EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

EĞİTİM AMAÇLI KULLANILACAK BİR ÜÇ EKSENLİ DİK İŞLEM CNC MAKİNASI DİZAYNI VE İMALATI

Nail AKÇURA

Tez Danışmanı : Doç. Dr. Musa ALCI

Mekatronik Anabilim Dalı

Sunuş Tarihi : 17.06.2015

Bornova-İZMİR

2015

Nail AKÇURA tarafından Yüksek Lisans tezi olarak sunulan "Eğitim Amaçlı Kullanılacak Bir Üç Eksenli Dik İşlem CNC Makinası Dizaynı ve İmalat" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 17.06.2015 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**:** Doç. Dr. Musa ALCI	**.....**
Raportör Üye	**:** Yrd. Doç. Dr. Mustafa ENGİN	**.....**
Üye	**:** Yrd. Doç. Dr. Levent ÇETİN	**.....**

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "Eğitim Amaçlı Kullanılacak Bir Üç Eksenli Dik İşlem Cnc Makinası Dizaynı Ve İmalatı" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

… /… / 2015

İmzası

Nail AKÇURA

ÖZET

EĞİTİM AMAÇLI KULLANILACAK BİR ÜÇ EKSENLİ DİK İŞLEM CNC MAKİNASI DİZAYNI VE İMALATI

AKÇURA, Nail

Yüksek Lisans Tezi, Mekatronik Anabilim Dalı
Tez Danışmanı: Doç. Dr. Musa ALCI
Haziran 2015, 74 sayfa

Günümüzde gerçekleştirilen üretimde torna, freze gibi geleneksel işlem makinaları üretim süresi, hız, maliyet, hassasiyet gibi avantajlarından dolayı otomatikleştirilerek CNC (Computer Numerical Controller) adını verdiğimiz makinalar haline getirilir. Bilgisayar destekli çizim (Computer Aided Drawing) programı ile tasarlanan ürünler bilgisayar destekli üretim (Computer Aided Manufacturing) ile bir üretim dosyası çıkartılarak ürünün ham materyalden işlenme yöntemi düzenlenir. Yardımcı programlar ile bu üretim dosyasında yer alan işlem kodları işlenerek CNC makinası üzerindeki motorlar kontrol edilir.

Bu tezde, üniversite eğitimine yardımcı olması amacıyla bir CNC tezgahı imal edilmiştir. CAM dosyasını işleyecek hazır bir program kullanılmak yerine Visual C# dilinde yeni bir yazılım yazılmış, böylece komut listeleri kullanıcı tarafından kontrol edilebilmiştir. Bilgisayar ara yüzünde olduğu gibi, ARM mikroişlemci mimarisi tabanlı Texas Instruments Stellaris LM3S811 geliştirme kartında da sık kullanılan komutlar için algoritmalar geliştirilmiştir. Uygulamalı eğitim olarak hem makina mühendisi hem elektronik mühendisi öğrencilerine öğretim materyali bir sistem haline getirilmiştir.

Anahtar sözcükler: CNC, imalat, CAD, CAM, 3 eksenli CNC.

ABSTRACT

DESIGN AND MANUFACTURING OF A 3-AXES VERTICAL CNC AT THE SERVICE OF EDUCATION

AKÇURA, Nail

MSc in Mechatronics Eng.
Supervisor: Assoc. Prof. Dr. Musa ALCI
June 2015, 74 pages

In today's manufacturing, held by traditional tool machines like lathes and milling machines, are advanced to automatic CNC (Computer Numerical Control) machines due to machining time speed, cost, and precision advantages. The products designed with Computer-aided design (Computer Aided Drawing) are progressed with the method of processing raw material via a production file created by computer-aided manufacturing software (Computer Aided Manufacturing). The motors on the machine are controlled using this production file with op-code processor software.

In this thesis, a CNC machine has been manufactured to aid university studies. Instead of using a ready-to-use product software for CAM process, a new software program was written in "Visual C#" language, thus the command list can be controlled by the user. Algorithms for frequently used commands were developed on ARM microprocessor architecture based Texas Instruments Stellaris LM3S811 evaluation board as we did in the computer interface. It fulfils the role of a teaching material for both mechanical and electronical engineering students.

Keywords: CNC, manufacturing, CAD, CAM, 3 axes CNC.

TEŞEKKÜR

Bu çalışma süresince bana maddi ve manevi olarak destek sağlayan tez danışmanım Doç. Dr. Musa ALCI'ya, değerli hocam Prof. Dr. Erol UYAR'a, Yar. Doç. Dr. Mustafa ENGİN'e, yine maddi ve manevi desteklerini esirgemeyen canım aileme, projemde yardımlarını esirgemeyen meslektaşlarım ve arkadaşlarım Mücahid CANDAN, Doğan Can KARATAŞ, Alican ALPKAYA, S. Alper OĞAN ve Ekrem YAVUZ'a teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

ŞEKİLLER DİZİNİ (devam)

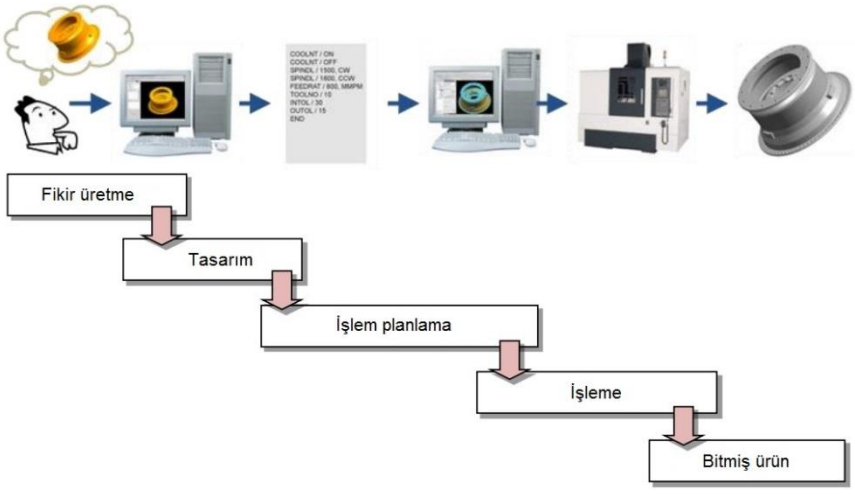
SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
AC	Alternative Current (Alternatif akım)
ARM	Acorn RISC Machine
CAD	Computer Aided Drawing (Bilgisayar destekli çizim)
CAM	Computer Aided Manufacturing (Bilgisayar destekli üretim)
CNC	Computer Numeric Controller (Bilgisayar sayımlı yönetim)
DC	Direct Current (Doğru akım)
IGBT	Insulated Gate Bipolar Transistor
MİB	Merkezi İşlem Birimi
MOSFET	Metal Oxide Semiconductor Field Effect Transistor (Metal Oksit Yarıiletkenli Alan Etkili Transistör)
NC	Numeric Controller (Sayımlı yönetim)
NVIC	Nested Vectored Interrupt Controller)İç içe Vektörlenmiş Kesme Denetleyicisi)
PLC	Programmable Logic Controller
PWM	Pulse Width Modulation

1. GİRİŞ

20. yüzyılın ikinci yarısından itibaren sanayide üretim, hızla gelişen endüstriyel teknolojilerin paralelinde seri üretime doğru kaymıştır. Bununla birlikte firmalar büyümüş, rijitleşmiş ve dinamik olan küresel talebe karşılık veremez hale gelmişlerdir. Bu yüzden firmaların odağı da müşteri odaklı, özel, yüksek kaliteli ve bol seçenekli üretim halini almıştır (Balic et al., 2006).

Konvansiyonel imalatın bel kemiği olan torna, freze gibi talaşlı imalat yöntemleriyle üretimde, endüstriyel gelişimle birlikte daha düşük insan faktörü, daha hızlı ve daha hassas üretim gibi taleplerin karşılaması amacıyla CNC (Bilgisayar Sayımlı Yönetim) (Computer Numerical Controller) makinaları geliştirilmiştir. Bu makinalar basitçe otomatikleştirilmiş imalat makinaları olarak özetlenebilir. CNC makinalar dünya çapında yaygınlaşmış ve endüstriyel üretimin vazgeçilmezi haline gelmiştir. CNC makinası kullanarak gerçekleştirilecek bir imalatta iş planı ele alındığında basitçe Şekil 1.1'deki gibi fikir üretiminden bitmiş ürün faslına kadarki aşamalar gösterilmiştir. Tasarımcının fikrini ortaya koyarak tasarımı gerçekleştirmesi, işlem için talimatları üretmesi ve son olarak iş parçasını işlemesi şeklinde üretim en basit şekilde özetlenebilir.



Şekil 1.1 Üretim aşamaları akışı (Anderberg, 2012).

Donanıma bakacak olursak, işleme aşamasında, makinaya otomatik kontrolü sağlamak amacıyla, makinaya direktif veren bütünleşik bir bilgisayar sistemi vardır. İşlem ve hareket direktifleri belirten komutlar G kodları olarak adlandırılır. Bilgisayar ve makine arasında komut işleme ve gönderme, haberleşme, çevre birimleri yönetme gibi işlemleri gerçekleştirmesi amacıyla mikroişlemci tabanlı

bir ara yüz kartı kullanılır. Bu kart da G kodlarını işleyerek makinanın işlem yapmasını sağlar. Bu anlamda disiplinler arası çalışma önem kazanmaktadır. Sistemde mekanik, elektrik, elektronik, bilgisayar ve kontrol dallarında öğeler bulunup bütünleşik olarak sistem tasarlanmış ve uygulanmıştır.

Bu tez çalışmasında, adım motorlara sahip bir üç eksenli dik işlem CNC makinası tasarlanmıştır. Bilgisayarda Visual C# programı ile insan ile bilgisayar arasında haberleşmeyi sağlayan bir ara yüz programı oluşturulmuştur. Makine ile bilgisayar arasında kontrolör ve ara yüz kartı olarak ARM işlemci tabanlı bir mikro denetleyici kart (Texas Instruments Stellaris LM3S811 geliştirme kartı) programlanmıştır. Projenin asıl amacı, açık kaynak bir sistem oluşturulup, hem elektrik-elektronik, hem de makine mühendisliği öğrencilerinin faydalanabileceği, eğitim görebileceği ve daha da geliştirebileceği açık mimari bir proje ortaya çıkarılmıştır.

1.1 Tez Ana Hatları

İkinci bölümde, çalışmayla ilgili literatür ve piyasa taramasıyla elde edilen bilgi ve sonuçlar literatür taraması olarak verilmiştir.

Üçüncü bölümde, CNC makinaların tanımı, tarihçesi, temel elemanları ve özellikleri incelenmiş, CNC programlama dili olan G-kodlarıyla ilgili temel detaylar ve örnek program çıktısı verilmiştir.

Dördüncü bölümde, mikrodenetleyicilerin tanımı, mikro işlemcilerin tarihi, temel çevre birimler, Cortex-M3 yapısal özellikleri ve Stellaris LM3S811 geliştirme kartının özelllikleri incelenmiştir.

Beşinci bölümde, C# programlama dili, tarihi, nesne yönelimli programlama hakkında incelemeler yapılmış, .NET yapısı, .NET çatısı altındaki bileşenler, Visual C# hakkında bilgiler verilmiş, program oluştururken oluşacak proje unsur ağacı incelenmiştir.

Altıncı bölümde, projenin oluşturulması aşamasındaki mekanik tasarım, elektriksel bağlantı, bilgisayar arayüzü ve elemanlarının görevleri, çalıştırma modları ele alınmıştır.

Yedinci bölümde, çalışmanın bir uygulamasına yer verilmiştir.

Sekizinci bölümde, çalışmaya ait sonuçlar ve tartışma bölümüne yer verilmiştir.

2. LİTERATÜR TARAMASI

Minhat et al. (2009), IEC 61499 standardındaki fonksiyon bloklarını kullanarak CNC algoritmaları kurmuş ve gömülü sistemlere uyarlamıştır.

Zuo et al. (2000), açık mimari bir CNC sistemi için belirli fonksiyonları çalıştıran SIC (Software Integrated Circuit) entegre programlar kullanarak bir platform tasarlamıştır.

Ma et al. (2006), nesne yönelimli programlama ile yine açık kaynak CNC sistemi geliştirmişlerdir.

Yuhan et al. (2003), açık mimari olarak çoklu görev üzerine çekirdek (kernel) geliştirmişlerdir.

Zhang et al. (2003), Linux işletim sistemi üzerinde açık mimari bir CNC yazılımı geliştirmişlerdir.

Morales-Velazquez et al. (2010), FPGA tabanlı çok ajanlı açık mimari CNC yazılımı geliştirmişlerdir.

Bir CNC makina mekanik, elektrik, elektronik, yazılım, kontrol ve mekatronik alanlarını kapsadığından interdisipliner bir alan olarak ele alınır. Bölüm 2.1'de mekanik gövde, hareket mekanizması ve uç işlevci; Bölüm 2.2'de kontrolör; Bölüm 2.3'te elektrik makinaları ve güç elektroniği; Bölüm 2.4'te hareket kontrolü; Bölüm 2.5'te yazılım konuları ele alınmıştır.

2.1 Mekanik

Piyasada farklı güç ve boyutlara sahip CNC makinalar mevcuttur. Masaüstünde durabilecek düşük boyutlu çalışma bölgesine sahip CNC makinalar olabileceği gibi, bir oda büyüklüğü ya da daha büyük boyutlara da ulaşabilir. İşlenecek materyalin cinsi ve boyutuna göre CNC tezgâhlar uygun işleme için sayılabilecek bazı tiplere sahiptirler. CNC makinalarda öne çıkan en dikkat çekici özellikler hassasiyet ve güvenilirliktir. Elemanların seçimi, tüm bu özellikler dikkate alınarak yapılır.

Dik işlemle işlenecek olan materyale göre boyutu ve gücü değişen CNC makinalarda, dayanımı düşük tahta, delrin gibi malzemeler için CNC Router olarak isimlendirilen, iş parçasının sabit ve uç işlevcinin hareketli bir araba üzerinde hareket ettiği sistemler tercih edilir. Maliyeti düşüktür. Metal gibi dayanımı yüksek ve işlemesi zor malzemeler için daha rijit gövdeli, çelik yapılı makinalar tercih edilir.

Hareket mekanizması yaygın olarak dişli sistem, lineer mil, kayış kasnak sistemleri gibi aktarma organlarıyla aktarılır. Bu aktarma organının seçiminde de hassasiyet ve taşınacak yük önem arz etmektedir. Hassasiyet açısından öne çıkan dezavantajlardan biri boşluktur (backlash). Dişli, zincir gibi elemanların mekaniğinden kaynaklı bazı boşluklar meydana gelmektedir ve özellikle harekete başlama noktalarında boşluk yapıları çok düşük hata paylı kontrole izin vermemektedir.

Yine materyale ve işleme kapasitesine göre freze motoru da farklılık gösterir. Kalıpçı taşlama motoru ile tahta ve benzeri malzemeleri işlemekte kullanılabilirken, 20-30 mm kalınlıktaki sert metalleri işlemekte yetersiz kalmakta ve güçlü bir freze motoruna ihtiyaç duyulmaktadır. Lazer kesim gibi makinalarda ise yine malzeme tipi ve kalınlığı, lazer ünitesinin gücünü ve tipini seçmekte önemli unsurlardır.

2.2 Kontrolör

Günümüz teknolojisinde NC sistemlerin neredeyse tamamında bilgisayar yada mikrodenetleyici tabanlı sistemler kullanılmaktadır.

PLC (Programlanabilir Mantık Kontrolörü)(Programmable Logic Controller) en yaygın olan sistemlerden biri olarak profesyonel makinalarda tercih edilen kontrolördür. Endüstriyel voltajda çalışıp, motor sürücü üniteleri ve sensörler gibi modüllerle bağlantı ve kontrolü kolaydır.

CNC makinalar üzerinde yer alan bilgisayarlar ise üzerlerinde çalışan bir işletim sistemi üzerinde özel bir program koşturularak hem kullanıcı arayüzü hem de kontrolör olarak görev görmektedir.

Ucuz bir çözüm isteyen üreticiler ise bilgisayara bağlanabilen ve motor kontrollerini de gerçekleştirebilen, mikroişlemci tabanlı arayüz kartları tercih

etmektedirler. Mach3 firmasının ürettiği kart da bunlara örnektir. Bu kart, lisanslı Mach3 bilgisayar programı ile de uyum içerisinde çalışır. ARM tabanlı mikro işlemciler, yüksek performans ve düşük güç tüketimleriyle günlük hayatımıza dahi girmiştir. Çalışmada tercih edilen kontrolördür.

2.3 Elektrik Makinaları ve Güç Elektroniği

CNC sistemlerinde en sık kullanılan iki tip motor vardır: Servo motor ve adım motor. Servo motorlar bütünleşik yapıları ve adım motora göre daha performanslı olmalarından dolayı tercih edilir, sürücüleriyle birlikte fiyatları eşdeğer adım motora göre çok daha fazladır. Ucuz çözümler için adım motorlar birebirdir. Sürüş teknikleri sayesinde yüksek hassasiyete sahiptir. Servo motor performansından dolayı profesyonel işlerde servo motor tercih edilmektedir. Bu konu Bölüm 3.4'te detaylı olarak incelenmiştir.

Motor sürücüleri kapasite ve yeteneklerine göre geniş bir yelpazede seçenek sunmaktadır. 1950'lerden sonra yarı iletken malzeme teknolojisinin gelişmesiyle güç elektroniği büyük bir atlama yapmıştır. Transistör, tristör, güç MOSFETi (Power MOS Field Transistor), IGBT (Insulated Gate Bipolar Transistor) gibi elemanlar, motor sürücülerde güç elektroniği elemanı olarak kullanılmaktadır. Her bir elemanın kendine has özellik ve verimi vardır. Güç MOSFETleri piyasada büyük bir paya sahiptir. Yüksek frekans, yüksek gerilim ve düşük güçlü uygulamalarda ucuz, yüksek verimlikli ve kolay sürülebilir olmasından dolayı tercih edilir. IGBT'ler, yüksek frekans, yüksek gerilim, yüksek güçlü uygulamalarda yüksek verim ve hızlı anahtarlama sağlayabilir. Bu güç elektroniği elemanları gerilim ve frekans kontrolünde kullanılırken, motor sürücülerde genellikle tercih edilen elemanlardır (Ay, 2004).

Mikro adım adım motor sürücüleri, adım motorlarda düşük maliyetli ve yüksek performanslı bir sürücü seçeneğidir. Konu Bölüm 3.4.4.2'de detaylı olarak incelenmiştir.

2.4 Hareket Kontrolü

Literatürde hareket kontrolüne dair birçok metod yer almaktadır. Buna göre temelde hız, pozisyon, tork değişkenleri kontrol edilmek istenen değişkenlerdir.

En yaygın yöntemlerden birisi kaskad hız kontrollü pozisyon kontrolüdür. Bu kontrolün sağlanabilmesi için sürekli olarak pozisyon sensöründen pozisyon bilgisini alınması gerekmektedir. Dış döngü olarak pozisyon hedefine varmaya çalışırken, diğer yandan bir iç döngü olarak da hız kontrolünü sağlar. Döngüler, hataya göre sürülen kontrol sistemleridir. İç döngüdeki hız kontrolünü sağlamakta en yaygın metodlardan PI ve PID, oransal, integral ve türevsel kontrol girdi ve katsayılarıyla kontrolü sağlamaktır. Buna benzer şekilde tork kontrolü de sağlanabilir. Pozisyon sensörünün bulunmadığı durumlarda bir gözlemci tasarlayarak kontrol etmek de mümkündür. Daha gelişmiş sürücülerde bulanık mantık, adaptif ve yapay sinir ağlarıyla hız kontrolü de gerçekleştirilmektedir. (Ay, 2004).

Adım motorlar, sürüş tekniği olarak geri beslemeye ihtiyaç duymadan fazlarının sıralı bir şekilde tetiklenmesiyle dönme miktarı sabit olan motorlardır. PWM (darbe genişlik modülasyonu)(Pulse Width Modulation) ile dönme yönü girdisi alarak sıranın ilerlemesini sağlayan motor sürücüleri mevcuttur. Saniyede gelen dalga sayısı ve dönme yönünün ayarlanmasıyla motor hızı ve yönü kolaylıkla kontrol edilebilir.

2.5 Yazılım

CNC yazılımı, komut setine göre komutları alarak gerekli fonksiyonları fiziksel olarak gerçekleştirmeye çalışan kontrolör olarak görülebilir. İki kısımdan oluşur: Bilgisayar ve kontrolör yazılımı.

Bilgisayar arayüzü herhangi bir programla yazılabileceği gibi, yazılımın interaktif bir yapıya sahip olması, arka plandaki kontrolör yazılım için önemlidir. Visual Basic, Visual C#, Java gibi form uygulamalarının kolayca yazılıp geliştirilebileceği diller bir CNC makine geliştirmek için yeterlidir. Windows, Linux işletim sistemleri bu sistemler için tercih edilen işletim sistemleridir.

Kontrolör yazılımı ise kontrolörün tipine göre değişiklik gösterebilir.

3. CNC

3.1 NC Tezgâhlar

Nümerik Kontrollü (NC)(Numerical Controller) tezgâhlar, konvansiyonel tezgâhların elektronik elemanlar yardımıyla otomatik kontrol edebilir hale getirilen makinalardır. Nümerik kontrolör, bir iş parçasını işlemek üzere makine üzerindeki komutlar doğrultusunda kontrolünü gerçekleştiren kontrol cihazıdır. Eğer bu makine, kontrol amaçlı olarak bir bilgisayar teknolojisiyle donatılmış ise, Bilgisayar Sayımlı Yönetim (CNC) (Computer Numerical Control) adını alır (Suh et al., 2008). CNC'ler, üzerlerindeki bilgisayarlarla G-kodları dediğimiz komutları işleyerek, makine üzerinde yer alan servo tabanlı motor ve aktüatörleri kontrol ederler ve iş parçasını işlerler.

CNC makinalar günümüz endüstrisinde birçok makinada kullanılmaktadır. Talaşlı imalat olarak adlandırılan, iş parçasından malzeme kaldırma veya eksiltme yöntemine dayanan torna, freze gibi konvansiyonel makinalar ve plazma, lazer kesim gibi sıradışı işleme yöntemlerine dayanan makinalar CNC haline getirilerek işlem hızlandırılır ve hassaslaştırılır. Sadece talaşlı imalat değil, pres makinası, kaynak gibi talaşsız imalat yöntemlerinin kullanıldığı makinalar da CNC makine haline getirilebilir. Masaüstü tipinden büyük boyutlu makinalara değişik boyutlarda CNC makinalar üretilmektedir. Bazı NC makine tipleri Şekil 3.1'de verilmiştir.

Şekil 3.1 a) Robot manipülatör, b) Freze, c) Torna makinası (Suh et al., 2008).

Kullanım alanları ise endüstriyel üretim, otomotiv, tarım, gıda, baskı ve matbaa gibi en temel noktalar başta gelirken, günlük yaşam ve ileri teknoloji ürünlerinin neredeyse hepsinin üretimlerinde kullanıldığından birçok sektörde kendine yer edinmiştir.

3.2 CNC Tezgâhların Tarihçesi

İlk NC makina, 1940'ların sonu ve 1950'lerin başlarında, yani 2. Dünya Savaşı'ndan sonra, MIT (Massachusetts Institute of Technology) işbirliğiyle John T. Parson tarafından geliştirilmiştir. Geliştirmedeki hedef ise, giderek daha karmaşık bir hal alan uçak ve hava araçlarının parçalarının üretiminde operatörlerin, ihtiyaç duyulan yüksek hassasiyetli üretim dolayısıyla yetersiz kalmasından çözüm üretmekti. İlk üretilen makinalarda fonksiyonlar yazılımsal olarak değil, donanımsal olarak vakum tüpleri, transistörler ve büyük boyutlu mantık devreleriyle sayıca yüksek miktarlarda kullanılarak gerçekleştirilmiştir. Bu da boyut olarak çok büyük bir yer kaplamaktaydı. Aktüatör olarak yağ basınçlı motor ve röleler kullanılmıştır. Komutlar, geliştirilen bir delikli kart sistemiyle makinaya verilmiştir. 1970'ler ve 80'lerde yarıiletken ve mikroişlemci teknolojilerinin gelişmesiyle bilgisayarların kullanımı yani CNC makinaya geçiş başlanmış, donanımsal NC yerini yazılımsal NC makinalara bırakmıştır. Kullanılan elemanlar da mikroişlemciler, elektronik teknolojiler, elektrik makinaları ve yazılıma dönmüştür (Albert, 2008; Suh et al., 2008).

3.3 CNC Sistemlerin Avantajları

Bir CNC makineyi diğer makinalardan ayıran en büyük özellik, otomatik olmasıdır. Bu nedenle işleme kalitesi, bir operatör ustasının yeteneğine, dikkatine ve hatalarına bağımlı değildir. Kalifiye insan gereksinimi yoktur. İşlem süresi de bir iş için neredeyse bellidir.

Müşterilerin giderek artan yüksek kalite ve hassasiyet ihtiyacını düşük hata payı ile süreklilik özelliği karşılayabilir. Makinanın performansı maksimize edebildiğinden, seri üretimde en kısa zamanda en yüksek verim ve miktarda üretim gerçekleştirilebilir. Bir iş parçasının üretiminde makine, yüzlerce defa çalışması halinde bile yine aynı hassasiyette işlem yapabilir. Üretim miktarı göz önüne alındığında üretim maliyetini azaltır (Kao et al., 2006).

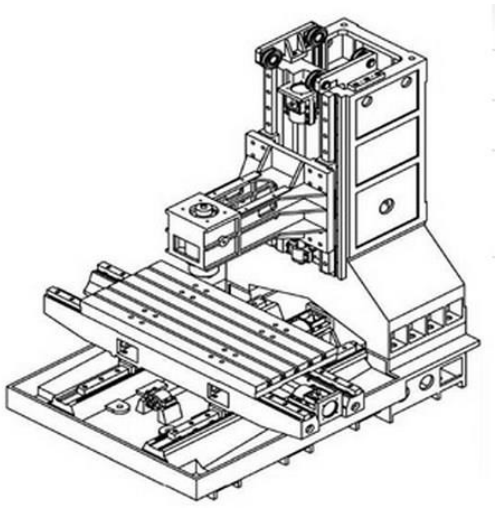
Makinanın esnekliği sayesinde, verilen G kodları değiştirilerek aynı ya da farklı bir iş parçasını farklı şekillerde işlememek mümkündür ve üretici her özel iş için farklı bir makineye ihtiyaç duymaz.

3.4 CNC Sistemlerin Temel Elemanları

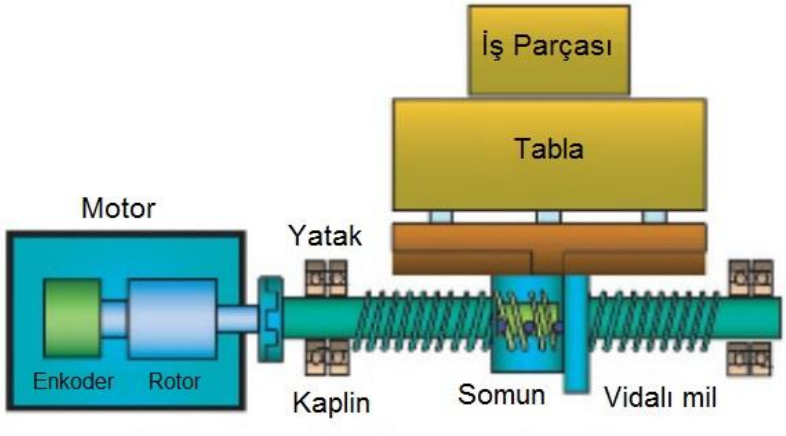
CNC makinalar birçok farklı makine tipini kapsasa da, genel olarak bu makinalarda temel sistem elemanları şöyledir: Mekanik sistem, bilgisayar, ara yüz kartı, aktüatörler, aktüatör sürücüleri, sensörler ve uç işlevcidir.

3.4.1 Mekanik sistem

CNC sistemlerde mekanik sistem makine gövdesi, tabla, hareketli eksenler ve bağlantı parçaları olarak tanımlanır. Gövde, diğer bütün elemanların üzerine bağlanacağı ana iskelettir, demir, alüminyum gibi materyaller dayanıklılıklarından dolayı gövdede tercih edilen materyallerdir. Tabla, iş parçasının konulacağı zemindir. Gövdeye monte edilir. İş parçasının üzerine bağlanabilmesi için bağlama aparatı ve bağlantı noktaları yer alabilir, yapılacak işe göre değişik özellikte olabilir. Hareketli eksenler, iş parçasının işlem görebilmesi için uç işlevcinin veya tablanın pozisyonunu değiştirmeye yarayan mekanizmalardır. Bunu genellikle, tahrik elemanından aldığı hareketi hareketli bir arabaya aktarıp pozisyonunu değiştirerek sağlar. Triger kayış ve kasnaklar, lineer mil kullanılan sistemler en yaygın hareket aktarma organlarıdır. Bağlantı parçaları ise gövdenin montaj ara parçaları, motor mili bağlantı parçaları gibi parça ve hırdavatlardan oluşur. Örnek bir makine gövdesi Şekil 3.2'de, hareket aktarma mekanizması ise Şekil 3.3'te verilmiştir.



Şekil 3.2 Örnek bir makine gövdesi (HV1060 modeli) (Smartech, 2015).



Şekil 3.3 Örnek bir hareket aktarma mekanizması (Fadal, 2008).

3.4.2 Bilgisayar

Kullanıcı ile makine arasında arayüz olup, G kodlarını işleme, makine değişkenlerinin değiştirilmesi ve gözlenmesi, makineye komut verilmesi gibi işlemleri gerçekleştirir. Bilgisayarda bu işlemleri gerçekleştirmek üzere bir işletim sistemi ve bir yazılım yer alır. Kullanıcı ile ara yüz klavye, fare, dokunmatik panel veya özel tasarlanmış panellerle sağlanabilir.

3.4.3 Ara yüz kartı

Makinenin üzerinde yer alan motor sürücüleri ve sensörler ile bilgisayar arasında ara yüz olarak kullanılan, mikroişlemci tabanlı elektronik karttır. Bilgisayardan gelen G komutlarını işler ve makinanın komutlara göre hareket etmesini sağlar.

3.4.4 Aktüatörler

CNC sistemlerinde aktüatörler makinadan makinaya farklılık gösterebildiği gibi genellikle 2 tip motor kullanılır: Servo motor ve adım motoru. Bu motorların tercihindeki en büyük sebep, yüksek hassasiyetli hareket ve sağladıkları yüksek momenttir.

3.4.4.1 Servo motor

Servo motorların temel mantığında, motorda yer alan sensörler vasıtasıyla motorun anlık pozisyon, hız gibi değerlerinin bilinebilmesi ve buna göre motorun uygun hareketinin sağlanmasıdır. Bu yapıları sayesinde istenilen hız ve pozisyon kontrolü hassas bir şekilde yapılabilir. Geri beslemeli kapalı döngü motor sistemleri isim olarak servo motor olarak adlandırılır. Motor ve geri besleme sensörleri kompakt bir yapıya sahiptir. Bu motorlar, hareketin denetlenmesini sağlayan mikrodenetleyici tabanlı motor sürücüler ile hareket ettirilir. Servo motorların başka bir tercih nedeni ise istenilen hızda istenilen momentte dönme hareketinin sağlanabilmesidir. Şekil 3.4'te örnek servo motor görselleri verilmiştir.

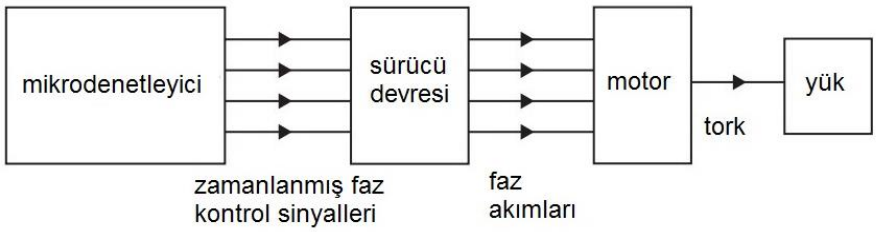
Şekil 3.4 Servo motor (Siemens, 2004).

Servo motorlar fırçalı ve fırçasız olmak üzere iki kategoriye ayrılır. Fırçalı motorda statorda (dış gövde) sabit mıknatıslar, rotorda (göbekte yer alan hareketli kısım) sargılar yer alır. Komütatör adı verilen ve akım geçişlerini sağlayan mil üzerinde yer alan fırça sistemine sahiptir. Fırçasız motorda tam tersi olarak rotorda sabit mıknatıs, rotorda ise sargılar yer alır. Pozisyon sensörleri ile hareket algılanarak gelecek hareket için ihtiyaç olan komutasyon elektronik sürücü ile sağlanır (Fjelde and Stamsø, 2011). Ayrıca servo motorları, sürme yöntemine göre AC ve DC servo olarak da gruplamak mümkündür. Bu yöntemlere göre pozisyon sensörü tipi, kontrol metodu ve sürücü tipi farklılık göstermektedir.

3.4.4.2 Adım motor

Adım motorlar, rotorlarında sabit mıknatıslara, statorlarında ise motor sargılarına sahip motorlardır. Rotorlarında elektriğe ihtiyaç olmadığından ve statorlarında DC elektrik kullanıldığından, fırçasız DC motorlardır. Statorlarındaki kutup sayısı çok yüksektir, rotorda ise çok dişli sabit mıknatıs yapısı vardır. Bu sayede bir turluk hareketi eşit aralıkta adımlarla tamamlar ve hassas pozisyon kontrolüne olanak sağlar. Stator sargılarının uygun enerjilendirilmesi durumunda ise motor milini bulunduğu konumda tutabilir ve böylece frenleme yapabilir. Şekil 3.5'te örnek adım motor görseli verilmiştir. Açık döngü çalışırlar, bir pozisyon sensörüyle geri beslemeye sahip değildir. Şekil 3.6'da açık döngü diyagramı verilmiştir. Fırçasız yapıya sahip olduklarından sargıların tetiklenmesini sağlayan harici bir elektronik tabanlı sürücüye ihtiyaç duyarlar. Uygun sargı tetiğiyle belirli sayıda adımın atılması şeklinde istenilen pozisyona gidiş gerçekleştirilir.

Şekil 3.5 Adım motor (Schneider Electric, 2012).



Şekil 3.6 Mikrodenetleyici bazlı açık döngü kontrol (Acarnely, 2002).

Üç tip adım motoru vardır: Değişken relüktanslı, sabit kutuplu ve hibrit adım motorları. Değişken relüktanslı motorlarda stator sargılara sahiptir ve rotor yumuşak demir gibi manyetik olmayan bir malzemeden üretilmiştir. Sabit kutuplu adım motorlarında rotor sabit kutuplu mıknatıslara sahiptir. Hibrit motor ise diğer

iki motor tipi prensibi de kullanılan bir motor tipidir (Wikipedia contributors, 2015f).

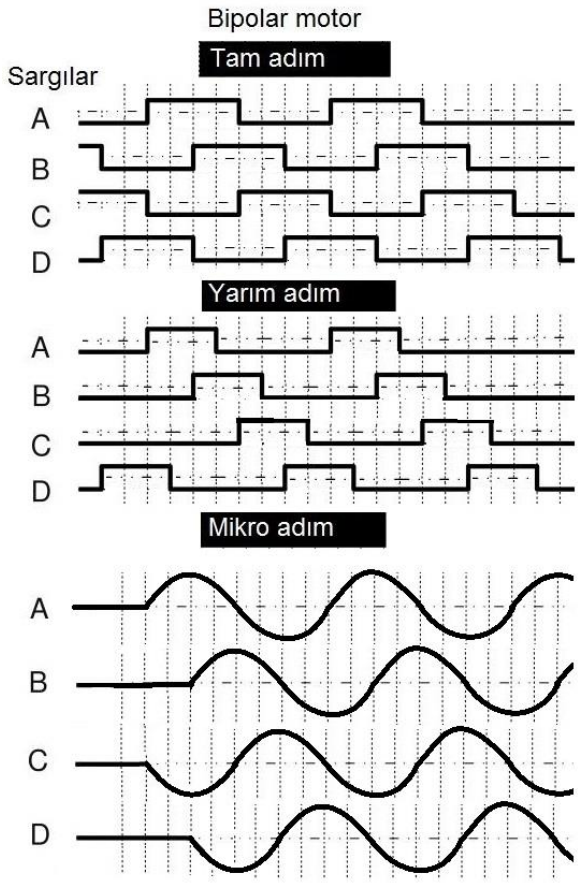
İki tip adım motor sargı tipi vardır: Bipolar ve ünipolar. Bipolar adım motorlarda her faz için tek sargı vardır. Her faz iki kablo ucuna sahiptir. Ünipolar motorlarda ise orta uçlu sargılara sahiptirler ve sargıların ortasında bir uç daha vardır. Bipolar motorda, sargı kutbunun terslenebilmesi için akım yönünün değiştirilmesi gerekmektedir, ünipolar motorda ise buna gerek yoktur. Bu açıdan ünipolar motorlar ve sürücülerinin karmaşıklığı düşük ve daha az maliyetlidir. Bipolar motorların ünipolar motorlara göre sargılarının kapladığı alan daha düşüktür, düşük hızlarda sağladığı moment daha yüksektir (Acarnely, 2002).

Adım motorlarda belirli bir yönde dönme hareketi, motor sargılarının belirli bir düzen ile polaritelerinin değiştirilmesiyle gerçekleştirilir. Polarite değişimlerinin gerçekleştirilmesi konusunda üç tip sürme yöntemi standart olarak kabul edilmektedir. Bunlar; tam adım, yarım adım ve mikro adım sürmedir. Bu sürme tekniklerinin farkları, fazlara verilen akımların dalgalarının şekilleri ve sıralarıdır (Acarnely, 2002). Şekil 3.7'de sürme tekniklerine dair sinyal şekilleri verilmiştir.

Tam adım sürme yönteminde, motorun kabaca temel dönüş adımını karşılamak üzere (örneğin, 1.8"lik açı hareketi ve 200 adımda tek turun tamamlanması) fazların tetiklenmesiyle gerçekleşir. İki türü vardır: tek-faz yöntemi ve çift-faz yöntemi. Tek-faz yönteminde, her bir adımda sadece tek fazenerjilendirilir. Çift-faz yönteminde ise, her bir adımda iki faz aynı anda enerjilendirilir, tek-faz yöntemine göre daha yüksek moment ve hız performansına sahiptir (Acarnely, 2002).

Yarım adım sürme yönteminde, motorun pozisyon çözünürlüğünü iki kat yükseltmek amacıyla, bir anlamda tam adım tek-faz ve çift-faz yöntemlerinin birleştirildiği ve temel adımlarda çift-faz yöntemiyle, ara adımlarda ise tek-faz yöntemiyle tahrik gerçekleştirilir(Acarnely, 2002).

Mikro adım sürme yönteminde, fazlara verilen akımın büyüklük ve yön kontrolü sağlanarak hareket sağlanır. Bu yöntemle, temel bir adım 256'ya kadar bölünerek çözünürlük arttırılmış olur. Ayrıca düşük hızlardaki geçiş yumuşatılır ve düşük hız rezonans etkisini küçültülür (Acarnely, 2002).



Şekil 3.7 Bipolar adım motor sürüş yöntemleri ve sinyalleri.

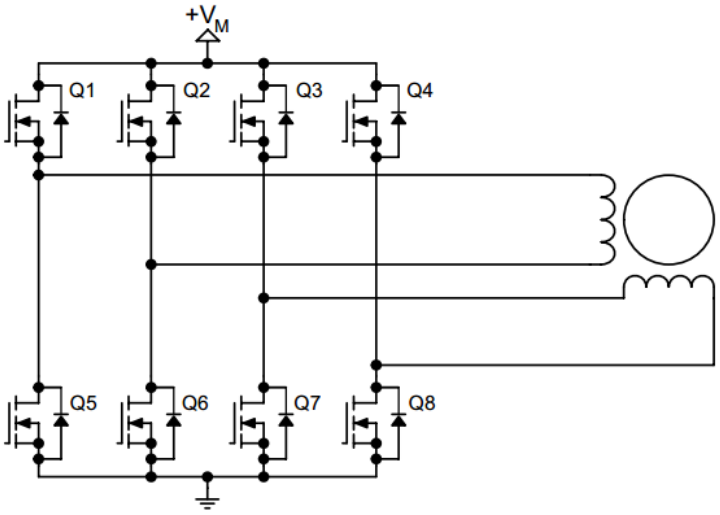
3.4.5 Aktüatör sürücüleri

CNC sistemlerinde aktüatörlerin istenilen pozisyona istenilen hızda hareket etmesi istenir. Bazı motor çeşitleri de motor uçlarına doğrudan elektrik verildiği takdirde harekete geçmezler. Aynı zamanda pozisyon sensörü gibi algılayıcılar yardımıyla kapalı döngü kontrole ihtiyaç duyan sistemlerde elektronik tabanlı bir donanıma gereklidir. Bu amaçlar doğrultusunda, aktüatör hareketini istenilen biçimde sağlayacak şekilde faz enerjilendirmelerinin kontrolünü gerçekleştiren motor sürücüleri kullanılmaktadır. Farklı motor türlerinin kontrolleri de farklı olduğundan, her motor türüne uygun farklı motor sürücüsü vardır. Sürülecek motorun gücüne göre de sürücü devresi elemanları değişeceğinden, sürücü seçiminde motorun gücü de önemlidir. Kullanıcının istediği senaryoyu ve kontrolü sürücüye, dahili ya da harici kontrolörler yoluyla gönderip motorun sürüşünü sağlar.

Adım motorların motor sürücüleri, sürme tekniğine göre farklılık gösterse de, temelde fazların önlerinde yer alan MOSFET veya transistör gibi yarı iletken

elemanlar vasıtasıyla enerjilendirilmesi ile gerçekleştirilir. Fazlara aç-kapat veya oransal kontrolle gerilimi ayarla-tersle gibi kontroller de bu devrelerle sağlanmaktadır.

Bipolar adım motorlarda, her bir fazdan iki uç çıkmaktadır ve uygun kontrol için iki yönden de akım geçirmek gerekmektedir. H köprüsü yöntemi bu iş için uygun bir yöntemdir. Anahtarlayıcı elemanların tek bir faz için uygun bağlantısıyla motor yönünün yanında faz geriliminin analog olarak ayarlanmasına da olanak sağlar (Wikipedia contributors, 2015ç). Şekil 3.8'de temel bir bipolar adım motor sürücü devresi verilmiştir.



Şekil 3.8 Bipolar adım motor sürücü (Silicon Labs, 2008).

3.4.6 Sensörler

Fiziksel büyüklüklerin algılanması ve ölçülmesinde kullanılan her mekanik veya elektronik, her türlü elemanı sensör olarak adlandırılır. Bu ölçümlere örnek olarak mesafe, ısı, sıcaklık, kuvvet, akım, gerilim, ışık gibi değerler verilebilir. Ölçüm değerleri ve ölçüm sonuçları çıktı olarak dijital veya analog olabilir. Örnek verecek olursak; bir limit anahtar sensörü, hareketli parça kendi mandalına baskı uygulayana kadar açık, baskı uygulandığında ise kapalıdır, dijital bir sensör olarak çalışır, fakat bir sıcaklık sensörü olan PT100, ölçeceği akışkanın sıcaklığına göre iç direnci değiştiğinden akışkanın tam sıcaklığı tespit edilebilir, analog bir sensör olarak çalışır.

CNC sistemlerinde kullanılabilecek sensörleri enkoder, anahtar, indüktif-kapasitif-fotoelektrik sensörler ve akım sensörü olarak sayabiliriz.

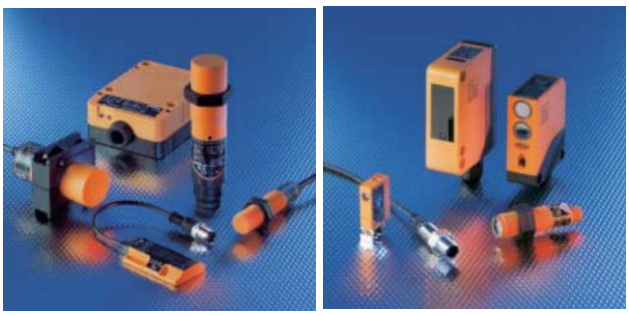
Enkoder, açısal pozisyonu algılamaya yarayan bir sensör türüdür. Birçok çeşidi bulunmaktadır. Çıkış olarak analog ya da dijital çıkış üretebilir. İçerisinde yer alan manyetik disk üzerinden ya da delikli plaka üzerindeki deliklerden foto diyotlar sayesinde okuma gerçekleştirir. Şekil 3.9'da örnek enkoder görselleri verilmiştir.

Şekil 3.9 Döner enkoder (Silicon Labs, 2008).

Sınır anahtarları, genellikle güvenlik amacıyla kullanılan, mekanik hareketin tespiti sonucu aç-kapat mantığıyla elektriksel çıkış üreten anahtarlardır. Çıkış sinyali ise mikrodenetleyici tabanlı kontrolöre veya motor girişini kontrol eden kontaktöre bağlıdır ve bu şekilde motor kontrolü sağlanır. Bir makinada bu sensör herhangi bir amaçla kullanılabileceği gibi, lineer kızakların sonlarına eklenen sensörlerin makinanın maksimum sınırlarını aşmasını önleme amaçlı CNC makinalarda uygulanabilir. Acil durdurma butonu ise normalde kapalı anahtar olarak güvenlikte rol alır. Şekil 3.10'da örnek sınır anahtar görselleri verilmiştir.

Şekil 3.10 Sınır anahtarı (Schneider Electric, 2007).

İndüktif-kapasitif-fotoelektrik sensörler, limit anahtarlarla aynı görevi görür ve bu işi temazsız olarak gerçekleştirir. Elektromanyetik ve fotoelektrik çeşitleri yaygındır. Elektromanyetik çeşitlerinin, kapasitif ve indüktif sensör olarak çeşitleri mevcuttur. İndüktif sensörler, sadece metal cisimlere tepki verirken, kapasitif ve fotoelektrik tipi sensörler ise normal malzemelerde de kullanılabilir. Çeşitlerine göre algılama mesafeleri değişkenlik gösterir (birkaç milimetreden birkaç santimetreye kadar). Fotoelektrik sensörlerde ise böyle bir sınır yoktur. Sistemde genellikle üzerindeki ışık kaynağı tarafından yayılan ışının reflektör özellikli bir yüzey tarafından alıcıya ulaştırılması esas alınır ve araya giren cisimle kesintiye uğrayan ışık iletimi, anahtarlama bazında sensörün aktif-pasif konumları olarak değerlendirilir. Bu sensörler de yine sınır anahtar görevinde kullanılabilir. Şekil 3.11'de örneklerine yer verilmiştir.



Şekil 3.11 İndüktif ve fotoelektrik sensörler (IFC Electronic, 2005).

Akım sensörü, bir kablodan ya da hattan geçen akımı ölçmeye yarayan sensördür. Makinanın elektrik besleme hattına, yüksek akım çekebilecek hatlara konan sensörlerle makine için veri toplama işlemi gerçekleştirilebilir ve güvenli çalışma ortamı sağlanabilir.

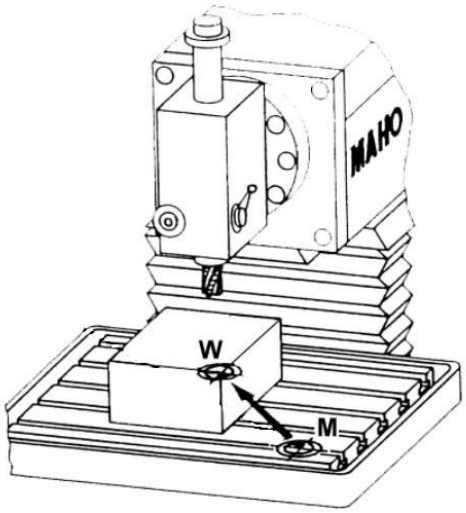
3.5 CNC Koordinat Sistemleri

CNC sistemlerinde genel koordinat sistemi olarak kartezyen koordinat sistemi kullanılmaktadır. Her makinede X, Y, Z koordinat eksenleri yer alırken, ek serbestlik derecesine sahip makinalarda A, B, C eksenleri de oryantasyon ekseni olarak yer alabilir

Bu koordinat sisteminde, X ve Y eksenleri taban eksenler, Z ekseni ise tabana dik eksen olarak yer alır. A, B ve C eksenleri ise, bu eksenlerin dönme eksenleri olarak tanımlanır (Uyar vd., 2014). Şekil 3.12'de 6 eksenli koordinat sistemi verilmiştir.

Şekil 3.12 6 eksenli koordinat sistemi (Fjelde and Stamsø, 2011).

Makinanın pozisyonu, makine tablası üzerinde tanımlanan koordinat sistemine göre uç işlevcinin koordinatı olarak tanımlanır. Makinanın orijin veya sıfır noktası, tablanın güney-batı köşesi veya programcının belirlediği herhangi bir nokta olabilir (Mechanical Guru, 2011). İş parçasının sıfır noktası da makinanın sıfır noktasının iş parçasının üzerinde belirlenen sıfır noktasına göre konumudur. Şekil 3.13'te makine ve iş parçasınını sıfır noktalarının tanımlanması gösterilmiştir.



Şekil 3.13 Makina koordinat sistemi M: makine, W: iş parçası sıfır noktası (Krar and Gill, 1999).

Koordinat sistemlerindeki diğer önemli bir nokta ise, uç işlevcinin pozisyonlamasının kendine veya tablaya göre tanımlanmasıdır. Tablaya göre pozisyonlamada (mutlak pozisyonlama) uç işlevcinin anlık ve gelecek pozisyonları bu koordinat sistemine göre ifade edilir. Kendine göre pozisyonlamada (artan pozisyonlama) anlık pozisyon sıfır olarak kabul edilir ve bu pozisyona göre gelecek pozisyon artımına göre kontrol sağlanır (Mechanical Guru, 2011).

3.6 G-Kodu

CNC makinalarında makinaya gönderilen ve makinayı kontrol amaçlı kullanılan yazılım diline verilen isimdir. Program komutları, kontrol birimi ile işlenerek makinanın hareketleri sağlanır. Bilgisayar destekli üretim (Computer Aided Machining) (CAM) programıyla yörünge ve talimatları belirlenen üretim işlemi, CAM programına dahil veya programdan hariç işleme sonrası (post-processor) programı tarafından işlenerek G kodu haline getirilir (Albert, 2008). Buna dair blok diyagramı Şekil 3.14'te verilmiştir.

Şekil 3.14 CNC makine işlem diyagramı (Hashim, 2012).

Komut sistemi harf ve sayılardan oluşur. Harf adresleri komut grubunu, sonrasında gelen sayılar ise komutun adresini gösterir. Harf adreslerinden bazıları aşağıdaki Ek 3'teki gibidir.

Bu komut sisteminde, satırlara yazılan komutlardan bazıları aynı satırda yer alabilirken, bazı komutlar ise sırayla işleme mantığından dolayı yeni bir satırda yer almalıdır.

3.6.1 Sıkça kullanılan bazı G komutları

3.6.1.1 G00 – Hızlı pozisyonlama

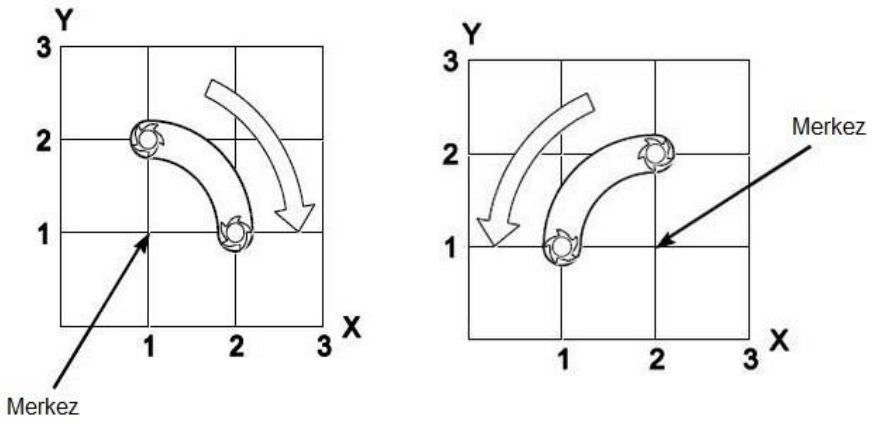
İki veya üç boyutta, hedef noktaya düz bir çizgi doğrultusunda mümkün olan maksimum hızda ilerlemedir. Kesme gibi işlemlerin olmadığı, kesme, dalma, kaynak atma bölgeleri arasında geçişlerde tercih edilir. X, Y, Z son pozisyonları belirtmek için kullanılan indislerdir (Haas Automation, 2006).

3.6.1.2 G01 – Lineer tahmin

G00 ile benzer bir işleve sahip olup, kullanıcı tarafından verilen hareket hızına göre hareket gerçekleşir. İşleme sırasında harekette kullanılır. Komutla birlikte, uç işlevcinin geçeceği noktalar hızdan yola çıkılarak zamana göre tahmin edilir. X, Y, Z son pozisyonları, F ise ilerleme hızını belirtmek için kullanılan indislerdir (Haas Automation, 2006).

3.6.1.3 G02, G03 – Dairesel tahmin

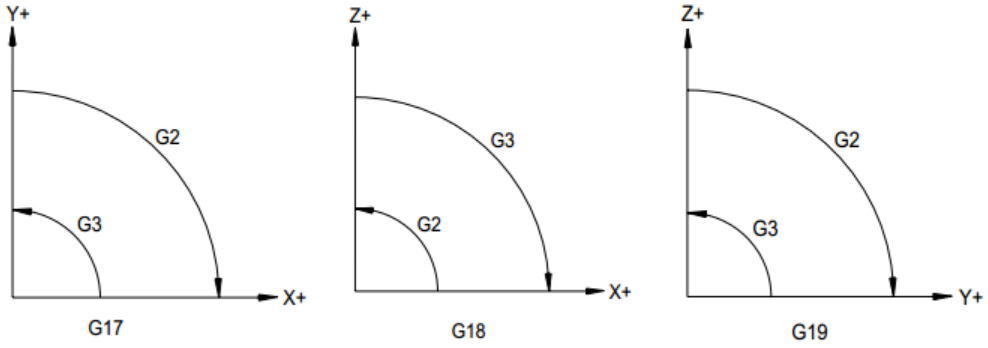
G01 ile benzer olup, bitiş noktasına doğrusal çizgi yerine belirtilen çap değerine sahip bir yay çizerek ilerler. Komutla birlikte, yay çizmek üzere iki nokta arası yörünge tahmin edilir ve yörünge üzerindeki geçiş noktaları tahmin etme yöntemiyle belirlenir. Bu belirlenen noktalar arasında yine lineer tahmin yöntemi uygulanarak çok kısa mesafeli lineer hareketlerle hareket gerçekleşir. Komutla birlikte çizilen yaylar, ilk nokta ile son nokta arasında G02'de saat yönünde, G03'te saat yönünün tersinde hareket edecek şekildedir. X, Y, Z son pozisyonları, R, I, J, K yayın çapı ve merkezini belirtmek için kullanılan indislerdir (Haas Automation, 2006). Şekil 3.15'te dönme hareketleri gösterilmiştir.



Şekil 3.15 G02 ve G03 komutlarına göre hareket (Tormach, 2015).

3.6.1.4 G17, G18, G19– Düzlemi seçimi

Temel işleme düzlemini G17 XY, G18 XZ, G19 YZ olarak ayarlar. İşleme düzlemi dairesel tahminli hareketlerde ve kesici uç kompanzasyonunda dikkate alınır (Haas Automation, 2006). Şekil 3.16'te seçilen düzlemin yay hareketlerine olan etkisi gösterilmiştir.



Şekil 3.16 Düzlem seçiminin yay hareketine olan etkisi (FADAL, 2003).

3.6.1.5 G20, G21– Uzunluk birimi seçimi

Uzunluk birimini G20 inç, G21 milimetre olarak ayarlar (Haas Automation, 2006).

3.6.1.6 G28– Ev pozisyonuna dönüş

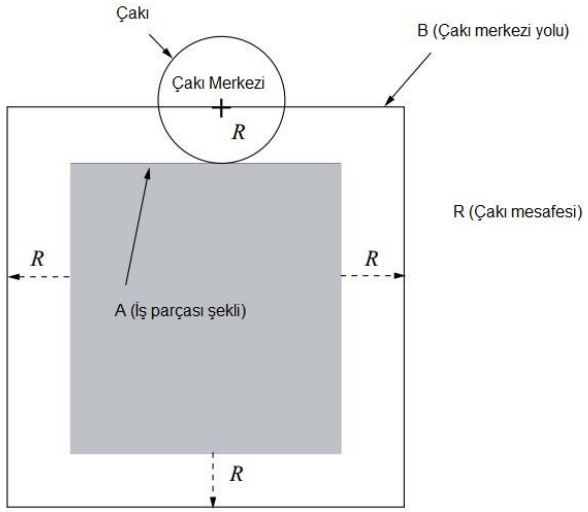
Uç işlevcinin sıfır pozisyonuna gitmesini sağlar (Haas Automation, 2006).

3.6.1.7 G40– Takım yarıçap kompanzasyon kapalı

G41 ve G42'nin tersi olarak çalışır. Uç işlevcinin pozisyonu, takımın işlem ucunun pozisyonu olarak algılandığından, işleme sırasında, iş parçası ve uç işlevci pozisyonu arasına takımın yarıçapına bağlı olarak bir mesafe eklenmelidir. G40'ın fonksiyonu ise, bu mesafe ayarını kapatmaktır (yarıçapı sıfır seçmek de denebilir) (Haas Automation, 2006).

3.6.1.8 G41, G42– Takım yarıçap kompanzasyon

Takımın kesimde G41 soldan, G42 sağdan yaklaşımlarında eklenecek mesafedir. D adresi kullanılarak yarıçap girişi yapılır (Haas Automation, 2006). Şekil 3.17'da kompanzasyonun etkilediği konum gösterilmiştir.



Şekil 3.17 Programlanmış yol ve mesafeli yol (FADAL, 2003).

3.6.1.9 G43, G44– Takım uzunluğu mesafe kompanzasyonu

Takımın Z ekseninde eklenecek kompanzasyon mesafesidir. H adresi kullanılarak mesafe girişi yapılır. Ayar çizgisine G43 negatif, G44 pozisif olarak eklenmesini sağlar (Haas Automation, 2006).

3.6.1.10 G49– Takım uzunluğu mesafe kompanzasyonu iptal

Takımın Z eksenindeki kompanzasyonu iptal eder (Haas Automation, 2006).

3.6.1.11 G52– Yerel koordinat sistemi

Makine koordinat sistemini tanımlanan mesafeler kadar kaydırır. İşlemler yerel koordinat sistemi üzerinden gerçekleşir (Haas Automation, 2006).

3.6.1.12 G53– Makine koordinat sistemi

Makinanın sıfır pozisyonudur (Haas Automation, 2006).

3.6.1.13 G54..G59– Çalışma koordinat sistemleri

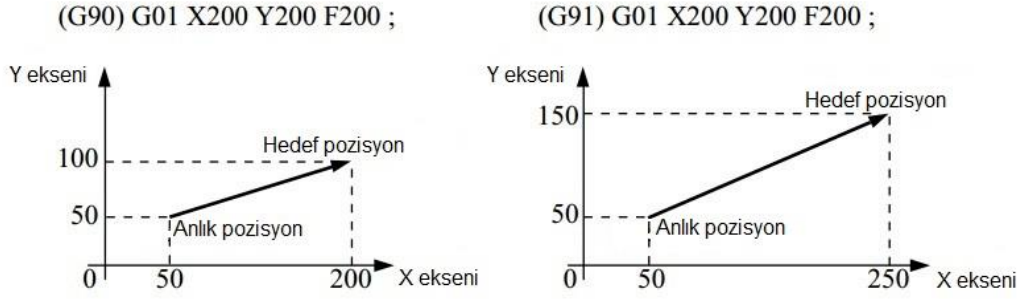
Makinanın sıfır pozisyona göre her eksene mesafe eklenmesine yarar (Haas Automation, 2006).

3.6.1.14 G90– Mutlak programlama

Pozisyonlama, makinanın sabit koordinat sistemine göredir (Haas Automation, 2006).

3.6.1.15 G91– Artan programlama

Pozisyonlama, bir önceki pozisyona göredir (Haas Automation, 2006). Şekil 3.18'de pozisyonlamanın harekete olan etkisi gösterilmiştir.



Şekil 3.18 G90 ile G91 arasındaki fark (FADAL, 2003).

3.6.2 Sıkça Kullanılan Bazı M Komutları

3.6.2.1 M00 – Zorlayıcı durdurma

Makine bu komutu okuduğunda, her ne olursa olsun programı durdurur (Haas Automation, 2006).

3.6.2.2 M01 – İsteğe bağlı durdurma

Makinenin bir buton yardımıyla durdurulması sağlanır (Haas Automation, 2006).

3.6.2.3 M02 – Program bitişi

Programın sonuna gelinmiştir (Haas Automation, 2006).

3.6.2.4 M03 – M04 Mil açık

Uç işlevci milini açar. M03 saat yönünde, M04 ise saat yönünün tersi yönde dönmesi anlamına gelir (Haas Automation, 2006).

3.6.2.5 M05 – Mili durdur

Uç işlevci milini durdurur (Haas Automation, 2006).

3.6.2.6 M06 – Otomatik takım değişimi

Operatöre ihtiyaç olmadan, magazinde tanımlı ve hazır olan takımı hâlihazırda kendi üzerindeki takımla değiştirmeye yarar (Haas Automation, 2006).

3.6.2.7 M07 – Hava soğutma açık

İşleme sırasında etrafa talaşlar dağılabilir. Ayrıca işlemde bir ısı ortaya çıkar. Bu ısı, takımın kalitesini, ömrünü ve işlem kalitesini etkiler. Talaşın çalışma bölgesinden temizlenmesi ve takımın soğutulması amacıyla basınçlı hava üflemesi komut ile aktif hale getirilir (Haas Automation, 2006).

3.6.2.8 M08 – Sıvı soğutma açık

İşleme sırasında ortaya çıkan ısının atılması amacıyla, soğutma sıvısı aktive edilir. Bu soğutma sıvısı su veya bor yağı gibi sıvılar olabilir (Haas Automation, 2006).

3.6.2.9 M09 –Soğutma kapalı

Aktif olan M07 veya M08'in kapatılması için kullanılır (Haas Automation, 2006).

3.6.2.10 M30 –Programı bitir ve programın başına dön

Programı bitirir ve programın başına döner (Haas Automation, 2006).

3.6.2.11 M98 –Alt program çağır

P adresiyle belirtilen alt programı çağırır (Haas Automation, 2006).

3.6.2.12 M99–Alt programı bitir

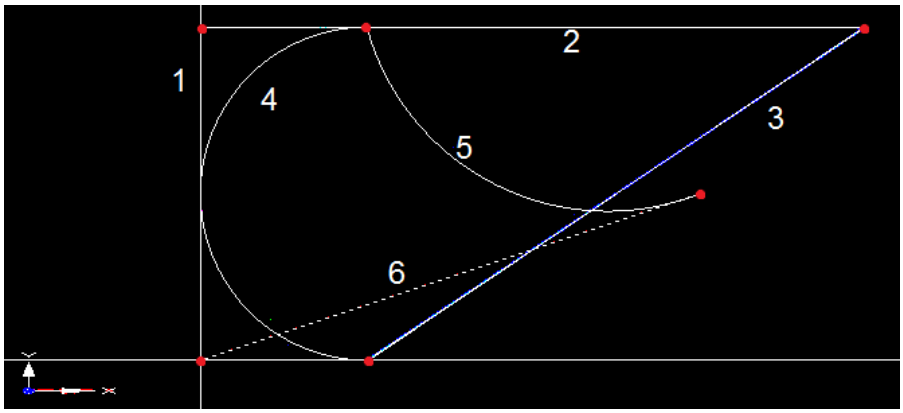
Çalıştırılan alt programdan çıkılıp ana programa geri döner (Haas Automation, 2006).

3.6.3 Bir program örneği

Örnek bir program aşağıdaki gibidir. Programın çıktısı Şekil 3.19'da verilmiştir.

```
%
O0000          (program no)
N1 G21         (mm birimi)
N2 G90         (mutlak pozisyonlama)
N3 G53         (anlık pozisyonu sıfır pozisyonu yap)
N4 G0 Y100     (Y100 noktasına hızlı hareket)(1 nolu hareket)
N5 G1 X200 F500        (X200 Y100 noktasına 500 hız ile hareket et)(2 nolu hareket)
N6 X50 Y0      (X50 Y0 noktasına 500 hız ile hareket et)(3 nolu hareket)
N7 G2 X50 Y100 R50     (X50 Y100 noktasına R50 yarıçapla yay çizerek git)(saat yönünde)(4
nolu hareket)
N8 G03 X150 Y50 R75    (X150 Y50 noktasına R75 yarıçapla yay çizerek git)(saat yönünün
tersinde)(5 nolu hareket)
N9 G28         (orijin noktasına geri dön)(6 nolu hareket)
N10 M30        (programı sonlandır)
```

%

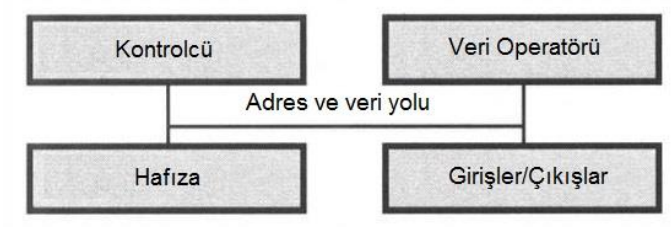


Şekil 3.19 Örnek program kodunun çıktısı.

4. MİKRODENETLEYİCİ

4.1 Mikrodenetleyici ve Tarihçesi

Mikrodenetleyici, üzerinde mikroişlemci ve çevre birimleri ile birlikte bütünleşik olarak üretilmiş elektronik elemandır. Tekrar programlanabilme özelliğine sahiptir. Gerçek zamanlı olarak dış ortamdan gelen girdileri programına göre işleyerek gerçek zamanlı çıkış üretebilir. İçine yazılan program sayesinde, otomatik olarak verilen görevleri yerine getirebilir. Genel amaçlıdır. Düşük maliyeti, boyutu, güç tüketimi ve yüksek işlem kapasitesi sayesinde günlük hayatımızda uygulama alanı çok geniştir. Otomobil, beyaz eşya, bilgisayar, televizyon, telefon, fotoğraf makinası, asansör, otomatik kapı sistemleri ve daha birçok alanı uygulama alanları olarak sayılabilir. Mikrodenetleyici üretimi ve geliştirmesi yapan dünya çapında birçok firma vardır (Wikipedia contributors, 2015e). Şekil 4.1'de basitleştirilmiş bilgisayar yapısına ait diyagram verilmiştir.



Şekil 4.1 Basitleştirilmiş bilgisayar yapısı (Lipovski, 1999).

Mikrodenetleyicilerin serüveni 1970'lerin başında 4-bitlik Intel 4004 mikroişlemcisi ile başlamıştır. Sonrasında daha gelişmiş modeller olan 8008 ve 8080 ile 8-bitlik modeller ortaya çıkarken, Motorola 6800 ve Texas Instruments TMS 1000 ile Intel'e rakip olmuştur. Teknolojilerik yeniliklerle birlikte (hafızanın hızlı yazılıp silinebilmesi, CMOS teknolojisinin oluşması, vs.) Intel 8051, 8086 (16-bit), 80186 (16-bit), 80286 (16-bit), 800386 (32-bit) (x86 ailesi), Pentium serisi ve 64-bitlik gelişmiş işlemcileriyle devam etmiştir. Motorola ise 6809, 68000 (16-bit), 68010, 68020, 68030 gibi modellerle gelişimine devam etmiştir ve mikroişlemci üretimini Freescale firmasına satmıştır. Bilgisayar ve mobil telefon üreticilerinin sıkça tercih ettiği mikroişlemciler olmuşlardır. Bu gelişim sırasında, Microchip PIC ve Atmel AVR serisi ile piyasaya girmiş ve günümüzde genellikle hobi amaçlı kullanılan seriyi oluşturmuştur. Yakın geçmişte geliştirilen, hakları ARM firmasına ait olan ARM mimariye sahip mikroişlemciler ise üretim hakkına sahip birçok firma tarafından üretilmekte ve avantajlarından dolayı tercih edilen

bu mimariye sahip mikroişlemcilerden dünya üzerinde satış rakamlarına bakıldığında on milyarlarca olduğu düşünülmektedir (Engin, 2013; Wikipedia Contributors, 2015d).

4.2 Mikrodenetleyici Elemanları

Günümüzde bir mikro denetleyicide standart olarak yer alan birimler mikroişlemci, salıngaç (osilatör), hafıza, aritmetik mantık birimi, giriş-çıkışlar, zamanlayıcılar ve sayıcılar, tut/karşılaştır, kesmeler ve haberleşme birimleri olarak söylenebilir.

Mikroişlemci (Merkezi İşlem Birimi)(MİB)(Central Processing Unit)(CPU), bir mikro denetleyicinin beyni olarak görev alır. Dışarıdan veriyi alır, komuta göre işler ve çıktı olarak sonucu dışarıya verir. Çevre birimlerle koordinasyon içindedir. Mikroişlemcileri, seferde veri işleyebilme kapasitesine göre 8, 16, 32, 64 bit gibi gruplandırmak da mümkündür. İşlemcinin işleme kapasitesinin artması, bir anlamda hızının da artması anlamına gelmektedir. Modern sistemlerde, özellikle bilgisayar, telefon gibi uygulama alanlarında, "çok çekirdekli" adı verilen ve birkaç mikroişlemciden oluşan ve paralel işleme tekniği uygulanabilen MİB'ler bulunmaktadır (Lipovski, 1999).

Salıngaç, periyodik olarak kare, sinüs, testere diş, üçgen dalga gibi sinyal üreten birimdir. Mikro denetleyicide temel olarak sistem saati üreteci olarak kullanılmaktadır. Sistem saati, mikro denetleyicinin kalp atışı olarak da algılanabilir, mikroişlemcinin kontrol biriminin çalışmasını sağlar. Sadece mikroişlemciye değil, diğer çevre birimlere de sinyal sağlar. Birimler, ön bölücü olarak adlandırılan bir birim kullanarak kendilerine gelen salıngaç sinyalinin frekansını ayarlayabilirler.

Hafıza birimi, verilerin tutulduğu birimdir. Hafıza birimi okunabilir-yazılabilir (RAM) ya da sadece okunabilir olabilir (ROM). ROM; PROM, EPROM, EEPROM gibi çeşitlere sahiptir. ROM üzerinde silinmesini istenilmeyen komut kodları gibi verileri tutabilir. EEPROM, elektrik sinyalleriyle silinebilir ve yeniden programlanabilir bir hafızadır. Program kodları bu birimde saklanmaya uygundur, ayrıca tekrar yazılabilir ve program değiştirilebilir. RAM hafıza, geçici hafıza olarak kullanılmaya uygundur. Enerjisi kesildiği takdirde içindeki veriyi kaybeder, ROM hafıza ise veriyi saklamaya devam eder. Çalışma hızı bakımından RAM hafıza çok daha yüksek çalışma hızına sahiptir. Ayrıca

çalışır durumda elde edilen ve saklanmak istenen veriler de ROM hafızaya kaydedilebilir. Hafıza birimleri temelde kaydedicilerden oluşur ve veriler kaydediciler içerisinde tutulur. Bazı kaydediciler özeldir ve mikro denetleyicinin ayarlamaları ile ilgili verileri içermektedir. İşlemci, hafıza birimleriyle daima iletişim halindedir. Ayrıca bellekler dahili veya harici olabilir (Lipovski, 1999).

Aritmetik Mantık Birimi (Arithmetic Logic Unit)(ALU), bir MİB'nin vazgeçilmez birimi olmak üzere, aritmetik, bit düzeyinde işlemler bu birimde gerçekleştirilir. Bir anlamda mikro denetleyicinin hesap makinası gibidir.

Giriş-çıkışlar, mikro denetleyicinin çevre birimleriyle ve dış dünyayla bağlantısıdır. Mikroişlemcide yer alan pinlerden bir kısmı sadece çevre birimlere bağlıdır. Geri kalan pinler ise genel amaçlı giriş-çıkış pinleri olarak adlandırılmaktadır. Giriş-çıkış pinleri birden çok fonksiyona sahip olabilir, mikro denetleyiciyi programlarken pinlerin kullanılmak istenilen fonksiyonu seçilir ve ayarlama yapılır. Pinler, mikroişlemcinin işlem kapasitesine (bit sayısına) dayanarak A, B, C, D gibi portlar olarak gruplandırılır. Dışarıdan dijital-analog veri-sinyal alma ve dışarıya gönderme gibi dış ortam ile tüm işlemler bu kanallar vasıtasıyla gerçekleştirilir.

Zamanlayıcılar, temel olarak zamanı ölçmek için kullanılırlar. Sayıcılar ise bir olayı veya zamanı saymak için kullanılır. Sayma ve zamanlama işlemlerini bu birimlerle halledilebilir. Bu iki birimi birbiriyle iç içe İki birimin de temel mantığında, atanan olayın gerçekleşme miktarını saymaktır. Zamanlayıcı, genellikle sabit frekanslı bir kaynağı (örneğin, sistem saatçisinden aldığı sinyal), sayıcı ise iç ya da dış bir kaynaktan aldığı sinyali ya da olayın gerçekleşme miktarını temel alarak sayma işlemi gerçekleştirir. Kapasiteleri sınırlı olduğundan (kaydedicilerin boyutları), menzil dışında kalan yüksek sayıcı veya zamanlayıcı değerlerine erişebilmek için ön bölücüler kullanılır. Ayrıca, darbe genişlik modülasyon (PWM)(Pulse Width Modulation) gibi periyodik sinyallerin üretilmesinde de zamanlayıcılar kullanılmaktadır (Valvano, 2011).

Tut/Karşılaştır (Capture/Compare) birimi, isminden de anlaşılabileceği gibi temel olarak karşılaştırma işlemi yapar. Bazı özel sinyallerin üretilmesinde (PWM), analog sinyallerin okunmasında, sayıcı ve zamanlayıcıların istenilen durumda tepki vermesinde kullanılır.

Kesme, bir olayın gerçekleşmesiyle birlikte ortaya çıkan uyarı sinyalidir. Mikro denetleyicide yer alan kesme birimi, ayarlanan kesmelere göre (örneğin zamanlayıcı kesmesi, sayıcı kesmesi, haberleşme kesmesi, dış kesme gibi), olayın gerçekleşmesiyle birlikte mikroişlemciye uyarı sinyali gönderir, mikro işlemci ise programa göre kesme işleyicisi (interrupt handler) ile belirli bir işlemi gerçekleştirebilir (Valvano, 2011).

Haberleşme birimi, dış ortam ile haberleşmeyi düzenlemekle görevlidir. Dış ortamdaki bigisayar, yazıcı, LCD, GPS gibi mikroişlemci tabanlı cihaz birim ile ayarlandığı haberleşme protokolüne göre veri alışverişi, hata denetleme gibi olayları gerçekleştirir. Mikro denetleyicilerde kullanılan bazı seri haberleşme protokolleri şöyledir: USART, SPI, I^2C (Valvano, 2012).

4.3 Mikroişlemci Mimari Yapıları

Firmaların ürettikleri mikroişlemci mimarileri birbirinden farklı olmasına rağmen, temel alınan iki tip mimari vardır: RISC ve CISC.

İndirgenmiş Komut Takımıyla Hesaplama (Reduced Instruction Set Computing)(RISC) mimarisinde, kullanılan komut listesi temel, genelde tek bir saat döngüsünde (clock cycle) tamamlanabilecek işlemleri operasyon kodları (Opcode) içerir. Komut sayısı azdır, cihaz üzerinde yürütülen makine diline çevirme işlemleri daha basittir. Yazılımın iyi yönetilebilmesi durumunda, daha efektif donanım kullanımı, daha düşük güç ve yıpranma elde edilebilir (Gerritsen, 1999). Günümüzde popüler olan ARM mimarisine sahip işlemcilerde geliştirilmiş bir RISC mimarisi kullanılmaktadır.

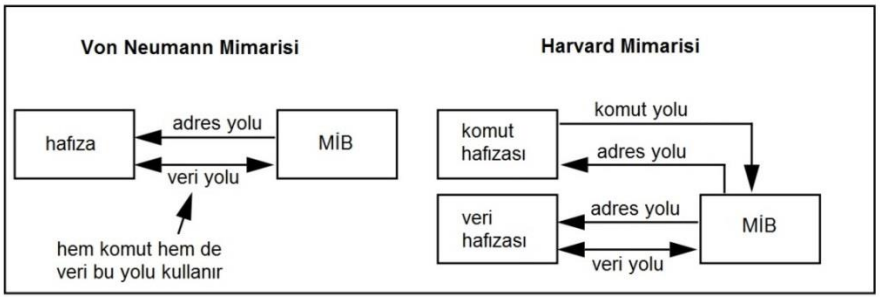
Kompleks Komut Takımıyla Hesaplama (Complex Instruction Set Computing)(CISC) mimarisinde amaç, bir görevi yazılabilecek en az satırla yazmak ve kod boyutunu düşürerek progaramcının işini kolaylaştırmaktır. Büyük boyutlu bir çalışmada, RISC mimariye göre yazılmış kodun boyutu ile karşılaştırıldığında CISC mimariyle yazılmış kodun boyutu daha düşüktür. Tanımlanan komut sayısı daha fazladır. Kullanılan komutlar birkaç döngüyü de kapsayabilmektedir (Gerritsen, 1999).

4.4 Mikro Denetleyici Mimari Yapıları

Mikro denetleyicilerde, çevre birimler arası bağlantılar farklılık gösterebilir ve bu farklılıklar mikro denetleyici içindeki haberleşmesinden, mikro denetleyici işlem hızı ve maliyetine kadar birçok faktörü etkiler. Ayrıca bu mimariler bilgisayarlarda da kullanıldığından, bilgisayar mimarisi olarak adlandırılabilir. Temel sayılabilecek iki tip bilgisayar mimarisi vardır: Von Neumann ve Harvard.

Von Neumann, Von Neumann tarafından geliştirilmiş bir mimaridir. Mikrodenetleyici tarihinin başlangıç süreçlerine bakıldığında, var olan bir sistemin geliştirilmesi noktasında uyumsuzlukların giderilmesi amacıyla tüm donanımın değiştirilmesi gerekmekteydi. Von Neumann ise, komut ve veri hafızasını tek bir hafızada toplayarak, geliştirilen sistemde değiştirilen tek birimin hafıza ya da hafıza içeriği olmasını sağlamıştır. Bu da maliyet açısından oldukça büyük bir avantaj sağlamıştır. Bu mimariye göre, MİB ile hafıza arasındaki haberleşmede, tek bir veri yolu üzerinden hem komut hem de veri alışverişi sağlanmaktadır. Yani, aynı anda veri yolunda birkaç işlem yapılamamaktadır. Ayrıca haberleşme hızındaki gecikme ve aynı anda komut-veri alışverişi gerçekleşmediğinden, işlemcinin hızını sekteye uğratmaktadır ve yüksek hızda mikrodenetleyici geliştirilmesinde büyük bir engeldir (Traylor, 2009).

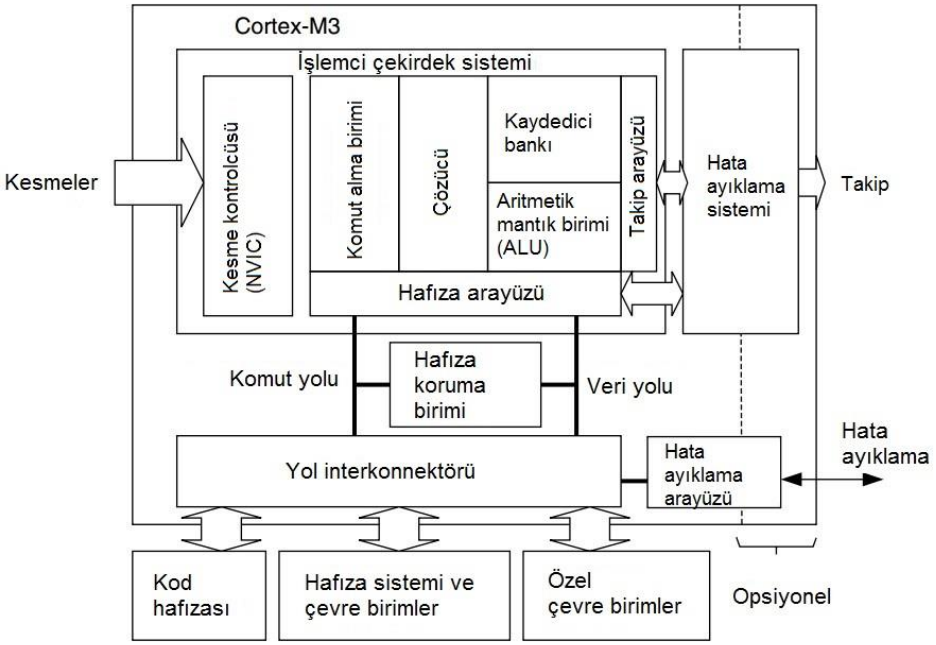
Harvard mimarisi, Harvard Üniversitesi'nde geliştirildiğinden ismini buradan almıştır. Veri yolları birbirinden bağımsız olacak şekilde, komut hafızası ve veri hafızası başta olmak üzere bütün veri yolları ayrıdır. Böylece MİB, aynı anda çoklu haberleşme yapabilmekte ve işlem hızı yükselmektedir. Günümüzde tercih edilen mimari yapıdır (Traylor, 2009). Şekil 4.2'de iki mimari arası fark gösterilmiştir.



Şekil 4.2 Von Neumann ve Harvard mimari yapısı (Traylor, 2009).

4.5 ARM Cortex-M3 Mimarisi

Cortex-M3 mimarisi 32 bit işlemci yapısına göre tasarlanmıştır. Harvard mimarisine sahip olup ayrı veri yolu ve komut yoluna sahiptir. 4 GB'a kadar hafızayı destekler. Hafıza Koruma Birimi'ne (Memory Protection Unit)(MPU) sahiptir. Hata ayıklama (debug) moduna sahiptir, bu sayede programcı geliştirme sırasında mikro denetleyici ve hafıza birimlerini gözlemleyebilir. En önemli özelliklerinden birisi olan İç içe Vektörlenmiş Kesme Denetleyicisi (Nested Vectored Interrupt Controller)(NVIC), kesmeleri yönetmektedir (Valvano, 2011). Şekil 4.3'te basitleştirilmiş Cortex-M3 yapısı verilmiştir.

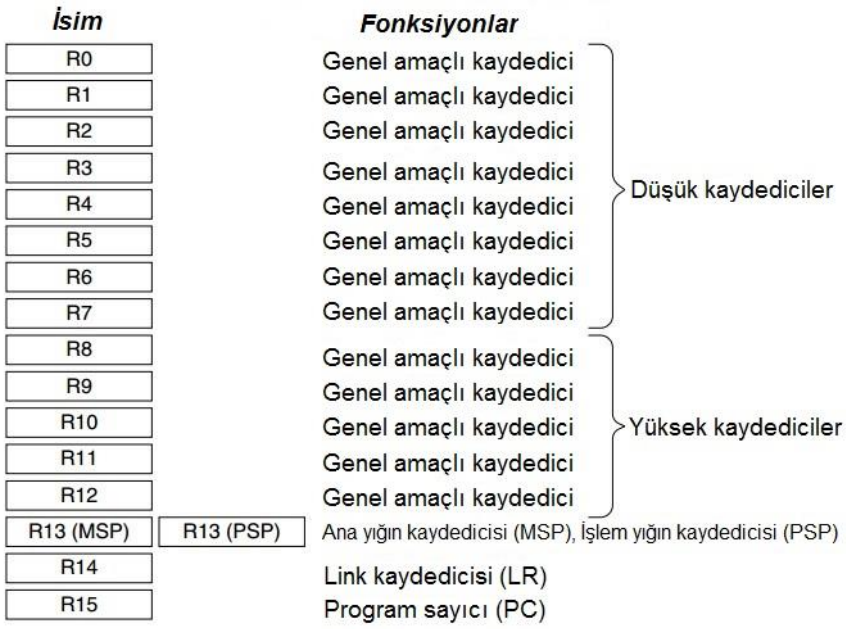


Şekil 4.3 Basitleştirilmiş Cortex-M3 yapısı (Yiu, 2009).

4.5.1 Kaydediciler (Registers)

Kaydediciler, içlerinde veri depolayan ve bitlerden oluşan en küçük hafıza birimleridir. Cortex-M3 mimarisinde, genel amaçlı 13 kaydedici vardır (R0-R12). Bunlar veri veya adres tutabilirler, bu birimleri programcı programında geçici hafıza olarak kullanabilir. R13 yığın işaretçisidir (stack pointer). Yığın, program döngüsünde ana programın dışına çıkan programda işlenen alt program, fonksiyon, kesme gibi aktif olan olaylar hakkında bilgi tutan yapılardır. R13 ise, yığının tepesini işaret eder. R14 kaydedicisi link kaydedicisi olup fonksiyonlar için dönüş adresini tutar. R15 kaydedicisi program sayıcı olarak kullanılıp,

sıradaki işlenecek komutu işaret eder. R0-R15 arasındaki tüm kaydediciler 32 bitlik boyuta sahiptir (Yiu, 2009). Şekil 4.4'te Cortex-M3'e ait kaydediciler ve görevleri verilmiştir.



Şekil 4.4 Cortex-M3 kaydedicileri (Yiu, 2009).

Bunun dışında kullanılan özel kaydediciler de vardır. PSR (Program Status Register)(Program Durum Kaydedicisi), birkaç durum kaydedicisinden oluşur. Bunlar APSR (Application Program Status Register)(Uygulama Program Durum Kaydedicisi), IPRS (Interrupt Program Status Register)(Kesme Program Durum Kaydedicisi), ve EPSR (Execution Program Status Register)(Uygulama Program Durum Kaydedicisi)'dir. Kısaca özetlersek, gerçekleşen kesmenin nerede gerçekleştiği, matematiksel ve aritmetik işlemlere ait taşma, negatif işaret gibi durum bayrakları bilgilerini içerir. Diğer özel kaydedicilerden PRIMAX, FAULTMASK, BASEPRI kaydedicileri, kesmeleri maskelemek için kullanılır. Kesmeleri etkin ve yetkisiz kılmak, kesme önceliklerini belirlemek gibi görevleri vardır. CONTROL kaydedicisinde imtiyazlı seviyeden kullanıcı seviyesine geçiş yapılabilir (Valvano, 2011). Şekil 4.5'te özel kaydediciler verilmiştir.

Şekil 4.5 Cortex-M3 özel kaydedicileri (Yiu, 2009).

4.5.2 Çalışma modları

İş parçacığı (Thread) yöntemi ve İşleyici (Handler) yöntemi olmak üzere iki çalışma yöntemi vardır. İş parçacığı yöntemi temelde normal çalışma yöntemidir ve ön planda çalışan program olarak algılanabilir, işleyici yöntemi ise kesme gerçekleştiğinde kesmelerin işletildiği yöntemdir ve arka planda çalışan program olarak algılanabilir (Valvano, 2011).

İmtiyaz seviyesi bakımından iki seviye vardır. Bunlar imtiyazlı seviye (previliged-level) ve kullanıcı seviyesidir (user-level). İmtiyazlı seviyede yazılım ile bütün işlem komutları ve kaynaklara erişilebilir, kullanıcı modunda ise sınırlı komut erişimi, kontrol bloklarına müdahale edememe, hafızaya ve çevre birimlere erişim engeli gibi sınırlamalarla karşılaşılır (Valvano, 2011).

4.5.3 İç içe vektörlenmiş kesme denetleyicisi (NVIC)

Kesme denetleyicisi, gerçekleşen kesmeleri yöneten birimdir. İç içe kesme desteği ile işlenen bir kesme olayı sürerken başka bir kesme işlemi oluştuğunda (kesmenin kesilmesi) devreye sokabilir. Vektörlenmiş kesme desteği ile adreslenenen kesme olaylarının kesme gerçekleştiğinde kullanıcıdan işlemek istediği koda geçmesi sağlanır. Kesme maskelemesi uygulayabilmesi ve kesmelerde gecikmeyi azaltması başlıca özellikleridir. Öncelik seviyesi de dinamik olarak değiştirilebilir (Yiu, 2009).

4.5.4 Hafıza haritası

4 GB destekli hafızada hafıza haritası Şekil 4.6'daki gibidir. Hafıza ataması başlangıçtan bitişe kadar birbirini takip eden bölümlerde belirlenen kapasitelere göre yapılmıştır.

511 MB	Markaya özel hafıza
1 MB	Özel çevre birim hafızası
1 GB	Dış aygıt
1 GB	Harici RAM
0,5 GB	Çevre Birimler
0,5 GB	SRAM
0,5 GB	Kod

Şekil 4.6 Cortex-M3 özel kaydedicileri (Yiu, 2009).

4.5.5 Veri yolu arabirimi

Mikro denetleyicide birkaç tür veri yolunun kullanılması hız açısından büyük bir avantaj sağlamaktadır. Veri yollarını görevlerine göre üçe ayırmak mümkündür: Kod hafızası, sistem ve özel çevrebirim veri yolları. Kod hafızası I-Code ve D-Code veri yolu olmak üzere iki ayrı veri yoluna sahiptir. I-Code komutların iletilmesinde, D-Code ise verilerin iletilmesinde kullanılan veri yoludur. Sistem veri yolları SRAM, harici RAM gibi birimlerle veri aktarımında kullanılmaktadır. Özel çevre birim veri yolları ise sadece hata ayıklama işlemlerinde kullanılan veri yollarıdır (Valvano, 2009).

4.5.6 Komut seti

Thumb-2 ismiyle geliştirilmiş komut seti kullanılmaktadır. Bu setin özelliği, 16 bitlik komutların yanında, ihtiyaçlardan dolayı geliştirilen 32 bitlik komutlar da içermesidir. Kullanım kolaylığı, performans ve kod boyutu açısından tercih edilmektedir (Yiu, 2009).

4.5.7 Hafıza koruma birimi (MPU)

Hafıza Koruma Birimi (Memory Protection Unit) opsiyonel olmakla birlikte, hafıza alanlarına erişim haklarını tanımlamak için kullanılan birimdir.

İmtiyazlı ve kullanıcı çalışma modlarında ayarlanan erişim izinleri çiğnendiğinde, hata kesmesine girerek istenilen kodlar çalıştırılabilir (Valvano, 2009).

4.5.8 Kesmeler ve istisnalar (Interrupts and exceptions)

Kesme sistemi direkt veya dışarıdan kesmelere izin vermektedir. Kesme özellikleri NVIC içerisine konulmuştur. Ayrıca efektif güç tüketimi için uyku modu ve derin uyku modu gibi bazı istisnalara da sahiptir (Valvano, 2011).

4.5.9 Hata ayıklama desteği (Debug)

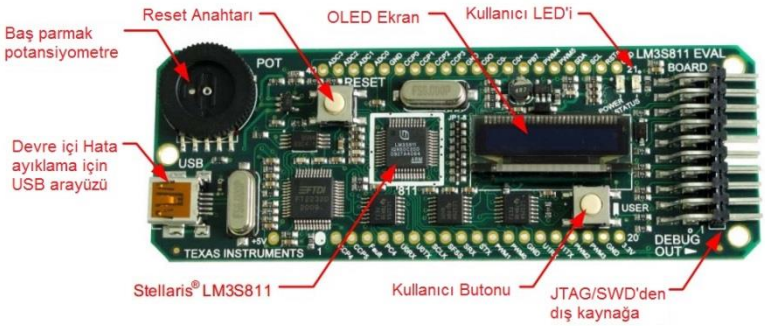
Kırılma noktaları, izleme noktaları, durdurma, adım adım işleme gibi özelliklerine sahip olan hata ayıklama desteği, kaydedici ve hafıza alanlarına da erişim sağlayarak yazılım geliştirmede gerçek zamanlı izleme ve kontrol desteği sağlanmaktadır. Bütün bu işlemler de Hata Ayıklama Erişim Noktası (Debug Access Point) (DAP) üzerinden gerçekleştirmektedir (Yiu, 2009).

4.5.10 Mimarinin karakteristik avantajları

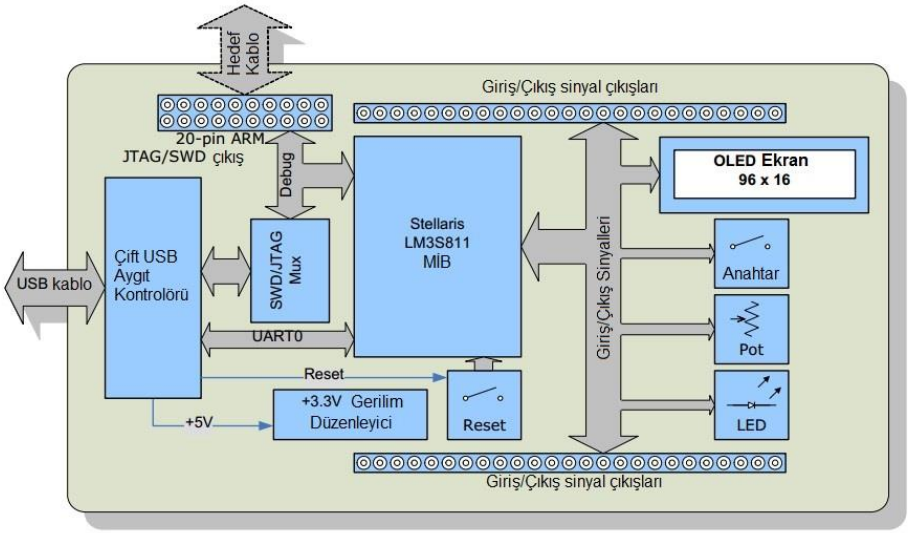
Kullanılan Thumb-2 komut seti, ayrık komut-veri yolu, yüksek saat frekansı performansını arttırması; 240'a kadar dış kesme desteği, vektörlenmiş kesme ile düşük kesme gecikmesi, bazı özel kaydedicilerin durumlarının otomatik güncellenmesi, NVIC ile kesme seviyesi belirleme özellikleri kesme idaresini geliştirmesi; düşük sayıda kapı sayısı, güç tasarruf modları güç tüketimini azalatması; programcının işini kolaylaştıran seri hata ayıklama ara yüzleri ve diğer sistem özellikleri ile Cortex-M3 tercih edilen bir mimari olmuştur (Valvano, 2011).

4.6 Texas Stellaris LM3S811 Geliştirme Kartı ve Bazı Çevre Birimleri

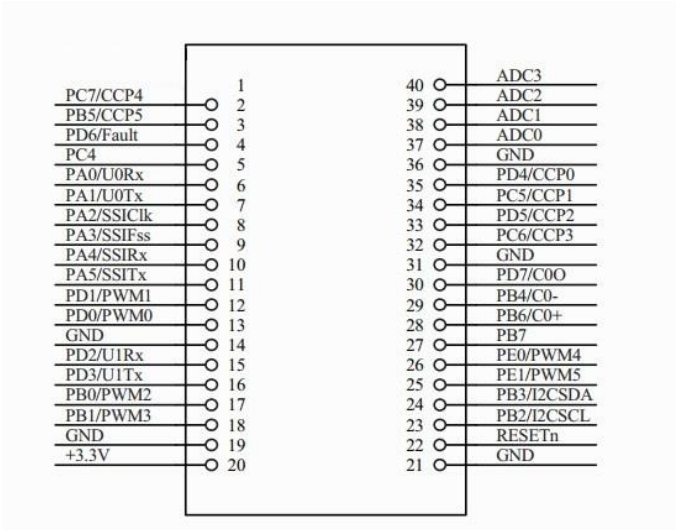
Çalışmada kullanılan LM3S811 geliştirme kartına ait görsel Şekil 4.7'de, blok diyagramı Şekil 4.8'de, pin çıkış diyagramı ise Şekil 4.9'da, mikrodenetleyici yüksek seviye blok diyagramı ise Ek 1'de verilmektedir.



Şekil 4.7 Geliştirme kartı görseli (Texas Instruments, 2010).



Şekil 4.8 LM3S811 geliştirme kartı blok diyagramı (Texas Instruments, 2010).



Şekil 4.9 Stellaris LM3S811 geliştirme kartı çıkış pinleri (Texas Instruments, 2010).

4.6.1 Genel amaçlı giriş/çıkışlar

LM3S811'de 1-32 adet genel amaçlı giriş/çıkış bulunur. Bunlar da 5 adet gruptan oluşup (A, B, C, D, E) her birinde 8 pin bulunmaktadır. Bu pinler ayarlanabilir olup kontrol kaydedicileri üzerinden fonksiyon ve ayar yapılandırması yapılabilir. Okuma ve yazma yapılabilmesi için de gruplara ait okuma ve yazma kaydediciler bulunmaktadır. Giriş-çıkış pinlerinin ayrıca yukarı çekme (pull-up), aşağı çekme (pull-down), analog gibi çalışma yöntemlerini ayarlamak için pin grubuna ait kaydediciler ayarlanabilir. Bunun dışında giriş-çıkış okuma-yazma frekansı, kesme algılaması, sinyal seçimi, toparlanma hızı gibi birkaç ayarı daha çeşitli kaydedicilerle gerçekleştirilir (Texas Instruments, 2014).

4.6.2 Zamanlayıcılar

LM3S811'de 3 adet 32 bitlik genel amaçlı zamanlayıcı/sayıcıya sahiptir. Bu zamanlayıcılar 2 adet 16 bitlik zamanlayıcı olarak da kullanılabilir, yani 6 adet 16 bitlik zamanlayıcı anlamına gelmektedir. Bir sinyali analogdan dijitale çevirmede kullanılabilir veya sayıcı olarak görev alabilirler. Periyodik, tek seferlik çalışma, aşağı sayma, yukarı sayma gibi çalışma yöntemleri vardır. Kaydediciler ile kullanılacak zamanlayıcıların çalışma yöntemleri ayarlanabilir (Texas Instruments, 2014).

Sistemdeki diğer zamanlayıcılar ise PWM üretmekte kullanılan PWM zamanlayıcısı ve sistemin kalp atışı gibi çalışan Sistem Zamanlayıcısı (SysTick)'dır (Texas Instruments, 2014).

4.6.3 Kesmeler

LM3S811'de kesmeler NVIC ile denetlenmektedir ve 26 kesmeyi desteklemektedir. Kesme öncelik seviyeleri 0-7 arasında olmakla birlikte, yüksek numaralı kesmenin önceliği daha düşüktür ve 0 numaralı kesme en yüksek öncelikli kesmedir. Önceliği yüksek olan bir kesme, gerçekleşmekte olan düşük öncelikli kesme işlemini kesintiye uğratıp kendi işlemini uygular, daha sonra düşük öncelikli kesme işlemi kaldığı yerden işlemine devam eder. Eğer önceliği yüksek olan kesme işlemi gerçekleşirken düşük öncelikli kesme gerçekleşirse, düşük öncelikli olan kesme gerçekleşmekte olan kesme işleminin bitmesini beklemek zorundadır. Bazı kesme çeşitleri şunlardır: Yeniden başlatma, veri yolu

hatası, kullanım hatası, sistem saati, dış, seri haberleşme, PWM üreteci, analog, zamanlayıcı, bekçi köpeği (watchdog) kesmeleri (Texas Instruments, 2014).

4.6.4 UART haberleşme

Seri haberleşme donanımı olan Evrensel Asenkron Alıcı/Verici (Universal Asynchronous Receiver/Transmitter)(UART), asenkron olarak baytlarca verinin seri alışverişine dayanan bir haberleşme sağlar. RS-232, RS-485 gibi haberleşme standartlarında kullanılır. Temelde alıcı (Rx) ve gönderici (Tx) veri pinleri ile bağlantı sağlar. Ayarlanabilir veri hızı, veri kontrolü gibi özelliklere sahiptir. Stellaris üzerinde 2 adet UART kanalı yer almaktadır (Texas Instruments, 2014).

4.6.5 Sistem zamanlayıcısı (SysTick)

Entegre olarak M3 içinde bulunan sistem zamanlayıcısı, otomatik yüklemeli 24-bitlik aşağı sayıcı bir zamanlayıcıdır. Otomatik yükleme değeri ayarlanarak otomatik yükleme süresi yani zamanlayıcının frekansı da ayarlanabilir. Anlık değer, ölçümleme, yükleme değeri, kontrol ve durum saklayıcıları vardır (Texas Instruments, 2014).

4.6.6 Darbe genişlik modülasyonu (PWM)

PMW bir modülasyon çeşidi olup, analog sinyalin dijital olarak enkode edilmesinde kullanılır. PWM sinyalinde bir kare dalga oluşturulur ve doluluk oranı (dalganın aktif olduğu sürenin dalganın periyoduna oranı) değiştirilerek analog değerin genliği değiştirilebilir. LM3S811 üzerinde 3 adet PWM üreteci ve her birinde ikişer adet olmak üzere toplamda 6 adet PWM çıkışı yer almaktadır. Kaydediciler ile bu sinyallerin doluluk oranı, frekansı, aktiflik-pasiflik, kesme, durum ayar ve kontrolleri yapılabilmektedir (Texas Instruments, 2014).

4.7 Programlama

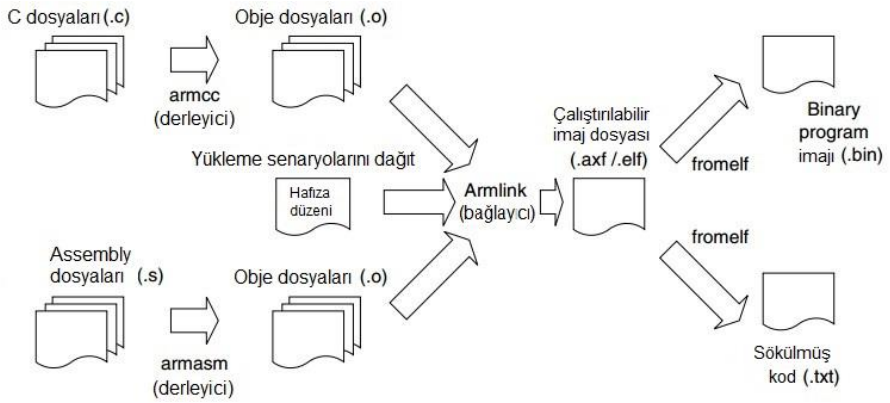
ARM tabanlı mikro denetleyicileri genel olarak makine (Assembly) dilinde programlamak mümkündür. Fakat üretici firmalar, çevre birimlerin kullanımını kolaylaştırmak amacıyla ARM Cortex-M tabanlı mikro denetleyicilerine özel olarak Çevre Birim Sürücü Kütüphanesi geliştirmişlerdir. C dilinde programlama yazılım geliştirme aygıt sürücülerini ve Cortex Mikrodenetleyici Yazılım Arayüzü Standardını (CMSIS)(Cortex Microcontroller Software Interface Standart)

kullanarak yapılabilir. C dilinde programlanabilen, anlaşılması kolay, etkin ve derleyiciyle uyum halinde hata tespiti yapabilen tasarım, büyük boyutlu program yazımında programcıya büyük kolaylık sağlamaktadır. Kütüphanenin uyumlu olduğu geliştirme ortamları şöyledir: Keil RealView Microcontroller Development Kit, CodeSourcery Sourcery G++ for Stellaris EABI, IAR Embedded Workbench, Code Red Technologies tools ve Texas Instruments Code Composer Studio. Çalışma yazılımı Keil platformunda geliştirilmiştir (Texas Instruments, 2012). Keil, gömülü sistemlerin geliştirilmesinde kullanılan bir platformdur. ARM, Cortex-M, 8051 gibi mikrodenetleyici ailelerine ait üyeler bu platformu kullanarak programlanabilir.

C programlama, "main" ismindeki temel fonksiyon ile başlar. Program her çalıştırılışında bu fonksiyonda yer alan komutları çalıştırır. Main içinden ise diğer farklı fonksiyonlar çalıştırılabilir. Programın başında yer alan "#include" ile, standart veya standart olmayan, bağlantı (header) olarak adlandırılan dosyalar dahil ederek, içeriklerinde yer alan fonksiyonları kullanıma sunar ve derleyebilir.

C programında değişkenler "int" (tamsayı), "double" (ondalıklı sayı), "float" (ondalıklı sayı), "boolean" (mantıksal doğru ya da yanlış) gibi değişken isimleriyle tanımlanır. Bu değişkenlerle aritmetik işlemler yapılabilir. Ayrıca "if" (eğer), "for" (için), "while" (iken) ve "switch" (anahtar) ifadeleriyle karşılaştırmaya dayalı işlemler ve döngüler de programlamanın temel yapısını oluşturmaktadır. Diziler (arrays) ve karakter dizileri (strings), verilerin ve metinlerin tek bir dizi isminde erişim ve işleme kolaylığı sağlar. Yapılar (structures) ile aynı veya farklı tipteki değişkenleri bir grupta toplayarak erişim ve işleme kolaylığı sağlar. İşaretçi (pointer), fonksiyon ve dizilerde değişkenin hafızadaki adresini gösteren ve programlamayı kolaylaştıran kuvvetli bir özelliktir.

Programı tamamlanan yazılımın mikro denetleyicide çalıştırılıncaya kadar izlenen aşamalar Şekil 4.10'daki gibidir. Kod üretirken temel olarak birleştirici (assembler), C derleyici (compiler), bağlayıcı (linker) ve binary dosya üretecine ihtiyaç vardır (Yiu, 2009).



Şekil 4.10 Tipik kod üretme akış diyagramı (Yiu, 2009).

5. VİSUAL C#

C#, Microsoft firmasının geliştirdiği, .NET çatısını kullanan bir programlama dilidir. C programlama dilinden türetilmiştir. Öğrenilmesi C ve C++ dillerini öğrenmekten daha kolaydır. Yazılımcıların uygulama yazarken çokça tercih ettikleri bir dildir. C++ ve Java dilleri ile bazı özellikleri aynıdır. Java gibi nesne yönelimli (object oriented) bir dildir ve geniş bir sınıf (class) kütüphanesine sahiptir. Bir bakımdan Java programlama diline rakip olarak da görülebilir. C#, bunun gibi önemli birçok özelliğe sahip bir dildir (Jones and Freeman, 2010). Programın geliştirme ortamı Microsoft'un Visual Studio programıdır, son sürümü Visual Studio 2015'tir. Visual C# ise, Visual Studio geliştirme ortamında C# proje uygulamasıdır. En önemli özelliği sunduğu kolay geliştirilebilir görsel ara yüz desteğidir.

5.1 C# Dilinin Tarihçesi

C# dilinin tarihine bakacak olursak, Microsoft firmasından Anders Hejlsberg, 1999 yılında Cool isminde (C Benzeri Obje Yönelimli Dil)(C-like Object Oriented Language) bir dil yaratmak üzere takımı ile yola çıkmıştır. 2000 yılında ise ismi C# olarak değiştirilmiştir. Zaman içinde gelişimlere de uğrayarak C# 1.0'dan başlayarak günümüzde C# 6.0 sürümü ortaya çıkmış, bu gelişimi sırasında da .NET çatısı da gelişim göstererek .NET Framework 1.0'dan başlayarak .NET Framework 4.6 sürümü ortaya çıkmıştır (Wikipedia contributors, 2015b).

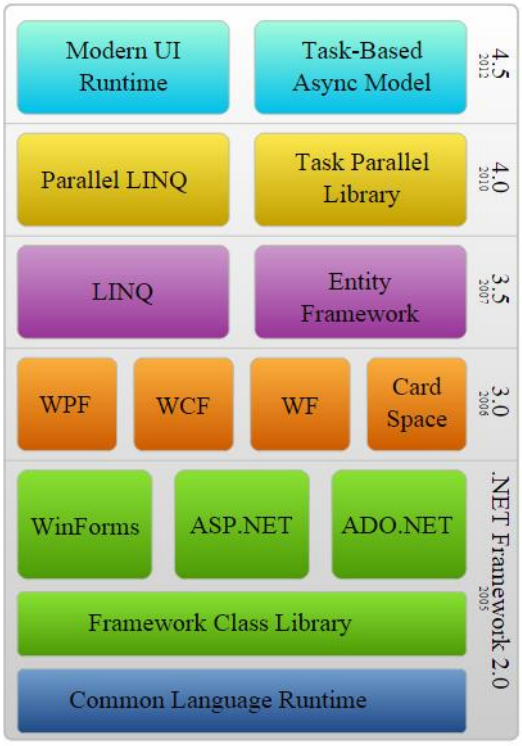
5.2 Nesne Yönelimli Programlama (Object Oriented Programming)

C gibi fonksiyonlar, rutinler, alt rutinlerin kullanıldığı prosedürel yöntemin özellikle büyük boyutlu çalışmalarda yetersiz kalmasından dolayı nesne yönelimli programlama yöntemine geçilmiştir. Bu programlama mantığı sınıf (class) adı verilen yapılar üzerine kurulmuştur. Sınıf, bir veri yapısı çeşidi (data structure) olarak görülebilir. İçerisinde farklı veri tipinde değişkenler ve kod parçalarından oluşan yöntemler (methods) barındırabilir. Soyut karakteristik olarak bir değişken tipi de denebilir. Nesne (object) ise bunun somutlaştırılmış halidir, sınıf tipinde bir değişken olarak düşünülebilir ve o sınıftan oluşturulmuş her nesne kendisine ait sınıf karakteristiğine sahiptir. Sınıf yapısı, hızlı programlama ve kolay bakım avantajlarına sahiptir. Programın hızlı yapılandırılmasında hiyerarşik yapısıyla

hem kendi içinde, hem de diğer sınıflarla ilişkisinde seviye etkileşimiyle etkilidir. Sınıf yapısında, enkapsülasyon özelliğiyle sınıfın içinde yer alan değişken ve yöntemlerin dışarıdan erişim seviyeleri ayarlanabilir olması programcıya büyük bir avantaj sağlar. Kütüphanelerde yer alan değişkenler ve yöntemler aynı isimde olabilir, bu çakışma sorunu ise isim alanı (namespace) ile çözülür (McMonnies, 2004).

5.3 .NET Çatısı (.NET Framework)

Microsoft tarafından geliştirilen, temelde Microsoft Windows üzerinde çalıştırılan yazılım yapısıdır. Birkaç yazılım diliyle uyum içerisinde geniş sınıf kütüphanelerine sahiptir. Windows, Mobil, Web uygulamaları geliştirilebilir. CLR (Common Language Runtime) ve FCL (Framework Class Library) olmak üzere iki temel bileşeni vardır. Uygulama çalıştırıldığında, derleme sonucu işletim sistemine uygun makine kodu üreten CLR devreye girer. FCL ise yazılımı yazarken yazılımcıya geniş bir kütüphane sunar. İçerisinde kullanıcı ara yüzü, veri tabanı, kriptolama, ağ uygulamaları geliştirme, nümerik algoritmalar ve web haberleşmesi gibi yazılımcının işini kolaylaştıran kütüphaneler sunmaktadır. Şekil 5.1'de .NET çatısına ait elemanlar verilmiştir.

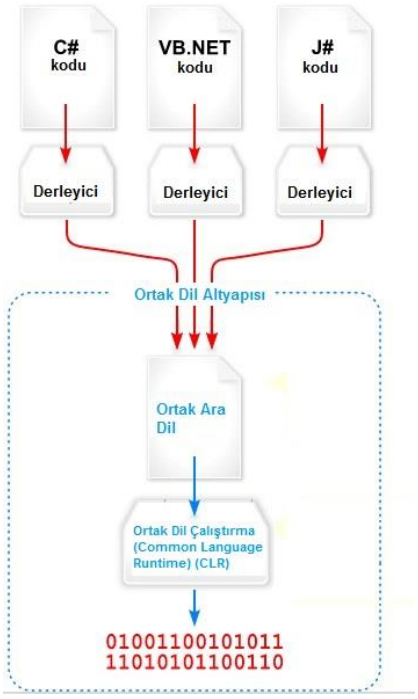


Şekil 5.1 .NET çatısı elemanları (Wikipedia Contributors, 2015a).

5.4 .NET Mimarisi

5.4.1 Ortak dil altyapısı (Common language infrastructure)

.NET, birden fazla yazılım dili için oluşturulan bir çatıdır. C#, Visual Basic .NET, C++ gibi desteklenen dillerin derlenmesi sonucu direkt olarak makine kodu üretilmez, bunun yerine her bir yazılım dilinin de ortak olduğu bir nötr dile çevrilir, ardından CLR ile işlenip makine kodu üretilir (Msdn training, 2001). Şekil 5.2'de .NET ortak dil altyapısının çalışma diyagramı verilmiştir.



Şekil 5.2 .NET ortak dil altyapısı (Wikipedia Contributors, 2015a).

5.4.2 Sınıf (Class) kütüphanesi

.NET standart sınıf kütüphane setine sahiptir. Bu kütüphane seti ikiye ayrılır: Çatı Sınıf Kütüphanesi (Framework Class Library)(FCL) ve Temel Sınıf Kütüphanesi (Base Class Library)(BCL). BCL'de temel seviye olan bazı sınıflar yer alır (zaman ve tarih sınıfları gibi), FCL ise geniş kütüphane setlerini kapsamaktadır (Windows Forms, ASP.NET, ADO.NET, WCF, WPF gibi)(Msdn training, 2001).

5.4.3 .NET çekirdeği

Yakın zamanda açık kaynak olarak sunulan bu özellik ile çapraz platform olarak Visual Studio programı kullanılarak sadece Microsoft Windows ve Windows Phone üzerinde değil, Linux, Android gibi farklı işletim sistemi üzerinde de çalışabilecek yazılımlar geliştirmek mümkün duruma getirilmiştir. Açık kaynak olması nedeniyle de, sadece Microsoft geliştiricileri değil, farklı platformlar üzerinde çalışan geliştiricilerin de .NET çatısını kullanarak yazılım geliştirmesine imkan sunulmuştur (Wikipedia contributors, 2015b).

5.5 Windows Forms

Windows Forms, grafik ara yüzü (GUI) oluşturmada kolaylık sağlayan .NET çatısında yer alan bir sınıf kütüphanesidir. GUI, kullanıcı ile bilgisayar arasında grafiksel bir etkileşim kurarak (form uygulaması) programı ergonomik ve kullanıcı dostu hale getirerek grafiksel çıktıların yazılımcı tarafından geliştirilmesini kolaylaştırmaktır. C++ ile oluşturulan grafik ara yüzlerin programlaması çok uzun ve karmaşık olduğundan bu kütüphane yaratılmıştır (Wikipedia contributors, 2015b).

Grafik elemanları sınıf bazında olup, her birinin karakteristik özellikleri ve yöntemleri vardır. Sıkça kullanılan bazı grafiksel elemanlar şunlardır: Metin kutusu, buton, etiket, zamanlayıcı, kontrol kutusu, karışım kutusu, liste kutusu, vb.

Windows Form sınıflarındaki tüm görsel elemanlar kontrol sınıfından türetilmiştir. Windows Form uygulamaları, olay sürümlü uygulamalar olup, olayın gerçekleşmesiyle birlikte belirlenen kodların çalıştırılması şeklindedir. Örneğin, grafiksel olarak yaratılmış bir butona tıklandığı takdirde butonda tıklama olayı sürülmüş olur ve bu olayın tetiklediği bloktaki kodlar çalıştırılır.

5.6 Visual C# Programlama

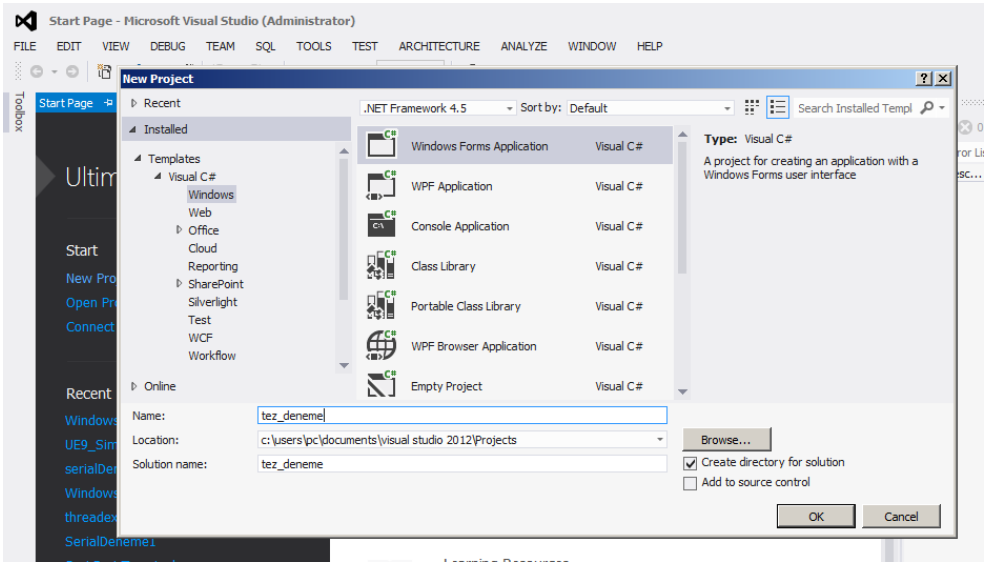
Visual Studio programında C# programlamada program bloğu, isim boşluğu (namespace) bloğunda yer alır. Bloğun üstünde “using” olarak başlayan ve dâhil edilmek istenen kütüphanelere yer verilir. İsim boşluğu bloğunun içerisinde “Program” isminde bir sınıf bloğu açılır, bu bloğun içerisine ise “Main” yöntemi yazılabilir. “Main” metodu programın giriş noktasıdır. Visual C# programında ise, standart bir programda öncelikle “Main” yöntemi içerisinde form değişkenleri ve

grafiklerinin tanımlanması ve formun ekrana yansıtılması sağlanır. Bu kısma ek kod da yazılabilir. Bütün bunlar “Program.cs” isminde bir dosyada gerçekleştirilir (Jones and Freeman, 2010).

Form elementlerinin dizaynı “Form.Designer.cs”, kodlaması ise “Form.cs” dosyasında gerçekleştirilir. “Form.cs” dosyasında kısmi sınıf olarak forma ait bir sınıf açılır ve ara yüze ait atanmış olaylar (butona tıklama, metin kutusunu doldurma gibi) gerçekleştiğinde çalıştırılacak komutlar yazılır (Jones and Freeman, 2010).

Form tasarımında, program ilk defa açıldığında karşımıza boş bir form gelir. İstenilen elemanlar alet kutusu penceresinden sürükle-bırak şeklinde eklenebilir ve boyutlandırılabilir. Formda yer alan bir elemanın üzerine tıklandığında, özellikler penceresinde elemana ait karakteristik özelliklerin hepsine erişmek mümkündür. Ayrıca aynı pencereden hızlı bir şekilde kod penceresine aktarılmak istenen ve o elemana ait olan olayı hızlı bir şekilde belirleyebiliriz (örneğin, fare ile çift tıklandığında olayı).

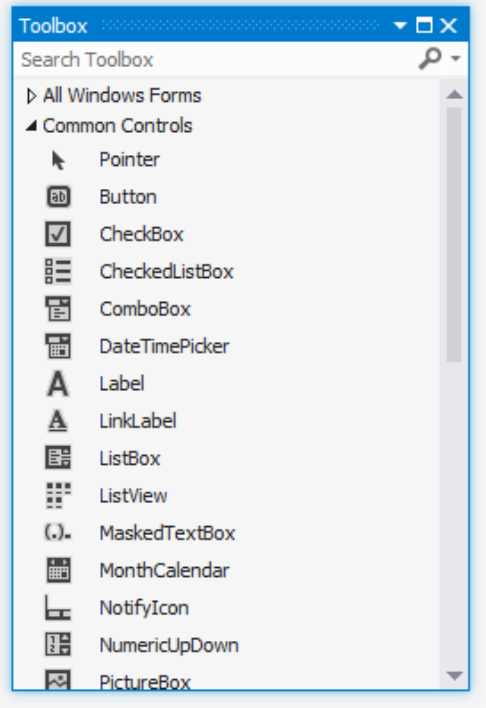
Visual Studio geliştirme ortamı Şekil 5.3’te verilmiştir. Yeni bir proje yaratma aşamasında proje türü, programlama dili gibi özellikler kullanıcıdan istenmektedir.



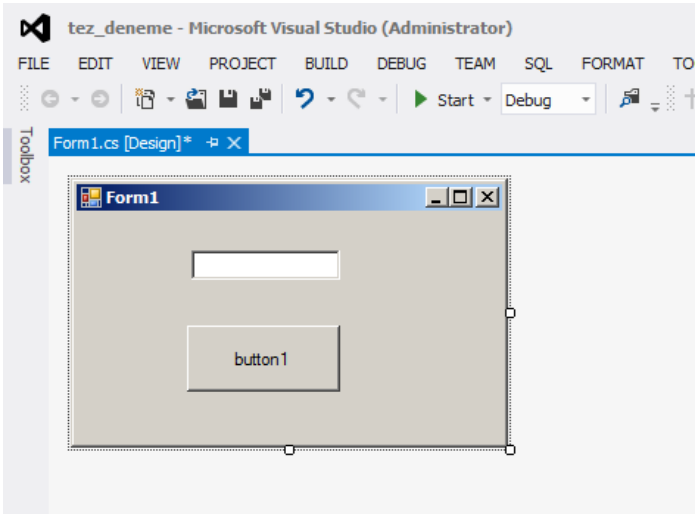
Şekil 5.3 Visual Studio yeni proje oluşturma.

Form uygulaması yaratıldığında karşılaşılan ilk ekran bir form tasarım ekranıdır. Form elemanları Şekil 5.4’teki gibi alet kutusundan seçilerek Şekil

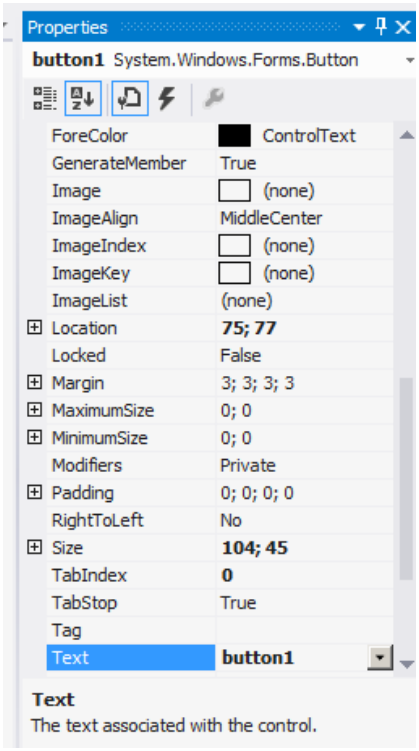
5.5'deki gibi sayfaya eklenir. Eklenen her elemanın Şekil 5.6'daki gibi özelliklerinin gözlemlenebileceği ve direkt değiştirilebileceği özellikler penceresi vardır.



Şekil 5.4 Visual C# alet kutusu.



Şekil 5.5 Visual C# formu.



Şekil 5.6 Visual C# özellikler penceresi.

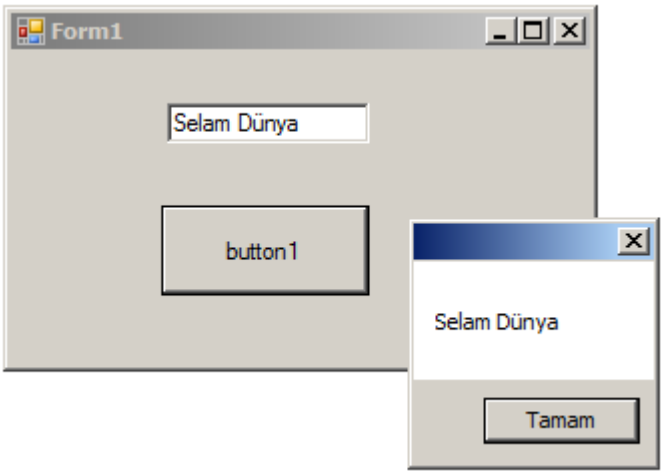
Kodlama ekranında yazılan Şekil 5.7'deki kodlar ile çalıştırılan programın çıktısı ise Şekil 5.8'de gösterilmiştir. Buna göre, butona basıldığı anda (buton tek tıklama olayı atanması) metin kutusunda yazılı olan metin, mesaj kutusu içerisinde ekranda gösterilir.

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

namespace tez_deneme
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show(textBox1.Text);
        }
    }
}
```

Şekil 5.7 Visual C# program kodu.



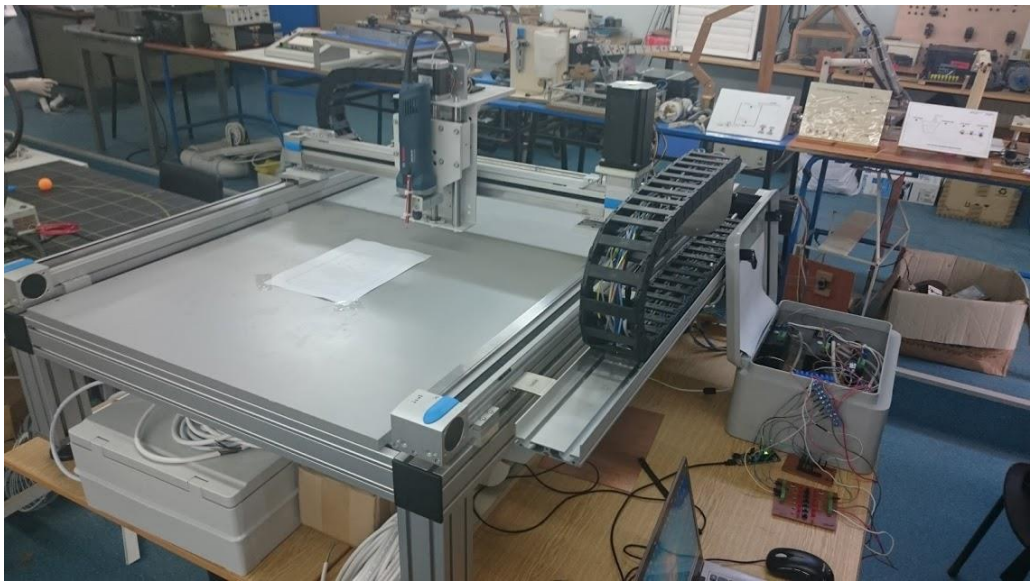
Şekil 5.8 Visual C# program çıktısı.

6. PROJENİN OLUŞTURULMASI

6.1 Mekanik Sistemin Tasarımı ve Özellikleri

Mekanik sistem ahşap, tahta gibi materyallerin işlenmesine uygun bir şekilde tasarlanmıştır. Makinanın çalışma bölgesi 916 × 666 × 96 mm^3'tür. İskelet alüminyum profillerden yapılmıştır. X ve Y ekseni kayış-kasnak mekanizmasıyla, Z ekseni ise lineer mil mekanizmasıyla hareket etmektedir. X ekseni motoru sabit olan gövdede monte olup hareketi ile hareketli arabayı (router) hareket ettirmektedir. Y ekseni hareketli arabanın üzerinde olup freze motorunun bağlı olduğu bloğu hareket ettirmektedir. Z ekseni ise lineer mil mekanizması ile freze motorunu aşağı-yukarı hareket ettirmektedir.

Freze motoru olarak 750 W gücünde bir kalıpçı taşlama motoru kullanılmıştır. Motor kabloları, hareketli kablo kanalları sayesinde hareketli arabanın pozisyonundan ve hareketlerinden etkilenmemektedir. Oluşturulan mekanik sistem Şekil 6.1'de verilmiştir.

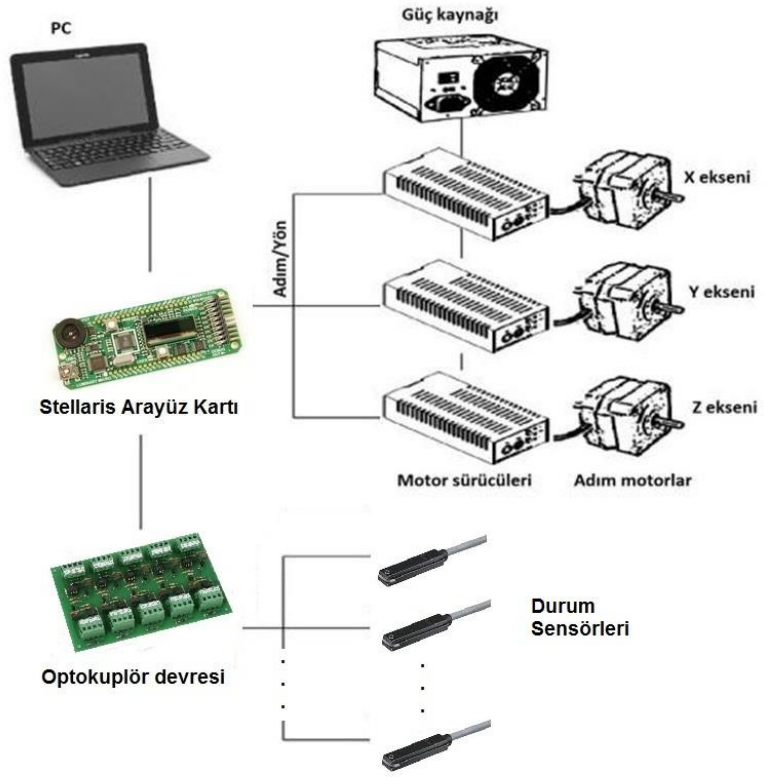


Şekil 6.1 Oluşturulan mekanik sistem.

Mekanik sistemin detaylı çizimi Ek 3'te verilmiştir.

6.2 Sistemin Elektrik-Elektronik Elemanları ve Bağlantıları

Sistemin genel bağlantı şeması Şekil 6.2'de verilmiştir.



Şekil 6.2 Sistemin genel bağlantı şeması.

Aktüatör motorları olarak NEMA standartlarında 2 adet 12,5 Nm (X ve Y ekseni), 1 adet 4,5 Nm (Z ekseni) bipolar adım motor kullanılmıştır. Bu motorların her biri 8 adet kablo ucuna sahiptir (4 faz). Motor sürücüleri mikro adım adım motor sürücüleri olup kontrol sinyali olarak PWM ve yön sinyali almaktadır. Üzerlerinde yer alan çözünürlük ayar anahtarları sayesinde hareket aktarma mekanizmalarındaki dönme/lineer hareket oranına göre 1 mikrometre hassasiyete kadar hareket komutu verilebilmektedir.

Güç kaynağı, motor sürücülerine motorları sürebilmek için besleme kaynağı görevi görmektedir. Motorlar 48 V'luk gerilimlereihtiyaç duymaktadır. Güç kaynağı ise 48 V 12 A'lik bir SMPS (Switch Mode Power Supply) güç kaynağı olup, tüm motorların aynı anda çekebilecekleri akım miktarını karşılayabilecek bir kapasiteye sahiptir.

Güvenlik amacıyla makinanın mekanik sınırların dışına çıkmaması için sınırlarda sensörler yer almaktadır. Bu sensör grubu manyetik ve indüktif sensörlerden oluşmaktadır. Bu sensörler metallere karşı duyarlı olup hareketli kısımlara monteli olan metal parçalara tepki vermektedir. Sensörler NPN

sensörlerdir, besleme ve çıkış gerilimleri pozitiftir. Manyetik sensörler normalde kapalı, indüktif sensörler ise normalde açık sensörlerdir. Yani manyetik sensörler cismi algıladıklarında 0 V, cismi algılamadıklarında ise pozitif besleme voltajı; indüktif sensörler ise cismi algıladıklarında pozitif besleme voltajı, cismi algılamadıklarında ise 0 V çıkı gerilimi üretirler. Fakat sensörlerin 15-30 V arasında bir çalışma gerilimi olduğundan, 3.3 V gerilimle çalışan mikrodenetleyici kartı ile gerilimleri uyuşmamaktadır. Bu sorunu gidermek için optokuplör devresi kullanılmıştır. Optokuplörler, iki elektronik elemanı izole etmek üzere tasarlanmış entegre elemanlardır. İçlerinde fotodiyot-fototransistör devre mantığı bulunmaktadır. Bu devre, sensörler ile mikrodenetleyiciyi birbirinden izole edip gerilim uyuşmazlığını da ortadan kaldırmaktadır.

Bilgisayar ve ara yüz kartı USB kablosu ile birbirine bağlanmıştır. Mikro denetleyici kart ise motor sürücülerine pin bağlantısı yapılarak bağlanmıştır. Sistemin detaylı bağlantı şeması verilmiştir. Mikrodenetleyici ile motor sürücülerine ait detaylı bağlantı şeması Ek 4'teki gibidir.

6.3 Yazılımlar

Yazılım iki platformdan oluşmaktadır: Bilgisayar ara yüzü ve mikro denetleyici ara yüzü.

Bilgisayar ara yüzünde, kullanıcı ile yazılım arasında iletişim için ağırlıklı olarak butonlar ve metin pencereleri mevcuttur. Kullanıcıdan butonlar ve metin kutuları sayesinde alınan girdiye göre atanan komut işlenir, makinanın karakteristiği ve algoritmaya göre mikro denetleyiciye uygun komutlar gönderilir.

Seri haberleşmeyle birbirine bağlı olan iki birim için ortak temel noktalardan biri geliştirilen haberleşme protokolüdür. Buna göre, örneğin "G01 X123" komutunun karşıya gönderilmesi veya karşıdan alınması noktasında, karşılaşılan bazı engellerin çözüm yolu olarak, komut metni "G01 X123OK" olarak gönderilir. Sonuna eklenen "OK" karakterleri, her iki birimin haberleşme kanalıyla gelen metnin ayıklanması esnasında tespit edildiğinde, o ana kadar aldığı metni komut olarak sorgulamaya ve işlemeye başlar.

CNC makinalarda hareket komutları olarak G kodları esas alınır. Çalışmada temel hareket komutları G00 ve G01 komutları ele alınmıştır. G02 ve G03 komutları ele alınmamıştır.

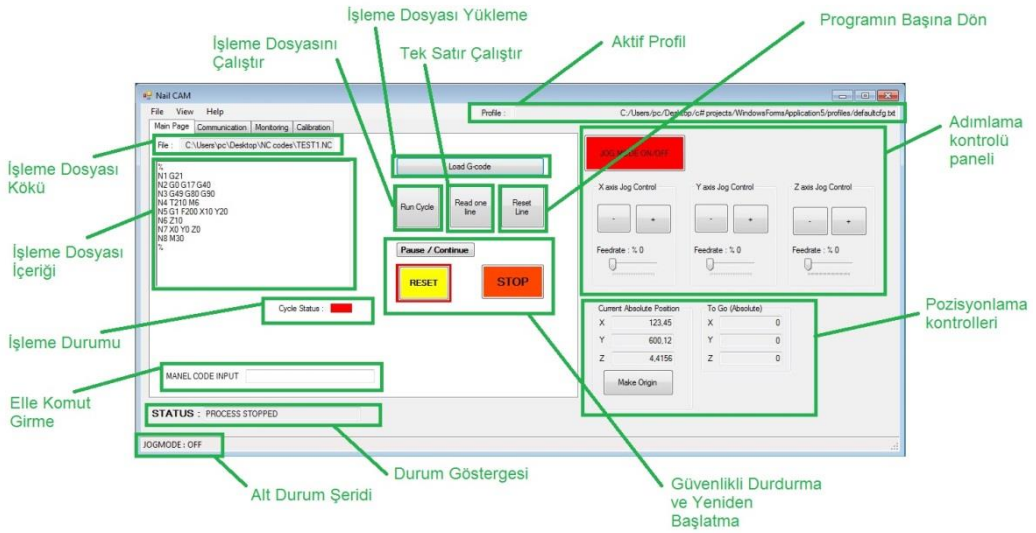
Standart G kodlarında karşılığı olmayan haberleşme protokolleri için M101, M102, M103, M104 ve M105 fonksiyonları geliştirilmiştir. M101, eksenlerde yer alan motorların dönüşleri ile dönüşlerinin yol açtığı ilerlemenin oranları olan ölçümleme katsayıları için kullanılmıştır. Bununla ilgili bilgisayar yazılımında otomatik ölçümleme yöntemi mevcuttur. M102, anlık mutlak pozisyonun eşleşmesi, bilgisayar ile mikro denetleyicide aynı anda aynı anlık mutlak pozisyon bilgisinin bulunması için geliştirilmiştir. M103, en yüksek besleme hızları için kullanılmıştır. M104, adımlama modunda kullanılmak üzere tasarlanmıştır. M105 ise limit anahtarların durum bilgisinin aktarılması için kullanılmıştır.

Sistemde güvenliği açısından bazı protokoller uygulanmaktadır. Örneğin, STOP butonuna basıldığında RESET butonu aktif hale gelir, RESET butonunu etkisizleştirmeden makine çalışmaz, RESET butonu başlangıçta sürekli aktiftir, parça işleme modunda diğer modlar çalışmaz, vb.

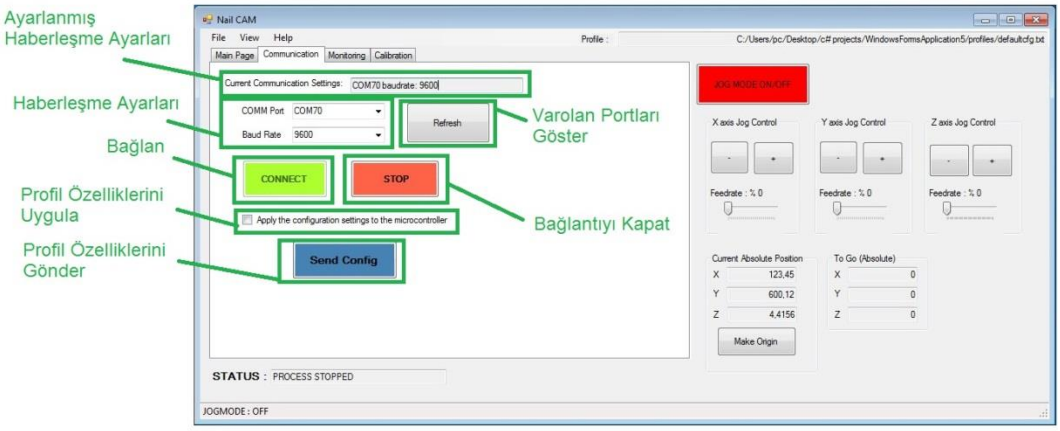
İki yazılıma da ait olan kodlar Ek 5 ve Ek 6'da verilmiştir.

6.3.1 Bilgisayar ara yüzü ve algoritması

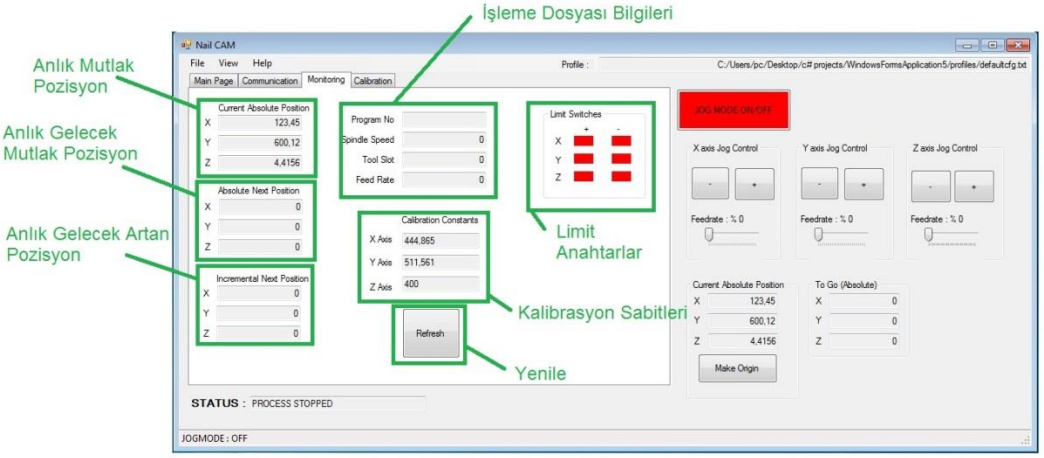
Bilgisayar grafik ara yüzü birkaç sekmeden (tab) oluşmaktadır. Bu sekmeler Ana sayfa, haberleşme, görüntüleme ve ölçülendirmedir. Bu grafik ara yüzler ve görevleri Şekil 6.3, 6.4, 6.5 ve 6.6'da verilmiştir.



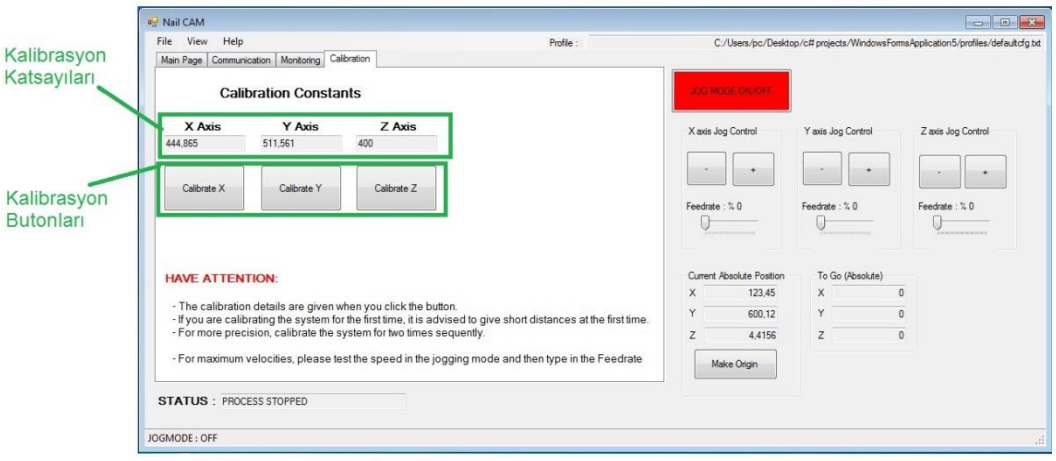
Şekil 6.3 Bilgisayar yazılımı ana sayfa arayüzü.



Şekil 6.4 Bilgisayar yazılımı haberleşme arayüzü.

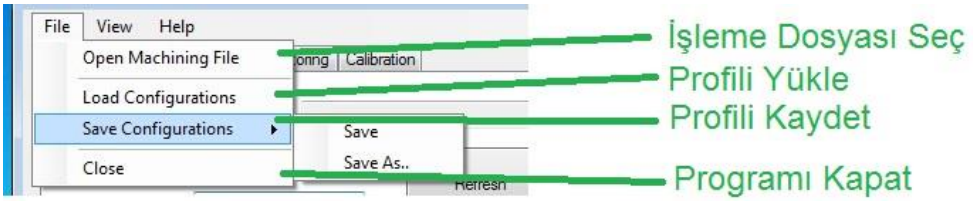


Şekil 6.5 Bilgisayar yazılımı görüntüleme arayüzü.

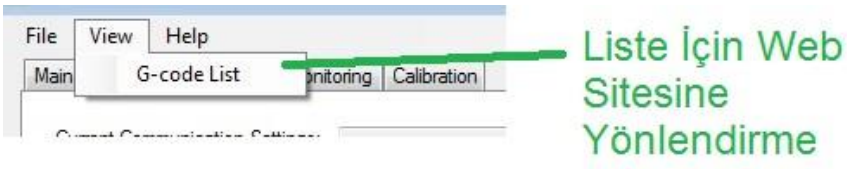


Şekil 6.6 Bilgisayar yazılımı kalibrasyon arayüzü.

Ayrıca ara yüzde menü kuşağı yer almaktadır. Menü kuşağı grafik ara yüzleri ve görevleri Şekil 6.7, 6.8 ve 6.9'da verilmiştir.



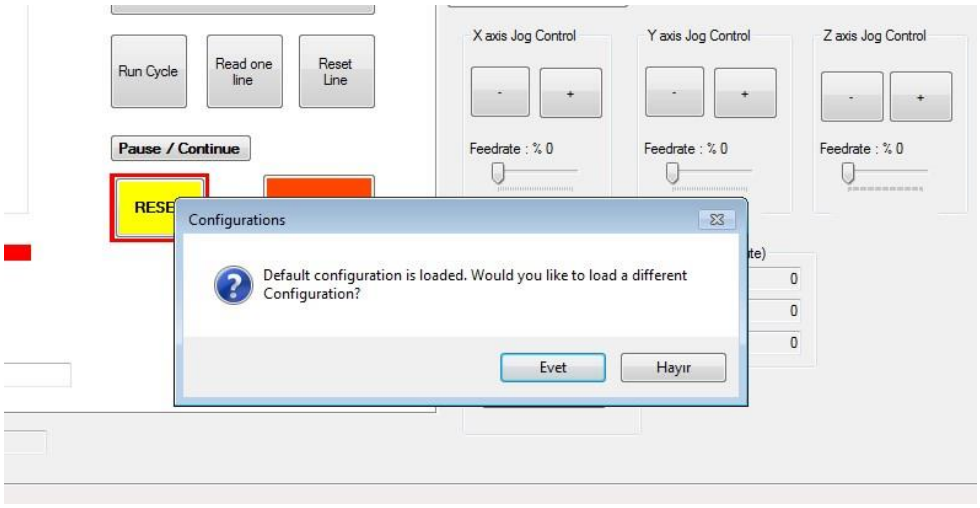
Şekil 6.7 Bilgisayar yazılımı menü kuşağı dosya sekmesi.



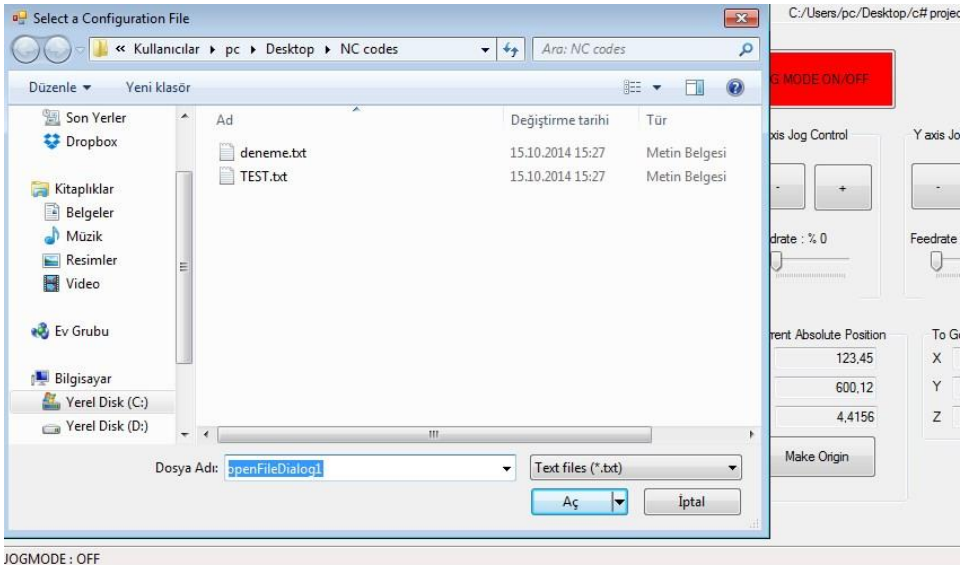
Şekil 6.8 Bilgisayar yazılımı menü kuşağı görüntüleme sekmesi.

Şekil 6.9 Bilgisayar yazılımı menü kuşağı yardım sekmesi.

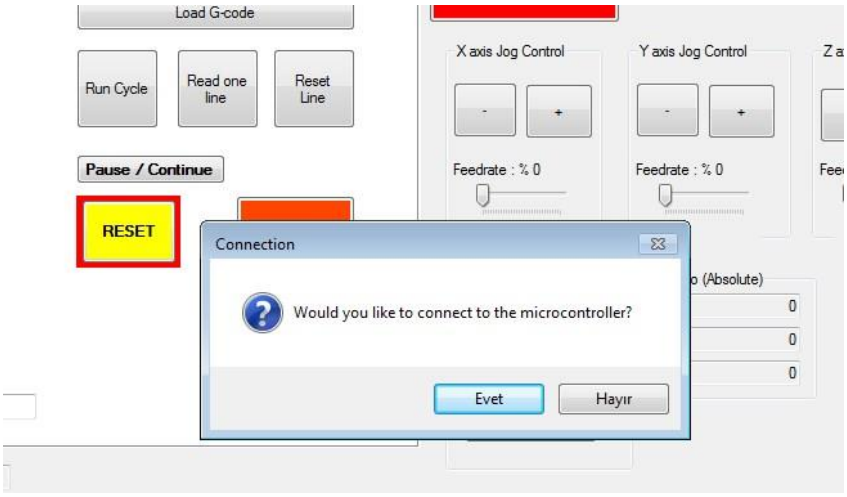
Programın başlatılmasıyla birlikte yazılım, kullanıcıya profil yükleme ve haberleşme bağlantısı için diyalog sunmaktadır. Profil dosyalarının içerisinde eksenlere ait ölçümleme katsayıları, anlık pozisyon ve maksimum hız bilgileri yer almaktadır. Şekil 6.10, 6.11 ve 6.12'de bununla ilgili diyalog pencerelerine yer verilmiştir.



Şekil 6.10 Bilgisayar yazılımı ilk kullanıcı yüklemesi.

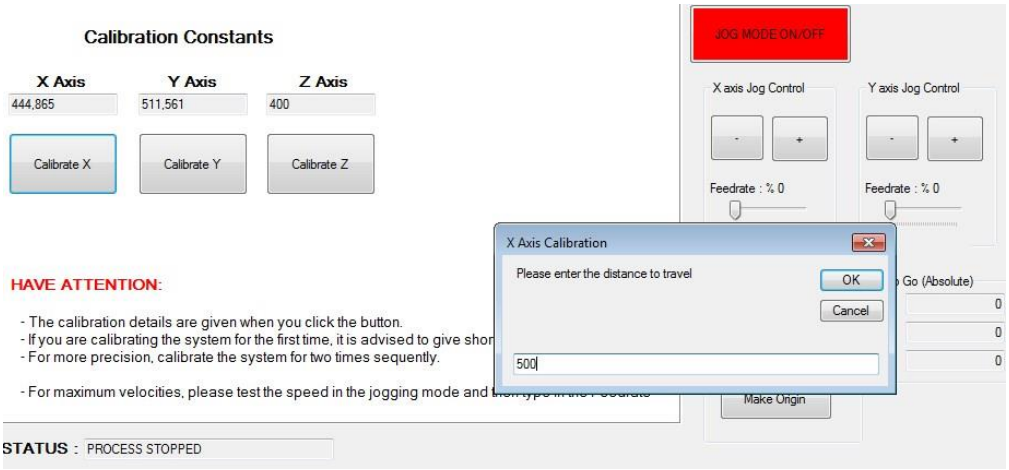


Şekil 6.2 Bilgisayar yazılımı kullanıcı yüklemesi.

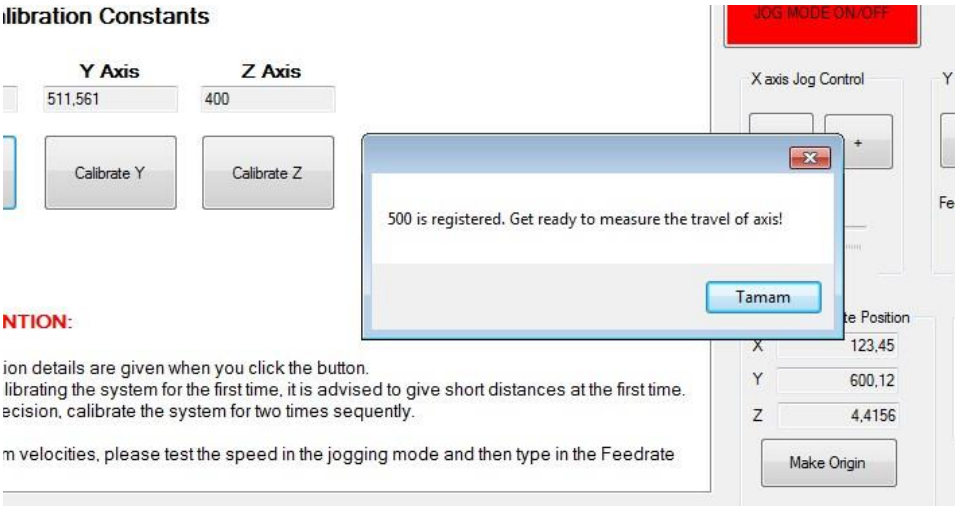


Şekil 6.3 Bilgisayar yazılımı ilk mikrodenetleyiciye bağlanması.

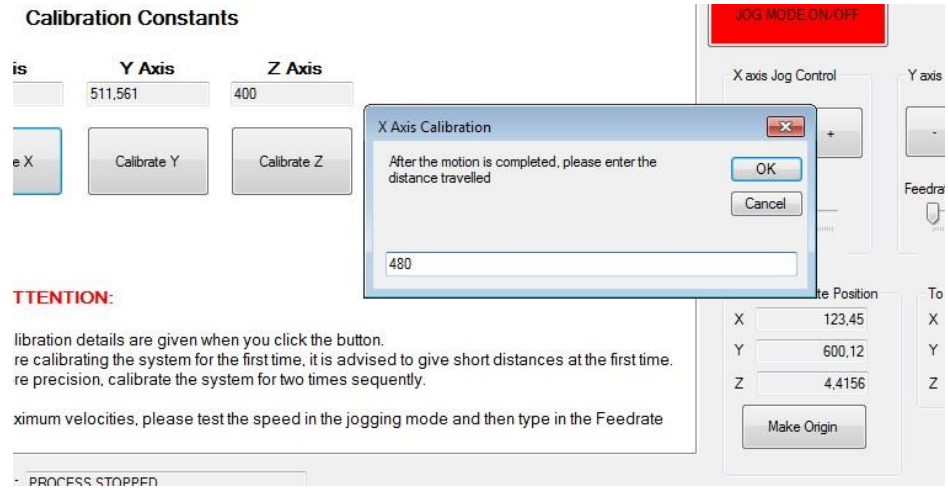
Programda yer alan ölçüleme sekmesinde de özel bir senaryo geliştirilmiştir. Kullanıcı, ölçüleme katsayılarını belirlemek için ölçüleme moduna girer ve katsayının hesaplanabilmesi için bir mesafe girer. Eksen girilen mesafe kadar hareket etmeye çalışır. Kullanıcı, makine hareketi tamamlandıktan sonra hareket edilen mesafeyi girer ve bilgisayar yazılımı da kabaca bir oranlama yaparak katsayıyı belirler. Şekil 6.13, 6.14, 6.15 ve 6.16'da bu moda ait aşamalar verilmiştir.



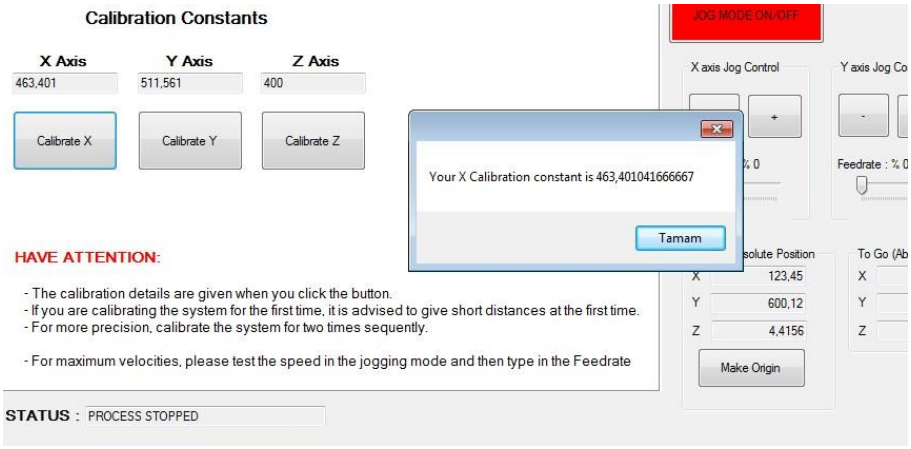
Şekil 6.4 Bilgisayar yazılımı 1. ölçümleme kademesi.



Şekil 6.54 Bilgisayar yazılımı 2. ölçümleme kademesi.



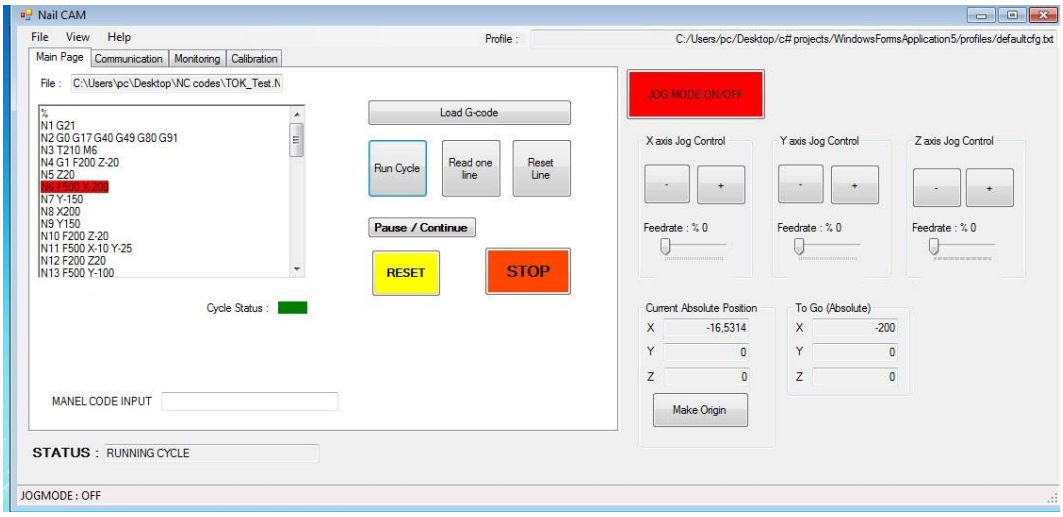
Şekil 6.6 Bilgisayar yazılımı 3. ölçümleme kademesi.



Şekil 6.7 Bilgisayar yazılımı 4. ölçümleme kademesi.

Makinenin profili olarak adlandırılan katsayılar ve değişkenlerin yer aldığı grup, programın başında da otomatik olarak yüklenebileceği gibi, yapılacak seçim ile de yükleme gerçekleştirilebilir. Bir profil dosyasında ölçüleme katsayıları, anlık pozisyon ve maksimum hız değerleri yer almaktadır.

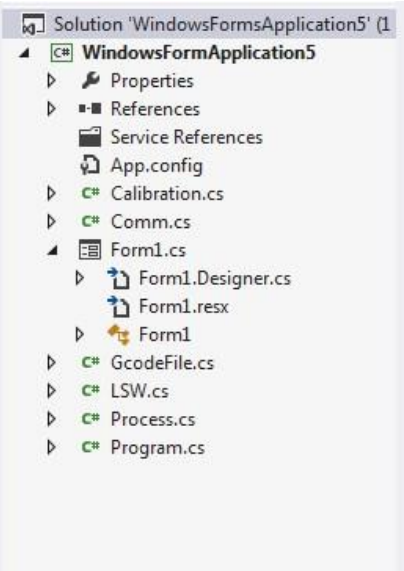
İşleme dosyasının yüklenmesinin ardından uygun çalışma koşulları sağlanmalı ve gerekirse orijin pozisyonu tekrar ayarlanmalıdır. “Run Cycle” veya “Read one line” butonlarından birisi kullanılarak işleme dosyasından komutlar işlenmeye başlar. Acil durumda “STOP” butonuna basmak işlemi sonlandırır. Buna ait görsel Şekil 6.17’de verilmiştir.



Şekil 6.8 Bilgisayar yazılımı işleme ekranı.

Adımlama modunda formun sağ tarafında yer alan panel kullanılır. Adımlama modu aktive edildikten sonra, iz çubuğu kullanılarak hareket hızı belirlenir. İstenilen yön butonuna basılarak belirtilen hızda hareket gerçekleştirilir. Fare butonu bıraktığı anda hareket sonlandırılır.

Visual Studio 2015 programında geliştirilen yazılıma ait çözüm unsur ağacı şekildeki gibidir. Programın isim başlığı “WindowsFormApplication5”’tir. “Main” fonksiyonu “Program.cs”, oluşturulan form uygulaması “Form1.cs”, oluşturulan yardımcı sınıflar ise “Calibration.cs”, “Comm.cs”, “GcodeFile.cs”, “LSW.cs” ve “Process.cs” adı altında oluşturulmuştur. Proje unsur ağacı Şekil 6.18’de verilmiştir.



Şekil 6.9 Bilgisayar yazılımı proje unsur ağacı.

Form dosyasında yazılan eylem tetiklemeli veya değil, bütün fonksiyonları ve kodları görevlerine göre bölgelere ayırmak mümkündür. Bunlar:

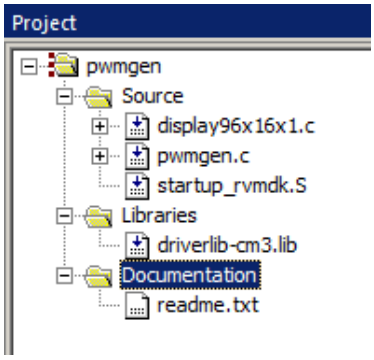
- Yapılar (structures)
- Kullanıcı tanımlı bazı evrensel değişkenler
- Sınıf (class) ve yapıların başlatılması
- Formun başlatılması
- Satır okuma-değerlendirme fonksiyon ve butonları
- Menü kuşağı butonları
- Haberleşme butonları
- Haberleşme ayarlama fonksiyonları
- Bazı pozisyonlama fonksiyonları
- Mikro denetleyici gönderme-alma ve komut değerlendirme fonksiyonları
- Elle komut girme fonksiyonu
- Profil okuma-yazma-gönderme
- Adımlama panel butonları ve kaydırıcıları
- Ölçüleme butonları
- Zamanlayıcı kontrolleri
- İşleme dosyası yükleme

- Maksimum hız
- İşleme döngüsü ve başla-durdur butonları

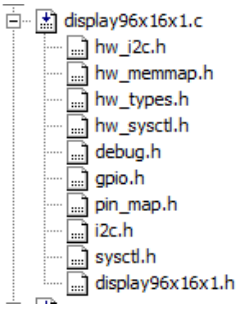
Mikrodenetleyici ile bilgisayar arasında basit bir haberleşme protokolü geliştirilmiştir. Buna göre, yine G komut temeline göre çalışan parçalar, gönderdikleri komut satırının ardından "OK" komutu ekler (Örneğin; "G01 X100 F300OK"). Bu komutu algılayan karşı taraf ise, gönderilen komut satırının sonuna geldiğini algılar ve veriyi değerlendirmeye alır.

6.3.2 Mikrodenetleyici programı algoritması

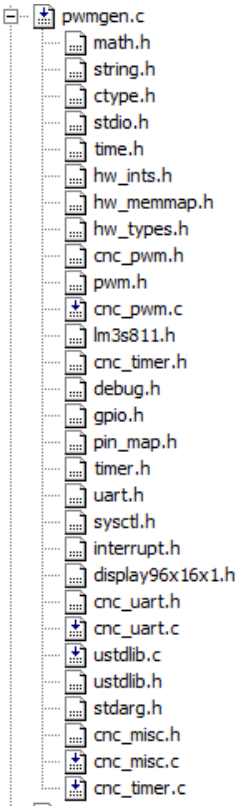
Mikro denetleyicideki yazılımı Keil programı üzerinde bir proje olarak yazılmıştır. Proje dosyasının ismi "pwmgen"dir. Proje dosyasında yer alan dosyalar ve kütüphaneler şekilde verilmiştir. Bunlardan bazıları kullanıcı tanımlı olup bazıları ise firmanın sunduğu hazır dosyalardır. Ana kod parçasının bulunduğu dosya ("main" fonksiyonunun bulunduğu dosya) "pwmgen.c" dosyasıdır. Proje dosya sistemine ait görseller Şekil 6.19, 6.20 ve 6.21'de verilmiştir.



Şekil 6.10 Mikrodenetleyici proje unsur ağacı.



Şekil 6.20 Mikrodenetleyici üzerindeki ekrana ait dosyalar.



Şekil 6.11 "pwmgen.c" dosyasının kullandığı dosya ve kütüphaneler.

Geliştirilen yazılımın hareket mekanizması zamanlayıcı kesmesi üzerine kurulmuştur. Bir komut için hareket süresi hesaplandıktan sonra zamanlayıcı süresi buna uygun olarak kurulur ve başlatılır. Zamanlayıcı kesmesi gerçekleştikten sonra hareket tamamlanmış olur.

Haberleşme mekanizması da aynı şekilde seri haberleşme kesmesi üzerinden gerçekleştirilmektedir. Haberleşme kanalından alınan veri, sistemi haberleşme kesmesine sokmaktadır ve gelen veri işlenerek gerekli görevler gerçekleştirilmektedir.

Sınır anahtarlar dış kesmeler olarak atanmıştır ve sınır anahtarların durumları değiştiğinde aktif hale geçmektedir.

"pwmgen.c" ana çalışma dosyasında yer alan fonksiyonlar ve görevleri aşağıdaki gibidir.

- **void sendPos(double x_cur, double y_cur, double z_cur):** Pozisyon bilgisini metin olarak bilgisaya gönderir
- **double strtodec(char *buf):** Metin tipindeki sayıyı ondalıklı sayı tipine dönüştürür.

- **void rtrim(char *str):** Metnin sağ boşluklarını kırpar.
- **void ltrim(char *str):** Metnin sol boşluklarını kırpar.
- **void trim(char *str):** Metnin sağ ve sol boşluklarını kırpar.
- **void Action (void):** Global değişkenleri değerlendirir ve talimatlara göre uygun hareketi yaptırır.
- **void EvalWords(char *str):** Kelimeyi değerlendirerek uygun değişkene atama yapar.
- **void EvalUart(char *str):** Seri haberleşme yoluyla gelen metni öncelikle kelimelere ayırarak değerlendirir (EvalWords), sonrasında uygun hareket talimatını verir (Action).
- **void UARTIntHandler(void):** Seri haberleşme kesmesidir, seri haberleşme yoluyla veri geldiğinde aktif olur, bilgisayardan gelen metni toplar ve değerlendirir (EvalUart).
- **void GPIOCIntHandler(void):** Kart üzerinde yer alan kullanıcı butonuna ait kesmedir, herhangi bir durumda motorları durdurur.
- **void GPIOAIntHandler(void):** X ve Y ekseni limit anahtarların bağlı olduğu pinlerde dış kesme meydana geldiğinde çalışır.
- **void GPIOBIntHandler(void):** Z ekseni limit anahtarlarının bağlı olduğu pinlerde dış kesme meydana geldiğinde çalışır.
- **void Timer0IntHandler(void):** Zamanlayıcı 0'ın kesmesidir, zamanlayıcı süresi dolduğunda aktif olur, hareket tamamlandığında yapılacak hareket ve protokolleri uygular.
- **int main(void):** Ana fonksiyondur, mikrodenetleyici çalıştırıldığında ilk çalışan fonksiyondur.

"main" fonksiyonuna ait algoritma sözde kod (pseudo code) şeklinde aşağıda verilmiştir.

Kesmelere izin ver
Sistem saatini ayarla
Ekranı aktif hale getir
Seri haberleşme ayarlarını yap
PWM ayarlarını yap
Motor yönleri ayarlama pinlerinin ayarlarını yap
Limit anahtar bağlantılarının ayarlarını yap
Kullanıcı butonunun ayarlarını yap

Zamanlayıcı ayarlarını yap
PWM kanallarını kapalı duruma getir
Sonsuz döngüye gir
Bitir

Kullanıcı tanımlı "cnc_pwm.c" dosyasında yer alan fonksiyonlar ve görevleri aşağıdaki gibidir.

- **void CNC_PWM_init(void):** PWM ayarlarını yapar ve başlatır.
- **void CNC_PWM_Enable(int pwm_channel):** İstenilen PWM kanalını aktive eder.
- **void CNC_PWM_Disable(int pwm_channel):** İstenilen PWM kanalını devre dışı bırakır.
- **void CNC_PWM_SetFreq(int pwm_channel, double freq):** İstenilen PWM kanalının PWM sinyalinin istenilen frekansa ayarlanmasını sağlar. Sistemin temel hareket fonksiyonlarından biridir. Örneğin, G01 ile X ekseninde F500 ile 600 ilerleme istenmektedir. Bu doğrultuda, öncelikle hareket süresi hesaplanır. Süre basit bir yol/hız denklemiyle bulunur. Hareket hızı eksene bağlı ölçümleme katsayısı ile çarpılarak fonksiyona gidilir ve PWM frekansı ayarlanır. Hareket hesaplanan süre boyunca çıkış olarak uygulanır.
- **void CNC_go_init(double x_val, double y_val, double z_val, double go_time, double x_calib, double y_calib, double z_calib):** Gerçekleştirilecek hareket öncesi zamanlayıcının, PWM frekansının ve hareket yönünün tespiti ve ayarlanması sağlanır.
- **void CNC_PWM_Dir(double x_go, double y_go, double z_go):** Hareket yönünü ayarlar.
- **void CNC_PWM_Dir_init(void):** Hareket yönlerini belirleyen pinleri ayarlar ve başlatır.

Kullanıcı tanımlı "cnc_uart.c" dosyasında yer alan fonksiyonlar ve görevleri aşağıdaki gibidir.

- **void CNC_UART_init(void):** UART0 kanalını ayarlar ve başlatır.
- **void UARTSend(const unsigned char *pucBuffer, unsigned long ulCount)**: Verilen metni UART0 kanalıyla yollar.

Kullanıcı tanımlı "cnc_timer.c" dosyasında yer alan fonksiyonlar ve görevleri aşağıdaki gibidir.

- **void CNC_Timer_init(void):** Zamanlayıcı 0 ile zamanlayıcı 2 ayarlarını yapar.
- **void CNC_Timer_Enable(int timer)**: İstenilen zamanlayıcıyı başlatır.
- **void CNC_Timer_Disable(int timer)**: İstenilen zamanlayıcıyı devre dışı bırakır.
- **void CNC_Timer_start(double sure, int timer)**: İstenilen zamanlayıcıyı verilen zaman değeri ile başlatır.
- **void CNC_go(double sure, double x_val, double y_val, double z_val)**: Zamanlayıcı 0'ı başlatır ve PWM çıkışlarını aktive eder.

Kullanıcı tanımlı "cnc_misc.c" dosyasında yer alan fonksiyonlar ve görevleri aşağıdaki gibidir.

- **void UserButton_init(void (*pfnIntHandler)(void)):** Kullanıcı butonunu ve kesmesini ayarlar.
- **void LSW_init(void (*pfnIntHandler1)(void), void (*pfnIntHandler2)(void)):** Limit anahtarların başlatılmasını sağlar.

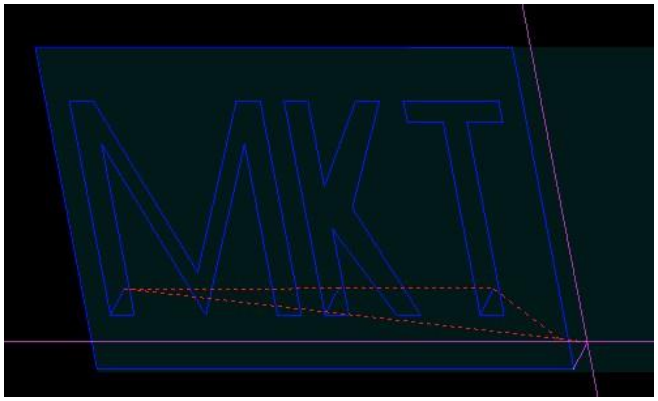
7. ÖRNEK ÇALIŞMA

Yapılan örnek çalışmada, CNC üzerine bağlanan bir kurşun kalem ile yazı yazdırılmıştır. Program işleme kodu aşağıdaki gibidir.

```
%
O0001 (Program no)
N1 G21                    (milimetre birimi)
N2 G0 G17 G40 G49 G91     (hızlı hareket, XY düzlemleri, uç işlevci kompanzasyon kapalı,
                          çakı uzunluğu kompanzasyon kapalı, artan pozisyonlama)
N3 T210 M6                (210 nolu çakı, M6 özellikli)
N4 G1 F200 Z20            (200 mm/dk hızla Z eksenini 20 mm yukarı çıkar)
N5 Z-20                   (200 mm/dk hızla Z eksenini 20 mm aşağı indir)
N6 F500 X-200             (500 mm/dk hızla X eksenini 200 mm negatif yönde götür)
N7 Y150                   (500 mm/dk hızla Y eksenini 150 mm pozitif yönde götür)
N8 X200                   (500 mm/dk hızla X eksenini 200 mm pozitif yönde götür)
N9 Y-150
N10 F200 Z20
N11 G0 X-190 Y25
N12 G1 F200 Z-20
N13 F500 Y100
N14 X10
N15 X30 Y-80
N16 X30 Y80
N17 X10
N18 Y-100
N19 X-10
N20 Y80
N21 X-30 Y-80
N22 X-30 Y80
N23 Y-80
N24 X-10
N24 F200 Z20
N25 G0 X90
N26 G1 F200 Z-20
N27 F500 Y100
N28 X10
N29 Y-40
N30 X20 Y40
N31 X10
N32 X-20 Y-50
N33 X20 Y-50
N34 X-10
N35 X-20 Y40
N36 Y-40
N37 X-10
N38 F200 Z20
N39 G0 X65
N40 G1 F200 Z-20
N41 F500 Y90
N42 X-15
N43 Y10
N44 X40
N45 Y-10
N46 X-15
N47 Y-90
```

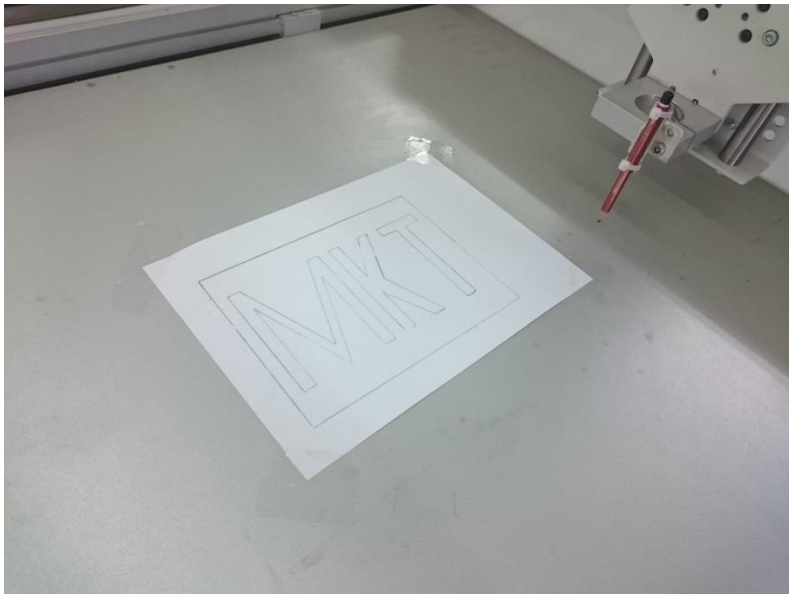
```
N48 X-10
N49 F200 Z20
N50 G0 X25 Y-25
N51 M30                    (Programı bitir)
%
```

Mach3 programı yardımıyla elde edilen, program işlem dosyasının örnek işleme simülasyonundan ekran çıktısı Şekil 7.1'deki gibidir. Buna göre başlangıç noktasında bulunan uç işlevci, mavi renkte gösterilen çizgiler üzerinden geçecektir.



Şekil 7.1 Mach3 simulasyon çıktısı.

İşleme dosyasının sistem üzerinde uygulamasının ardından elde edilen çıktı Şekil 7.2'deki gibidir.



Şekil 7.2 Örnek kod deney çıktısı.

8. SONUÇ VE TARTIŞMA

Bu çalışma kapsamında ileride geliştirmeye açık, mühendis adaylarının eğitimine katkıda bulunarak daha donanımlı olarak mezun etmeye yönelik bir açık kaynak CNC makinası yazılımı oluşturulmaya çalışılmıştır. Bu konuda da başarıya ulaşıldığı düşünülmektedir. Örnek çalışmada görüldüğü gibi G komutları doğrultusunda makine kontrol edilebilmektedir.

Eğitim konusunda bu şekilde bir çalışma bulunmadığından, bu alanda kendini geliştirmek isteyen öğrenciler için açık kaynak olarak geliştirilen yazılım ilk sayılabilecek türdendir.

Bu çalışmayla görülmüştür ki, çalışmanın daha fazla ve daha hızlı ilerleyebilmesi için ciddi anlamda disiplinler arası çalışmanın gerçekleştiği bir ekip çalışmasına ihtiyaç duyulmaktadır.

Çalışmada düşük maliyeti dolayısıyla aktüatör olarak adım motorlar kullanılmıştır. Daha verimli kullanım için gelişmiş motor sürücü kullanımı veya servo motor kullanımının daha uygun olacağı düşünülmektedir. Kullanılan adım motorların çok hassas işler için uygun olmadığı ve performans konusunda servo motorların daha iyi olduklarından Bölüm 3.4.4'te bahsedilmişti.

Yapılan çalışmadaki ihtiyaçlardan birisi de G02 ve G03 komutlarının algoritmalarının yer almamış olmasıdır. Çalışmadaki bir sonraki aşamanın bu yönde olması planlanmaktadır.

Çalışma daha da geliştirilerek bu alanda ülkemizde henüz bulunmayan yerli bir CNC yazılımı üretilmiş olacaktır.

KAYNAKLAR DİZİNİ

Acarnley, P., 2002, Stepping Motors: A Guide to Theory and Practice, Institution of Engineering and Technology, Forth Edition, 155p.

Albert, A., 2008, Understanding CNC Routers, FPInnovations, First Edition, 116p.

Anderberg, S., 2012, Methods for improving performance of process planning for CNC machining - An approach based on surveys and analytical models, Thesis For The Degree Of Doctor Of Philosophy, Department of Materials and Manufacturing Technology, Chalmers Universıty Of Technology, Göteborg, Sweden, 172p.

Ay, G. M., 2004, High Precision CNC Motor Control, Master of Science Thesis in Mechanical Engineering, Middle East Technical University, Ankara, 176s.

Balic, J., Kovacic, M., N. and Vaupotic, B., 2006, Intelligent programming of CNC turning operations using genetic algorithm, Journal of Intelligent Manufacturing, 17:331-340pp.

Engin, M., 2013, Mikroişlemciler, Mekatronik Mühendisliği Mikrodenetleyiciler dersi ders notları , 134s.

FADAL, 2003, User Manual, Flint Machine Tools Inc., 531p.

Fjelde, B. and Stamsø, A., 2011, New Approach to produce special purpose visual aid Glasses, Master's theses in Mechatronics, University of Agder, 76p.

Gerritsen, A., 1999, CISC vs RISC, Dr. Farid Farahmand ES310 Microcontrollers Lecture notes, Sonama University, https://www.sonoma.edu/users/f/farahman/sonoma/courses/es310/resources/module_5_-_cisc_vs_risc.doc, 15p.

Haas Automation, 2006, Programming Workbook, Haas Automation Inc., 137p.

Hashim, N. S. B., 2002, Design of Mini CNC Machine, Bachelor Thesis of Engineering in Manufacturing, Universiti Malaysia Pahang, Malaysia, 49p.

IFM Electronic, 2005, Catalogue, IFM Electronic article no. 7511084, 55p.

Jones, A. and Freeman, A., 2010, Visual C# 2010 Recipes: A Problem-Solution Approach, Apress, First Edition, 1016p.

KAYNAKLAR DİZİNİ (devam)

Kao, Y. C., Cheng, H. Y. and Chen, Y., 2006, Development of a Virtual Controller Integrating Virtual and Physical CNC, Material Science Forum, Vols 505-507, 631-636pp.

Krar, S. and Gill, A., 1999, Computer Numerical Control Programming Basics, Industrical Press, 42p.

Lipovski, J., 1999, Introduction to Microcontrollers: Architecture, Programming, and Interfacing of the Motorola 68Hc12, Academic Press, First Edition, 394p.

Ma, X. B., Han, Z. Y., Wang, Y. Z. and Fu, H. Y., 2007, Development of a PC-based open architecture software-CNC system, Chinese Journal of Aeronautics, 20(3), 272-281pp.

McMonnies, A., 2004, Object Oriented Programming in VB.Net, Addison Wesley, First Edition, 696p.

Mechanical Guru, 2011, "C.N.C (Computer Numeric Controller), Mechanical Guru'nun blog sayfası, http://sumitshrivastva.blogspot.com.tr/2011/11/cnc-computer-numeric-control.html, (Erişim tarihi: 20 Nisan 2015)

Minhat, M., Vyatkin, V., Xu, X., Wong, S. and Al-Bayaa, Z., 2009, A novel open CNC architecture based on STEP-NC data model and IEC 61499 function blocks, Robotics and Computer-Integrated Manufacturing, 25(3), 560-569pp.

Morales-Velazquez, L., de Jesus Romero-Troncoso, R., Osornio-Rios, R. A., Herrera-Ruiz, G. and Cabal-Yepez, E., 2010, Open-architecture system based on a reconfigurable hardware–software multi-agent platform for CNC machines, Journal of Systems Architecture, 56(9), 407-418pp.

Msdn training, 2001, Introduction to C# Programming for the Microsoft .NET Platform (Prerelease) Workbook, Microsoft Corp., 812p.

Pepperl+Fuchs, 2007, Rotary Encoder Catalog North America Edition, Pepperl+Fuchs Inc., 112p.

Schneider Electric, 2007, Limit Swithes, Schneider Electric Motion, 246p.

Schneider Electric, 2012, Stepper Motors datasheet, Schneider Electric Motion, 6p.

Siemens, 2004, Servo Motors for Simovert Masterdrivers, Siemens AG, 224p.

KAYNAKLAR DİZİNİ (devam)

Silicon Labs, 2008, AN155: Stepper Motor Reference Design, Silicon Laboratories, Rev. 1.1, 36p.

Smartech, 2015, "CNC machine frame/body; HV1060", Smartech Machinery & Equipment, http://zzsmartech.en.alibaba.com/product/521058738-213158018/CNC_machine_frame_body_HV1060.html (Erişim tarihi: 15 Nisan 2015)

Suh, S.-H., Kang, S.-K., Chungi D.-H. and Straud, I., 2008, Theory and Design of CNC Systems, Springer-Verlag London, 456p.

Texas Instruments, 2010, Stellaris LM3S811 Evaluation Board User's Manuel, Texas Instruments Inc., 35p.

Texas Instruments, 2012, Stellaris Peripheral Driver Library, Texas Instruments Inc., 516p.

Texas Instruments, 2014, Stellaris LM3S811 Microcontroller Data Sheet, Texas Instruments Inc., 595p.

Tormach, 2015, "G02 and G03 – Arc at feed rate", Tormach Inc, http://www.tormach.com/g02_g03.html (Erişim tarihi: 15 Nisan 2015)

Traylor, R. L., 2009, A Brief view of Computer Architecture, Roger L. Traylor ECE112 Lecture notes, Oregon State University, http://web.engr.oregonstate.edu/~traylor/ece112/lectures/comp_arch.pdf, 4p.

Uyar, E., Akçura, N., Yavuz, E. ve Candan, M., 2014, 3 Eksenli CNC İşleme Tezgahı Tasarımı ve Kontrolü, TOK 2014, Kocaeli Üniversitesi, Bildiri no:16, 4s.

Valvano, J. W., 2011, Embedded Systems: Real-Time Interfacing to Arm Cortex-M Microcontrollers, CreateSpace Independent Publishing Platform, Second Edition, 600p.

Valvano, J. W., 2012, Embedded Systems: Real-Time Operating Systems for Arm Cortex-M Microcontrollers, CreateSpace Independent Publishing Platform, Second Edition, 448p.

Wikipedia contributors, 2015, ".NET Framework", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/.NET_Framework (Erişim tarihi: 15 Nisan 2015)

KAYNAKLAR DİZİNİ (devam)

Wikipedia contributors, 2015, "C Sharp", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/C_Sharp_(programming_language) (Erişim tarihi: 15 Nisan 2015)

Wikipedia contributors, 2015, "G-code", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/G-code (Erişim tarihi: 15 Nisan 2015)

Wikipedia contributors, 2015, "H Bridge", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/ H_bridge (Erişim tarihi: 15 Nisan 2015)

Wikipedia contributors, 2015, "Microcontroller", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Microcontroller (Erişim tarihi: 15 Nisan 2015)

Wikipedia contributors, 2015, "Microprocessor", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Microprocessor (Erişim tarihi: 15 Nisan 2015)

Wikipedia contributors, 2015, "Stepper motor", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/ Stepper_motor (Erişim tarihi: 15 Nisan 2015)

Yiu, J., 2009, The Definitive Guide to the ARM Cortex-M3, Newnes, Second Edition, 479p.

Yuhan, W., Jun, H. and Ye, L., 2003, Study on a reconfigurable model of an open CNC kernel, Journal of materials processing technology, 138(1), 472-474pp.

Zhang, C., Wang, H. and Wang, J., 2003, An USB-based software CNC system, Journal of materials processing technology, 139(1), 286-290pp.

Zuo, J., Chen, Y. P., Zhou, Z. D., Nee, A. Y. C., Wong, Y. S. and Zhang, Y. F., 2000, Building Open CNC Systems with Software IC Chips Based on Software Reuse, The International Journal of Advanced Manufacturing Technology, Volume 16, Issue 9, 643-648pp.

ÖZGEÇMİŞ

Ad Soyad : Nail Akçura

Doğum tarihi : 03/05/1989

Doğum yeri : İzmir (TR)

Adres : 1833 sokak no:1/6 Karşıyaka/İzmir Turkey

Telefon :(+90) 5542007771

E-mail : nailakcura@gmail.com

Eğitim :

- 2012-… Ege Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Anabilim Dalı
- 2012–2012 İstanbul Teknik Üniversitesi Mekatronik Mühendisliği Bölümü Yüksek Lisans
- 2007–2012 Kocaeli Üniversitesi Mekatronik Mühendisliği Bölümü CGPA: 3.07/4
- 2003–2007 İzmir Karşıyaka Anadolu Lisesi: 4.92/5

İş deneyimi :

- 2015–2015 İzmir Katip Çelebi Üniversitesi Mühendislik-Mimarlık Fakültesi Mekatronik Müh. Bölümü
- 2012–2014 Ege Üniversitesi Fen Bilimleri Enstitüsü Mekatronik A.B.D.
- 2011–2012 İzgen Mühendislik / Kocaeli

Çalışmalar :

- Erol Uyar, Nail Akçura, Ekrem Yavuz, Mücahid Candan, "3 Eksenli CNC İşleme Tezgahı Tasarımı ve Kontrolü", TOK 2014, Kocaeli Üniversitesi, Bildiri no:16, 4s.
- Nail Akçura, Erol Uyar, Mücahid Candan, Ekrem Yavuz, "Çevre Haritalandırma ve Obje Geçme Algoritması Entegreli Bir Mobil Aracın Bulanık Mantık ile İç Mekan Navigasyon Kontrolü", TOK 2014, Kocaeli Üniversitesi, Bildiri no:122, 4s.
- Mücahid Candan, Erol Uyar, Nail Akçura, Ekrem Yavuz, "Remote Controlled Electro-Pneumatical Climbing Robot For Cleaning Of Skyscrapers", International Journal of Computer and Information Technology, Volume 04 – Issue 01, January 2015.
- Erol Uyar, Mücahid Candan, Ekrem Yavuz, Nail Akçura, "Düzlemsel Elektro-Pnömatik Manipülatör Tasarimi Ve Kontrolü", HPKON 2014, İstanbul.
- Erol Uyar, Mücahid Candan, Ekrem Yavuz, Nail Akçura, "Düzlemsel Elektro-Pnömatik Manipülatör Tasarimi Ve Kontrolü", Mühendis ve Makina, Cilt 56 Sayı 662 s:56-62, Mart 2015.

EKLER

Ek 1 Stellaris LM3S811 Mikrodenetleyici Yüksek Seviye Blok Diyagramı

Ek 2 G kodu harf tanımları

Ek 3 Mekanik Tasarım

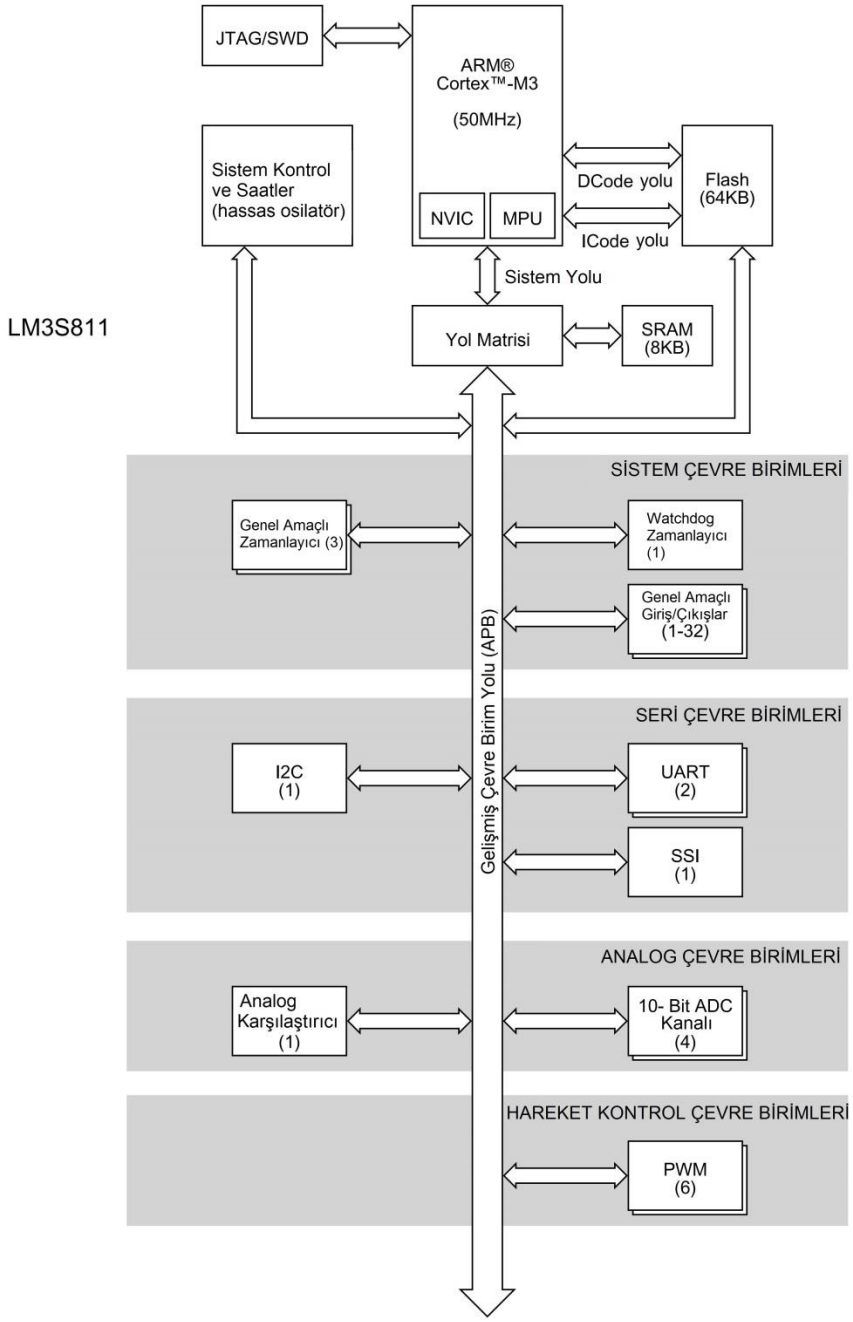
Ek 4 Sistemin detaylı elektrik bağlantı şeması

Ek 5 Mikrodenetleyici programı kodları

Ek 6 Bilgisayar programı kodları

Ek 1 Stellaris LM3S811 Mikrodenetleyici Yüksek Seviye Blok Diyagramı

(Texas Instruments, 2010)

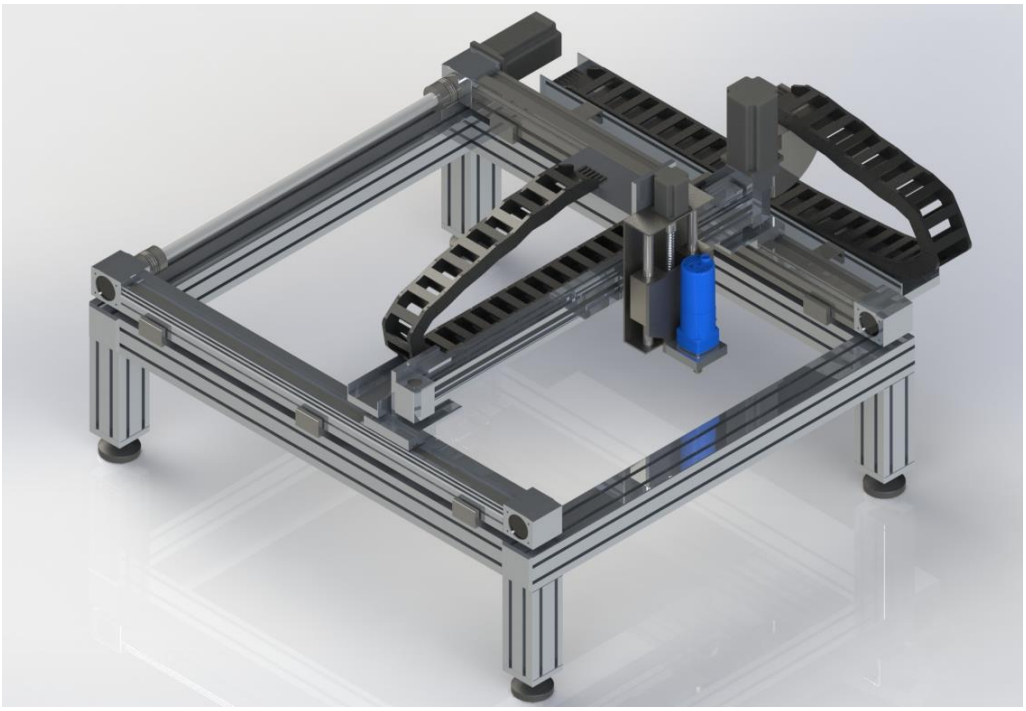


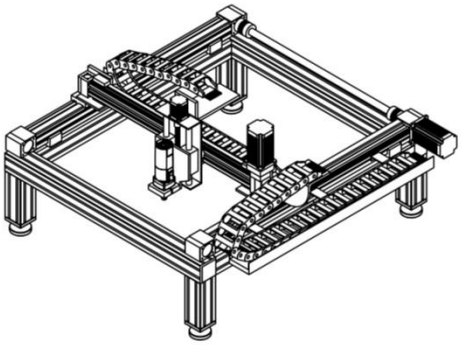
Ek 2 G kodu harf tanımları (Wikipedia contributors, 2015c).

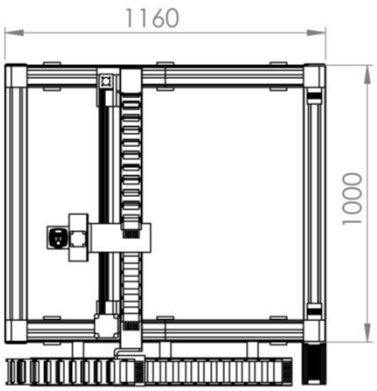
Harf	Tanım
A	A ekseni (X ekseninin dönme ekseni) pozisyonu
B	B ekseni (Y ekseninin dönme ekseni) pozisyonu
C	C ekseni (Z ekseninin dönme ekseni) pozisyonu
D	Kesici kompanzasyonu için kullanılan çap veya mesafe
F	İlerleme hızı
G	Hazırlık komutları için adres
H	Takım uzunluğu mesafesi
I	G02 ve G03 komutları için X ekseninde yay merkezi
J	G02 ve G03 komutları için Y ekseninde yay merkezi
K	G02 ve G03 komutları için Z ekseninde yay merkezi
M	Çeşitli fonksiyonlar
N	Satır numarası
O	Program ismi
R	Yay çapı
S	Uç işlevci dönüş hızı
T	Uç seçimi
U	X eksenine göre artımlı eksen
V	Y eksenine göre artımlı eksen
W	Z eksenine göre artımlı eksen
X	X ekseni pozisyonu
Y	Y ekseni pozisyonu
Z	Z ekseni pozisyonu

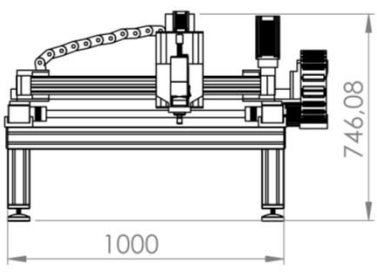
Ek 3 Mekanik Tasarım

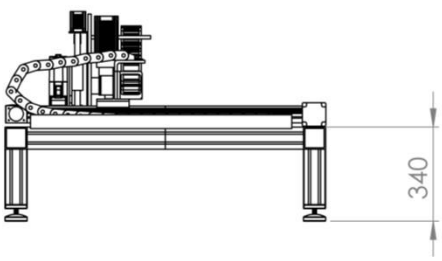
CNC makinasının modellenmesi ve makine parçalarının boyutlandırılması aşağıdaki gibidir.



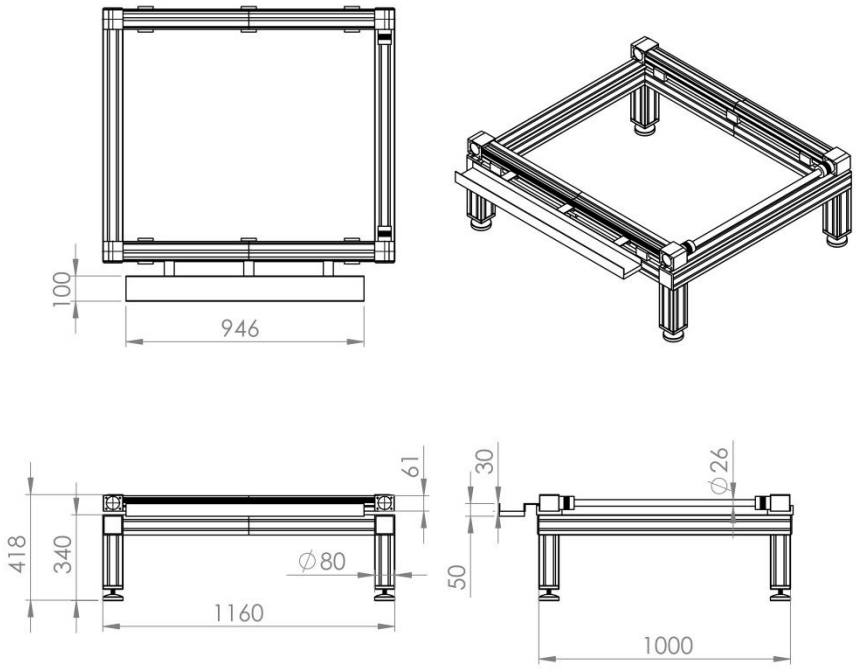




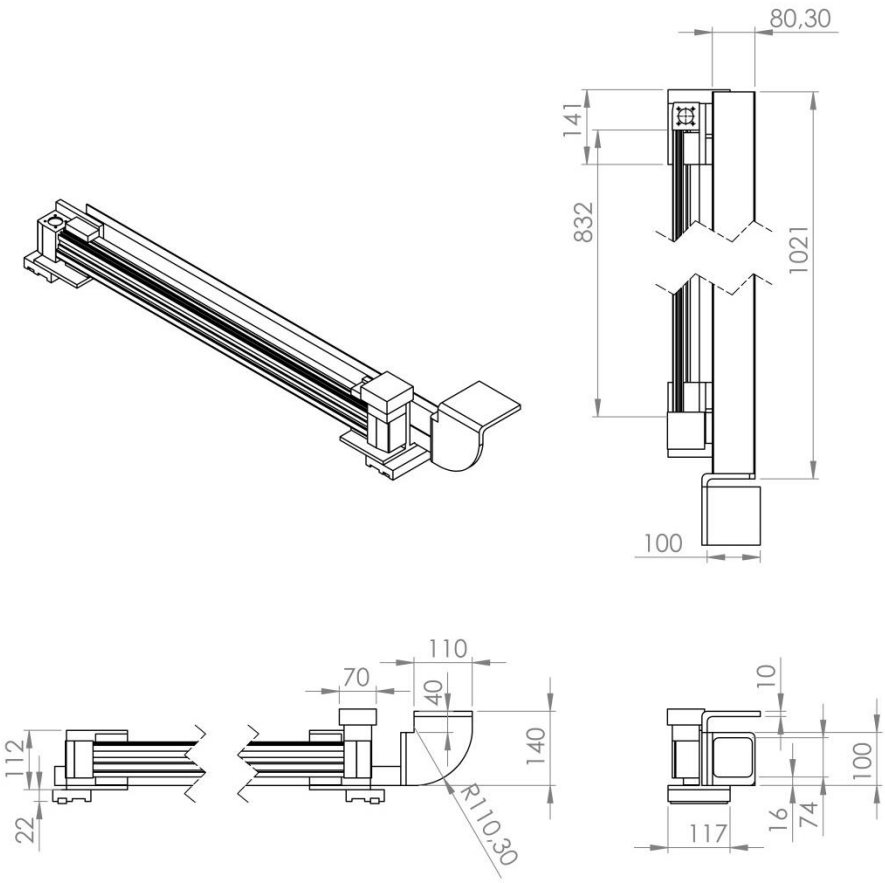




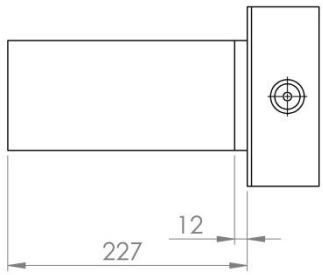
Makinanın alt gövdesi aşağıdaki gibidir.

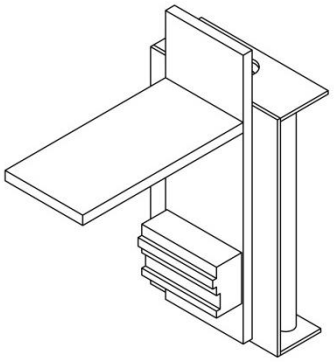


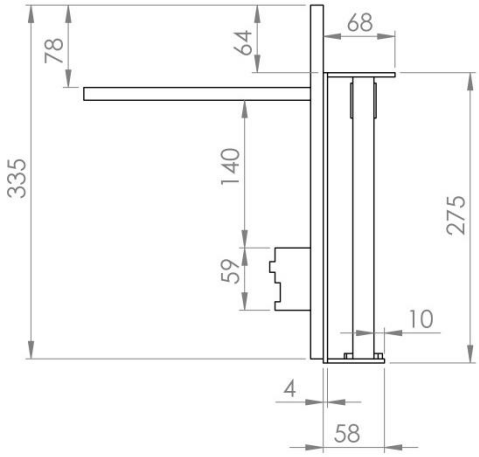
Makinanın hareketli arabası aşağıdaki gibidir.

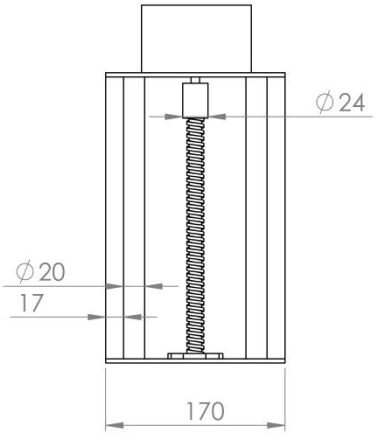


Z ekseni lineer mil mekanizması aşağıdaki gibidir.

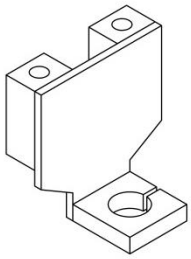


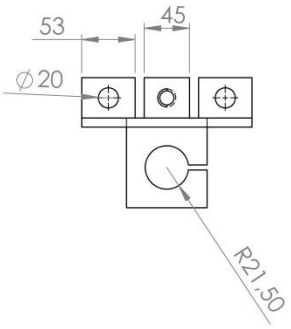


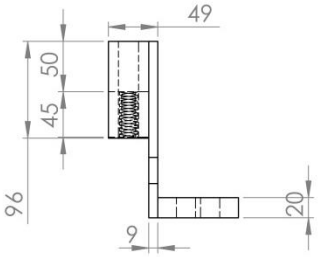


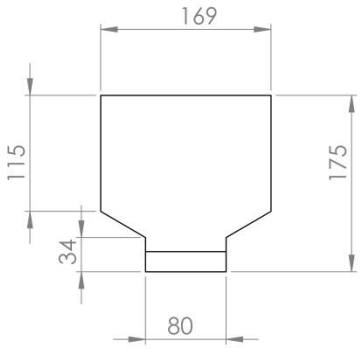


Z ekseni lineer mil mekanizması ve uç işlevci bağlantı gövdesi aşağıdaki gibidir.

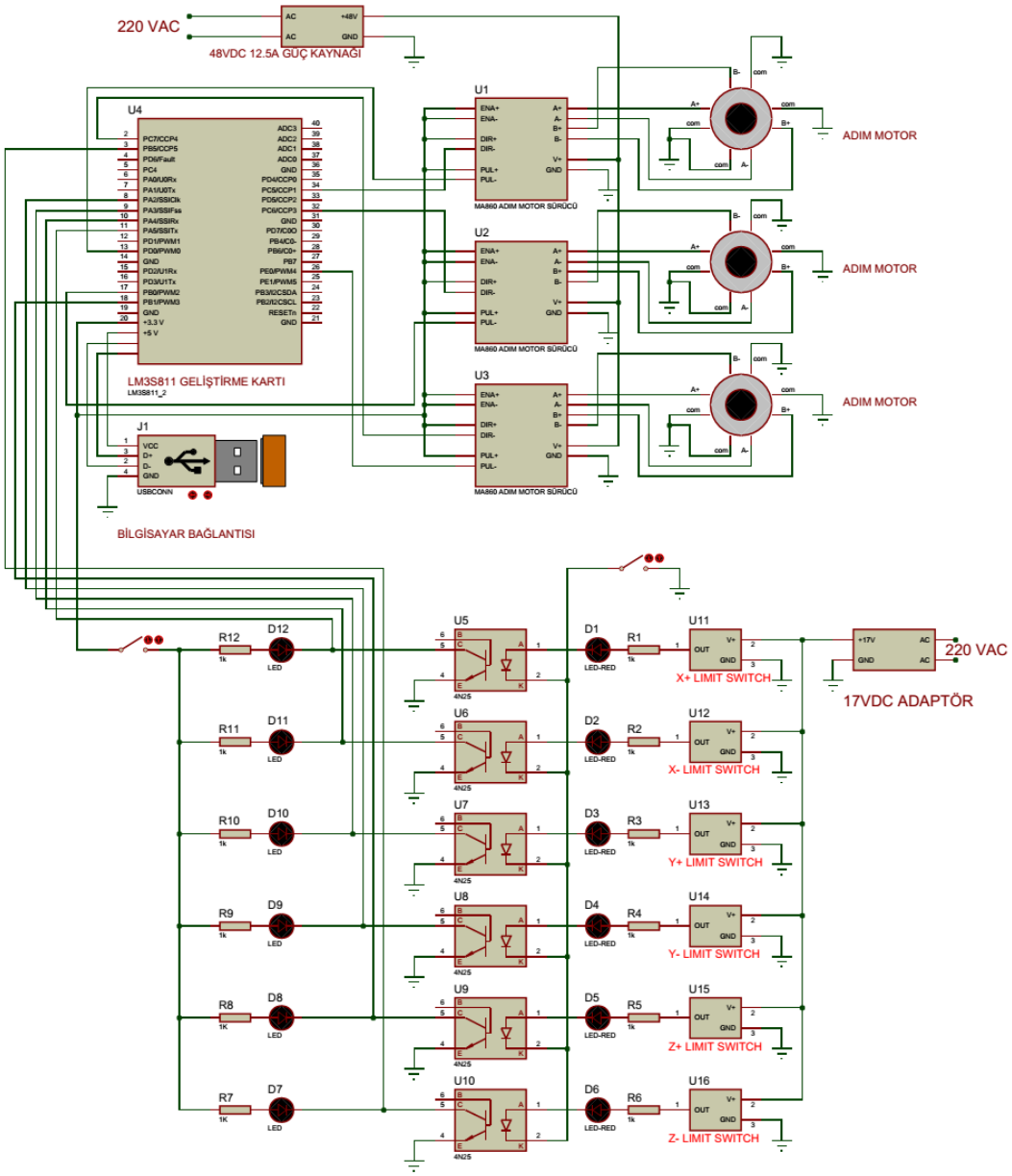








Ek 4 Sistemin detaylı elektrik bağlantı şeması



Ek 5 Mikrodenetleyici programı kodları

```
//*****************************************************************************
//
// pwmgen.c – Bu program CNC makinede ara yüz programı olarak kullanılmaktadır.
//
//*****************************************************************************
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <stdio.h>
#include <time.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/cnc_pwm.h"
#include "driverlib/cnc_pwm.c"
#include "driverlib/cnc_uart.h"
#include "driverlib/cnc_uart.c"
#include "driverlib/cnc_misc.h"
#include "driverlib/cnc_misc.c"
#include "driverlib/cnc_timer.h"
#include "driverlib/cnc_timer.c"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/pwm.h"
#include "driverlib/uart.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "drivers/display96x16x1.h"

int temp_deneme = 5;
int gmode = -1, hmode = -1, mmode = -1;
double spindle_speed = 0.0;
double x_value = 0.0, y_value = 0.0, z_value = 0.0;
double r_value = 0.0, s_value = 0.0, t_value = 0.0;
double f_value = 10.0, h_value = 1.0;
double pos_vector = 0.0, temp_vector = 0.0;
double go_time = 0.0;
double unit_conv = 1.0;
int abs_mode = 1;
double x_calib = 444.865, y_calib = 511.561, z_calib = 400.0;
double x_maxfeed = 50000,y_maxfeed = 50000,z_maxfeed = 30000;
double x_current = 0.0, y_current = 0.0, z_current = 0.0;
int x_flag = 0, y_flag = 0, z_flag = 0;
double temp_d = 0.0;
double d1 = 0.0;
unsigned long g_ulMode = 0;
unsigned long flag = 0;
unsigned long g_ulFlags;
char UART_temp[100], temp_sure[10], wasd[10];
int UART_i = 0;
int send_message = 0;
long lsw_pinstate= 0;
double joggingtime, joggingtime1;
double jog_x_speed, jog_y_speed, jog_z_speed;
unsigned long timer1_val, timer1_val1, timer1_val2, timer1_val3;
```

```
double timer1_speed;
volatile unsigned long ulLoop;
double tmr1_val1, tmr1_val2, tmr1_val3;
double stop_tmr1_val1, stop_tmr1_val2, stop_tmr1_val3;
int gstop;

// Sürücü kütüphanelerinde hata halinde çağırılan hata rutini
#ifdef DEBUG
void
__error__(char *pcFilename, unsigned long ulLine)
{
}
#endif

// Pozisyon bilgisini metin olarak bilgisayara gönderir
void sendPos(double x_cur, double y_cur, double z_cur)
{
        char str_pos2[100];
        sprintf(str_pos2,"M102 X%.3f Y%.3f Z%.3f\n", x_cur, y_cur, z_cur);
        UARTSend((unsigned char *)str_pos2, strlen(str_pos2));
}

// Limit anahtar durum bilgisini gönderir
void sendLSW (long lsw)
{
        char str_lsw[100];
        usprintf(str_lsw,"M105 X%d\n", lsw);
        UARTSend((unsigned char *)str_lsw, strlen(str_lsw));
}

// Metin tipindeki sayıyı “double” tipine dönüştürür
double strtodec(char *buf)
{
   double total=0;
   double integer=0;
   double fraction=0;
   int decflag=0;
   int L=0;
   int dec_ind=0;
   char temp;
   int i=0;
        L = strlen(buf);

   for (i=0;i<L;i++)
   {
    if (buf[i] == '.' || buf[i] == ',')
    {
      decflag = 1;
      dec_ind = i;
    }
   }
   temp = dec_ind+48;

   if (decflag == 1)
   {
    for (i=0;i<dec_ind;i++)
    {
      temp = buf[i]-48;
      integer = integer+(temp)*pow(10,(dec_ind-i)-1);
    }
```

```
      for (i=1;i<L-dec_ind;i++)
      {
        temp = buf[i+dec_ind]-48;
        fraction = fraction+(temp)*pow(10,(-1*i));
      }
    }
    else
      for (i=0;i<strlen(buf);i++)
      {
        temp = buf[i]-48;
        integer = integer+(temp)*pow(10,L-i-1);
      }

    total = integer+fraction;
    return total;

}

// Metnin sağ boşluklarını kırpar
void rtrim(char *str)
{
  size_t n;
  n = strlen(str);
  while (n > 0 && isspace((unsigned char)str[n - 1])) {
    n--;
  }
  str[n] = '\0';
}

// Metnin sol boşluklarını kırpar
void ltrim(char *str)
{
  size_t n;
  n = 0;
  while (str[n] != '\0' && isspace((unsigned char)str[n])) {
    n++;
  }
  memmove(str, str + n, strlen(str) - n + 1);
}

// Metnin sağ ve sol boşluklarını kırpar
void trim(char *str)
{
  rtrim(str);
  ltrim(str);
}

// Global değişkenleri değerlendirir ve talimatlara göre hareketi yaptırır
void Action (void)
{
 send_message = 1;
 UART_i = 0;
 switch(gmode)
    {
      case -1:  break;
      case  0:  break;
      case  1: send_message = 0;
                 if( abs_mode == 1)       // mutlak mod
                {
                  temp_vector = 0.0;
```

```
                if (x_flag == 1)  temp_vector += pow(x_value - x_current,2);
                if (y_flag == 1)  temp_vector += pow(y_value - y_current,2);
                if (z_flag == 1)  temp_vector += pow(z_value - z_current,2);
                pos_vector = unit_conv * sqrt(temp_vector);       //  gidilecek toplam mesafe
                go_time = 60 * pos_vector / f_value;      //  mm/dk - mm/sn dönüşümü
                Display96x16x1StringDraw("              ",1, 1);
               usprintf(temp_sure,"%d", (int)(x_value));
               Display96x16x1StringDraw(temp_sure,20, 1);
               usprintf(temp_sure,"%d", (int)(y_value));
               Display96x16x1StringDraw(temp_sure,40, 1);
               usprintf(temp_sure,"%d", (int)(z_value));
               Display96x16x1StringDraw(temp_sure,60, 1);
               usprintf(temp_sure,"%d", (int)(pos_vector));
               Display96x16x1StringDraw(temp_sure,80, 1);
              usprintf(temp_sure,"%d", (int)(go_time));
             Display96x16x1StringDraw(temp_sure,1, 1);
              CNC_go_init(x_value - x_current, y_value - y_current, z_value - z_current, go_time,
x_calib, y_calib, z_calib);
              CNC_go(go_time, x_value - x_current, y_value - y_current, z_value - z_current);
             stop_tmr1_val1 = ((double)TimerValueGet(TIMER0_BASE,
TIMER_A))/(((double)SysCtlClockGet())*100);
              }
              else                // artan mod
              {
               pos_vector = unit_conv * sqrt( pow(x_value,2) + pow(y_value,2) + pow(z_value,2) );
               go_time = pos_vector / (f_value * 0.016667);
              CNC_go_init(x_value, y_value, z_value, go_time, x_calib, y_calib, z_calib);
              CNC_go(go_time, x_value, y_value, z_value);
             }
             break;


          case  1:  send_message = 0;
                    if( abs_mode == 1)          // mutlak mod
                    {
                              temp_vector = 0.0;
                              if (x_flag == 1)  temp_vector += pow(x_value - x_current,2);
                              if (y_flag == 1)  temp_vector += pow(y_value - y_current,2);
                              if (z_flag == 1)  temp_vector += pow(z_value - z_current,2);
                              pos_vector = unit_conv * sqrt(temp_vector) ;          //  toplam mesafe
                              go_time = 60 * pos_vector / f_value;
                              Display96x16x1StringDraw("              ",1, 1);
                              usprintf(temp_sure,"%d", (int)(x_value));
                              Display96x16x1StringDraw(temp_sure,20, 1);
                              usprintf(temp_sure,"%d", (int)(y_value));
                              Display96x16x1StringDraw(temp_sure,40, 1);
                              usprintf(temp_sure,"%d", (int)(z_value));
                              Display96x16x1StringDraw(temp_sure,60, 1);
                              usprintf(temp_sure,"%d", (int)(pos_vector));
                              Display96x16x1StringDraw(temp_sure,80, 1);
                              usprintf(temp_sure,"%d", (int)(go_time));
                              Display96x16x1StringDraw(temp_sure,1, 1);
                              CNC_go_init(x_value - x_current, y_value - y_current, z_value -
z_current, go_time, x_calib, y_calib, z_calib);
                              CNC_go(go_time, x_value - x_current, y_value - y_current, z_value -
z_current);
                              stop_tmr1_val1 = ((double)TimerValueGet(TIMER0_BASE,
TIMER_A)/((double)SysCtlClockGet()));

                                                  }
```

```
                else    // artan mod
                {
                        pos_vector = unit_conv * sqrt( pow(x_value,2) + pow(y_value,2) +
pow(z_value,2) );
                        go_time = pos_vector / (f_value * 0.016667);
                        CNC_go_init(x_value, y_value, z_value, go_time, x_calib, y_calib,
z_calib);
                        CNC_go(go_time, x_value, y_value, z_value);
                }
                break;

           default:  // UARTSend((unsigned char *)"DONE\n",5);
           break;
        }

        switch (mmode)
        {
          case -1: break;
          case 104: send_message = 0;
                     if (x_flag == 1)
                     {
                        if (x_value == 0)
                       {
                                Display96x16x1StringDraw("OFF",50, 0);
                                CNC_PWM_Disable(0);
                                CNC_Timer_Disable(2);
                                tmr1_val2 = ((double)TimerValueGet(TIMER1_BASE,
TIMER_A))/(((double)SysCtlClockGet())*100);
                                tmr1_val3 = tmr1_val1 - tmr1_val2;
                                x_current += tmr1_val3 * x_calib * timer1_speed * 13.587;
                                sendPos(x_current, y_current, z_current);
                       }
                       else
                       {
                                CNC_PWM_Dir(x_value, 0,0);
                                timer1_speed =  x_value / 60.0;
                                Display96x16x1StringDraw("ON ",50, 0);
                                CNC_PWM_SetFreq(0, fabs(x_value)*x_calib);
                                TimerLoadSet(TIMER1_BASE, TIMER_A, SysCtlClockGet()*1000);
                                tmr1_val1 = ((double)TimerValueGet(TIMER1_BASE,
TIMER_A))/(((double)SysCtlClockGet())*100);
                                CNC_PWM_Enable(0);
                                CNC_Timer_Enable(2);
                                jog_x_speed = x_value/16.667;
                                x_value = 0;
                       }
                     }
                     if (y_flag == 1)
                     {
                                if (y_value == 0)
                                {
                                        Display96x16x1StringDraw("OFF",50, 0);
                                        CNC_PWM_Disable(2);
                                        CNC_Timer_Disable(2);
                                        tmr1_val2 = ((double)TimerValueGet(TIMER1_BASE,
TIMER_A))/(((double)SysCtlClockGet())*100);
                                        tmr1_val3 = tmr1_val1 - tmr1_val2;
                                        y_current += tmr1_val3 * y_calib * timer1_speed * 11.725;
                                        sendPos(x_current, y_current, z_current);
```

```
                        }
                        else
                        {
                                CNC_PWM_Dir(0, y_value,0);
                                timer1_speed =  y_value / 60.0;
                                Display96x16x1StringDraw("ON ",50, 0);
                                CNC_PWM_SetFreq(2, fabs(y_value)*y_calib);
                                TimerLoadSet(TIMER1_BASE, TIMER_A,
SysCtlClockGet()*1000);

                                tmr1_val1 = ((double)TimerValueGet(TIMER1_BASE,
TIMER_A))/(((double)SysCtlClockGet())*100);
                                CNC_PWM_Enable(2);
                                CNC_Timer_Enable(2);
                                jog_y_speed = y_value/16.667;
                                y_value = 0;
                        }
                }
                if (z_flag == 1)
                {
                        if (z_value == 0)
                        {
                                Display96x16x1StringDraw("OFF",50, 0);
                                CNC_PWM_Disable(4);
                                CNC_Timer_Disable(2);
                                tmr1_val2 = ((double)TimerValueGet(TIMER1_BASE,
TIMER_A))/(((double)SysCtlClockGet())*100);
                                tmr1_val3 = tmr1_val1 - tmr1_val2;
                                z_current += tmr1_val3 * z_calib * timer1_speed * 15.08;
                                sendPos(x_current, y_current, z_current);
                        }
                        else
                        {
                                CNC_PWM_Dir(0, 0, z_value);
                                timer1_speed =  z_value / 60.0;
                                Display96x16x1StringDraw("ON ",50, 0);
                                CNC_PWM_SetFreq(4, fabs(z_value)*z_calib);
                                TimerLoadSet(TIMER1_BASE, TIMER_A,
SysCtlClockGet()*1000);
                                tmr1_val1 = ((double)TimerValueGet(TIMER1_BASE,
TIMER_A))/(((double)SysCtlClockGet())*100);
                                CNC_PWM_Enable(4);
                                CNC_Timer_Enable(2);
                                jog_z_speed = z_value/16.667;
                                z_value = 0;
                        }
                }
                break;
          default: break;
          }
        x_flag = 0; y_flag = 0; z_flag = 0;
        Display96x16x1StringDraw("          ",70, 0);
        usprintf(wasd,"%d", (int)x_current);
        Display96x16x1StringDraw(wasd,70, 0);
        if (send_message == 1)
        {
          sendPos(x_current, y_current, z_current);
          for(ulLoop = 0; ulLoop < 200000; ulLoop++);
          UARTSend((unsigned char *)"DONE\n",5);
        }
```

```
}

// Kelimeyi değerlendirerek uygun değişkene atama yapar
void EvalWords(char *str)
{
        if (strcmp(str, "OKEY")==0 )            // Kod satırı tamam değilse
        {
          Display96x16x1StringDraw("OK!",10, 1);
          usprintf(temp_sure,"%d", (gmode));
          Display96x16x1StringDraw(temp_sure,40, 0);
          Action();
        }
        else if (strcmp(str, "G00")==0 || strcmp(str, "G0")==0 )     // Hızlı hareket
        {
          gmode = 1; mmode = -1; f_value = 1000;
        }
        else if (strcmp(str, "G01")==0 || strcmp(str, "G1")==0 )     // Verilen hıza göre hareket
        {
          gmode = 1; mmode = -1;
        }
        else if (strcmp(str, "G02")==0 || strcmp(str, "G2")==0 )   // Dairesel hareket
        {
          gmode = 2;
        }
        else if (strcmp(str, "G03")==0 || strcmp(str, "G3")==0 )   // Dairesel hareket
        {
          gmode = 3;
        }
        else if (strcmp(str, "G20")==0 )                    // İnç birimi
        {
          unit_conv = 0.0393700787;
        }
        else if (strcmp(str, "G21")==0 )                    // Milimetre birimi
        {
          unit_conv = 1.0;
        }
        else if (strcmp(str, "G90")==0 )                    // Mutlak mod
        {
          abs_mode = 1;
        }
        else if (strcmp(str, "G91")==0 )                    // Artan mod
        {
          abs_mode = 0;
        }
        else if (strcmp(str, "M00") == 0 || strcmp(str, "M0") == 0 || strcmp(str, "M1") == 0 ||
strcmp(str, "M01") == 0 || strcmp(str, "M30") == 0)
        {
          CNC_PWM_Disable(0); CNC_PWM_Disable(2); CNC_PWM_Disable(4);
          CNC_Timer_Disable(1); CNC_Timer_Disable(2);
        }
        else if (strcmp(str, "M101")==0 )                   // Kullanıcı tanımlı (kalibrasyon)
        {
          mmode = 101;
          gmode = -1;
        }
        else if (strcmp(str, "M102")==0 )                   // Kullanıcı tanımlı (anlık pozisyon)
        {
          mmode = 102;
          gmode = -1;
        }
```

```
else if (strcmp(str, "M103")==0 )                // Kullanıcı tanımlı (maksimum hız)
{
  mmode = 103;
  gmode = -1;
}
else if (strcmp(str, "M104")==0 )                // Kullanıcı tanımlı (adımlama modu)
{
  mmode = 104;
  gmode = -1;
}
else if (strcmp(str, "M105")==0 )                // Kullanıcı tanımlı (limit anahtar)
{
  mmode = 105;
  gmode = -1;
}
else
{
  if (str[0]=='S')                  // Uç işlevci hızı
  {
            char speed_str[10]; int i;
            for(i = 1; i< strlen(str);i++)
            {
                      speed_str[i-1] = str[i];
            }
            spindle_speed = strtodec(speed_str);
  }
  else if (str[0]=='X')             // X pozisyonu girişi
  {
            char x_str[10]; int i;
            if(str[1] == '-')
            {
                      for(i = 2; i< strlen(str); i++)
                      {
                                x_str[i-2] = str[i];
                      }
                      x_value = strtodec(x_str) * -1.0;
            }
            else
            {
                      for(i = 1; i< strlen(str);i++)
                      {
                                x_str[i-1] = str[i];
                      }
                      x_value = strtodec(x_str);
            }

            if (mmode == 101)           x_calib = x_value;
            else if (mmode == 102) x_current = x_value;
            else if (mmode == 103) x_maxfeed = x_value;
            else x_flag = 1;
  }
  else if (str[0]=='Y')             // Y pozisyonu girişi
  {
            char y_str[10]; int i;
            if(str[1] == '-')
            {
                      for(i = 2; i< strlen(str); i++)
                      {
                                y_str[i-2] = str[i];
                      }
```

```
                y_value = strtodec(y_str) * -1.0;
        }
        else
        {
                for(i = 1; i< strlen(str);i++)
                {
                        y_str[i-1] = str[i];
                }
                y_value = strtodec(y_str);
        }

        if (mmode == 101)       y_calib = y_value;
        else if (mmode == 102) y_current = y_value;
        else if (mmode == 103) y_maxfeed = y_value;
        else y_flag = 1;
}
else if (str[0]=='Z')                   // Z pozisyonu girişi
{
        char z_str[10]; int i;

        if(str[1] == '-')
        {
                for(i = 2; i< strlen(str); i++)
                {
                        z_str[i-2] = str[i];
                }
                z_value = strtodec(z_str) * -1.0;
        }
        else
        {
                for(i = 1; i< strlen(str);i++)
                {
                        z_str[i-1] = str[i];
                }
                z_value = strtodec(z_str);
        }

        if (mmode == 101)       z_calib = z_value;
        else if (mmode == 102) z_current = z_value;
        else if (mmode == 103) z_maxfeed = z_value;
        else z_flag = 1;
}
else if (str[0]=='R')                   // R pozisyonu girişi
{
        char r_str[10]; int i;
        for(i = 1; i< strlen(str);i++)
        {
                r_str[i-1] = str[i];
        }
        r_value = strtodec(r_str);
}
else if (str[0]=='F')                   // İlerleme hızı girişi
{
        char f_str[10]; int i;
        for(i = 1; i< strlen(str);i++)
        {
                f_str[i-1] = str[i];
        }
        f_value = strtodec(f_str);
}
```

```
        else if (str[0]=='T')                   // Alet girişi
        {
                char t_str[10];
                int i;
                for(i = 1; i< strlen(str);i++)
                {
                        t_str[i-1] = str[i];
                }
                t_value = strtodec(t_str);
        }
        else {}
      }
}

// Seri haberleşmeyle gelen metni öncelikle kelimelere ayırarak değerlendirir (EvalWords),
// sonrasında uygun hareket talimatını verir (Action).
void EvalUart(char *str)
{
        char *pch;
        pch = strtok(str, " ");
        while (pch!= NULL)
        {
                trim(pch);
                EvalWords(pch);
                pch = strtok(NULL, " ");
        }
        Action();
}


// Seri haberleşme kesmesidir, seri haberleşme yoluyla veri geldiğinde aktif olur, bilgisayardan
// gelen metni toplar ve değerlendirir (EvalUart)
void UARTIntHandler(void)
{
   unsigned long ulStatus;
   ulStatus = UARTIntStatus(UART0_BASE, true);      // Kesme durumunu alır
   UARTIntClear(UART0_BASE, ulStatus);              // Kesme durumunu temizler
   while(UARTCharsAvail(UART0_BASE))
  // Okunacak karakter kalmayıncaya kadar sonsuz döngüde oku
   {
        UART_temp[UART_i] = UARTCharGet(UART0_BASE);
        if(UART_i >= 1)
        {
                if (UART_temp[UART_i - 1] == 'O' && UART_temp[UART_i] == 'K')
                {
                        UART_temp[UART_i-1] = '\0';
                        Display96x16x1StringDraw(UART_temp, 6, 1);
                        EvalUart(UART_temp);
                        UART_i = -1;
                }
        }
        UART_i++;
   }
}

// Kart üzerinde yer alan kullanıcı butonuna ait kesmedir, herhangi bir durumda motorları
// durdurur.
void GPIOCIntHandler(void)
{
   GPIOPinIntClear(GPIO_PORTC_BASE, GPIO_PIN_4);    // Kesme durumunu temizle
```

```
   CNC_PWM_Disable(0);
   CNC_PWM_Disable(2);
   CNC_PWM_Disable(4);
}

// Limit anahtarların bağlı olduğu pinlerin kesmeleri
void GPIOAIntHandler(void)
{
   char str_lsw[100];
   GPIOPinIntClear(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 |
GPIO_PIN_5);
   CNC_PWM_Disable(0); CNC_PWM_Disable(2); CNC_PWM_Disable(4);
   CNC_Timer_Disable(1);
   Display96x16x1StringDraw("A INT", 6, 1);
   Display96x16x1StringDraw("ok INT", 6, 0);
   lsw_pinstate = GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_5 )/32;
   lsw_pinstate += GPIOPinRead(GPIO_PORTB_BASE,  GPIO_PIN_1 );
   lsw_pinstate += GPIOPinRead(GPIO_PORTA_BASE,  GPIO_PIN_2 | GPIO_PIN_3 |
GPIO_PIN_4 | GPIO_PIN_5);
   usprintf(wasd,"%d", lsw_pinstate);
   Display96x16x1StringDraw("   ",60, 0);
   Display96x16x1StringDraw(wasd,60, 0);
   sendLSW(lsw_pinstate);
   usprintf(str_lsw,"M105 X%d\n", lsw_pinstate);
   UARTSend((unsigned char *)str_lsw, strlen(str_lsw));
   if (gmode == 0 || gmode == 1)
  {
        switch (abs_mode)
        {
            case 1 :   stop_tmr1_val2 = ((double)TimerValueGet(TIMER0_BASE,
TIMER_A)/(((double)SysCtlClockGet())));
                       stop_tmr1_val3 = (stop_tmr1_val1 - stop_tmr1_val2);
                       gstop = 1;
                       if (gmode == 1)
                       {
                            x_current += (x_value - x_current)*(stop_tmr1_val3 /stop_tmr1_val1);
                            y_current += (y_value - y_current)*(stop_tmr1_val3 /stop_tmr1_val1);
                            z_current += (z_value - z_current)*(stop_tmr1_val3 /stop_tmr1_val1);
                       }
                       sendPos(x_current, y_current, z_current);
                       break;
            case 0 :  stop_tmr1_val2 = ((double)TimerValueGet(TIMER0_BASE,
TIMER_A)/(((double)SysCtlClockGet())));
                       stop_tmr1_val3 = (stop_tmr1_val1 - stop_tmr1_val2);
                       gstop = 1;
                       if (gmode == 1)
                       {
                            x_current += x_value *(stop_tmr1_val3 /stop_tmr1_val1);
                            y_current += y_value *(stop_tmr1_val3 /stop_tmr1_val1);
                            z_current += z_value *(stop_tmr1_val3 /stop_tmr1_val1);
                       }
                       sendPos(x_current, y_current, z_current);
                       x_value = 0; y_value = 0; z_value = 0; break;
            default : break;
          }
    }
    UARTSend((unsigned char *)"ERRORA\n",7);
}

// Limit anahtarların bağlı olduğu pinlerin kesmeleri
```

```
void GPIOBIntHandler(void)
{
        char str_lsw[100];
   GPIOPinIntClear(GPIO_PORTB_BASE, GPIO_PIN_1 | GPIO_PIN_5 );
        CNC_PWM_Disable(0); CNC_PWM_Disable(2); CNC_PWM_Disable(4);
        CNC_Timer_Disable(1);
        Display96x16x1StringDraw("A INT", 6, 1);
        Display96x16x1StringDraw("ok INT", 6, 0);
        lsw_pinstate = GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_5 )/32;
        lsw_pinstate += GPIOPinRead(GPIO_PORTB_BASE,  GPIO_PIN_1 );
        lsw_pinstate += GPIOPinRead(GPIO_PORTA_BASE,  GPIO_PIN_2 | GPIO_PIN_3 |
GPIO_PIN_4 | GPIO_PIN_5);
        usprintf(wasd,"%d", lsw_pinstate);
        Display96x16x1StringDraw("   ",60, 0);
        Display96x16x1StringDraw(wasd,60, 0);
        sendLSW(lsw_pinstate);
        usprintf(str_lsw,"M105 X%d\n", lsw_pinstate);
        UARTSend((unsigned char *)str_lsw, strlen(str_lsw));
        if (gmode == 0 || gmode == 1)
        {
            switch (abs_mode)
            {
                case 1 :  stop_tmr1_val2 = ((double)TimerValueGet(TIMER0_BASE,
TIMER_A)/(((double)SysCtlClockGet())));
                          stop_tmr1_val3 = (stop_tmr1_val1 - stop_tmr1_val2);
                          gstop = 1;
                          if (gmode == 1)
                          {
                            x_current += (x_value - x_current)*(stop_tmr1_val3 /stop_tmr1_val1);
                            y_current += (y_value - y_current)*(stop_tmr1_val3 /stop_tmr1_val1);
                            z_current += (z_value - z_current)*(stop_tmr1_val3 /stop_tmr1_val1);
                          }
                          sendPos(x_current, y_current, z_current);
                          break;
                case 0 :  stop_tmr1_val2 = ((double)TimerValueGet(TIMER0_BASE,
TIMER_A)/(((double)SysCtlClockGet())));
                          stop_tmr1_val3 = (stop_tmr1_val1 - stop_tmr1_val2);
                          gstop = 1;
                          if (gmode == 1)
                          {
                            x_current += x_value *(stop_tmr1_val3 /stop_tmr1_val1);
                            y_current += y_value *(stop_tmr1_val3 /stop_tmr1_val1);
                            z_current += z_value *(stop_tmr1_val3 /stop_tmr1_val1);
                          }
                          sendPos(x_current, y_current, z_current);
                          x_value = 0; y_value = 0; z_value = 0; break;
                          default : break;
                }
        }
        UARTSend((unsigned char *)"ERRORB\n",7);
}


// Zamanlayıcı 0'ın kesmesidir, zamanlayıcı süresi dolduğunda aktif olur, hareket
// tamamlandığında yapılacak hareket ve protokolleri uygular
void Timer0IntHandler(void)
{
   TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
   IntMasterDisable();
   CNC_PWM_Disable(0);
```

```
    CNC_PWM_Disable(2);
    CNC_PWM_Disable(4);
    if (gstop ==0)
   {
         if (gmode == 0 || gmode == 1)
        {
           switch (abs_mode)
           {
                    case 1 : x_current = x_value; y_current = y_value; z_current = z_value; break;
                    case 0 : x_current += x_value; y_current += y_value; z_current += z_value;
                             x_value = 0; y_value = 0; z_value = 0; break;
                    default : break;
           }
        }
      Display96x16x1StringDraw("DONE!",1, 0);
      UARTSend((unsigned char *)"DONE\n",5);
      sendPos(x_current, y_current, z_current);
   }
   else
   {
         gstop = 0;
         x_flag = 0; y_flag = 0; z_flag = 0;
   }
  IntMasterEnable();
}

// Ana fonksiyondur, mikro denetleyici çalıştırıldığında ilk çalışan fonksiyondur
int main(void)
{
   IntMasterEnable();
    // Sistem saatinin ayarlar
    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_6MHZ);
    SysCtlPWMClockSet(SYSCTL_PWMDIV_1);
    // Ekranı başlatır
    Display96x16x1Init(false);
   CNC_UART_init();       // haberleşmeyi ayarlar
   CNC_PWM_init();        // PWM kanallarını ayarlar
   CNC_PWM_Dir_init(); // hareket yönü pinlerini ayarlar
   LSW_init(GPIOAIntHandler, GPIOBIntHandler); // limit anahtarları ayarlar
   UserButton_init(GPIOCIntHandler);        // kullanıcı butonunu ayarlar
   CNC_Timer_init();                  // zamanlayıcıları ayarlar
   CNC_PWM_Disable(0);            // PWM çıkışlarını etkisiz kıl
   CNC_PWM_Disable(2);
   CNC_PWM_Disable(4);
    while(1)                          // sonsuz döngü
    {  }
}
```

```
//*****************************************************************************
//
//        cnc_pwm.c  - Kullanıcı tanımlı bir kütüphanedir. CNC için PWM ve yön pinlerinin
//                     ayarlanması ve kontrolünde kullanılır
//
//*****************************************************************************
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/lm3s811.h"
#include "driverlib/cnc_pwm.h"
#include "driverlib/cnc_timer.h"
#include <math.h>
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/pwm.h"
#include "driverlib/uart.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "drivers/display96x16x1.h"

// PWM ayarlarını yapar ve başlatır
// PWM0, PWM2 ve PWM4 kanallarını ayarlar, çıkış pinleri D0, B0 ve E0'dır. Bunlara ait yön
// pinleri ise C5, C6 ve C7'dir.  Kanalları aktif hale getirmek için "CNC_PWM_Enable"
// fonksiyonu gereklidir.
void CNC_PWM_init(void)
{
        volatile unsigned long ulLoop;
        SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOC;    // yönlendirme portu açılır
        ulLoop = SYSCTL_RCGC2_R;        // zaman kazanmak için birkaç gereksiz okuma
        GPIO_PORTC_DIR_R |= 0xE0;
        GPIO_PORTC_DEN_R |= 0xE0;

        SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM);              // çevre birimleri aktive et
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);            // PWM0
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);            // PWM2
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);            // PWM4

        GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);    // D0
        GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_0);    // B0
        GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_0);    // E0

        // PWM'leri PWMGEN modülleriyle eşleştir
        PWMGenConfigure(PWM_BASE, PWM_GEN_0, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);
        PWMGenPeriodSet(PWM_BASE, PWM_GEN_0, SysCtlClockGet()/ 100 );
        PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, SysCtlClockGet()/ 200 );
        PWMGenEnable(PWM_BASE,  PWM_GEN_0 );

        PWMGenConfigure(PWM_BASE, PWM_GEN_1, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);
        PWMGenPeriodSet(PWM_BASE, PWM_GEN_1, SysCtlClockGet()/ 100 );
        PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, SysCtlClockGet()/ 200 );
        PWMGenEnable(PWM_BASE,  PWM_GEN_1 );
```

```
        PWMGenConfigure(PWM_BASE, PWM_GEN_2, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);
        PWMGenPeriodSet(PWM_BASE, PWM_GEN_2, SysCtlClockGet()/ 100 );
        PWMPulseWidthSet(PWM_BASE, PWM_OUT_4, SysCtlClockGet()/ 200 );
        PWMGenEnable(PWM_BASE,  PWM_GEN_2 );
}

// İstenilen PWM kanalını aktive eder
// PWM0, PWM2 ve PWM4 kanalları D0, B0 ve E0 pinlerine bağlıdır.
// pwm_channel değişkeni, PWMx üzerindeki x değerini ifade eder.
void CNC_PWM_Enable(int pwm_channel)
{
        switch (pwm_channel){
          case 0:   PWMOutputState(PWM_BASE, PWM_OUT_0_BIT, true);   break;
          case 2:   PWMOutputState(PWM_BASE, PWM_OUT_2_BIT, true);   break;
          case 4:   PWMOutputState(PWM_BASE, PWM_OUT_4_BIT, true);   break;
          default:    break;
        }
}

// İstenilen PWM kanalını devre dışı bırakır
// PWM0, PWM2 ve PWM4 kanalları D0, B0 ve E0 pinlerine bağlıdır.
// pwm_channel değişkeni, PWMx üzerindeki x değerini ifade eder.
void CNC_PWM_Disable(int pwm_channel)
{
        switch (pwm_channel){
          case 0: PWMOutputState(PWM_BASE, PWM_OUT_0_BIT, false); break;
          case 2: PWMOutputState(PWM_BASE, PWM_OUT_2_BIT, false); break;
          case 4: PWMOutputState(PWM_BASE, PWM_OUT_4_BIT, false); break;
          default:    break;
        }
}

// İstenilen PWM kanalınıın PWM sinyalinini istenilen frekansa ayarlanmasını sağlar
void CNC_PWM_SetFreq(int pwm_channel, double freq)
{
        switch (pwm_channel)
        {
          case 0:   PWMGenPeriodSet(PWM_BASE, PWM_GEN_0, (unsigned long)
SysCtlClockGet()/ freq );
                    PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, (unsigned long)
SysCtlClockGet()/ (2*freq) );         break;
          case 2:   PWMGenPeriodSet(PWM_BASE, PWM_GEN_1, (unsigned long)
SysCtlClockGet()/ freq );
                    PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, (unsigned long)
SysCtlClockGet()/ (2*freq) ); break;
          case 4:   PWMGenPeriodSet(PWM_BASE, PWM_GEN_2, (unsigned long)
SysCtlClockGet()/ freq );
                     PWMPulseWidthSet(PWM_BASE, PWM_OUT_4, (unsigned long)
SysCtlClockGet()/ (2*freq) ); break;
          default:  break;
          }
}

// Gerçekleştirilecek hareket öncesi zamanlayıcının, PWM frekansının ve hareket yönünün
// tespiti ve ayarlanması sağlanır
void CNC_go_init(double x_val, double y_val, double z_val, double go_time, double x_calib,
double y_calib, double z_calib)
{
        double x_speed = fabs(x_val) / go_time;
```

```
        double y_speed = fabs(y_val) / go_time;
        double z_speed = fabs(z_val) / go_time;
        CNC_PWM_Dir(x_val, y_val, z_val);
        CNC_PWM_SetFreq(0, x_speed * x_calib);
        CNC_PWM_SetFreq(2, y_speed * y_calib);
        CNC_PWM_SetFreq(4, z_speed * z_calib);
}

// Hareket yönünü ayarlar
// PWM0, PWM2 ve PWM4 yönlerini C5, C6 ve C7 pinlerini değiştirerek ayarlarlar
// Örnek : eğer (x_go > 0 ) pin_c5->high, değilse pin_c5->low
void CNC_PWM_Dir(double x_go, double y_go, double z_go)
{
        if (x_go >= 0) GPIO_PORTC_DATA_R |= 0x20;
        else GPIO_PORTC_DATA_R &= ~(0x20);
        if (y_go >= 0) GPIO_PORTC_DATA_R |= 0x40;
        else GPIO_PORTC_DATA_R &= ~(0x40);
        if (z_go >= 0) GPIO_PORTC_DATA_R |= 0x80;
        else GPIO_PORTC_DATA_R &= ~(0x80);
}

// Hareket yönlerini belirleyen pinleri ayarlar ve başlatır
void CNC_PWM_Dir_init(void)
{
         SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
        GPIODirModeSet(GPIO_PORTC_BASE, GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7,
GPIO_DIR_MODE_OUT);
}
```

```
//*****************************************************************************
//
//        cnc_uart.c  - Kullanıcı tanımlı bir kütüphanedir. Seri haberleşmenin
//                      ayarlanması ve kontrolünde kullanılır
//
//*****************************************************************************
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/cnc_pwm.h"
#include "driverlib/cnc_uart.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/pwm.h"
#include "driverlib/uart.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "drivers/display96x16x1.h"
#include "utils/ustdlib.c"

// UART0 kanalını ayarlar ve başlatır
void CNC_UART_init()
{
   SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
   GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
   UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 9600,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
   IntEnable(INT_UART0);
   UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);
}

// Verilen metni UART0 kanalıyla yollar
void UARTSend(const unsigned char *pucBuffer, unsigned long ulCount)
{
   while(ulCount--)                    // gönderilecek karakter varken döngüde kal
   {
        UARTCharPut(UART0_BASE, *pucBuffer++);      // Sıradaki karakteri UART'la gönder
   }
}
```

```
//*****************************************************************************
//
//        cnc_timer.c  - Kullanıcı tanımlı bir kütüphanedir. Zamanlayıcıların
//                       ayarlanması ve kontrolünde kullanılır
//
//*****************************************************************************
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/cnc_pwm.h"
#include "driverlib/cnc_uart.h"
#include "driverlib/cnc_misc.h"
#include "driverlib/cnc_timer.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/pwm.h"
#include "driverlib/uart.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "drivers/display96x16x1.h"

// Zamanlayıcı 0 ile Zamanlayıcı 1'in ayarlarını yapar
// Başlatmak için CNC_Timer_Start veya CNC_Timer_Enable gereklidir.
void CNC_Timer_init(void)
{
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
        TimerConfigure(TIMER0_BASE, TIMER_CFG_ONE_SHOT);
        TimerConfigure(TIMER1_BASE, TIMER_CFG_ONE_SHOT_UP);

        TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
        TimerLoadSet(TIMER0_BASE, TIMER_A, SysCtlClockGet());
        TimerLoadSet(TIMER1_BASE, TIMER_A, SysCtlClockGet() / 2);

        TimerLoadSet(TIMER1_BASE, TIMER_A, SysCtlClockGet()*1000);
        TimerLoadSet(TIMER1_BASE, TIMER_A, 1000);

        IntEnable(INT_TIMER0A);
        IntEnable(INT_TIMER1A);
        TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
        TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
}

// İstenilen zamanlayıcıyı başlatır
// timer -> zamanlayıcı 0 için timer=1, zamanlayıcı 1 için timer=2
void CNC_Timer_Enable(int timer)
{
        if (timer==1)
          TimerEnable(TIMER0_BASE, TIMER_A);
        else if (timer==2)
           TimerEnable(TIMER1_BASE, TIMER_A);
}

// İstenilen zamanlayıcıyı devre dışı bırakır
// timer -> zamanlayıcı 0 için timer=1, zamanlayıcı 1 için timer=2
void CNC_Timer_Disable(int timer)
```

```
{
        if (timer==1)
          TimerDisable(TIMER0_BASE, TIMER_A);
        else if (timer==2)
            TimerDisable(TIMER1_BASE, TIMER_A);
}

// İstenilen zamanlayıcıyı verilen zaman değeri ile başlatır
void CNC_Timer_start(double sure, int timer)
{
        if (timer==1){
            TimerLoadSet(TIMER0_BASE, TIMER_A, (unsigned long)(SysCtlClockGet()*sure));
             CNC_Timer_Enable(timer);
        }
        else if (timer==2){
            TimerLoadSet(TIMER0_BASE, TIMER_A, (int)(SysCtlClockGet()*sure));
        }
}

// Zamanlayıcı 0'ı başlatır ve PWM çıkışlarını aktive eder
void CNC_go(double sure, double x_val, double y_val, double z_val)
{
        if (x_val != 0)
          CNC_PWM_Enable(0);
        if (y_val != 0)
          CNC_PWM_Enable(2);
        if (z_val != 0)
          CNC_PWM_Enable(4);
        CNC_Timer_start(sure, 1);
}
```

```
//*****************************************************************************
//
//        cnc_misc.c  - Kullanıcı tanımlı bir kütüphanedir. Çeşitli durumların
//                      ayarlanması ve kontrolünde kullanılır
//
//*****************************************************************************
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/cnc_pwm.h"
#include "driverlib/cnc_uart.h"
#include "driverlib/cnc_misc.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/pwm.h"
#include "driverlib/uart.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "drivers/display96x16x1.h"

// Kullanıcı butonunu ve kesmesini ayarlar
void UserButton_init(void (*pfnIntHandler)(void))
{
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
        GPIOPinTypeGPIOInput(GPIO_PORTC_BASE, GPIO_PIN_4);
        GPIOPortIntRegister(GPIO_PORTC_BASE,pfnIntHandler);
        GPIOPinIntClear(GPIO_PORTC_BASE,GPIO_PIN_4);
        GPIOIntTypeSet(GPIO_PORTC_BASE, GPIO_PIN_4, GPIO_FALLING_EDGE);
        GPIOPinIntEnable(GPIO_PORTC_BASE, GPIO_PIN_4);
        IntEnable(INT_GPIOC);
}

// Limit anahtar ayarları
void LSW_init(void (*pfnIntHandler1)(void), void (*pfnIntHandler2)(void) )
{
          SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
          GPIOPinTypeGPIOInput(GPIO_PORTA_BASE,   GPIO_PIN_2 | GPIO_PIN_3 |
GPIO_PIN_4 | GPIO_PIN_5);
          GPIOPortIntRegister(GPIO_PORTA_BASE, pfnIntHandler1);
          GPIOPinIntClear(GPIO_PORTA_BASE,  GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 |
GPIO_PIN_5);
   GPIOIntTypeSet(GPIO_PORTA_BASE,   GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 |
GPIO_PIN_5, GPIO_BOTH_EDGES);
          GPIOPinIntEnable(GPIO_PORTA_BASE,    GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4
| GPIO_PIN_5);
          IntEnable(INT_GPIOA);

          SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
          GPIOPinTypeGPIOInput(GPIO_PORTB_BASE,   GPIO_PIN_1 | GPIO_PIN_5 );
          GPIOPortIntRegister(GPIO_PORTB_BASE, pfnIntHandler2);
          GPIOPinIntClear(GPIO_PORTB_BASE,  GPIO_PIN_1 | GPIO_PIN_5 );
   GPIOIntTypeSet(GPIO_PORTB_BASE,   GPIO_PIN_1 | GPIO_PIN_5 ,
GPIO_BOTH_EDGES);
          GPIOPinIntEnable(GPIO_PORTB_BASE,    GPIO_PIN_1 | GPIO_PIN_5 );
          IntEnable(INT_GPIOB);
}
```

Ek 6 Bilgisayar programı kodları

```
///
/// Program.cs
///

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;

namespace WindowsFormsApplication5
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

```
///
/// Form1.cs
///

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using Microsoft.VisualBasic;


namespace WindowsFormsApplication5
{
    public partial class Form1 : Form
    {

        #region structures      // Yapılar
        // Anlık mutlak pozisyon koordinatları
        public struct Current
        {
            public double x, y, z;
        }
        // Gelecek mutlak pozisyon koordinatları
        public struct NextAbs
        {
            public double x, y, z, r;
        }
        // Gelecek artan pozisyon koordinatları
        public struct NextInc
        {
            public double x, y, z, r;
        }
        // Bayraklar
        public struct Flag
        {
            public int x, y, z;
        }
        // Hareket hızları
        public struct Feedrate
        {
            public double x, y, z;
        }
        // Gidilecek pozisyon
        public struct ToGo
        {
            public double x, y, z;
        }

        public struct Position  // Pozisyon yapılarının tek bir yapıda toplanması
        {
            public Current current;
            public NextAbs nextabs;
            public NextInc nextinc;
            public Flag flag;
            public Feedrate feedrate;
            public ToGo togo;
        }
        #endregion

        #region some user defined variables     // Bazı kullanıcı tanımlı değişkenler
        public string RxString;
        public string[] str;
        public string myString;
        public double xNew;
        public double yNew;
        public double zNew;
        public double rNew;
```

```
        public string[] ports;
        public bool estop_stat = false;
        public string indata = "";
        public int colorcharno1 = 0;
        public int colorcharno2 = 0;
        public int jogclick_ctr = 0;
        public int jogmode = 0;
        public int xjogincmode = 0;
        public int xjogdecmode = 0;
        public int yjogincmode = 0;
        public int yjogdecmode = 0;
        public int zjogincmode = 0;
        public int zjogdecmode = 0;
        public int rungcode = 0;
        public int timer_tmp = 0;
        public double timer_tmp_xcur = 0;
        public double timer_tmp_xnew = 0;
        public double timer_tmp_xspe = 0;
        public bool timer2_open = false;

        #endregion

        // Sınıf ve yapıların başlatılması
        #region initialization of the classes and structures

        LSW lsw = new LSW();                        // Sınır anahtarlar
        Comm comm =new Comm();    // Metin kutularından seri haberleşme parametreleri
        GcodeFile gcodefile = new GcodeFile();    // İşleme dosyası özellikleri
        Position pos;                               // Tüm pozisyon koordinatları
        Calibration calib = new Calibration();    // Ölçüleme sabitleri
        Process process= new Process();     // İşlemde kullanılan işlem değişkenleri

        #endregion

        #region Form initializations    // Formun başlatılması

        public Form1()
        {
            InitializeComponent();
            InitVariables();     // kullanıcı tanımlı bazı değişkenlerin başlatılması
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            comm_Refresh();       // Varolan haberleşme portları listesini yenile
            RefreshTextBoxes();   // Metin kutularını yenile
            LSWcontrol();                 // Sınır anahtarlar kontollerini yap
            LoadConfig(1);        // Profili yükle
        }

        private void Form1_Shown(object sender, EventArgs e)
        {
             // Profil seçimi ve haberleşme ekranının getir
            DialogResult result3 = MessageBox.Show("Default configuration is loaded. Would
you like to load a different Configuration?", "Configurations",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (result3 == DialogResult.Yes)
                LoadConfig(0);
            DialogResult result2 = MessageBox.Show("Would you like to connect to the
microcontroller?", "Connection",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (result2 == DialogResult.Yes)
            {
                tabControl1.SelectTab(1);
            }
            Timer1_init();
            timer2.Enabled = true;
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (serialPort1.IsOpen) serialPort1.Close();
        }
```

```
        private void InitVariables()
        {
            // Yapı değişkenlerinin başlatılması
            gcodefile.Chosen_File = "";     // işleme dosyasının yolu
            process.line = 0;               // işleme satır numarası
            process.cmd_flag = false;       // komut aktif bayrağı
            process.spindleSpeed = 0;       // uç işlevci hızı
            comm.baudrate = "9600";          // haberleşme bant hızı
            process.gmode = -1;              // g modu numarası
            process.mmode = -1;              // m modu numarası
            process.posType = -1;            // pozisyonlama türü
            gcodefile.unitconv = 1;          // uzunluk birimi
        }

        #endregion

 // Satır okuma-değerlendirme fonksiyon ve butonları
        #region Line reading - evaluating functions and buttons

        private void readoneline_Click(object sender, EventArgs e) // tek satır oku
        {
            readoneline1();
        }

        private void readoneline1() // tek satır oku
        {
            if (NCFileTextBox.Lines.Length > process.line)// prog. sonuna gelmemişse
            {
                myString = NCFileTextBox.Lines[process.line]; // satırı oku
                try
                {
                        // okunan satırın arkaplanını boya
                    NCFileTextBox.Select(colorcharno1,
(NCFileTextBox.Lines[process.line].Length + 1));
                    NCFileTextBox.SelectionBackColor = Color.Red;
                    NCFileTextBox.Select(0, colorcharno1);
                    NCFileTextBox.SelectionBackColor = Color.Transparent;
                    colorcharno1 += myString.Length + 1;
                }
                catch
                { }
                process.line++;         // satır numarasını artır
                colorcharno2 = colorcharno1;
                ReadmyLine2(myString);  // satırımı oku
                EvalMyLine();           // satırımı değerlendir

            }
            else
            {
                process.runcycle = false;       // çalışmayı durdur
                timer2_open = false;            // zamanlayıcıyı durdur
                process.stop = true;            // tek satır oku
                MessageBox.Show("File ended!"); // dosyanın sonu mesajı ver
            }

                // Eğer okuma başarılıysa
            if (process.gmode != -1 && (xNew != 0 || yNew != 0 || zNew != 0))
            {
                MessageBox.Show("X = " + xNew.ToString() + " Y = " + yNew.ToString() + " Z
= " + zNew.ToString());
            }
            serialPort1.Write(myString + "OK");   // satır mesajını gönder
            RefreshTextBoxes();                 // metin kutularını yenile
        }

        private void EvalMyLine() // satırı değerlendir
        {
            if (process.gmode == 1)     // G01 modundaysa
            {
                if (process.posType == 1)       // mutlak pozisyonlamaysa
                {
                    // gidilecek toplam mesafeyi bul
                    double tmp_dist = 0;
```

```
                    if (pos.flag.x == 1) tmp_dist += Math.Pow(pos.nextabs.x -
pos.current.x, 2);
                    if (pos.flag.y == 1) tmp_dist += Math.Pow(pos.nextabs.y -
pos.current.y, 2);
                    if (pos.flag.z == 1) tmp_dist += Math.Pow(pos.nextabs.z -
pos.current.z, 2);
                    tmp_dist = Math.Pow(tmp_dist, 0.5);
                    // hareket süresini hesapla
                    process.sure = (tmp_dist / process.feedrate) * 60;
                    // eksenlerin hareket hızını hesapla
                    if (pos.flag.x == 1) pos.feedrate.x = (pos.nextabs.x - pos.current.x)
/ process.sure;
                    else pos.feedrate.x = 0;
                    if (pos.flag.y == 1) pos.feedrate.y = (pos.nextabs.y - pos.current.y)
/ process.sure;
                    else pos.feedrate.y = 0;
                    if (pos.flag.z == 1) pos.feedrate.z = (pos.nextabs.z - pos.current.z)
/ process.sure;
                    else pos.feedrate.z = 0;

                    pos.flag.x = 0;
                    pos.flag.y = 0;
                    pos.flag.z = 0;

                    pos.togo.x = pos.nextabs.x; pos.togo.y = pos.nextabs.y; pos.togo.z =
pos.nextabs.z;
                }
                else if (process.posType == 0)  // artan pozisyonlama ise
                {
                    double tmp_dist1 = 0;
                    if (pos.flag.x == 1) tmp_dist1 += Math.Pow(pos.nextinc.x, 2);
                    if (pos.flag.y == 1) tmp_dist1 += Math.Pow(pos.nextinc.y, 2);
                    if (pos.flag.z == 1) tmp_dist1 += Math.Pow(pos.nextinc.z, 2);
                    tmp_dist1 = Math.Pow(tmp_dist1, 0.5);
                    process.sure = (tmp_dist1 / process.feedrate) / 0.016667;
                    pos.feedrate.x = (pos.nextinc.x) / process.sure;
                    pos.feedrate.y = (pos.nextinc.y) / process.sure;
                    pos.feedrate.z = (pos.nextinc.z) / process.sure;
                    pos.flag.x = 0;  pos.flag.y = 0;  pos.flag.z = 0;
                    pos.togo.x = pos.current.x + pos.nextinc.x;
                    pos.togo.y = pos.current.y + pos.nextinc.y;
                    pos.togo.z = pos.current.z + pos.nextinc.z;
                }
            }

        }

        private void ReadmyLine2(string codeline)   // satırımı oku
        {
            ClearPos();
            pos.flag.x = 0; pos.flag.y = 0; pos.flag.z = 0;
            codeline = codeline.Trim(); // kenarlarındaki boşluklarını kırp
            string[] words = codeline.Split(' ');       // tek satır oku
            // boşluk karakterlerinden kelimelere böl
     foreach (string word in words)
            {
                EvalWord(word); // kelimeyi değerlendir
            }

            if (process.mmode == 101 || process.mmode == 102 || process.mmode == 103 ||
process.mmode == 104 || process.mmode == 105) process.mmode = -1;

        }

        private void EvalWord(string word)      // kelimeyi değerlendir
        {
            if (word[0] == '%') process.cmd_flag = true;  // prosesi başlat
// parametreyi ve işlemi belirle, gerekli komutu gerçekleştir
            else if (word[0] == 'G' || word[0] == 'M')
            {
                if (word.CompareTo("G0") == 0 || word.CompareTo("G00") == 0) {
process.gmode = 1; process.mmode = -1; process.feedrate = 1000; }           //rapid
movement
```

```
                else if (word.CompareTo("G1") == 0 || word.CompareTo("G01") == 0) {
process.gmode = 1; process.mmode = -1; }           // linear movement
                else if (word.CompareTo("G20") == 0) { process.gmode = -1;
gcodefile.unitconv = 2.54; }       // Inch Units
                else if (word.CompareTo("G21") == 0) { process.gmode = -1;
gcodefile.unitconv = 1; }          // Metric Units
                else if (word.CompareTo("G90") == 0) { process.gmode = -1; process.posType
= 1; }        // Absolute Positioning
                else if (word.CompareTo("G91") == 0) { process.gmode = -1; process.posType
= 0; }        // Incremental Positioning
                else if (word.CompareTo("G92") == 0) { process.gmode = -1; MakeOrigin(); }
// Make Origin
                else if (word.CompareTo("G50") == 0) { process.gmode = 50; process.scale_x
= 1; process.scale_y = 1; process.scale_z = 1; } // Scaling
                else if (word.CompareTo("G51") == 0) { process.gmode = 51; }
                else if (word.CompareTo("M0") == 0 || word.CompareTo("M00") == 0) {
process.mmode = 0; process.line = NCFileTextBox.Lines.Length; }            // program stop
                else if (word.CompareTo("M1") == 0 || word.CompareTo("M01") == 0) {
process.mmode = 1; }            // optional program stop
                else if (word.CompareTo("M2") == 0 || word.CompareTo("M02") == 0) {
process.mmode = 2; process.line = NCFileTextBox.Lines.Length; }            // program end
                else if (word.CompareTo("M3") == 0 || word.CompareTo("M03") == 0) {
process.mmode = 3; }            // spindle on clockwise
                else if (word.CompareTo("M4") == 0 || word.CompareTo("M04") == 0) {
process.mmode = 4; }            // spindle on counter clockwise
                else if (word.CompareTo("M5") == 0 || word.CompareTo("M05") == 0) {
process.mmode = 5; }            // spindle stop
                else if (word.CompareTo("M6") == 0 || word.CompareTo("M06") == 0) {
process.mmode = 6; }            // tool change
                else if (word.CompareTo("M8") == 0 || word.CompareTo("M08") == 0) {
process.mmode = 8; }            // coolant on
                else if (word.CompareTo("M9") == 0 || word.CompareTo("M09") == 0) {
process.mmode = 9; }            // coolant off
                else if (word.CompareTo("M10") == 0) process.mmode = 10;       // clamps
on
                else if (word.CompareTo("M11") == 0) process.mmode = 11;       // clamps
off
                else if (word.CompareTo("M30") == 0) { process.mmode = 30; process.line =
NCFileTextBox.Lines.Length; }      // end of program, reset to start
                else if (word.CompareTo("M98") == 0) process.mmode = 98;       // call
subordinate command
                else if (word.CompareTo("M99") == 0) process.mmode = 99;       // return
from subordinate command
                else if (word.CompareTo("M101") == 0) { process.mmode = 101; }       //
calibration
                else if (word.CompareTo("M102") == 0) { process.mmode = 102; }      //
current pos
                else if (word.CompareTo("M103") == 0) { process.mmode = 103; }     // max
feedrates
                else if (word.CompareTo("M104") == 0) { process.mmode = 104; }   //
jogging
                else if (word.CompareTo("M105") == 0) { process.mmode = 105; }   // LSW
control

            }
            else if (word[0] == 'X')
            {
                if (process.gmode == 51) process.scale_x =
Convert.ToDouble(word.TrimStart('X').Replace('.', ','));
                else if (process.mmode == 101) calib.x =
Convert.ToDouble(word.TrimStart('X').Replace('.', ','));
                else if (process.mmode == 102) pos.current.x =
Convert.ToDouble(word.TrimStart('X').Replace('.', ','));
                else if (process.mmode == 103) process.maxfeed_x =
Convert.ToDouble(word.TrimStart('X').Replace('.', ','));
                else if (process.mmode == 105)
                {
                    lsw.lsw_state = Convert.ToInt32(word.TrimStart('X').Replace('.',
','));
                    lsw.lsw_state_calc();
                }
                else if ((process.gmode == 0 || process.gmode == 1) && process.posType ==
1) { pos.nextabs.x = Convert.ToDouble(word.TrimStart('X').Replace('.', ',')) *
process.scale_x; pos.flag.x = 1; }
```

```
                else if ((process.gmode == 0 || process.gmode == 1) && process.posType ==
0) { pos.nextinc.x = Convert.ToDouble(word.TrimStart('X').Replace('.', ',')) *
process.scale_x; pos.flag.x = 1; }

            }
            else if (word[0] == 'Y')
            {
                if (process.gmode == 51) process.scale_y =
Convert.ToDouble(word.TrimStart('Y').Replace('.', ','));
                else if (process.mmode == 101) calib.y =
Convert.ToDouble(word.TrimStart('Y').Replace('.', ','));
                else if (process.mmode == 102) pos.current.y =
Convert.ToDouble(word.TrimStart('Y').Replace('.', ','));
                else if (process.mmode == 103) process.maxfeed_y =
Convert.ToDouble(word.TrimStart('Y').Replace('.', ','));
                else if ((process.gmode == 0 || process.gmode == 1) && process.posType ==
1) { pos.nextabs.y = Convert.ToDouble(word.TrimStart('Y').Replace('.', ',')) *
process.scale_y; pos.flag.y = 1; }
                else if ((process.gmode == 0 || process.gmode == 1) && process.posType ==
0) { pos.nextinc.y = Convert.ToDouble(word.TrimStart('Y').Replace('.', ',')) *
process.scale_y; pos.flag.y = 1; }
            }
            else if (word[0] == 'Z')
            {
                if (process.gmode == 51) process.scale_z =
Convert.ToDouble(word.TrimStart('Z').Replace('.', ','));
                else if (process.mmode == 101) calib.z =
Convert.ToDouble(word.TrimStart('Z').Replace('.', ','));
                else if (process.mmode == 102) pos.current.z =
Convert.ToDouble(word.TrimStart('Z').Replace('.', ','));
                else if (process.mmode == 103) process.maxfeed_z =
Convert.ToDouble(word.TrimStart('Z').Replace('.', ','));
                else if ((process.gmode == 0 || process.gmode == 1) && process.posType ==
1) { pos.nextabs.z = Convert.ToDouble(word.TrimStart('Z').Replace('.', ',')) *
process.scale_z; pos.flag.z = 1; }
                else if ((process.gmode == 0 || process.gmode == 1) && process.posType ==
0) { pos.nextinc.z = Convert.ToDouble(word.TrimStart('Z').Replace('.', ',')) *
process.scale_z; pos.flag.z = 1; }
            }
            else if (word[0] == 'O')
            {
                gcodefile.programNo = Convert.ToInt16(word.TrimStart('O').Replace('.',
','));
                programNotextBox.Text = gcodefile.programNo.ToString();
            }
            else if (word[0] == 'F') process.feedrate =
Convert.ToDouble(word.TrimStart('F').Replace('.', ','));
            else if (word[0] == 'S') process.spindleSpeed =
Convert.ToDouble(word.TrimStart('S').Replace('.', ','));
            else if (word[0] == 'T') process.tool = Convert.ToInt16(word.TrimStart('T'));
        }

        private void readonelineButton_Click(object sender, EventArgs e)
        {
            if (lsw.error == false && process.stop == false)
                readoneline1(); // tek satır oku
            else
                MessageBox.Show("Check the faults (Reset or LSW button)");
        }

        #endregion

        #region Strip Buttons   // Menü kuşağı butonları

        private void openStripMenuItem_Click(object sender, EventArgs e)
        {
            LoadGcode();        // işleme kodunu yükle
        }

        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ActiveForm.Close(); // formu kapat
        }
```

```
 // referans internet sitesine git
        private void viewImagesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("http://www.cncezpro.com/gcodes.cfm");
            System.Diagnostics.Process.Start("http://www.cncezpro.com/mcodes.cfm");

        }

        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Author\nNail Akcura\n----------\nAll rights reserved\nEge
University\nMechatronics Engineering\n2014");
        }
        #endregion

        #region COM PORT Buttons        // Haberleşme butonları

 // haberleşme özelliklerini uygula
        private void commFormApplybutton_Click(object sender, EventArgs e)
        {
            comm.portno = portnocomboBox.Text;
            comm.baudrate = baudratecomboBox.Text;
            try
            {
                textBox3.Text = comm.portno + " baudrate: " + comm.baudrate;
                serialPort1.PortName = comm.portno;
                serialPort1.BaudRate = Convert.ToInt32(comm.baudrate);
                serialPort1.Parity = Parity.None;
                serialPort1.StopBits = StopBits.One;
                serialPort1.DataBits = 8;
                serialPort1.Handshake = Handshake.None;
                serialPort1.ReadTimeout = -1;
                serialPort1.WriteTimeout = -1;
                serialPort1.DtrEnable = true;
                serialPort1.RtsEnable = true;
                serialPort1.DataReceived += new
SerialDataReceivedEventHandler(serialPort1_DataReceived);
                serialPort1.Open();
                if (commCheckBox.Checked == true)
                    SendConfig();
            }

            #region exceptions
            catch (ArgumentException)
            {
                MessageBox.Show("invalid comm port");
            }

            catch(FormatException)
            {
                MessageBox.Show("invalid comm baudrate");
            }
            catch (IOException)
            {
                MessageBox.Show("invalid port");
            }
            #endregion

            if (serialPort1.IsOpen)
            {
                commFormApplybutton.Enabled = false;
                commFormStopbutton.Enabled = true;
            }


        }

         // haberleşmeyi durdur
        private void commFormStopbutton_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Close();
                commFormApplybutton.Enabled = true;
```

```
                commFormStopbutton.Enabled = false;
            }
        }

        #endregion

        #region COM PORT set functions          // haberleşme fonksiyonları

         // varolan haberleşme kanalları listesini yenile
        private void commRefresh_Click(object sender, EventArgs e)
        {
            portnocomboBox.Items.Clear();
            baudratecomboBox.Items.Clear();
            comm_Refresh();
        }

         // varolan haberleşme kanalları listesini yenile
        private void comm_Refresh()
        {
            ports = SerialPort.GetPortNames();

            // adding available ports to the list
            foreach (string port in ports)
            {
                portnocomboBox.Items.Add(port);
            }
            // adding boudrates
            baudratecomboBox.Items.Add("2400");
            baudratecomboBox.Items.Add("4800");
            baudratecomboBox.Items.Add("9600");
            baudratecomboBox.Items.Add("19200");
            baudratecomboBox.Items.Add("384000");
            baudratecomboBox.Items.Add("115200");
            baudratecomboBox.SelectedIndex = 2;
        }

 // haberleşme verisi alındığında aktif ol
        private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs
e)
        {
            try
            {
                indata = serialPort1.ReadLine(); // gelen veriyi al
                ReadmyLine2(indata);                    // protokole göre veriyi oku
                if (process.runcycle == true && indata == "DONE")
                {
                    timer2_open = true;
                }
            }

            #region exceptions
            catch (InvalidOperationException)
            {     }
            catch (TimeoutException)
            {      }
            catch (IOException)
            {
                MessageBox.Show("Beklenmedik kapanma");
                if (serialPort1.IsOpen)
                {
                    serialPort1.Close();
                    commFormApplybutton.Enabled = true;
                    commFormStopbutton.Enabled = false;
                }
            }
            #endregion
        }

        #endregion

        #region Some Positioning Functions      // bazı pozisyonlama fonksiyonları

        private void button3_Click(object sender, EventArgs e)
        {
```

```
            LSWcontrol();              // sınır anahtarları kontrol et
        }

        private void LSWcontrol()
        {
            ZposLSWlabel.BackColor = lsw.ZposLimSw == true ? Color.Lime : Color.Red;
            ZnegLSWlabel.BackColor = lsw.ZnegLimSw == true ? Color.Lime : Color.Red;
            YposLSWlabel.BackColor = lsw.YposLimSw == true ? Color.Lime : Color.Red;
            YnegLSWlabel.BackColor = lsw.YnegLimSw == true ? Color.Lime : Color.Red;
            XposLSWlabel.BackColor = lsw.XposLimSw == true ? Color.Lime : Color.Red;
            XnegLSWlabel.BackColor = lsw.XnegLimSw == true ? Color.Lime : Color.Red;
        }

        private void ClearPos() // gelecek pozisyonları temizle
        {
            pos.nextabs.x = 0; pos.nextabs.y = 0; pos.nextabs.z = 0;
            pos.nextinc.x = 0; pos.nextinc.y = 0; pos.nextinc.z = 0;
        }

        private void MakeOrigin()       // anlık pozisyonu orijin yap
        {
            pos.current.x = 0; pos.current.y = 0;  pos.current.z = 0;
            RefreshTextBoxes();
        }

        private void RefreshTextBoxes() // metin kutularını yenile
        {
            feedratetextBox.Text = process.feedrate.ToString();
            tooltextBox.Text = process.tool.ToString();
            spindleSpeedtextBox.Text = process.spindleSpeed.ToString();
            if (pos.current.x == 0) { xCurrenttextBox.Text = pos.current.x.ToString();
xCurrenttextBox1.Text = pos.current.x.ToString(); }
            else { xCurrenttextBox.Text = Math.Round(pos.current.x, 4).ToString();
xCurrenttextBox1.Text = Math.Round(pos.current.x, 4).ToString(); }
            if (pos.current.y == 0) { yCurrenttextBox.Text = pos.current.y.ToString();
yCurrenttextBox1.Text = pos.current.y.ToString(); }
            else { yCurrenttextBox.Text = Math.Round(pos.current.y, 4).ToString();
yCurrenttextBox1.Text = Math.Round(pos.current.y, 4).ToString(); }
            if (pos.current.z == 0) { zCurrenttextBox.Text = pos.current.z.ToString();
zCurrenttextBox1.Text = pos.current.z.ToString(); }
            else { zCurrenttextBox.Text = Math.Round(pos.current.z, 4).ToString();
zCurrenttextBox1.Text = Math.Round(pos.current.z, 4).ToString(); }
            if (pos.nextabs.x == 0) { xAbsolutetextBox.Text = pos.nextabs.x.ToString(); }
else { xAbsolutetextBox.Text = Math.Round(pos.nextabs.x, 4).ToString(); }
            if (pos.nextabs.y == 0) { yAbsolutetextBox.Text = pos.nextabs.y.ToString(); }
else { yAbsolutetextBox.Text = Math.Round(pos.nextabs.y, 4).ToString(); }
            if (pos.nextabs.z == 0) { zAbsolutetextBox.Text = pos.nextabs.z.ToString(); }
else { zAbsolutetextBox.Text = Math.Round(pos.nextabs.z, 4).ToString(); }
            if (pos.nextinc.x == 0) { xIncremettextBox.Text = pos.nextinc.x.ToString(); }
else { xIncremettextBox.Text = Math.Round(pos.nextinc.x, 4).ToString(); }
            if (pos.nextinc.y == 0) { yIncremettextBox.Text = pos.nextinc.y.ToString(); }
else { yIncremettextBox.Text = Math.Round(pos.nextinc.y, 4).ToString(); }
            if (pos.nextinc.z == 0) { zIncremettextBox.Text = pos.nextinc.z.ToString(); }
else { zIncremettextBox.Text = Math.Round(pos.nextinc.z, 4).ToString(); }
            CalibXtextBox.Text = Math.Round(calib.x, 4).ToString();
            CalibXtextBox2.Text = Math.Round(calib.x, 4).ToString();
            CalibYtextBox.Text = Math.Round(calib.y, 4).ToString();
            CalibYtextBox2.Text = Math.Round(calib.y, 4).ToString();
            CalibZtextBox.Text = Math.Round(calib.z, 4).ToString();
            CalibZtextBox2.Text = Math.Round(calib.z, 4).ToString();
            MaxfeedXTextBox.Text = process.maxfeed_x.ToString();
            MaxfeedYTextBox.Text = process.maxfeed_y.ToString();
            MaxfeedZTextBox.Text = process.maxfeed_z.ToString();
            togoXtextbox.Text = Math.Round(pos.togo.x, 4).ToString();
            togoYtextbox.Text = Math.Round(pos.togo.y, 4).ToString();
            togoZtextbox.Text = Math.Round(pos.togo.z, 4).ToString();
        }

         // metin kutularını yenile
        private void Refreshbutton_Click(object sender, EventArgs e)
        {
            RefreshTextBoxes();
        }
```

```
 // anlık pozisyonu orijin yap
        private void MakeOriginbutton_Click(object sender, EventArgs e)
        {
            MakeOrigin();
            serialPort1.Write("M102 X0 Y0 Z0OK");
        }

        #endregion

        #region Manuel Gcode Input      // elle komut girme

         // elle komut metin kutusunda “enter” tuşuna basılınca komutu çalıştır
        private void GcodeInputTextBox_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {
                process.posType = 1;
                process.cmd_flag = true;
                ReadmyLine2(GcodeInputTextBox.Text);
                EvalMyLine();
                serialPort1.Write(GcodeInputTextBox.Text + "OK");
                RefreshTextBoxes();
            }
        }

        #endregion

         // profil oku-yükle-gönder
        #region Config Read-Load-Send
        private void loadConfigurationsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LoadConfig(0); // profili gönder
        }

        private void LoadConfig(int defaultcfg)  // profili yükle
        {
            if (defaultcfg == 1)
            {
                gcodefile.Config_File = "C:/Users/pc/Desktop/c#
projects/WindowsFormsApplication5/profiles/defaultcfg.txt";
                StreamReader myFile = new StreamReader(gcodefile.Config_File);
                ReadConfig(myFile);
                myFile.Close();
            }
            else
            {
                openFileDialog1.Title = "Select a Configuration File";
                openFileDialog1.InitialDirectory = "C:/Users/pc/Desktop/";
                openFileDialog1.Filter = "Text files|*.txt";
                if (openFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    gcodefile.Config_File = openFileDialog1.FileName;
                    StreamReader myFile = new StreamReader(gcodefile.Config_File);
                    ReadConfig(myFile);

                    myFile.Close();
                }
                else
                {      }
            }
            ProfileTextBox.Text =  gcodefile.Config_File.ToString();
        }

         // profili dosyadan oku
        private void ReadConfig(StreamReader cfgfile)
        {
            while (cfgfile.EndOfStream == false)
            {
                string tempstr = cfgfile.ReadLine();
                string[] words = tempstr.Split(' ');
                int wordflag =0;
                int commandid = 0;
                foreach (string word1 in words)
                {
```

```
                    if (wordflag == 0)
                    {
                        switch (word1)
                        {
                            case "Calibration_X": commandid = 1; break;
                            case "Calibration_Y": commandid = 2; break;
                            case "Calibration_Z": commandid = 3; break;
                            case "Current_X": commandid = 4; break;
                            case "Current_Y": commandid = 5; break;
                            case "Current_Z": commandid = 6; break;
                            case "Maxfeed_X": commandid = 7; break;
                            case "Maxfeed_Y": commandid = 8; break;
                            case "Maxfeed_Z": commandid = 9; break;
                        }
                        wordflag = 1;
                    }
                    else
                    {
                        switch (commandid)
                        {
                            case 1: calib.x = Convert.ToDouble(word1); break;
                            case 2: calib.y = Convert.ToDouble(word1); break;
                            case 3: calib.z = Convert.ToDouble(word1); break;
                            case 4: pos.current.x = Convert.ToDouble(word1); break;
                            case 5: pos.current.y = Convert.ToDouble(word1); break;
                            case 6: pos.current.z = Convert.ToDouble(word1); break;
                            case 7: process.maxfeed_x = Convert.ToDouble(word1); break;
                            case 8: process.maxfeed_y = Convert.ToDouble(word1); break;
                            case 9: process.maxfeed_z = Convert.ToDouble(word1); break;
                        }
                        wordflag = 0;
                    }
                }
            }
            RefreshTextBoxes();
        }

         // profili gönder
        private void SendConfigButton_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen == true) SendConfig();
        }

 // profili gönder
        private void SendConfig()
        {
            serialPort1.Write("M101 X" + Math.Round(calib.x, 4).ToString() + " Y" +
Math.Round(calib.y, 4).ToString() +
    " Z" + Math.Round(calib.z, 4).ToString() + " M102 X" + Math.Round(pos.current.x,
4).ToString() + " Y" +
     Math.Round(pos.current.y, 4).ToString() + " Z" + Math.Round(pos.current.z,
4).ToString() + " M103 X" + Math.Round(process.maxfeed_x, 4).ToString() + " Y" +
     Math.Round(process.maxfeed_y, 4).ToString() + " Z" + Math.Round(process.maxfeed_z,
4).ToString() + "OK");
        }
        #endregion

         // adımlama paneli
        #region Jogging Panel Buttons and Scrolls
        private void JogButton_Click(object sender, EventArgs e)
        {
            JogButtonControl();
        }
        private void JogButtonControl()
        {
            jogclick_ctr++;
            if ((jogclick_ctr % 2) == 1)
            {
                JogButton.BackColor = Color.Green;
                process.jogmode = true;
                toolStripStatusLabel1.Text = "JOGMODE : ON ";
                StatusTextBox.Text = "JOGMODE ON";
            }
            else
```

```
            {
                JogButton.BackColor = Color.Red;
                process.jogmode = false;
                toolStripStatusLabel1.Text = "JOGMODE : OFF";
                StatusTextBox.Text = "";
            }
        }
        private void XtrackBar_Scroll(object sender, EventArgs e)
        {
            label37.Text = "Feedrate : % " + XtrackBar.Value.ToString();
        }
        private void YtrackBar_Scroll(object sender, EventArgs e)
        {
            label38.Text = "Feedrate : % " + YtrackBar.Value.ToString();
        }
        private void ZtrackBar_Scroll(object sender, EventArgs e)
        {
            label39.Text = "Feedrate : % " + ZtrackBar.Value.ToString();
        }
        private void XjogincButton_MouseDown(object sender, MouseEventArgs e)
        {
            string jog_str;
            if (process.jogmode == true && process.stop == false)
            {
                process.xjogincmode = true;
                label40.BackColor = Color.Red;
                jog_str = (XtrackBar.Value).ToString();
                serialPort1.Write("M104 X" + jog_str + "OK");
            }
            else
            {
                MessageBox.Show("JOGMODE is OFF or PROCESS is STOPPED or LSW is ACTIVE.
Please Turn On the Jog Mode or Check the Error!");
                process.xjogincmode = false;
            }
        }
        private void XjogincButton_MouseUp(object sender, MouseEventArgs e)
        {
            if (process.jogmode == true && process.stop == false)
            {
                process.xjogincmode = false;
                label40.BackColor = Color.Green;
                serialPort1.Write("M104 X0OK");
            }

        }
        private void XjogdecButton_MouseDown(object sender, MouseEventArgs e)
        {
            string jog_str;

            if (process.jogmode == true)
            {
                process.xjogdecmode = true;
                label40.BackColor = Color.Red;
                jog_str = (XtrackBar.Value).ToString();
                serialPort1.Write("M104 X-" + jog_str + "OK");
            }
            else
            {
                MessageBox.Show("JOGMODE is OFF or PROCESS is STOPPED or LSW is ACTIVE.
Please Turn On the Jog Mode or Check the Error!");
                process.xjogdecmode = false;
            }
        }
        private void XjogdecButton_MouseUp(object sender, MouseEventArgs e)
        {
            if (process.jogmode == true && process.stop == false)
            {
                process.xjogdecmode = false;
                label40.BackColor = Color.Green;
                serialPort1.Write("M104 X0OK");
            }
        }
        private void YjogincButton_MouseDown(object sender, MouseEventArgs e)
```

```
        {
            string jog_str;
            if (process.jogmode == true)
            {
                process.yjogincmode = true;
                label40.BackColor = Color.Red;
                jog_str = (YtrackBar.Value).ToString();
                serialPort1.Write("M104 Y" + jog_str + "OK");
            }
            else
            {
                MessageBox.Show("JOGMODE is OFF or PROCESS is STOPPED or LSW is ACTIVE.
Please Turn On the Jog Mode or Check the Error!");
                process.yjogincmode = false;
            }
        }
        private void YjogincButton_MouseUp(object sender, MouseEventArgs e)
        {
            if (process.jogmode == true && process.stop == false)
            {
                process.yjogincmode = false;
                label40.BackColor = Color.Green;
                serialPort1.Write("M104 Y0OK");
            }

        }
        private void YjogdecButton_MouseDown(object sender, MouseEventArgs e)
        {
            string jog_str;
            if (process.jogmode == true)
            {
                process.yjogdecmode = true;
                label40.BackColor = Color.Red;
                jog_str = (YtrackBar.Value).ToString();
                serialPort1.Write("M104 Y-" + jog_str + "OK");
            }
            else
            {
                process.yjogdecmode = false;
                MessageBox.Show("JOGMODE is OFF or PROCESS is STOPPED or LSW is ACTIVE.
Please Turn On the Jog Mode or Check the Error!");
            }
        }
        private void YjogdecButton_MouseUp(object sender, MouseEventArgs e)
        {
            if (process.jogmode == true && process.stop == false)
            {
                process.yjogdecmode = false;
                label40.BackColor = Color.Green;
                serialPort1.Write("M104 Y0OK");
            }

        }
        private void ZjogincButton_MouseDown(object sender, MouseEventArgs e)
        {

            string jog_str;
            if (process.jogmode == true)
            {
                process.zjogincmode = true;
                label40.BackColor = Color.Red;
                jog_str = (ZtrackBar.Value).ToString();
                serialPort1.Write("M104 Z" + jog_str + "OK");
            }
            else
            {
                process.zjogincmode = false;
                MessageBox.Show("JOGMODE is OFF or PROCESS is STOPPED or LSW is ACTIVE.
Please Turn On the Jog Mode or Check the Error!");
            }
        }
        private void ZjogincButton_MouseUp(object sender, MouseEventArgs e)
        {
            if (process.jogmode == true && process.stop == false)
```

```
            {
                process.zjogincmode = false;
                label40.BackColor = Color.Green;
                serialPort1.Write("M104 Z0OK");
            }

        }
        private void ZjogdecButton_MouseDown(object sender, MouseEventArgs e)
        {
            string jog_str;
            if (process.jogmode == true)
            {
                process.zjogdecmode = true;
                label40.BackColor = Color.Red;
                jog_str = (ZtrackBar.Value).ToString();
                serialPort1.Write("M104 Z-" + jog_str + "OK");
            }
            else
            {

                process.zjogdecmode = false;
                MessageBox.Show("JOGMODE is OFF or PROCESS is STOPPED or LSW is ACTIVE.
Please Turn On the Jog Mode or Check the Error!");
            }

        }
        private void ZjogdecButton_MouseUp(object sender, MouseEventArgs e)
        {
            if (process.jogmode == true && process.stop == false)
            {
                process.zjogdecmode = false;
                label40.BackColor = Color.Green;
                serialPort1.Write("M104 Z0OK");
            }

        }
        #endregion

        #region Calibration Buttons
        private void XCalibbutton_Click(object sender, EventArgs e)
        {
            string xcalib_str1 = Interaction.InputBox("Please enter the distance to
travel", "X Axis Calibration","1", 500, 300);
            serialPort1.Write("G01 F1 X" + Convert.ToDouble(xcalib_str1) + "OK");
            MessageBox.Show(xcalib_str1 + " is registered. Get ready to measure the travel
of axis!");
            string xcalib_str2 = Interaction.InputBox("After the motion is completed,
please enter the distance travelled", "X Axis Calibration", "1", 500, 300);
            calib.x *= Convert.ToDouble(xcalib_str1) / Convert.ToDouble(xcalib_str2);
            MessageBox.Show("Your X Calibration constant is " + calib.x);
            serialPort1.Write("H01 P" + calib.x +"OK");
            RefreshTextBoxes();
        }
        private void YCalibbutton_Click(object sender, EventArgs e)
        {
            string ycalib_str1 = Interaction.InputBox("Please enter the distance to
travel", "Y Axis Calibration", "1", 500, 300);
            serialPort1.Write("G01 F1 Y" + Convert.ToDouble(ycalib_str1) + "OK");
            MessageBox.Show(ycalib_str1 + " is registered. Get ready to measure the travel
of axis!");
            string ycalib_str2 = Interaction.InputBox("After the motion is completed,
please enter the distance travelled", "Y Axis Calibration", "1", 500, 300);
            calib.y *= Convert.ToDouble(ycalib_str1) / Convert.ToDouble(ycalib_str2);
            MessageBox.Show("Your Y Calibration constant is " + calib.y);
            serialPort1.Write("H02 P" + calib.y + "OK");
            RefreshTextBoxes();
        }
        private void ZCalibbutton_Click(object sender, EventArgs e)
        {
            string zcalib_str1 = Interaction.InputBox("Please enter the distance to
travel", "Z Axis Calibration", "1", 500, 300);
            serialPort1.Write("G01 F1 Z" + Convert.ToDouble(zcalib_str1) + "OK");
            MessageBox.Show(zcalib_str1 + " is registered. Get ready to measure the travel
of axis!");
```

```
            string zcalib_str2 = Interaction.InputBox("After the motion is completed,
please enter the distance travelled", "Z Axis Calibration", "1", 500, 300);
            calib.z *= Convert.ToDouble(zcalib_str1) / Convert.ToDouble(zcalib_str2);
            MessageBox.Show("Your Z Calibration constant is " + calib.z);
            serialPort1.Write("H03 P" + calib.z + "OK");
            RefreshTextBoxes();
        }


        #endregion

        #region TIMER Control // zamanlayıcı kontrolü


        private void Timer1_init()      // zamanlayıcı 1'i başlat
        {
            timer_tmp_xcur = 0;
            timer_tmp_xnew = 200;
            timer_tmp_xspe = 1;
            timer1.Interval = 10;
            timer1.Enabled = true;
        }

         // zamanlayıcı 1'in süresi her dolduğunda hareket, mod ve metin kutusu
         // güncellemelerini yap
        private void timer1_Tick(object sender, EventArgs e)
        {
            RefreshTextBoxes();
            #region Jog mode timer updates
            if (process.jogmode == true)
            {
                if (process.xjogincmode == true && lsw.XposLimSw == false)
                {
                    pos.current.x += XtrackBar.Value * 0.01667;
                    RefreshTextBoxes();
                }
                else if (process.xjogdecmode == true && lsw.XnegLimSw == false)
                {
                    pos.current.x -= XtrackBar.Value * 0.01667;
                    RefreshTextBoxes();
                }
                else if (process.yjogincmode == true && lsw.YposLimSw == false)
                {
                    pos.current.y += YtrackBar.Value * 0.01667;
                    RefreshTextBoxes();
                }
                else if (process.yjogdecmode == true && lsw.YnegLimSw == false)
                {
                    pos.current.y -= YtrackBar.Value * 0.01667;
                    RefreshTextBoxes();
                }
                else if (process.zjogincmode == true && lsw.ZposLimSw == false)
                {
                    pos.current.z += ZtrackBar.Value * 0.01667;
                    RefreshTextBoxes();
                }
                else if (process.zjogdecmode == true && lsw.ZnegLimSw == false)
                {
                    pos.current.z -= ZtrackBar.Value * 0.01667;
                    RefreshTextBoxes();
                }
            }
            #endregion

            #region G-mode timer updates
            if (process.stop == false &&  process.pause == false)
            {
                if (process.sure > 0)
                {
                    if (lsw.error == false)
                    {
                        if (process.gmode == -1)
                         { }
                         else if (process.gmode == 0)
```

```
                    {
                        label45.BackColor = Color.Green;
                        if (process.posType == 1)
                        {
                            pos.current.x += pos.feedrate.x * 0.01667;
                            pos.current.y += pos.feedrate.y * 0.01667;
                            pos.current.z += pos.feedrate.z * 0.01667;
                            RefreshTextBoxes();
                        }
                    }
                    else if (process.gmode == 1)
                    {
                        label45.BackColor = Color.Green;
                        if (process.posType == 1)
                        {
                            pos.current.x += pos.feedrate.x * 0.01667;
                            pos.current.y += pos.feedrate.y * 0.01667;
                            pos.current.z += pos.feedrate.z * 0.01667;
                            RefreshTextBoxes();
                        }
                        else if (process.posType == 0)
                        {
                            pos.current.x += pos.feedrate.x * 0.01667;
                            pos.current.y += pos.feedrate.y * 0.01667;
                            pos.current.z += pos.feedrate.z * 0.01667;
                            RefreshTextBoxes();
                        }
                    }

                    if (process.gmode != 1)
                    {
                        label45.BackColor = Color.Red;
                    }
                    process.sure -= 0.01*1.667;
                    pauseCycleCheckBox.Text = Math.Round(process.sure,4).ToString();
                }

            }
            else
            {
                 MessageBox.Show("lsw error occured");
                 process.stop = true;
                 process.runcycle = false;
                 process.sure = 0;
             }
          }
        }

        #endregion

        #region LSW timer updates

        if (lsw.XposLimSw == true) XposLSWlabel.BackColor = Color.Red;
        else XposLSWlabel.BackColor = Color.Green;
        if (lsw.XnegLimSw == true) XnegLSWlabel.BackColor = Color.Red;
        else XnegLSWlabel.BackColor = Color.Green;
        if (lsw.YposLimSw == true) YposLSWlabel.BackColor = Color.Red;
        else YposLSWlabel.BackColor = Color.Green;
        if (lsw.YnegLimSw == true) YnegLSWlabel.BackColor = Color.Red;
        else YnegLSWlabel.BackColor = Color.Green;
        if (lsw.ZposLimSw == true) ZposLSWlabel.BackColor = Color.Red;
        else ZposLSWlabel.BackColor = Color.Green;
        if (lsw.ZnegLimSw == true) ZnegLSWlabel.BackColor = Color.Red;
        else ZnegLSWlabel.BackColor = Color.Green;
        LSWstate.Text = lsw.lsw_state.ToString();
        #endregion

         // işleme dosyasının otomatik ilerlemesindeki zamanlayıcı 2
        private void timer2_Tick(object sender, EventArgs e)
        {
            if (process.stop == false && process.runcycle == true && process.pause ==
false && timer2_open == true && lsw.error == false)
            {
                readoneline1();
```

```
                timer2_open = false;
            }
        }

        public static bool tick3 = false;

         // durum metin kutusunda durumun tespiti ve gösterilmesi
        private void timer3_Tick(object sender, EventArgs e)
        {
            if (process.stop == true)
            {
                StatusTextBox.Text = "PROCESS STOPPED";

                if (tick3 == false)
                {
                    tick3 = true;
                    panel1.BackColor = Color.Red;
                }
                else
                {
                    tick3 = false;
                    panel1.BackColor = Color.Transparent;
                }
            }
            else
            {
                if (process.runcycle == true && process.pause == false) StatusTextBox.Text
= "RUNNING CYCLE";
                else if (process.runcycle == true) StatusTextBox.Text = "CYCLE PAUSED";
                else if (process.jogmode == true) StatusTextBox.Text = "JOGGING MODE ON";
                else StatusTextBox.Text = "READY TO USE";
            }
        }


        #endregion

        #region Save config (profile)   // profili kaydet

         // profili kaydet
        private void SaveConfig(int savetype)
        {
            string cfgtext;

            cfgtext = "Calibration_X " + calib.x.ToString() + Environment.NewLine;
            cfgtext += "Calibration_Y " + calib.y.ToString() + Environment.NewLine;
            cfgtext += "Calibration_Z " + calib.z.ToString() + Environment.NewLine;
            cfgtext += "Current_X " + pos.current.x.ToString() + Environment.NewLine;
            cfgtext += "Current_Y " + pos.current.y.ToString() + Environment.NewLine;
            cfgtext += "Current_Z " + pos.current.z.ToString() + Environment.NewLine;
            cfgtext += "Maxfeed_X " + process.maxfeed_x.ToString() + Environment.NewLine;
            cfgtext += "Maxfeed_Y " + process.maxfeed_y.ToString() + Environment.NewLine;
            cfgtext += "Maxfeed_Z " + process.maxfeed_z.ToString() + Environment.NewLine;
            if (savetype == 0)
            {
                File.WriteAllText(ProfileTextBox.Text, cfgtext);
            }
            else
            {
                saveFileDialog1.Title = "Select a Profile file";
                saveFileDialog1.InitialDirectory = "C:/Users/pc/Desktop/";
                saveFileDialog1.Filter = "Text files|*.txt";

                if (saveFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    string cfgfilename = saveFileDialog1.FileName;
                    File.WriteAllText(cfgfilename, cfgtext);
                }
            }
        }

         // aynı dosyaya kaydet
        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            SaveConfig(0);
        }

         // farklı dosyaya kaydet
        private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveConfig(1);
        }

        #endregion

        #region Load G-code     // işleme dosyasını yükle

        private void LoadGcodeFileButton_Click(object sender, EventArgs e)
        {
            LoadGcode();// işleme dosyasını yükle
        }

         // işleme dosyasını yükle
        private void LoadGcode()
        {
            openFileDialog1.Title = "Select a Machining File";
            openFileDialog1.InitialDirectory = "C:/Users/pc/Desktop/";
            openFileDialog1.Filter = "NC files|*.nc|Text files|*.txt";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                gcodefile.Chosen_File = openFileDialog1.FileName;
                StreamReader myFile = new StreamReader(gcodefile.Chosen_File);
                NCFileTextBox.LoadFile(gcodefile.Chosen_File,
RichTextBoxStreamType.PlainText);
                textBox2.Text = gcodefile.Chosen_File;
                MessageBox.Show(gcodefile.Chosen_File);
                process.line = 0;
            }
            else
            {     }
        }

        #endregion

        #region Max Feedrates // maksimum hızlar

        private void MaxfeedXTextBox_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {
                process.maxfeed_x = Convert.ToDouble(MaxfeedXTextBox.Text);
                serialPort1.Write("M103 X" + process.maxfeed_x.ToString() + "OK");
            }

        }
        private void MaxfeedYTextBox_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {
                process.maxfeed_y = Convert.ToDouble(MaxfeedYTextBox.Text);

            }
        }
        private void MaxfeedZTextBox_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {
                process.maxfeed_z = Convert.ToDouble(MaxfeedZTextBox.Text);
            }
        }

        #endregion

 // komut döngüsü ve başla-dur butonları
        #region G code Cycle and Start-Stop Buttons

          // döngüyü duraklat
```

```
        private void CyclePausecheckBox_CheckedChanged(object sender, EventArgs e)
        {
            if (CyclePausecheckBox.Checked == true)
            {
                process.pause = true;
            }
            else if (CyclePausecheckBox.Checked == false)
            {
                process.pause = false;
            }
        }

         // döngüyü başlat
        private void RunGCode_Click(object sender, EventArgs e)
        {
            process.runcycle = true;
            timer2_open = true;
        }

          // reset butonu
        private void ResetButton_Click(object sender, EventArgs e)
        {
            if (process.runcycle == true) {process.line--; process.runcycle = false;}
            process.stop = false;
            panel1.BackColor = Color.Transparent;
        }

        private void Reset_Click(object sender, EventArgs e)
        {
            process.line = 0;
            NCFileTextBox.Select(0, 10000);
            NCFileTextBox.SelectionBackColor = Color.Transparent;
            colorcharno1 = 0;
        }

         // durdurma
        private void Stopbutton_Click(object sender, EventArgs e)
        {
            process.stop = true;
            process.sure = 0;
            serialPort1.Write("M30" + "OK");
        }

        private void PauseButton_Click(object sender, EventArgs e)
        {
            rungcode = 0;
        }

        #endregion
    }
}
```

```
///
/// Calibration.cs
///

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication5
{
    class Calibration
    {
        public double x { get; set; }
        public double y { get; set; }
        public double z { get; set; }
        public Calibration()
        {
            x = 100;    // ilk ölçekleme kaysayıları
            y = 100;
            z = 100;
        }
    }
}
```

```
///
/// Comm.cs
///

using System;
using System.Collections.Generic;
using System.IO.Ports;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication5
{
    class Comm
    {
        public string portno;       // haberleşme port no
        public string baudrate;     // haberleşme bant hızı
                public Comm()
        {
            this.portno = "";
            this.baudrate = "";
        }
    }
}
```

```
///
/// GcodeFile.cs
///

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication5
{
    class GcodeFile
    {
        public string Chosen_File;  // seçilen dosya
        public int programNo;       // program numarası
        public double unitconv;     // uzunluk birimi
        public string Config_File;  // profil dosyası
        public GcodeFile()
        {
            this.Chosen_File = "";
            this.programNo = 0;
            this.unitconv = 0;
            this.Config_File = "defaultcfg.txt";
        }
    }
}
```

```
///
/// LSW.cs
///

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication5
{
    class LSW
    {
        public bool XposLimSw;
        public bool XnegLimSw;
        public bool YposLimSw;
        public bool YnegLimSw;
        public bool ZposLimSw;
        public bool ZnegLimSw;
        public int lsw_state=0;
        public bool error;
        public void lsw_state_calc()
        {
            if (lsw_state != 0) error = true;
            int lsw_tmp = lsw_state;
            if (lsw_tmp / 32 == 1) XposLimSw = true; else XposLimSw = false;
            lsw_tmp = lsw_tmp % 32;
            if (lsw_tmp / 16 == 1) XnegLimSw = true; else XnegLimSw = false;
            lsw_tmp = lsw_tmp % 16;
            if (lsw_tmp / 8 == 1) YposLimSw = true; else YposLimSw = false;
            lsw_tmp = lsw_tmp % 8;
            if (lsw_tmp / 4 == 1) YnegLimSw = true; else YnegLimSw = false;
            lsw_tmp = lsw_tmp % 4;
            if (lsw_tmp / 2 == 1) ZposLimSw = true; else ZposLimSw = false;
            lsw_tmp = lsw_tmp % 2;
            if (lsw_tmp == 1) ZnegLimSw = true; else ZnegLimSw = false;
        }
        public LSW()
        {
            this.XposLimSw = false;
            this.XnegLimSw = false;
            this.YposLimSw = false;
            this.YnegLimSw = false;
            this.ZposLimSw = false;
            this.ZnegLimSw = false;
            this.lsw_state = 0;
            this.error = false;
        }
    }
}
```

```
///
/// Process.cs
///

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication5
{
    class Process
    {
        public int line;            // satır numarası
        public double spindleSpeed; // uç işlevci hızı
        public int tool;            // takım no
        public double feedrate;     // ilerleme hızı
        public bool cmd_flag;       // programın başlama durumu (1 ise başladı, 0 ise
                                        // başlamadı)
        public int gmode;           // g modu temsilcisi
        public int mmode;           // m modu temsilcisi
        public int posType;         // pozisyon tipi (1 ise mutlak, 0 ise artan)
        public int gmodeflag;     // g modu durumu
        public double sure;       // süre
        public double maxfeed_x;    // maksimum hızlar
        public double maxfeed_y;
        public double maxfeed_z;
        public bool pause;                // durma
        public bool error;                // hata
        public bool stop;                 // durma
        public bool runcycle;     // döngü durumu
        public bool jogmode;      // adımlama modu durumu
        public bool xjogincmode, yjogincmode, zjogincmode; // artan adımlamaları
        public bool xjogdecmode, yjogdecmode, zjogdecmode; // azalan adımlamalar

        public Process()
        {
            this.line = 0;
            this.spindleSpeed = 0;
            this.tool = 0;
            this.feedrate = 0;
            this.cmd_flag = false;
            this.gmode = 0;
            this.mmode = 0;
            this.posType = 1;
            this.gmodeflag = 0;
            this.sure = 0;
            this.maxfeed_x = 1;
            this.maxfeed_y = 1;
            this.maxfeed_z = 1;
            this.scale_x = 1;
            this.scale_y = 1;
            this.scale_z = 1;
            this.pause = false;
            this.error = false;
            this.stop = true;
            this.runcycle = false;
            this.jogmode = false;
            this.xjogincmode = false;
            this.yjogincmode = false;
            this.zjogincmode = false;
            this.xjogdecmode = false;
            this.yjogdecmode = false;
            this.zjogdecmode = false;
        }
    }
}
```