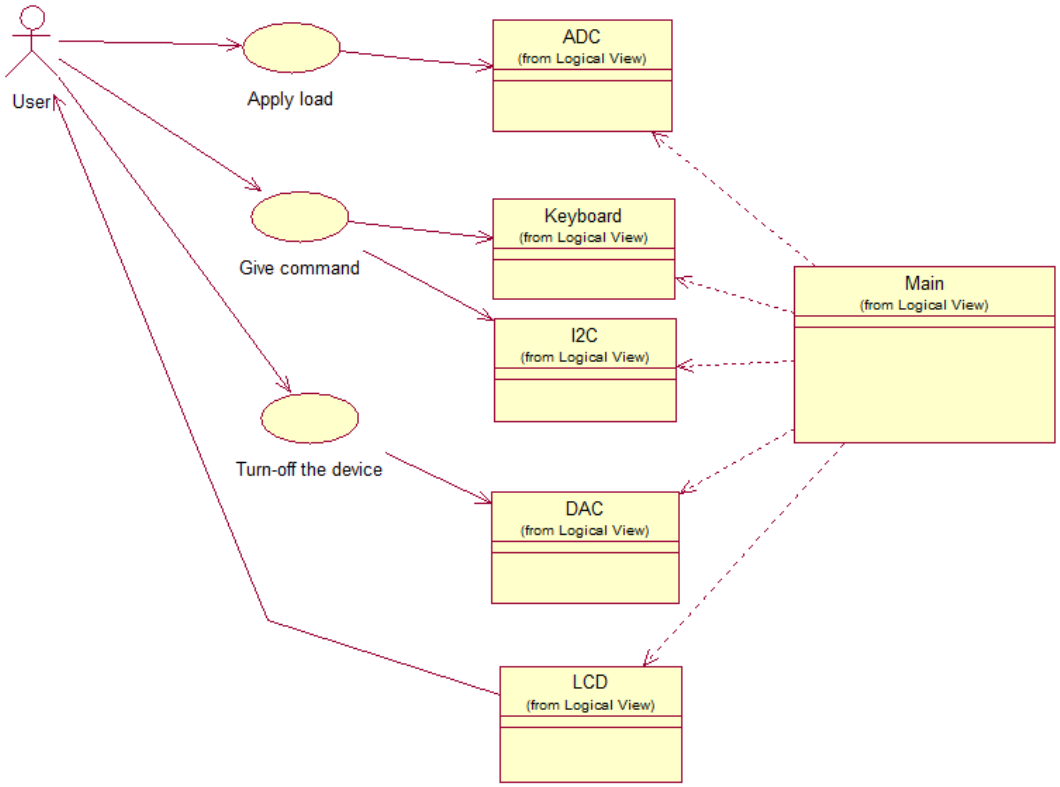
Şekil 3.2: MCU yerlesik bellegi sinif (class) diyagrami.

Kodun genel islevi, gorsel olarak bir dizi kullanim senoryosu ile belirlenmistir. Ornegin cihazin ilk acilis anindan itibaren komutlar icin hazir hale gelisi suresinde meydana gelen etkilesimler, Sekil 5.3 uzerinde gosterilmistir. (Butun component, logic, usecase ve sequence diyagramlari IBM Ration Rose [4] ile cizilmistir)



Şekil 3.3: Acilis sekansi icin kullanim ornegi (use case) diyagrami.

Kullanicinin cihazi acmasi anindan itibaren, ilk olarak Main sinifi baslatilir ve sirasi ile LCD ekran, klavye ve dijital-analog cevirici aktif hale gelir. Eger, onceden kaydedilmis akim ve gerilim degerleri varsa, bunlar hafizadan okunarak cikis degerleri ayarlanir. Bu uc asamali baslangic sekansindan sonra sistem uyarilara hazir hale gelir. Bu uyari cikis uclarina bir yuk direnci baglanmasi olabilecegi gibi, klavyeden veya bilgisayardan bir komut verilmesi seklinde de olabilir.



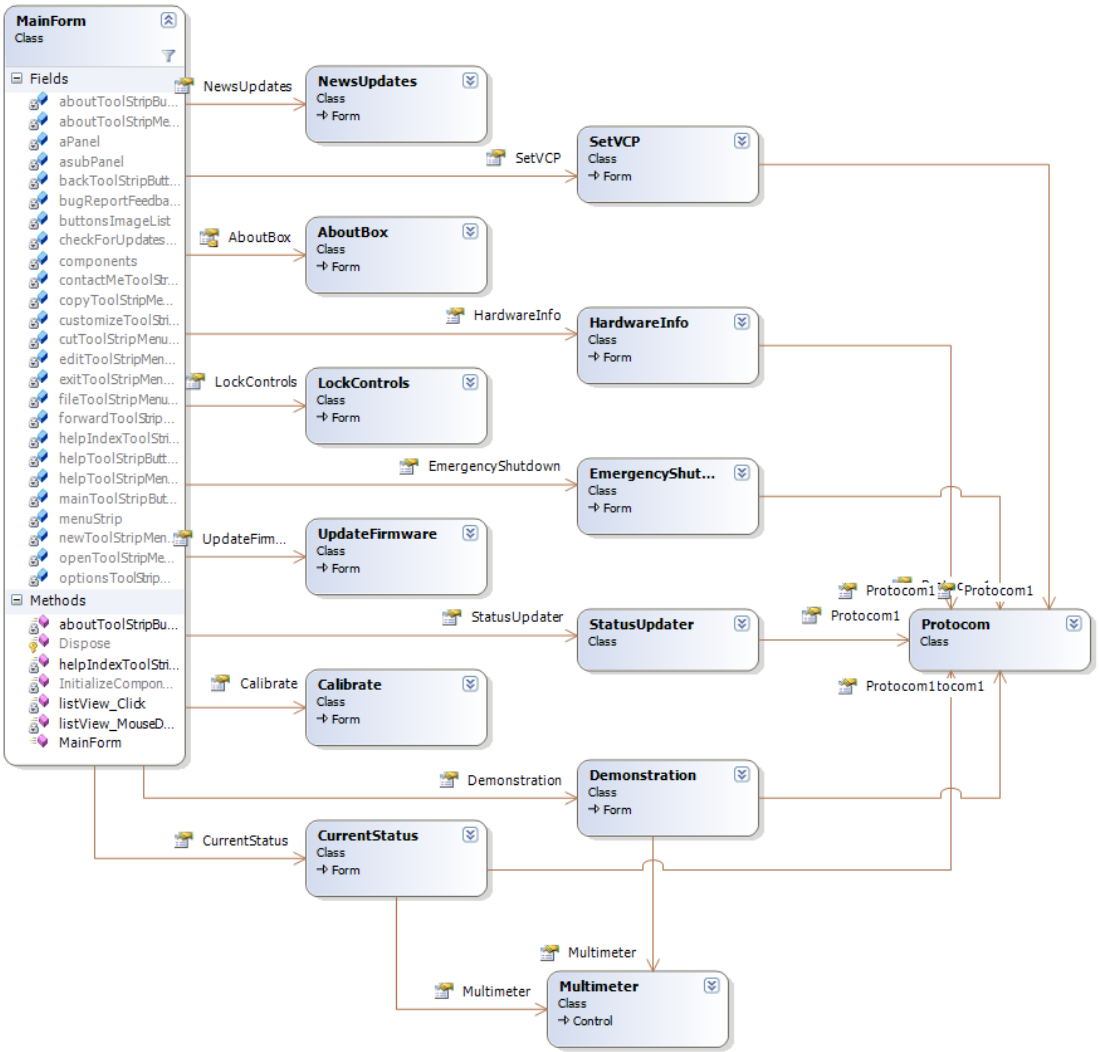
Şekil 3.4: Sistemin dis dunya ile etkilesimleri icin kullanim ornegi diyagrami.

Yukaridaki sekilde gorulecegi uzere, kullanicinin sistem ile her etkilesimi farkli bir bilesen tarafindan degerlendirilerek sistemde bir degisiklige yol acmaktadir. Bu, verilen bir komuta uygun olarak akimin degistirilmesi olabilecegi gibi buyuk bir yuk baglanmasiyla kisa devre korumasinin aktive olmasi seklinde gelisebilir. Diyagramda, sistemin kullaniciya veri aktarma yolu olarak sadece LCD ekran gosterilmis olsada,

bir sonraki kisimda gorulecegi uzere, kumanda yazilimi kullanilarak cihazdan mevcut yuk durumu ile ilgili cok detayli bilgiler alinabilmektedir.

Bölüm 4 Kontrol Yazılımı Tasarımı

Butunu ile lineer bir guc kaynagi olmasi icin tasarlanan bu sistemin kontrolu icin en pratik yol cihaz uzerindeki klavye ve verileri gozlemlemek icin LCD ekran olsa bile, bu basit ozelliklerin cok otesinde gereksinimler icin bir masaustu kontrol yazilimi kullanmak kacinilmaz hale gelmektedir. Tasarlanan boyle bir yazilimin en onemli ozelligi dogal olarak kullanici ile etkilesimi maximuma cikarirken, otomatik kontrol islevini de eksiksiz yerine getirebilmek olmaktadir.



Şekil 4.1: Kontrol paneli sinif diyagrami.

Elektronik sistem ile benzer isme sahip olmasi icin “Prototip B Kontrol Paneli” olarak adlandirilan kontrol yaziliminin butunu ile moduler yapisi Sekil 6.1 uzerinde rahatlikla gorulmektedir. Birbiri ile butunu ile instance iliskisi olan siniflar, bir otomobilin parcalari gibi bir araya gelerek islevsel bir butun olusturmaktadir. Tamami ile nesne-iliskili (object-oriented) olarak C# programlama dile ve Microsoft Visual Studio 2008 ile dizayn edilen yazilim [7], .NET Framework 3.5 SP1’in ozellikle nesne tabanli sistemler icin sundugu butun olanaklari kullanmaktadir. (Sekil 6.2 uzerindeki islevi gerceklestiren modul’un kaynak kodu EK 1 icinde bulunabilir. Yazilimin tamami 13500 satir [270 sayfa] ‘i astigi icin tez dosyasi ile birlikte sunulan CD icinde bulunabilir). Kullanilan programlama teknikleri acisindan da oldukca zengin olan program, temel olarak asagida listelenen programlama modellerin kullanmaktadir:

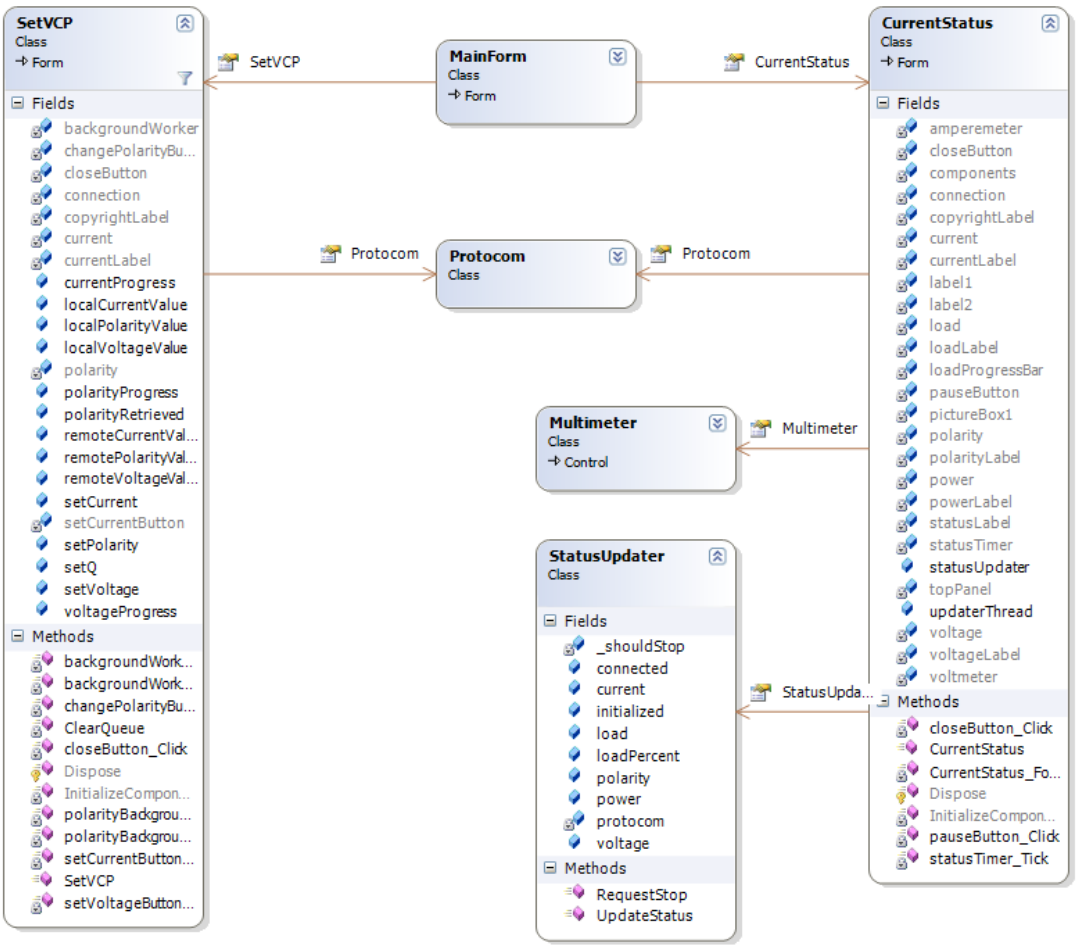
- .NET Framework 3.5 ile sunulan birinci sinif multithreading siniflar ile elektronik devre ile non-blocking iletisim. Yani kontrol panelinde bilgiler surekli olarak kullaniciya sunulurken arka pilanda cihaz ile iletisim araliksiz olarak surmekte ve bu iletisim sirasinda kontrol paneli kullanici girislerine surekli olarak cevap vermeye hazir durumda olmaktadir.

- Ozellikle Timer kontrolu ile periyodik interruplar ile baglantinin devamliligi surekli test edilmektedir. Kullanici arayuzu ve iletisim icin kullanilan islemcilere ek olarak 3. bir eszamanli islemici olarak calisan timer interrup’lar ile cihaz ile baglantinin kesilmesi halinde varsayilan is yarida kesilerek baglanti sekansinda geri donulmektedir ki bu da endustriyel kalitedeki yazilimlarda bile sıkca rastlanan veri kayiplarini sifira indirir.

- Queue ve stack veri yapilarinda tutulan komutlar sirasi ile devreye gonderilerek sonuclar arka planda islenir. Bu kullanicinin cihaza gonderilen komutlarin sonuclarini beklemeden seri bir sekilde komut verebilmesini saglamaktadir.

- Ozellikle Windows.Forms sinifinin oldukca gelismis grafik istemci ozellikleri kullanilarak program icinde kullaniciya daha fazla veri monoton hale getirilmeden sunulabilmektedir.

Butun bunlarin yaninda C# dilinin esnekliklerini de gostermek icin oldukca iyi bir ornek olan bu program, ucretsiz olan Visual C# 2008 Express Edition [7] kaynak kodunda degisiklikler yapilarak yeniden derlenebilir.



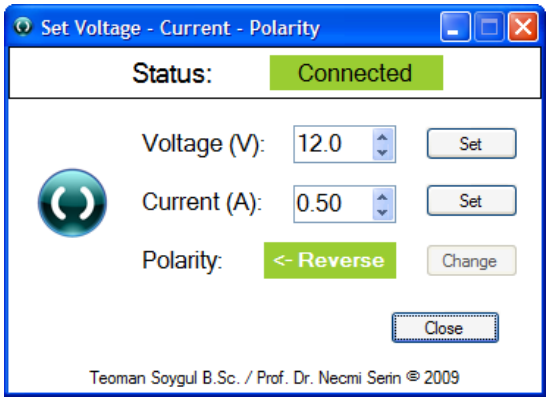
Şekil 4.2: Kontrol yazilimin en onemli parcasi olan SetVCP formunun sinif diyagrami.

Sekil 6.2 Prototip B Kontrol Paneli yazilimin en onemli parcasi olan SetVCP (Set Voltage Current Polarity) sinifi gorulmektedir. Bu sinif, Protocom sinifini kullanarak cihaz ile sorunsuz iletisimi saglamaktadir. Protocom sinifi ile kullanici tarafindan girilen komutlar arka planda islenirken, SetVCP sinifi tarafindan olusturulan kullanici arayuzu kullanarak yeni komutlar verilmeye devam edilebilir. Veri iletisim yolunun sinirli bant genisligine sahip olmasi nedeni ile veri yapisi olarak depolanan komutlar sirasi ile yerine getirilir. Butun bu komutlarin sonucunda olanlar ise CurrentStatus sinifi tarafindan takip edilerek kullaniciya grafik arayuzu ile gosterilir. Burada code reuse ‘a guzel bir ornek olarak Procom sinifinin iki farkli sinif tarafindan farkli amaclar ile kullanimi gorulmektedir. Butun bu sinif iletisimlerinin tek bir program altinda toplanmis haliyle kontrol yazilimi asagidaki seklini alir.

Şekil 4.3: Butunu ile Prototip B Kontrol Paneli yazilimi.

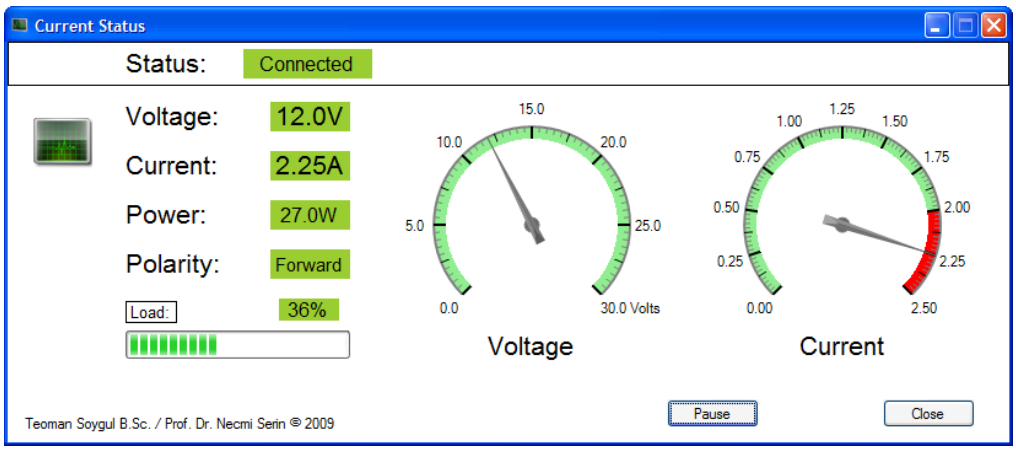
Sekilde gorulen ana ekran, diger butun modullerin biraraya getirildigi ve yardimci hizmetlerin verildigi ana formdur. Ekranin ortasinda gorulen butonlar, sistemin geri

kalan ozelliklerini kullanmak uzere alt modulleri acar. Ornegin Sekil 6.2 de sinif diyagrami gosterilen ve moduler yapisina deginilen Set Voltage Current Polarity modulunun grafik arayuzu asagidaki gibidir.



Şekil 4.4: SetVCP modulunun grafik ara yuzu.

Bu forma ersimek icin ana menuden yesille ikon ile gosterilen "Set Voltage – Current" tusuna basmak yeterlidir. Modul acildigi anda cihazin bilgisayara bagli olup olmadigini kontrol eder ve baglanti yapilmissa, devreye komut gondermek icin gereken alanlar aktiflesir. Bu modul ile verilen komutlarin sonucu, "Current Status" modulu ile Sekil 6.5 'de goruldugu uzere gozlemlenebilir. Burada asil yuk direnci uzerinden alinan olcumler grafiksel olarak gosterilmektedir.

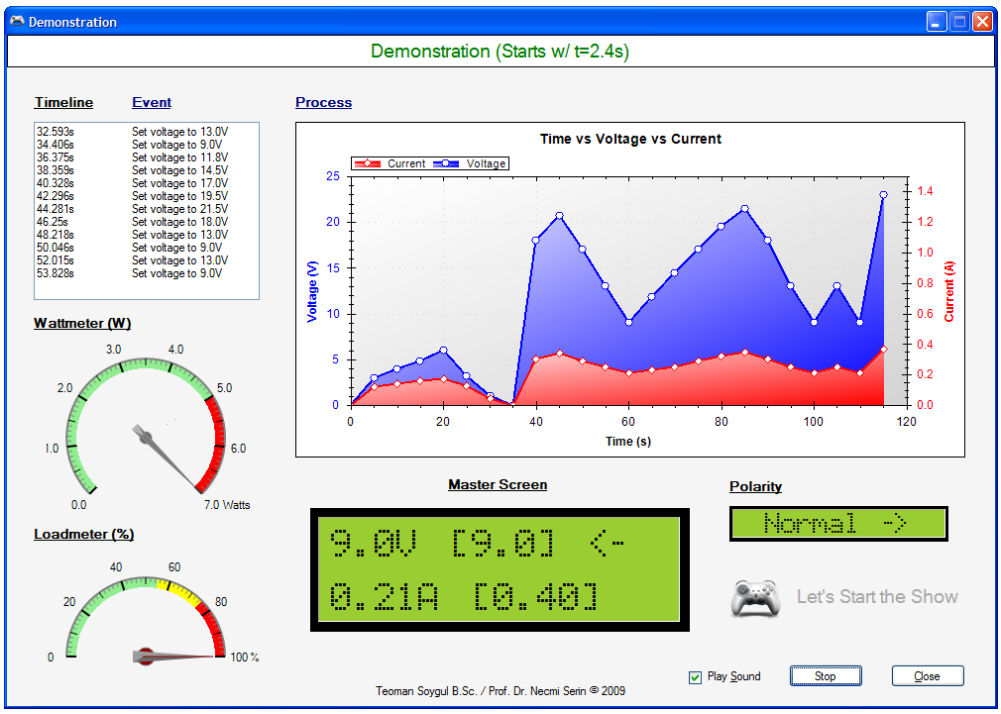


Şekil 4.5: Current Status modulunun grafik ara yuzu.

“Update Firmware” ve “Calibration” modullerinin kullanimi oldukca kolay oldugundan dolayi burada aciklanmamakla birlikte, yerlesik bellek guncellemelerinden sonra Sekil 6.6 ‘daki gibi “Hardware Info” modulu ile son yapilan degisiklikleri onamakta fayda faydir.

Şekil 4.6: Hardware Information modulunun grafik ara yuzu.

Gereken butun moduller hakkinda genis bilgiye “Help” menusunden kolaylikla ulasibilir. Bunun yaninda burda tanitilan son modul, devrenin yeteneklerini gostermek acisindan oldukca bilgilendirici olan “Demonstration” moduludur.



Şekil 4.7: Demonstration modulunun grafik ara yuzu.

Sekil 6.7’de gorulecegi uzere, bu modul uzerinde LCD panelde gorulen verilen gercek zamanli olarak takip edilebilir. Bu projenin replikasinin insaasi sirasinda bu modul gereken butun islevlik testleri icin kullanilmak icin tasarlanmistir. Gerektiginde farkli ayarlar ile farkli konfigurasyonlarin denemesine izin veren bu yazilim sayesinde, benzeri bir devrenin gelismis modellerinin yapimi veya benzeri islevi olan farkli devreler icin de kullanilabilir.

Bölüm 5 Sonuç

Bu deneysel calismanin sonucu itibari ile hedeflenen lineer guc kaynagi devresi beklenilenin cok otesinde bir performans ile calismaktadir. Bu proje ile geleneksel olarak analog parcalara atfedilen bircok gorevin tek basina bir mikrokontrolur ile yapilabilecegi gosterilmis olmaktadir. Devrenin kararliliginin yaninda cok onemli miktarda ozellistirilme imkani sunmasi, gelecek tasarimlar icin de bir referans teskil etmektedir. Ozellikle bilgisayar kontrolu, kontrol paneli yazilimi ile cok uyumlu calismakla birlikte, yerlesik bellek dahilindeki kalibrasyon ve guncelleme secenekleri cok uzun omurlu bir kullanimi olanakli kilmaktadir. Benzeri bir tasarim endustriyel amaclarla yapilacak olsa idi, bu secenekeler cok iyi servis imkanlarinin sunulmasina da olanak saglayacak oldugu kesindir. Genel itibari ile lineer guc kaynaklari icin mikrokonrolurlu mimari oldukca etkin olabilecegi gosterilmistir. Dijital kisim oldukca iyi iken, benzeri bir guc kaynaginin verimliligini maksimuma cikarmak icin daha yuksek frekanslarda calisan switch-mode uygulamalarina da gecmeye bir on hazirlik olabilir. Bu tasarimi gelecekte gelistirmek amaci ile ozellikle nispeten guncel olan switch-mode teknolojisi de entegre olarak kullanilabilir. Bu sayede ana hedeflerden biri olan minimum maliyet ve boyut amacina bir adim daha yaklasilabilir.

Ek 1 Uygulama Kodu

Bu kisimda, C dili ile yazilmis olup, mikroislemcinin butun islevlerini tanimlayan kod sunulmaktadir. Burada, islemcinin sistemin diger elemanlari ile iletisim kurmasi icin gereken kurallar, disaridan gelen uyarimlara vermesi gereken tepkiler ve sistemin genelinin islevi belirlenmektedir. Bu kod, butun kodun en kritik one sahip kismi oldugu icin burada sunulmustur. Butunu, oldukca fazla yer kaplayacak olmasindan dolayi ancak tez ile birlikte gelen CD icinde sunulmustur.

C Kodu

main.c

```
// Chip type           : ATmega8
// Clock frequency     : 4.0MHz (intenal oscillator)

#include <avr/io.h>
#include <inttypes.h>
#include <avr/interrupt.h>
#define F_CPU 4000000UL  // 4 MHz
#include <util/delay.h>
#include "lcd.h"
#include "dac.h"
#include "kbd.h"
#include "analog.h"
#include "avr_compat.h"
#include "hardware_settings.h"
#include "i2c_avr.h"
#include <stdlib.h> // atoi
#include <string.h>
#include <avr/eeprom.h>

#define SWVERSION "V1.2 (20.4.2009)"

static int set_val[2];
static int measured_val[2];
static char i2c_buf[MAX_BUF_LEN+1];
// the set values but converted to ADC steps
static int set_val_adcUnits[2];
static unsigned char bpress=0;
static int polarity=0; // polarity is normal

void delay_ms(unsigned int ms)
/* delay for a minimum of <ms> */
{
        // we use a calibrated macro. This is more
        // accurate and not so much compiler dependent
```

```
        // as self made code.
        while(ms){
                _delay_ms(0.96);
                ms--;
        }
}

// Convert a integer which is representing a float into a string.
// decimalpoint_pos sets the decimal point after 2 pos: e.g 74
becomes "0.74"
// The integer may not be larger than 10000.
// The integer must be a positive number.
// spacepadd can be used to add a leading speace if number is less
than 10
static void int_to_ascii(int inum,char *outbuf,signed char
decimalpoint_pos,signed char spacepadd){
      signed char i,j;
      char chbuf[8];
      j=0;
      while(inum>9 && j<7){
            // zero is ascii 48:
            chbuf[j]=(char)48+ inum-((inum/10)*10);
            inum=inum/10;
            j++;
            if(decimalpoint_pos==j){
                  chbuf[j]='.';
                  j++;
            }
      }
      chbuf[j]=(char)48+inum; // most significant digit
      decimalpoint_pos--;
      while(j<decimalpoint_pos){
            j++;
            chbuf[j]='0';
      }
      if (spacepadd && j > (decimalpoint_pos+2)){
            // no leading space padding needed
            spacepadd=0;
      }
      if(decimalpoint_pos==j){
            j++;
            chbuf[j]='.';
            j++;
            chbuf[j]='0'; // leading zero
      }
      if (spacepadd){
            j++;
            chbuf[j]=' '; // leading space padding: "9.50" becomes "
9.50"
      }
      // now reverse the order
      i=0;
      while(j>=0){
            outbuf[i]=chbuf[j];
            j--;
            i++;
      }
      outbuf[i]='\0';
}

// convert voltage values to adc values, disp=10 is 1.0V
```

```
static int disp_u_to_adc(int disp){
      return((int)(disp * 102.3) / (ADC_REF * U_DIVIDER));
}
// calculate the needed adc offset for voltage drop on the
// current measurement shunt (the shunt has about 0.5 Ohm =1/2 Ohm)
static int disp_i_to_u_adc_offset(int disp){
      return(disp_u_to_adc(disp/20));
}
// convert adc values to voltage values, disp=10 is 1.0V
// disp_i_val is needed to calculate the offset for the voltage drop
over
// the current measurement shunt
static int adc_u_to_disp(int adcunits,int disp_i_val){
      int adcdrop;
      adcdrop=disp_i_to_u_adc_offset(disp_i_val);
      if (adcunits < adcdrop){
            return(0);
      }
      adcunits=adcunits-adcdrop;
      return((int)(((adcunits /102.3)* ADC_REF * U_DIVIDER)+0.6));
}
// convert adc values to current values, disp=10 needed to be printed
// by the printing function as 0.10 A
static int disp_i_to_adc(int disp){
      return((int) (((disp * 10.23)* I_RESISTOR) / ADC_REF));
}
// convert adc values to current values, disp=10 needed to be printed
// by the printing function as 0.10 A
static int adc_i_to_disp(int adcunits){
      return((int) (((adcunits* ADC_REF)/(10.23 * I_RESISTOR))+0.6));
}

static void store_permanent(void){
      int tmp;
      signed char changeflag=1;
      lcd_clrscr();
      if (eeprom_read_byte((uint8_t *)0x0) == 19){
            changeflag=0;
            // ok magic number matches accept values
            tmp=eeprom_read_word((uint16_t *)0x04);
            if (tmp != set_val[1]){
                  changeflag=1;
            }
            tmp=eeprom_read_word((uint16_t *)0x02);
            if (tmp != set_val[0]){
                  changeflag=1;
            }
      }
      if (changeflag){
            lcd_puts_P("Settings Stored");
            eeprom_write_byte((uint8_t *)0x0,19); // magic number
            eeprom_write_word((uint16_t *)0x02,set_val[0]);
            eeprom_write_word((uint16_t *)0x04,set_val[1]);
      }else{
            if (bpress> 5){
                  // display software version after long press
                  lcd_puts_P(SWVERSION);
                  lcd_gotoxy(0,1);
                  lcd_puts_P("Teoman Soygul");
            }else{
                  lcd_puts_P("Already Stored");
```

```
            }
      }
      delay_ms(200);
}

// check the keyboard
static unsigned char check_buttons(void){
      if (check_u_button(&(set_val[1]))){
            if(set_val[1]>U_MAX){
                  set_val[1]=U_MAX;
            }
            return(1);
      }
      if (check_i_button(&(set_val[0]))){
            if(set_val[0]>I_MAX){
                  set_val[0]=I_MAX;
            }
            return(1);
      }
      if (check_store_button()){
            store_permanent();
            return(2);
      };
      return(0);
}

void check_i2c_interface(void){
      if (i2c_get_received_data(i2c_buf)){
            if (i2c_buf[0]=='i'){
                  if (i2c_buf[1]=='=' && i2c_buf[2]!='\0'){
                        set_val[0]=atoi(&i2c_buf[2]);
                        if(set_val[0]>I_MAX){
                              set_val[0]=I_MAX;
                        }
                        if(set_val[0]<0){
                              set_val[0]=0;
                        }
                        i2c_send_data("ok");
                  }else{
                        int_to_ascii(measured_val[0],i2c_buf,2,0);
                        strcat(i2c_buf,"A");
                        i2c_send_data(i2c_buf);
                  }
            }else if (i2c_buf[0]=='s'){
                  store_permanent();
                  i2c_send_data("ok");
            }else if (i2c_buf[0]=='p') {
                  // change polarity here
                  if (i2c_buf[1]=='=') {
                        if(i2c_buf[2]=='n'){
                              cbi(PORTB,PB3); // off so relay is off
and polarity is normal
                              polarity = 0;
                              i2c_send_data("ok");
                        }
                        else if(i2c_buf[2]=='r'){
                              sbi(PORTB,PB3); // on so relay is on
and polarity is reverse
                              polarity = 1;
                              i2c_send_data("ok");
                        }
```

```
                        else {
                              i2c_send_data("err");
                        }
                  }else {
                        if(polarity==0){
                              i2c_send_data("nrm");
                        }
                        else {
                              i2c_send_data("rev");
                        }
                  }
            }else if (i2c_buf[0]=='v'){
                  lcd_gotoxy(0,0);
                  lcd_puts_P(SWVERSION);
                  lcd_gotoxy(0,1);
                  lcd_puts_P("Teoman Soygul");
                  i2c_send_data(SWVERSION);
                  delay_ms(2000);
            }else if (i2c_buf[0]=='u'){
                  if (i2c_buf[1]=='=' && i2c_buf[2]!='\0'){
                        set_val[1]=atoi(&i2c_buf[2]);
                        if(set_val[1]>U_MAX){
                              set_val[1]=U_MAX;
                        }
                        if(set_val[1]<0){
                              set_val[1]=0;
                        }
                        i2c_send_data("ok");
                  }else{
                        int_to_ascii(measured_val[1],i2c_buf,1,0);
                        strcat(i2c_buf,"V");
                        i2c_send_data(i2c_buf);
                  }
            }else{
                  i2c_send_data("err");
            }
      }
}

int main(void)
{
      char out_buf[20+1];
      measured_val[0]=0;
      measured_val[1]=0;
      init_dac();
      lcd_init(LCD_DISP_ON);
      init_kbd();
      set_val[0]=15;set_val[1]=50; // 150mA and 5V
      if (eeprom_read_byte((uint8_t *)0x0) == 19){
            // ok magic number matches accept values
            set_val[1]=eeprom_read_word((uint16_t *)0x04);
            set_val[0]=eeprom_read_word((uint16_t *)0x02);
      }
      // I2C also called TWI, slave address=3
      i2c_init(3,1,0);
      sei();
      i2c_send_data("on");
      init_analog();

      while (1) {
```

```
            // current
            measured_val[0]=adc_i_to_disp(getanalogresult(0));
            set_val_adcUnits[0]=disp_i_to_adc(set_val[0]);
            set_target_adc_val(0,set_val_adcUnits[0]);
            // voltage

      measured_val[1]=adc_u_to_disp(getanalogresult(1),measured_val[0
]);

      set_val_adcUnits[1]=disp_u_to_adc(set_val[1])+disp_i_to_u_adc_o
ffset(measured_val[0]);
            set_target_adc_val(1,set_val_adcUnits[1]);

            // voltage
            lcd_clrscr();
            int_to_ascii(measured_val[1],out_buf,1,1);
            lcd_puts(out_buf);
            lcd_puts("V ");
            int_to_ascii(set_val[1],out_buf,1,1);
            lcd_putc('[');
            lcd_puts(out_buf);
            lcd_putc(']');
            if (!is_current_limit()){
                  // put a marker to show which value is currenlty
limiting
                  lcd_puts("<-");
            }

            check_i2c_interface();

            // current
            lcd_gotoxy(0,1);
            int_to_ascii(measured_val[0],out_buf,2,0);
            lcd_puts(out_buf);
            lcd_puts("A ");
            int_to_ascii(set_val[0],out_buf,2,0);
            lcd_putc('[');
            lcd_puts(out_buf);
            lcd_putc(']');
            if (is_current_limit()){
                  // put a marker to show which value is currenlty
limiting
                  lcd_puts("<-");
            }
            //dbg
            //int_to_ascii(is_dacval(),out_buf,0,0);
            //lcd_puts(out_buf);
            check_i2c_interface();

            // the buttons must be responsive but they must not
            // scroll too fast if pressed permanently
            if (check_buttons()==0){
                  // no buttons pressed
                  delay_ms(100);
                  bpress=0;
                  check_i2c_interface();
                  check_buttons();
                  delay_ms(150);
            }else{
                  // button press
                  if (bpress > 11){
```

```
                        // somebody pressed permanetly the
button=>scroll fast
                        delay_ms(10);
                        check_i2c_interface();
                        delay_ms(40);
                    }else{
                        bpress++;
                        delay_ms(100);
                        check_i2c_interface();
                        delay_ms(150);
                    }
                }
        }
        return(0);
}
```

Ek 2 Kontrol Paneli Kaynak Kodu

Bu kisimda sunulan kodu dosyasi C# dili ile yazilmistir. Grafik arayuzleri tasarlamak ve bircok Windows programlama teknigini derinlemesine kullanmak icin oldukca faydali olan bu dil ile, guc kaynagi ile bilgisayar arasindaki butun iletisim saglanmaktadir. Ticari yazilim kalitesinde oaln bu program ile, cihazin butun yonleri oldukca iyi tasarlanmis bir grafik arayuzu ile oldukca rahat bir sekilde kullanilabilir. Alet uzerindeki kontrol panelinde olmayan onlarda yeni ozellik de bu yazilim ile sunulmaktadir. Burada yazilimin sadece en kritik parcasi sunulmaktadir. Asli 13740 satir (275 sayfa) olan ve butunu benim tarafimdan yazilmis olan bu kod, butun haliyle tez ile birlikte gelen CD icinde sunulmustur.

C# Kodu

SetVCP.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace Prototype_B_Control_Panel.Forms
{
      public partial class SetVCP : Form
      {
            private Protocom protocom = new Protocom();
            // Declare the fields as public static for thread safety
            public static Queue setQ = new Queue();
            public static bool setVoltage, voltageProgress,
setCurrent, currentProgress, setPolarity, polarityProgress,
polarityRetrieved = false;
            // Local values which may not be set as of yet
            public static string localVoltageValue,
localCurrentValue, localPolarityValue = "";
            // Remote values which are real time device values
            public static string remoteVoltageValue,
remoteCurrentValue, remotePolarityValue = "";

            public SetVCP()
            {
                  InitializeComponent();
                  polarityRetrieved = false;
```

```
                setQ.Clear();
                setVoltage = false;
                setCurrent = false;
                setPolarity = false;
                polarityBackgroundWorker.RunWorkerAsync();
            }

            private void polarityBackgroundWorker_DoWork(object
sender, DoWorkEventArgs e)
            {

                while (polarityRetrieved == false)
                {
                    remotePolarityValue = protocom.GetPolarity();
                    localPolarityValue = remotePolarityValue;
                    if (localPolarityValue == "normal")
                    {
                        polarityRetrieved = true;

      polarityBackgroundWorker.ReportProgress(100);
                    }
                    else if (localPolarityValue == "reverse")
                    {
                        polarityRetrieved = true;

      polarityBackgroundWorker.ReportProgress(100);
                    }
                    else
                    {
                        polarityRetrieved = false;

      polarityBackgroundWorker.ReportProgress(0);
                    }
                }
            }

            private void
polarityBackgroundWorker_ProgressChanged(object sender,
ProgressChangedEventArgs e)
            {
                if (e.ProgressPercentage == 100)
                {
                    statusLabel.Text = "Connected";
                    statusLabel.BackColor = Color.YellowGreen;
                    if (remotePolarityValue == "normal")
                    {
                        polarity.Text = "Normal ->";
                    }
                    else
                    {
                        polarity.Text = "<- Reverse";
                    }
                    changePolarityButton.FlatStyle =
FlatStyle.Standard;
                    changePolarityButton.Enabled = true;
                    if(setQ.Count != 0)
backgroundWorker.RunWorkerAsync();
                }
                else if (e.ProgressPercentage == 0)
                {
                    statusLabel.Text = "Disconnected";
```

```
                        statusLabel.BackColor = Color.Red;
                  }
            }

            private void backgroundWorker_DoWork(object sender,
DoWorkEventArgs e)
            {
                  // First in first out
                  string s = "";
                  if (polarityRetrieved == false)
                  {
                        // If there is nothing in the queue, do
nothing
                  }
                  else
                  {
                        // Continue running while there is something
in the queue
                        while (setQ.Count != 0)
                        {
                              // Remove one from the queue
                              s = setQ.Dequeue().ToString();
                              if (s == "voltage")
                              {
                                    while (setVoltage)
                                    {
                                          setVoltage = false;
                                          // Progress stareted
                                          voltageProgress = true;
                                          if
(protocom.SetVoltage(localVoltageValue) == true)
                                          {
                                                // Job done

      backgroundWorker.ReportProgress(100, "voltage");
                                          }
                                          else
                                          {
                                                // Connection problem

      backgroundWorker.ReportProgress(0, "voltage");
                                          }
                                          // Progress completed
                                          voltageProgress = false;
                                    }
                              }
                              else if (s == "current")
                              {
                                    while (setCurrent)
                                    {
                                          setCurrent = false;
                                          // Progress stareted
                                          currentProgress = true;
                                          if
(protocom.SetCurrent(localCurrentValue) == true)
                                          {
                                                // Job done

      backgroundWorker.ReportProgress(100, "current");
                                          }
                                          else
```

```
                                        {
                                              // Connection problem

      backgroundWorker.ReportProgress(0, "current");
                                        }
                                        // Progress completed
                                        currentProgress = false;
                                  }
                            }
                            else if (s == "polarity")
                            {
                                  while (setPolarity)
                                  {
                                        setPolarity = false;
                                        // Progress stareted
                                        polarityProgress = true;
                                        if
(protocom.SetPolarity(localPolarityValue) == true)
                                        {
                                              // Job done

      backgroundWorker.ReportProgress(100, "polarity");
                                        }
                                        else
                                        {
                                              // Connection problem

      backgroundWorker.ReportProgress(0, "polarity");
                                        }
                                        // Progress completed
                                        polarityProgress = false;
                                  }
                            }
                      }
                }
            }

            private void backgroundWorker_ProgressChanged(object
sender, ProgressChangedEventArgs e)
            {
                  if (e.UserState.ToString() == "voltage")
                  {
                        if (e.ProgressPercentage == 100)
                        {
                              statusLabel.Text = "Voltage Set";
                              statusLabel.BackColor =
Color.YellowGreen;
                        }
                        else
                        {
                              ClearQueue();
                        }
                  }
                  else if (e.UserState.ToString() == "current")
                  {
                        if (e.ProgressPercentage == 100)
                        {
                              statusLabel.Text = "Current Set";
                              statusLabel.BackColor =
Color.YellowGreen;
                        }
```

```
                        else
                        {
                              ClearQueue();
                        }
                  }
                  else if (e.UserState.ToString() == "polarity")
                  {
                        if (e.ProgressPercentage == 100)
                        {
                              statusLabel.Text = "Polarity Set";
                              statusLabel.BackColor =
Color.YellowGreen;
                        }
                        else
                        {
                              ClearQueue();
                        }
                  }
            }

            private void ClearQueue()
            {
                  statusLabel.Text = "Disconnected";
                  statusLabel.BackColor = Color.Red;
                  setQ.Clear();
                  setVoltage = false;
                  setCurrent = false;
                  setPolarity = false;
                  localPolarityValue = remotePolarityValue;
                  if (localPolarityValue == "normal")
                  {
                        polarity.Text = "Normal ->";
                  }
                  else
                  {
                        polarity.Text = "<- Reverse";
                  }
            }

            private void closeButton_Click(object sender, EventArgs
e)
            {
                  this.Close();
            }

            private void setVoltageButton_Click(object sender,
EventArgs e)
            {
                  if (voltage.Value.ToString().Contains("."))
                  {
                        string tmp = (voltage.Value*10).ToString();
                        localVoltageValue = tmp.Substring(0,
tmp.Length - 2);
                  }
                  else
                  {
                        localVoltageValue =
(voltage.Value*10).ToString();
                  }
```

```
                  // If value is being transmitted to the device
already
                  if (voltageProgress == true)
                  {
                        setVoltage = true;
                  }
                  // If the value is already in the setQ
                  else if (setVoltage == true)
                  {
                        // Do nothing as the new value will be used
automaticly
                  }
                  // If none of the above, create a new queue entry
in setQ and run backgroundWorker
                  else
                  {
                        setQ.Enqueue("voltage");
                        setVoltage = true;
                        if (backgroundWorker.IsBusy == false)
backgroundWorker.RunWorkerAsync();
                  }
            }

            private void setCurrentButton_Click(object sender,
EventArgs e)
            {
                  if (current.Value.ToString().Length == 1)
                  {
                        localCurrentValue = (current.Value *
100).ToString();
                  }
                  else if
(current.Value.ToString().Substring(current.Value.ToString().Length -
2, 1) == ".")
                  {
                        localCurrentValue =
(Convert.ToDecimal(current.Value.ToString()) * 100).ToString();
                  }
                  else
                  {
                        string tmp = (current.Value *
100).ToString();
                        localCurrentValue = tmp.Substring(0,
tmp.Length - 3);
                  }

                  // If value is being transmitted to the device
already
                  if (currentProgress == true)
                  {
                        setCurrent = true;
                  }
                  // If the value is already in the setQ
                  else if (setCurrent == true)
                  {
                        // Do nothing as the new value will be used
automaticly
                  }
                  // If none of the above, create a new queue entry
in setQ
                  else
```

```
                  {
                        setQ.Enqueue("current");
                        setCurrent = true;
                        if (backgroundWorker.IsBusy == false)
backgroundWorker.RunWorkerAsync();
                  }
            }

            private void changePolarityButton_Click(object sender,
EventArgs e)
            {
                  if (localPolarityValue == "normal")
                  {
                        localPolarityValue = "reverse";
                        polarity.Text = "<- Reverse";
                  }
                  else
                  {
                        localPolarityValue = "normal";
                        polarity.Text = "Normal ->";
                  }

                  // If value is being transmitted to the device
already
                  if (polarityProgress == true)
                  {
                        setPolarity = true;
                  }
                  // If the value is already in the setQ
                  else if (setPolarity == true)
                  {
                        // Do nothing as the new value will be used
automaticly
                  }
                  // If none of the above, create a new queue entry
in setQ
                  else
                  {
                        setQ.Enqueue("polarity");
                        setPolarity = true;
                        if (backgroundWorker.IsBusy == false)
backgroundWorker.RunWorkerAsync();
                  }
            }
      }
}
```

Ek 3 Test Kodu

Bu kisimda, bu tezin deneysel kismi icin hem yazilim hem de donanim islevsellik testleri icin kullanilan yazilimin kaynak kodu bulunmaktadir. Bu kod, ayni zamanda kontrol paneli yazilimina da entegre edilerek kullanim kolayligi saglanmistir. Bolum 6'ya donulerek bu yazilimin dizayni ve kullanimi hakkinda daha genis bilgiye erisilebilecegi gibi, Bolum 8'de bu tez icin uygulanan testler, bir replika icin de derlenerek ayri bir yazilim olarak kullanilabilir. Bu kod da kontrol paneli modulleri ile uyumlu olarak C# ile yazilmis olup, derlemek icin Microsoft Visual Studio 2008 gereklidir [7]. (VS2008, Express Edition adi altinta ucretsiz olarak da sunulmaktadir). Testlerin manuel olarak tekrarlanmasi icin kod icindeki komutlar 'protocom' yazilimi ile ayrica kullanilabilir.

C# Kodu

Demonstration.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace Prototype_B_Control_Panel.Forms
{
      public partial class SetVCP : Form
      {
            private Protocom protocom = new Protocom();
            // Declare the fields as public static for thread safety
            public static Queue setQ = new Queue();
            public static bool setVoltage, voltageProgress,
setCurrent, currentProgress, setPolarity, polarityProgress,
polarityRetrieved = false;
            // Local values which may not be set as of yet
            public static string localVoltageValue,
localCurrentValue, localPolarityValue = "";
            // Remote values which are real time device values
            public static string remoteVoltageValue,
remoteCurrentValue, remotePolarityValue = "";

            public SetVCP()
            {
                  InitializeComponent();
```

```
                  polarityRetrieved = false;
                  setQ.Clear();
                  setVoltage = false;
                  setCurrent = false;
                  setPolarity = false;
                  polarityBackgroundWorker.RunWorkerAsync();
            }

            private void polarityBackgroundWorker_DoWork(object
sender, DoWorkEventArgs e)
            {

                  while (polarityRetrieved == false)
                  {
                        remotePolarityValue = protocom.GetPolarity();
                        localPolarityValue = remotePolarityValue;
                        if (localPolarityValue == "normal")
                        {
                              polarityRetrieved = true;

      polarityBackgroundWorker.ReportProgress(100);
                        }
                        else if (localPolarityValue == "reverse")
                        {
                              polarityRetrieved = true;

      polarityBackgroundWorker.ReportProgress(100);
                        }
                        else
                        {
                              polarityRetrieved = false;

      polarityBackgroundWorker.ReportProgress(0);
                        }
                  }
            }

            private void
polarityBackgroundWorker_ProgressChanged(object sender,
ProgressChangedEventArgs e)
            {
                  if (e.ProgressPercentage == 100)
                  {
                        statusLabel.Text = "Connected";
                        statusLabel.BackColor = Color.YellowGreen;
                        if (remotePolarityValue == "normal")
                        {
                              polarity.Text = "Normal ->";
                        }
                        else
                        {
                              polarity.Text = "<- Reverse";
                        }
                        changePolarityButton.FlatStyle =
FlatStyle.Standard;
                        changePolarityButton.Enabled = true;
                        if(setQ.Count != 0)
backgroundWorker.RunWorkerAsync();
                  }
                  else if (e.ProgressPercentage == 0)
                  {
```

```
                        statusLabel.Text = "Disconnected";
                        statusLabel.BackColor = Color.Red;
                  }
            }

            private void backgroundWorker_DoWork(object sender,
DoWorkEventArgs e)
            {
                  // First in first out
                  string s = "";
                  if (polarityRetrieved == false)
                  {
                        // If there is nothing in the queue, do
nothing
                  }
                  else
                  {
                        // Continue running while there is something
in the queue
                        while (setQ.Count != 0)
                        {
                              // Remove one from the queue
                              s = setQ.Dequeue().ToString();
                              if (s == "voltage")
                              {
                                    while (setVoltage)
                                    {
                                          setVoltage = false;
                                          // Progress stareted
                                          voltageProgress = true;
                                          if
(protocom.SetVoltage(localVoltageValue) == true)
                                          {
                                                // Job done

      backgroundWorker.ReportProgress(100, "voltage");
                                          }
                                          else
                                          {
                                                // Connection problem

      backgroundWorker.ReportProgress(0, "voltage");
                                          }
                                          // Progress completed
                                          voltageProgress = false;
                                    }
                              }
                              else if (s == "current")
                              {
                                    while (setCurrent)
                                    {
                                          setCurrent = false;
                                          // Progress stareted
                                          currentProgress = true;
                                          if
(protocom.SetCurrent(localCurrentValue) == true)
                                          {
                                                // Job done

      backgroundWorker.ReportProgress(100, "current");
                                          }
```

```
                                        else
                                        {
                                              // Connection problem

      backgroundWorker.ReportProgress(0, "current");
                                        }
                                        // Progress completed
                                        currentProgress = false;
                                  }
                            }
                            else if (s == "polarity")
                            {
                                  while (setPolarity)
                                  {
                                        setPolarity = false;
                                        // Progress stareted
                                        polarityProgress = true;
                                        if
(protocom.SetPolarity(localPolarityValue) == true)
                                        {
                                              // Job done

      backgroundWorker.ReportProgress(100, "polarity");
                                        }
                                        else
                                        {
                                              // Connection problem

      backgroundWorker.ReportProgress(0, "polarity");
                                        }
                                        // Progress completed
                                        polarityProgress = false;
                                  }
                            }
                      }
                }
            }

            private void backgroundWorker_ProgressChanged(object
sender, ProgressChangedEventArgs e)
            {
                  if (e.UserState.ToString() == "voltage")
                  {
                        if (e.ProgressPercentage == 100)
                        {
                              statusLabel.Text = "Voltage Set";
                              statusLabel.BackColor =
Color.YellowGreen;
                        }
                        else
                        {
                              ClearQueue();
                        }
                  }
                  else if (e.UserState.ToString() == "current")
                  {
                        if (e.ProgressPercentage == 100)
                        {
                              statusLabel.Text = "Current Set";
                              statusLabel.BackColor =
Color.YellowGreen;
```

```
                        }
                        else
                        {
                              ClearQueue();
                        }
                  }
                  else if (e.UserState.ToString() == "polarity")
                  {
                        if (e.ProgressPercentage == 100)
                        {
                              statusLabel.Text = "Polarity Set";
                              statusLabel.BackColor =
Color.YellowGreen;
                        }
                        else
                        {
                              ClearQueue();
                        }
                  }
            }

            private void ClearQueue()
            {
                  statusLabel.Text = "Disconnected";
                  statusLabel.BackColor = Color.Red;
                  setQ.Clear();
                  setVoltage = false;
                  setCurrent = false;
                  setPolarity = false;
                  localPolarityValue = remotePolarityValue;
                  if (localPolarityValue == "normal")
                  {
                        polarity.Text = "Normal ->";
                  }
                  else
                  {
                        polarity.Text = "<- Reverse";
                  }
            }

            private void closeButton_Click(object sender, EventArgs
e)
            {
                  this.Close();
            }

            private void setVoltageButton_Click(object sender,
EventArgs e)
            {
                  if (voltage.Value.ToString().Contains("."))
                  {
                        string tmp = (voltage.Value*10).ToString();
                        localVoltageValue = tmp.Substring(0,
tmp.Length - 2);
                  }
                  else
                  {
                        localVoltageValue =
(voltage.Value*10).ToString();
                  }
```

```
                  // If value is being transmitted to the device
already
                  if (voltageProgress == true)
                  {
                        setVoltage = true;
                  }
                  // If the value is already in the setQ
                  else if (setVoltage == true)
                  {
                        // Do nothing as the new value will be used
automaticly
                  }
                  // If none of the above, create a new queue entry
in setQ and run backgroundWorker
                  else
                  {
                        setQ.Enqueue("voltage");
                        setVoltage = true;
                        if (backgroundWorker.IsBusy == false)
backgroundWorker.RunWorkerAsync();
                  }
            }

            private void setCurrentButton_Click(object sender,
EventArgs e)
            {
                  if (current.Value.ToString().Length == 1)
                  {
                        localCurrentValue = (current.Value *
100).ToString();
                  }
                  else if
(current.Value.ToString().Substring(current.Value.ToString().Length -
2, 1) == ".")
                  {
                        localCurrentValue =
(Convert.ToDecimal(current.Value.ToString()) * 100).ToString();
                  }
                  else
                  {
                        string tmp = (current.Value *
100).ToString();
                        localCurrentValue = tmp.Substring(0,
tmp.Length - 3);
                  }

                  // If value is being transmitted to the device
already
                  if (currentProgress == true)
                  {
                        setCurrent = true;
                  }
                  // If the value is already in the setQ
                  else if (setCurrent == true)
                  {
                        // Do nothing as the new value will be used
automaticly
                  }
                  // If none of the above, create a new queue entry
in setQ
                  else
```

```
                  {
                        setQ.Enqueue("current");
                        setCurrent = true;
                        if (backgroundWorker.IsBusy == false)
backgroundWorker.RunWorkerAsync();
                  }
            }

            private void changePolarityButton_Click(object sender,
EventArgs e)
            {
                  if (localPolarityValue == "normal")
                  {
                        localPolarityValue = "reverse";
                        polarity.Text = "<- Reverse";
                  }
                  else
                  {
                        localPolarityValue = "normal";
                        polarity.Text = "Normal ->";
                  }

                  // If value is being transmitted to the device
already
                  if (polarityProgress == true)
                  {
                        setPolarity = true;
                  }
                  // If the value is already in the setQ
                  else if (setPolarity == true)
                  {
                        // Do nothing as the new value will be used
automaticly
                  }
                  // If none of the above, create a new queue entry
in setQ
                  else
                  {
                        setQ.Enqueue("polarity");
                        setPolarity = true;
                        if (backgroundWorker.IsBusy == false)
backgroundWorker.RunWorkerAsync();
                  }
            }
      }
}
```

Kaynaklar

[1] *Atmel AVR ATmega8 PDIP*. Atmel, 2009.
< http://en.wikipedia.org/wiki/File:ATmega8_01_Pengo.jpg >

[2] Anonim. *Crowbar (circuit)*. Wikipedia the Free Encyclopedi, 2009.
< http://en.wikipedia.org/wiki/Crowbar_(circuit) >

[3] *Proteus VSM Suite v7.4 SP3.* Labcenter Electronics, 2009.
< http://www.labcenter.co.uk/ >

[4] *IBM Rational Rose Enterprise v2003.06 r15*. IBM, 2008.
< http://www.ibm.com/software/rational/rose >

[5] *Atmel AVR Studio v4.16.* Atmel Corporation, 2009.
< http://www.atmel.com/products/AVR/ >

[6] *WinAVR v2009.03.13.* WinAVR Project, 2009.
< http://winavr.sourceforge.net/ >

[7] *Microsoft Visual Studio 2008 SP1*. Microsoft Corporation, 2009.
< http://go.microsoft.com/fwlink/?LinkId=88616 >

[8] Base Digital Regulatory Circuit, Guido Socher, 2005. < http://www.linuxfocus.org/English/September2005/article389.shtml >

[9] Ideas on polarity switch, macro record & replay functionality. Prof. Necmi Serin, 2009.