T.C.
ONDOKUZ MAYIS ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

LEGO MİNDSTORMS NXT 2.0® KİTİNİ KULLANARAK 3B KESİM PLATFORMU GERÇEKLENMESİ

LİSANS BİTİRME TEZİ

Proje Öğrencileri:
Emin Tarık BAKİ
Büşra DEMİR
Funda KARADENİZ
Sefa YILDIZ

Tezin Savuma Tarihi : 10 Haziran 2013

Tez Danışmanı : Doç. Dr. Erdal KILIÇ

Özet

21. Yüzyılda Robotik, Bilgisayar Bilimi içerisinde çok güçlü bir alan haline geldi. Günümüzde kullanışlı mühendislik araçlarını tasarlayan, bunları gerçek dünyaya uyarlayan ve bunu düşük maliyet ve kısa sürede yapan birçok robotik kit piyasaya sürüldü. Bu kitler özellikle genç kesimin hem donanım hem de yazılım yeteneklerini geliştirmesine ciddi anlamda yardımcı oluyorlar. Öğrenciler karmaşık mühendislik problemlerini çözmeyi bu programlanabilir, akıllı ve otonom robotları kullanarak kolayca ve rahatça öğreniyorlar.
Lego® firmasının üretmiş olduğu Lego Mindstorms® robotik kiti kullanıcılara parça parça dinamik ve statik nesneler tasarlamak için imkân veren bu robot kitlerinden bir tanesidir. Bu kitin zengin programlama kütüphanesi ve görsel-bütünleşik programlama ortamı, insanların hayal ettikleri şeyleri kolayca tasarlamalarına ve onların, tasarlanan bu nesneleri test etmesine olanak sağlıyor. Buna rağmen bazı karmaşık problemler söz konusu olduğunda da Lego®'nun kendi görsel programlama ortamı yetersiz kalıyor. Java ve C# gibi temel programlama dillerine kıyasla, dosyadan veri okuma ve bunu işleme gibi önemli bazı işlevleri Lego®'nun görsel ara yüzü kullanıcıya sağlayamıyor. Bu kısıtlamalar doğal olarak kullanıcının daha gerçekçi denemeler yapmasını engelliyor. Temel programlama dilleri ve Lego®'nun kendi Uygulama Programlama Arabirimi(API) kombinasyonu kullanıcıya üst düzeyde öğrenme deneyimi için daha iyi bir çerçeve sağlıyor.

Bu projenin amacı günümüzde ve gelecekte teknoloji dünyasında çok önemli bir yere sahip olacak olan 3B kesim platformu için bir örnek tasarlamaktır. Bu projede donanım olarak Lego Mindstorms NXT 2.0® robotik kiti ve Lego® parçalarını, yazılım olarak ise C# programlama dilini ve gerekli kütüphaneleri kullanarak hem robotun motor kontrolünü sağlayacak, hem de 3B kesim işlevini yapacak olan örnek bir 3B kesim platformu tasarlanacaktır.

Anahtar Kelimeler

3B Kesim Platformu, Robotik, Lego Mindstorms NXT 2.0®, Blender, C#, WPF

İçindekiler Tablosu

1. GİRİŞ

1.1. Problem Tanımı ve Motivasyon

Robotik kitler kendi görsel programlama arabirimiyle kullanıcıya sunulurlar. Bu kitlerin üreticileri, görsel programlamanın kullanıcıya daha rahat mühendislik projeleri tasarlamaları ve uygulamaları için bir ortam hazırladıklarına inanırlar. Onların programlama kaynak eğitimleri çoğunlukla görsel arabirim üzerinedir. Bu arabirimlerin içerisinde belirli kütüphanelerle tanımlanan, robotik kiti kontrol etmek için kullanıcıya görsel bloklar tanımlanır. Basit mühendislik problemleri için çoğu zaman bu bloklar yeterli olur. Bu görsellik kolay gruplama ve birleştirme yetisini kullanıcıya kazandırır. Bunun dışında daha karmaşık projelere gelecek olursak, görsel programlamanın kısıtlamaları projelerin başarısı için ciddi sıkıntılara neden olur. Özellikle veri okuma/yazma uygulama bloklarının eksikliği, motor kontrolü için kullanıcıyı sürekli aynı blokları birbirine bağlamaya zorlar. Basit programlama mantıklarından döngü kurmak, değişken tanımında çok ciddi bir problem haline geliyor. Örnek olarak az değişkenli bir döngü fonksiyonu JAVA, C# gibi bir programlama dilinde kolayca tanımlanabilirken, sürükle/bırak programlama doğası gereği görsel programlamada çok karmaşık, uzun bir hale gelir.

Robotik kitlerin bu görsel arabirim karmaşasına son vermek için, Robotik kitlere alt seviye uygulama programlama ara yüzleri tasarlamak çok mantıklı bir çözümdür. Lego Mindstorms NXT 2.0® için C dilini temel alan çok detaylı tanımlanmış bir API bulunuyor. Bu API'yi diğer dil ve arabirimlerde kullanmak için birçok ara dil tanımlanmıştır. Lego®'nun web sitesinde Java, C#, Ruby, JavaScript başta olmak üzere birçok programlama dili ile yazılmış arabirimler bulunmaktadır. Java ve C# dilleri bu diller arasında proje için en uygun iki dil olarak belirlendi. Bu iki dil akademik ve ticari olarak çok sık kullanılan iki dil olmak üzere proje öğrencileri tarafından C# dili proje için temel dil olarak seçildi. Aynı proje Java programlama dili için benzer nesne-yönelimli programlama mantığı ve model-view controller(MVC) mimarisi kullanılarak tasarlanabilir. Bu önemli sebepleri göz önünde bulundurarak C# dilinde tasarlanacak bir ara yüze karar verildi.

1.2. Bölümlerin Organizasyonu

Bu projede 3B kesim platformu tasarımı için 3 ana adım izlenecek. İlk adım bu projenin amacına uygun kolayca programlanabilir bir donanım bulunması, ikinci adım 3B nesne tasarlama aracı bulunması ve son olarak ise C# dili ile donanımı kontrol edecek ara yüzün gerçeklenmesi.

İkinci bölüm projenin donanım kısmına odaklanacak ve basit yazılım şemaları hakkında bilgi verecek. Arduino[3] gibi elektronik prototip dizaynları programlanabilir beyin ve motorların sıfırdan tasarlanmasını gerektirdiği ve gereken zamanın bitirme tezi zamanından uzun olacağı öngörülerek böyle bir tasarım bu projede kullanılmadı. Diğer taraftan günümüzde robot marketi endüstrisi ve araç-kiti serileri göz önüne alındığında Lego Mindstorms® kutusu içerisinde beyin (brick), motorlar, sensörler, birleştirme parçaları ve kendi yazılımını içerdiği için bu bitirme projesi için uygun görüldü. Bu kit kolayca programlanabilir ve kurulabilir olduğu için seçildi. Bu kit için birden fazla dil için tasarlanmış programlama kütüphaneleri de bulunmaktadır. Ayrıca bu projenin ana temalarından bir tanesi ise Lego Mindstorms NXT 2.0® kitinin Lego® firmasının ürettiği sadece çocukların oynayacağı bir oyuncak olmadığını,

karmaşık bir mühendislik probleminin de çözülebildiği bir kit olabileceğini göstermektir. Lindh & Holgersson[6], 2007 yılında yayınlamış oldukları "Öğrencilerin mantıklı problemleri çözmesi üzerinde Lego®'nun etkileri" adlı makalede bu konu hakkında detaylı bir bilgilendirme vermiştir.

Üçüncü bölümde ise projede 3B nesnelerin ve kesim platformunun nasıl tasarlandığı açıklanacaktır. 3B nesne tasarlama aracı olarak, bir açık kaynak, özgür yazılım olan Blender[4] yazılımı seçilmiştir. Bu bölümün son kısmında hangi dosya türünün 3B kesim platformu arabirimiyle etkileşimde olduğu hakkında detaylı bilgi verilecektir. Bu bölümde ayrıca projenin yazılım ayağı hakkında ayrıntılı bilgi verilecektir. "Hangi yazılım arabirimleri Lego Mindstorms® Brick (beyin) ile birlikte çalışabilir?" sorusuna cevap verilecek, bununla birlikte projede tercih edilen C# programlama dilinde yazılmış olan AForge.NET[2] arabirimi açıklanacaktır. Son olarak bu bölümde proje kodlarından örneklere yer verilecektir.

Son bölüm projede karşılaşılan zorluklar hakkında bilgilendirme yapılacak, bu zorlukları aşmak için hangi tekniklerin uygulandığı anlatılacak, projenin kapanış ve özeti bu bölümde yapılacaktır. Bu bölümün son kısmında ileriye dönük fikirler, çalışmalar ve üst seviye gerçeklemeler hakkında bilgi verilecektir.

1.3. Kısıtlamalar

Bu projede bazı kısıtlamalar aşağıda sıralanmıştır:

- İlk kısıtlama iç bükey nesnelerin üst yarılarını kesmekle alakalı olacaktır. 3B kesme platformu bir adet matkap ucuyla delme işlemi yapacağı için, tasarım planlamasına göre iç bükey nesnelerin üst yarı düzlemlerini kesemeyecektir. Kesim cihazı nesneleri yukarıdan aşağıda doğru keseceği için tek noktalı bir matkap ucu nesnelerin sadece dış bükey kısımlarını kesebilecektir.
- İkinci kısıtlama kesme mekanizması ile ilgili olacaktır. 3B kesim cihazı nokta-nokta kesim mekanizmasına ve standart serve motorlara sahip olduğu için bir nesnenin tam kesimi dikkate değer bir şekilde zaman alacaktır.
- Son kısıtlama ise kesilecek nesnenin boyutları ile alakalı olacaktır. 3B kesme platformu 4,8 cm * 6,8 cm'yi geçmeyecek şekilde nesneleri kesmek için tasarlanmıştır. Bu birimler Y ekseninin genişliği ve X ekseninin referans noktasını göz önünde alarak hesaplanmıştır. Bu boyut ileriye dönük farklı tasarımlarda genişletilebilir ve daha büyük nesnelerin kesilmesine olanak sağlayacak platformalar tasarlanabilir.

1.4. Amaçlar ve Projenin Yapısı

Projemizin bir temel hedefi ve birkaç alt-hedefi bulunmaktadır.

- Temel hedef 3B kesim platformunu Lego Mindstorms® robotik kiti ve gerekli program kaynakları kullanarak tasarlamak ve doğru çalışmasını sağlamaktır.
- İlk alt-hedef "Lego Mindstorms® nasıl çalışır?" sorusunda cevap vermek ve birkaç örnek tasarım gerçekleyerek hangi parçaların birbiriyle uyumlu olduğuna ve bu projede 3B kesim platformunun nasıl tasarlanabileceği hakkındaki sorulara cevap vermektir. Bu aşamada Lego Mindstorms® içerisinden çıkan NXT-G® yazılımı başlangıç seviyesinde programlama için yeterli olacaktır. Bu alt-amaç içerisinde Lego Mindstorms® kitinin içerisinden çıkan ve çıkmayan Lego® parçalarını da sabit platform gereği projeye dâhil etmeyi içermektedir.

- İkinci alt-hedefimiz 3B nesne tasarımını Blender yazılımını kullanarak tasarlamak, tasarlanan bu nesneleri koordinat düzlemindeki çıktılarını Lego Mindstorms® Brick parçasının anlayacağı şekle çevirebilmektir. Ayrıca alınan koordinat noktalarının normalizasyonu da bu alt-amacın içerisinde olacaktır.
- Üçüncü ve son alt-hedef NXT-G® programlama dilinden bir adım ileriye giderek; proje için kullanacağımız C# dili ile bütünleşme sağlamak olacaktır. Referans kütüphanelerini kullanarak Lego Mindstorms NXT®'nin Brick parçasıyla iletişim kurabilecek şekilde bir ara yüz tasarlamaktır. Ayrıca bu ara yüzün tasarlanan farklı 3B nesneler için kalibrasyon seçeneğini de içermesi öngörülmektedir.

Projenin son kısmı ise planlanan hedeflerle, çıkan sonuçların karşılaştırılması ve projenin yapımı esnasında karşılaşılan zorluklar üzerinde bir değerlendirme yapılıp, projenin ileriye dönük geleceği ile ilgili bilgi ve fikirleri içerecektir.

2. ALTYAPI

2.1 Robotik kit olarak Lego Mindstorms NXT 2.0®

2.1.1 Lego Mindstorms®'a giriş

Bu bölümde Lego Mindstorms 2.0® kitinin bir Lego® fikri olarak nasıl çalıştığı hakkında bilgi verilecektir. Lego Mindstorms NXT-G® ve robotik kite destek sağlayan API kütüphaneleri hakkında bilgi verilecektir. Son bölümde mühendislik problemi olarak 3B kesim platformu ele alınacak ve proje metodolojisi hakkında detaylı bilgi verilecektir.

2.1.2 Lego Mindstorms® nasıl çalışır?

Lego® farklı projeler, farklı amaçlar için küçük parça setleri üreten bir oyuncak firmasıdır. Bu setler ile küçük arabalardan büyük evlere kadar birçok oyuncağı yapmak mümkündür. Lego® aldığınız setin içinden çıkan parçalar ve parça birleştirme talimatları doğrultusunda o projeyi gerçeklemenizi mümkün kılar. Firma ilk olarak statik(durgun) nesneler üzerinde projeler geliştirdi ancak zamanla daha dinamik(hareketli) nesne tasarımları (dişli, motorlar, sensörler, hareket eden nesneler) üretmeye başladı. Günümüzde bunun en gözle görülür örneği Lego Mindstorms® araç-kiti serisidir.

3D kesim platformunu tasarlamak için bir Brick(beyin), üç eksen (x,y,z) için üç farklı motor yeterli olacaktır. Ayrıca z-eksenini kontrol etmek için bir sensöre ihtiyaç duyulacaktır. Proje planlamamız doğrultusunda bir adet Lego Mindstorms NXT 2.0® kiti projemiz için yeterli olacaktır. Bunların dışında Lego Mindstorms NXT 2.0® kitinin içinden çıkmayan ama platformun kararlılığı için gerekli bazı başka Lego® parçalarına ihtiyaç duyulacaktır.

Aşağıda Lego Mindstorms® kitinin çalışma prensipleri verilmiştir:

- Bir beyin kendisine bağlanan tüm motor ve sensörleri kontrol eder.
- Beyin, NXT-G yazılımı kullanarak veya 3. Parti frameworkler sayesinde programlanabilirdir.
- Bu beyin *Universal Serial Bus(USB)(kablolu)* veya *Bluetooth(kablosuz)* ile kontrol edilebilir.
  Ek-A içerisinde Lego Mindstorms NXT 2.0® kitinin temel parçalarının işlevleri hakkında detaylı bilgiyi bulabilirsiniz.

Lego® Robotik kitini bu kadar başarılı yapan ne? [7]

- Öncelikle, ucuz elektronik parçalar bu robotu karşılanabilir kılıyor.
- İkinci olarak, Lego®'nun beyni her zaman iyi bir mekanik örnekleme aracı olmuştur.
- Örnek olarak; Massachusetts Institute of Technology (MIT) araştırmacıları yıllardır Lego® beyin parçalarını ve küçük bilgisayarları ilginç yöntemlerle birbirine bağlayan projelere imza atıyorlar.

2.2. Robotik kiti görsel programlama

NXT kitinin yazılımı, National Instruments' LabVIEW[11] yazılımı temel alınarak oluşturulmuştur. Ayrıca, Microsoft Robotic Studio (MSRS) [10] adı altında *Microsoft* firması bütün robot çeşitlerini destekleyen bir platform geliştirmiştir. MSRS ayrıca görsel programlama ortamı olan Görsel Programlama Dili MVPL'yi geliştirmiştir. İki programlama ortamı da kullanıcı dostu, ana ekranda simgelerin sürükle ve bırak tekniğiyle çalışmasını sağlayan bir ara yüze sahiptir.

2.2.1 Lego'nun® NXT için sağlamış olduğu görsel programlama dili nedir?

Lego Mindstorms® kiti içerisinde NXT-G programlama yazılımı ya da diğer bir adıyla Labview for Lego Mindstorms yazılımı bulunur. Lego®, NXT kitin içindeki gömülü yazılımını açık kaynak projesi olarak yayınlamıştır. Bununla birlikte gayri resmi olarak diğer bazı firmalar tarafından çıkarılmış, beyni kontrol eden arayüzler vardır. Bunlar, *NBC*[12] (Next Byte Codes, asamble sözdizimine sahip basit bir dil), *NXC*[12] (Tam olarak C değil, C'nin bir alt dili NBC'nin geliştirmiş bir hali) ve *leJOS NXJ*[9] (Lego Mindstorms'un beyninin içerisindeki Java ile yazılmış temel dilin yerine kullanılabilecek bir dil). Bunların dışında Microsoft firması da son olarak Microsoft Robotic Developer Studio olan dördüncü sürümü yakınlarda piyasaya çıkarmıştır.

Lego Mindstorms® kitinin kendi NXT-G yazılımı bazı örneklemeler için çok kullanışlı bir yazılımdır. Başlangıç seviyesindeki tasarımlar için çok yardımcı küçük görevler bloklar sayesinde tasarlanmaktadır. Örnek kaynak kodlar Lego®'nun kendi web sitesinde mevcuttur. Bu sebepten dolayı bu yazılım motor ve sensör kontrol çalışma prensiplerini anlamak için iyi bir başlangıç olmaktadır. Ama program bazı kısıtlamalara sahiptir. Program, Lego Mindstorms® beyninde derlendiği için hafıza sıkıntısına yol açmaktadır. 3B nesnelerin koordinatlarının okunup motorlara iletilmesi için gereken çalışma süreci çok hantal işlemektedir. Ayrıca koordinatları alacağımız Wavefront (.obj) dosya türü de bu görsel programlama dili tarafından tanınmamaktadır.

2.3. Robotik kitin API Programlaması

2.3.1.Robotik kitin üreticisinin sağladığı çekirdek API

Bu bölümde projenin API kısmını anlatacak ve Lego Mindstorms® yazılımının tarihi hakkında giriş yapılacaktır. Bölümün bir kısımda çekirdek kütüphanelerini entegre etmek için kullanılan ara yüz hakkında bilgi verilecektir. *GhosAPI.dll* ve motorların komut setleri detaylı bir şekilde ele alınacaktır.

Proje boyunca, Lego Mindstorms® NXT beyin parçasıyla iletişim kurmak için, Lego® ve üçüncü parti birimlerin paylaştığı kütüphaneler yardımıyla sıfırdan bir Windows® masaüstü uygulaması yazıldı. Lego firması Mindstorms® kiti geliştiricileri ve kullanıcıları için "*GhostAPI.dll*" adında *C#, Visual Basic, C++* ve *Java* gibi farklı dillere yönelik yazılım geliştirme kiti (*SDK*) desteği sağlıyor. Lego® şimdiye kadar iki tane farklı Robotik kit seti piyasaya sürdü. Bunlardan *RCX,* Lego®'nun kısıtlı işlevlere sahip olan ilk robotik kitidir. Sonrasında piyasaya sürülen ikinci nesil kit ise daha çok özellik ve daha gelişmiş kapasiteye sahip olan *NXT*'dir. Lego®, bu kitler için Lego® derlenmiş komut setleri (*LASM*) adı altında komutlar içeren bir özel derleme dili geliştirdi. Bu komut setleri araç üstü programlar tarafından

çevrilip çalıştırılabiliyor ve beyin temel parçasına bir bir işleniyordu. Lego®'nun beyin kaynaklı uygulamalarının iki modu aşağıda özetlenmiştir.

- Mod 1: Programın çalışması için bütün LASM bit kodları derlenip Lego® beyin parçasında indiriliyor. Bu modda, komutlar beyine bütün bir program olarak gönderiliyor ve beynin işlemcisi tarafından bağımsız olarak çalıştırılıyor. Projenin görsel NXT programı kullanarak gerçeklenen örnekleme aşamasında detaylı olarak bu mod kullanıldı. Bu modda kitin beyin parçasının sanal makinesi bir program çalıştırırken, bellekte yer alan ".RXE" formatındaki şifrelenmiş dosyaları okuyarak, 32kB'lık RAM(Rastgele erişimli bellek) havuzunda programa yer ayırır. Bu havuzun ilklenmesinden sonra, program aktif ve çalışmaya hazır hale gelir. Çalışma esnasında, programın ana komutları ya bu RAM havuzunda takas yapar ya da RAM'de saklanan değerleri göz önüne alarak Girdi/Çıktı işlemi gerçekler.
- Mod 2: Her bir LASM bit kodu beyine kişisel bilgisayar tarafından USB portu ya da Bluetooth teknolojisi üzerinden istek olarak gönderilir. Programın temel mantığı, komutlar ana bilgisayarda kalır ve tek bir komut olarak gerçeklenmesi beklenen işlev beyin parçasına gönderilir. 3D kesim platformunun ara yüz program desteği bu modu kullanıyor. NXT gömülü yazılımı Bluetooth iletişimi için efendi/köle(master/slave) ilişkisini kullanan seri bağlantı noktasını kullanıyor. Herhangi bir direk komut NXT köle gömülü yazılımı tarafından çevrilip, işlevler haline dönüştürülüyor. Bu durumda, ana bilgisayardaki komutlar, NXT beyin parçasına Bluetooth paketleri olarak gönderiliyor.

İki mod için de, kişisel bilgisayar, Lego® beyin LASM komutlarını derleyip işliyor. Bu derlenme ve işleme aşamalarını kolaylaştırmak için Lego® Win32 DLL'leri (*Dinamik bağlantı kütüphaneler*) tasarlandı. Temel DLL, "*GhostAPI.dll*" LASM komutlarını sıraya koymak ve işlemek için alt seviye bir işlevlik gösteriyor. Bu DLL RS-232 iletişimi için "*PbkCOMM32.dll*" ve USB iletişimi için ise "*PbkUsbPort.dll*" adlı iki tane alt DLL'yi kullanıyor. Bu projede *Microsoft.NET uyumlu çalışabilme katmanında*, beyin ile iletişim için üçüncü parti bir arabirim olan "*AForge.NET*" arabirimini kullanılıyor. Bu kısımda iletişimin C# sınıfları tarafından nasıl yapıldığı ve yerleşik kod çağrılarının nasıl yönetildiği anlatılacaktır. Örnek olarak Bluetooth iletişim teknolojisi için, *Microsoft .NET*'in kendi "*System.dll"* (System.IO.Ports isim uzayı) kütüphanesi tarafından sağlanan bazı sınıflar kullanılır. "*SerialPort*" sınıfı ve "*StopBits*", "Parity" sayıcıları gibi.

2.3.2 Çekirdek API için Üçüncü parti arabirim

AForge.Net[2] frameworkü bilgisayar mühendisliği, robotik, bilgisayar vizyonu ve yapay zekâ gibi alanlarda çalışmak için tasarlanmış açık kaynaklı bir API'dir. Tüm framework Visual Studio çözümlerinden bağımsız olarak C# dilinde yazılmıştır. Bütün çözümler belirli bilgisayar mühendisliği konularında özelleşmiş örnek kodları içerirler. Bütün bu kodlar çekirdek kütüphane olan "*AForge.dll*" DLL'sini kullanırlar. Bu DLL çekirdek veri türlerini ve bazı *Thread* sınıflarını içerir. Bu çekirdek DLL'nin tepesinde, robotik ile alakalı veri türleri ve ara yüzlerin bulunduğu başka bir DLL olan "*AForge.Robotics.Lego.DLL*" kütüphanesi bulunur. Bu DLL NXTBrick ve RCXBrick olmak üzere iki tane ara sınıfı içerir.

"AForge.Robotics.Lego.DLL" Lego®'nun Win32 "GhostAPI.dll" ara sınıfı olan "*GhostAPI*"'yi içerir. Bu, C#'ın *"DllImport"* özniteliğini *"extern"* anahtarıyla yerleşik DLL ile bağlantı kurmak için kullanır. Destek sağlanan "*GhostAPI.dll*" işlevleri aşağıdadır:

1. **GhCreateStack:** İletişim yığıtı oluşturur.
2. **GhSelectFirstDevice:** İlk uygun aracı bulur.
3. **GhOpen:** Seçilen aracı kullanıma açar.
4. **GhClose:** Seçilen aracı kullanıma kapatır.
5. **GhSetInterleave:** "*interlave*" komut setlerini çalıştırır ve indirme kuyruğuna ekler.
6. **GhSetWaitMode:** Güncel bildirim durumunu bekleme durumuna alır.
7. **GhCreateCommandQueue:** Komut kuyruğu oluşturur, dönüş değerini tutar.
8. **GhDestroCommandQueue:** Komut kuyruğunu serbest bırakır.
9. **GhAppendCommand:** Verilen komutu, komut kuyruğuna ekler.
10. **GhExecute:** Verilen komutu, kuyrukta onaylar ve çalıştırır.
11. **GhGetFirstCommand:** Kuyruğun ilk komutunu alır.
12. **GhGetCommandReplyLen:** Komutun cevap uzunluğunu alır.
13. **GhGetCommandReply:** Komutun cevabını alır.

Tipik bir "*GhGreateStack*" işlevi aşağıdadır. Diğer metodlar benzer şekilde çağırılır.

```
[DllImport ("GhostAPI.dll")]
        public extern static uint uint GhCreateStack(
            string port,
            string protocol,
            string session,
            out IntPtr IntPtr stack );
```

NXT bit kodu "*AForge.Robotics.Lego.DLL*" içerisinde C# sayacı olarak "*NXTCommand*" sınıfı içerisinde tanımlanır. Lego Mindstorms® NXT beyin tarafından desteklenen komut tipleri için bir tane, sistem komutları için bir tane ve direk komutlar için de bir tane olmak üzere toplamda üç tane sayıcı tanımlanır.

Komut Tipleri:

```
internal enum NXTCommandType
    {
        /// <summary>
        /// Direct command, which requires reply.
        /// </summary>
        DirectCommand = 0x00,

        /// <summary>
        /// System command, which requires reply.
        /// </summary>
        SystemCommand = 0x01,

        /// <summary>
        /// Reply command received from NXT brick.
        /// </summary>
        ReplyCommand = 0x02,

        /// <summary>
        /// Direct command, which does not require reply.
        /// </summary>
        DirectCommandWithoutReply = 0x80,
```

```
        /// <summary>
        /// System command, which does not require reply.
        /// </summary>
        SystemCommandWithoutReply = 0x81
    }
```

AForge.NET tarafından desteklenen NXT Sistem Komut Tipleri: AForge.Net alt komut olarak aşağıdaki sistem komutlarını geliştirmiştir.

```
internal enum NXTSystemCommand
    {
        /// <summary>
        /// Get firmware version of NXT brick.
        /// </summary>
        GetFirmwareVersion = 0x88,

        /// <summary>
        /// Set NXT brick name.
        /// </summary>
        SetBrickName = 0x98,

        /// <summary>
        /// Get device information.
        /// </summary>
        GetDeviceInfo = 0x9B
    }
```

AForge.NET tarafından desteklenen NXT Direk Komut Tipleri: AForge.Net alt komut olarak aşağıdaki direk komutlarını geliştirmiştir.

```
  internal enum NXTDirectCommand
    {
        /// <summary>
        /// Keep NXT brick alive.
        /// </summary>
        KeepAlive = 0x0D,

        /// <summary>
        /// Play tone of specified frequency.
        /// </summary>
        PlayTone = 0x03,

        /// <summary>
        /// Get battery level.
        /// </summary>
        GetBatteryLevel = 0x0B,

        /// <summary>
        /// Set output state.
        /// </summary>
        SetOutputState = 0x04,

        /// <summary>
        /// Get output state.
        /// </summary>
        GetOutputState = 0x06,

        /// <summary>
        /// Reset motor position.
        /// </summary>
        ResetMotorPosition = 0x0A,

        /// <summary>
        /// Set input mode.
        /// </summary>
        SetInputMode = 0x05,

        /// <summary>
        /// Get input values.
```

```
        /// </summary>
        GetInputValues = 0x07,

        /// <summary>
        /// Get status of the Low Speed bus.
        /// </summary>
        LsGetStatus = 0x0E,

        /// <summary>
        /// Write to the Low Speed bus.
        /// </summary>
        LsWrite = 0x0F,

        /// <summary>
        /// Read from the Low Speed bus.
        /// </summary>
        LsRead = 0x10,

        /// <summary>
        /// Reset input scaled value.
        /// </summary>
        ResetInputScaledValue = 0x08
    }
```

"*NXTBrick*" sınıfı içerisinde tanımlanan üst seviye fonksiyonlar, NXT Brick ile iletişim kurmak için alt seviye komutları kullanır. Gerçekleme esnasında bir üst seviye fonksiyon için, sistem ve direk komutların (cevaplı veya cevapsız) bir kombinasyonu kullanılmıştır. Beyine Bluetooth üzerinden komut yollamak için, "*NXTBrick*" sınıfı içerisindeki "*SendCommand*" işlevi kullanılmıştır. "*NXTBrick*" sınıfı içerisinde bulunan private "*communicationInterface*" üyesi "*INXTCommunicationInteface*" tipini ve "*GetMessage*", "*SendMessage*", "*Connect*", "*Disconnect*" gibi alt seviye iletişim metotlarını barındırır.

```
public bool SendCommand( byte[] command, byte[] reply )
        {
            bool result = false;

            lock ( sync )
            {
                // check connection
                if ( communicationInterface == null )
                {
                    throw new NullReferenceException( "Not connected to NXT brick" );
                }

                // send message to NXT brick
                if ( communicationInterface.SendMessage( command, command.Length ) )
                {
                    // notifies clients if any
                    if ( MessageSent != null )
                    {
                        MessageSent( this, new CommunicationBufferEventArgs( command )
);
                    }

                    if ( ( command[0] == (byte)
NXTCommandType.DirectCommandWithoutReply ) ||
                         ( command[1] == (byte)
NXTCommandType.SystemCommandWithoutReply ) )
                    {
                        result = true;
                    }
                    else
                    {
                        int bytesRead;
                        // read message
                        if ( communicationInterface.ReadMessage( reply, out bytesRead
) )
                        {
                            // notifies clients if any
                            if ( MessageRead != null )
```

```
                            {
                                MessageRead( this, new CommunicationBufferEventArgs(
reply, 0, bytesRead ) );
                            }

                            // check that reply corresponds to command
                            if ( reply[1] != command[1] )
                                throw new ApplicationException( "Reply does not
correspond to command" );

                            // check for errors
                            if ( reply[2] != 0 )
                            {
                                if ( reply[2] == 221 )
                                {
                                    throw new ApplicationException( "It seems that a
wrong sensor type is connected to the corresponding port" );
                                }
                                else
                                {
                                    throw new ApplicationException( "Error occurred in
NXT brick. Error code: " + reply[2].ToString( ) );
                                }
                            }

                            result = true;
                        }
                    }
                }
            }
            return result;
        }
```

Lego Mindstorms® motorlarını kontrol etmek ve uygun NXT sensörlerinden veri okumak için birçok üst seviye işleve ihtiyaç duyulur. Bu işlevler "*NXTBrick*" ara sınıfı içerisinde tanımlanmıştır. Örnek olarak; beyin ile bağlantı kurmak/koparmak, ya da robotun içerisindeki motorları kontrol etmek için AForge.NET tarafından üst seviye işlevler tasarlanmıştır.

"Connect" İşlevi:

```
public bool Connect( string portName )
        {
            lock ( sync )
            {
                if ( communicationInterface != null )
                    return true;

                // create communication interface,
                communicationInterface = new SerialCommunication( portName );
                // connect and check if NXT is alive
                if ( ( communicationInterface.Connect( ) ) && ( IsAlive( ) ) )
                    return true;

                Disconnect( );
            }
            return false;
        }
```

"Disconnect" İşlevi:

```
  public void Disconnect( )
        {
            lock ( sync )
            {
                if ( communicationInterface != null )
                {
                    communicationInterface.Disconnect( );
                    communicationInterface = null;
                }
```

```
            }
        }
```

“SetMotorState” İşlevi:

```
public bool SetMotorState( Motor motor, MotorState state, bool waitReply )
        {
            byte[] command = new byte[12];

            // prepare message
            command[0] = (byte) ( ( waitReply ) ? NXTCommandType.DirectCommand :
NXTCommandType.DirectCommandWithoutReply );
            command[1] = (byte) NXTDirectCommand.SetOutputState;
            command[2] = (byte) motor;
            command[3] = (byte) state.Power;
            command[4] = (byte) state.Mode;
            command[5] = (byte) state.Regulation;
            command[6] = (byte) state.TurnRatio;
            command[7] = (byte) state.RunState;
            // tacho limit
            command[8]  = (byte) ( state.TachoLimit & 0xFF );
            command[9]  = (byte) ( ( state.TachoLimit >> 8 ) & 0xFF );
            command[10] = (byte) ( ( state.TachoLimit >> 16 ) & 0xFF );
            command[11] = (byte) ( ( state.TachoLimit >> 24 ) & 0xFF );

            return SendCommand( command, new byte[3] );
        }
```

Üstteki kod parçasında “*CommunicationBufferEventArgs*” etkinlik argümanı “*AForge.DLL*” içinde, “*NullReferenceException*” ve “*ApplicationException*” adlı iki istisna durum da Microsoft.NET “*mscorelib.dll*” içerisinde tanımlanmıştır. “*SerialCommunication*”, “*AForge.Robotics.Lego.DLL*” içerisindeki bir başka sınıftır.

Proje için yazılan 3B kesim platformu uygulaması “*NXTBrick*” sınıfının bir nesnesini oluşturacak ve onun “*Connect*”, “*SetMotorState*”, “*GetMotorState*” ve “*GetSensorValue*” vb. gibi public metotlarını kullanacaktır.

2.4. Mühendislik Projesi Olarak 3B Kesim Platformu seçmek

Mühendisliğin birçok alanında, 3B nesne tasarımı her zaman zorlu bir problem olmuştur. 3B çizim yapabilmek ve bu çizimi kesecek bir makineyi üretmek hem yazılım hem de donanım yeteneğini gerektirir. CNC (Bilgisayar Sayısal Denetim) makinesi icat edilmeden önce, 3B kesim süreci elle matkaplar vasıtasıyla yapılıyordu. Bugünlerde CNC tezgâhları, otomatikleştirilmiş ve insan hatasından bağımsız hale getirilmiş en önemli kesim cihazları olarak yerini korumaktadır. CNC tezgâhlarını tasarlamak için ciddi bir bilgi ve birikim gerekiyor. Bu sebepten ötürü 3B kesim makinesi uygun bulundu. Karmaşık ve bir o kadar da büyüleyici bu mühendislik eserini tasarlamak için, proje içerisinde donanım kısmında Lego Mindstorms NXT 2.0® robotik kiti ve yazılım kısmı için API temelli programlama esas alındı.

2.5. Metodoloji

2.5.1.Proje Yönetim Metodolojisi

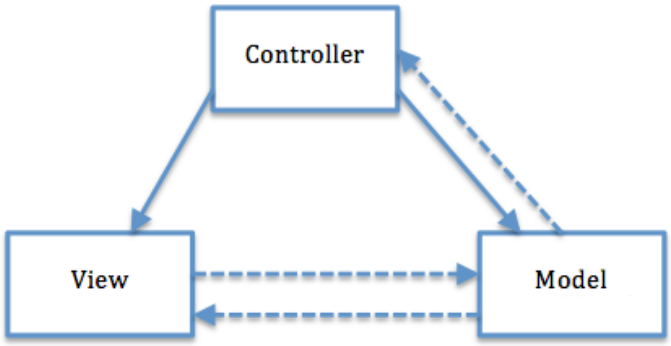
Bu proje boyunca tekrarlanan proje yönetim metodolojisi kullanıldı. Yani proje boyunca, aylık düzen göz önüne alınarak her ayın ortasında aylık raporlar oluşturulup projenin bir sonraki ayının planlaması çıkarıldı. Çıkartılan planlama esas alınarak

proje takip edildi. Projede yazılım/donanım işbirliği ve belgeleme süreci olabildiğince birlikte götürülmeye çalışıldı.

2.5.2. Geliştirme Metodolojisi ve Mimari

Bu proje boyunca teste dayalı programlama metodolojisi takip edildi. Test durumları nihai kodu yazmadan önce gerçeklendi. Özellikle kalibrasyon aşamasında 3B kesim matkabının platforma zarar vermemesi için farklı test koşulları geliştirildi.

.NET frameworkü bize Model-View-Controller(MVC) **(bkz. Şekil-1)** yapısını kullanarak yazılım geliştirme imkânı verdi. MVC tasarım yolları programı üç temel katmana ayırıyor. Bu farklı katmanlar yazılımın farklı problemlerini ele aldı; örnek olarak sunum, veri yönlendirmesi ve iş mantığı birbirinden farklı olarak tasarlandı. Bu yazılım mimarisini daha basit hale getirmeye yaradı. Modern uygulamalarda, bu yol uygulamanın mantıksal yapısını kurmak için kullanılan temel bir yoldur. Bu sebepten dolayı birden fazla denetleyici aynı ara yüz için yazılabilir. Özellikle test bazlı geliştirme mimarisinde MVC mimarisi birbirinden bağımsız test durumlarını yazmak için önemlidir.



Şekil-1 : Model-view-controller düzeni

3. PROBLEMİN ÇÖZÜMÜ

Bu bölümde 3B kesim platformunun tasarlanması aşamasındaki prototip ve 3B nesne örnekleri anlatılacaktır. Ayrıca NXT-G yazılımı ile kesme hareketini gerçekleyen bloklar bu bölümde tasarlanacaktır ve sonrasında proje boyunca tasarlanan API kaynak kodları verilecektir. Bölümün son kısmında 3B kesme işlevini gerçekleyecek ara yüz hakkında detaylı bilgi verilecek ve test kodu gerçeklenecektir.

3.1 3B Kesim Platformu Tasarımı

3.1.1 Prototip Tasarlanması

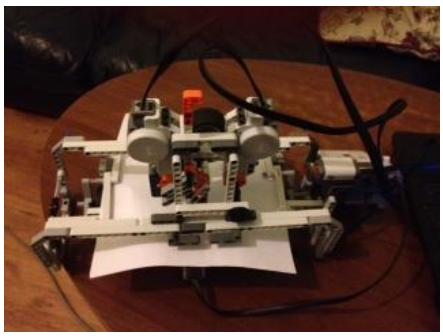
3B kesim platformu tasarlanırken, birden fazla seçenek mevcuttur. Özellikle z-ekseninin entegrasyonu aşamasında platformun düzeni çok önem kazanıyor. Lego Mindstorms NXT 2.0® destek kitabındaki bazı prototipler proje esnasında 3B kesim platformunu tasarlamak için gerçeklendi. Bu bölümde, farklı prototip örneklerinin 3B kesim platformunu tasarlarkenki katkısı detaylıca ele alınacak.

3.1.1.1 İlk Prototip: Yazıcı

Tasarım örneği "*The Unofficial Lego Mindstorms NXT 2.0 Inventors Guide*"[14] kitabın 16. Bölümünde bulunan "*The Printer*" talimatları doğrultusuna gerçeklenmiştir. (*Resim 2.1, Resim 2.2*)

Bu prototip neden tasarlandı?
Bu prototipin tasarlanma mantığında Lego® kitinde hareket eden parçaların farklı yönlerdeki birbiriyle uyumu hakkında bilgi vermesi için tasarlandı

Resim 2.1-Yazıcı prototipi ön kısım



Resim 2.2-Yazıcı prototipi arka kısım

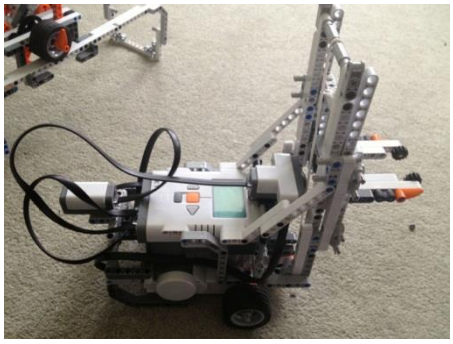
3.1.1.2. İkinci Prototip : Forklift (Çatallı Kaldırıcı)

Talimatlar www.nxtprograms.com[13] adresinden alındı.

Bu prototip neden tasarlandı?

Bu prototip iki eksen (x-ekseni ve y-ekseni) hakkında projemize bilgi versin diye tasarlandı. (*Resim 2.3, Resim 2.4*). Bu prototip motorların farklı yöndeki hassas

uyumu üzerine odaklıdır. Lego Mindstorms NXT 2.0® görsel programlama platformundaki blok tasarımları için bu prototipin çok katkısı olmuştur.



Resim 2.3-Forklift prototip sağ kısım

Resim 2.4-Forklift prototip üst kısım

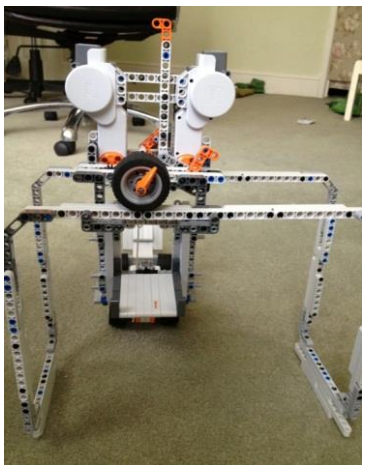
3.1.1.3. Üçüncü Prototip: Birleştirilmiş Prototip

Bu prototip üç farklı eksenin (x, y, z) önceki prototipler göz önünde bulundurularak birleştirilmiş halidir. Aşağıdaki resimlerde görüldüğü üzere*(Resim 2.5,Resim 2.6),* platform y ekseni üzerinde hareket ederken, araç x ve z eksenlerinde hareket etmek üzere tasarlanmıştır.

Bu prototip neden tasarlandı?

Bu prototipi tasarlamak istenmesinin amacı üç ekseni birlikte hareket ettirecek bir düzenek tasarlamaktı ama yine fotoğraflarda görüldüğü üzere *(Resim 2.5,Resim 2.6),* düzenek sabit değildi. Bu yüzden daha sağlam bir düzenek arayışına girildi.

Resim 2.5-Birleştirilmiş prototip ön kısım



Resim 2.6-Birleştirilmiş prototip arka kısım

3.1.2. 3B kesim platformu prototipi

Belirli amaçlar çerçevesinde tasarlanan üç prototip, hangi parçaların birbiriyle uyuştuğu ve koordinat düzleminde bu parçaların uyumu hakkında bilgi verdi. Ancak bu platformlar proje için önemli olan kararlılık problemini bir türlü çözemedi. Bu yüzden daha kararlı Lego® parçaları sipariş edildi. Ulaşılan farklı kaynaklar sonucunda, Lego®'nun birden fazla statik ve sağlam parçaya sahip olduğu fark edildi.

Ayrıca, tasarım aşamasında platformun üzerinde üç boyutlu hareket için ve yuvarlak tekerleklerin dönebilmesi için gereken dişli parçalar sipariş verildi. Bu dişliler hem kararlılık sıkıntısını çözebilmek adına hem de her bir boyuttaki hareket için önemli bir sıkıntıyı çözmeye yardımcı oldular.

Bu görev tanımının en zorlayıcı kısmı platform tasarlamak için gerekli olan hangi parçanın eksik olduğunu projenin nasıl bir platform olacağını görmeden, herhangi bir talimat veya şema(ör. oyuncak araba kılavuz şeması) yardımı olmadan ortaya bir prototip çıkartmak oldu. Bu zorluğu şöyle bir örnekle açıklayabiliriz: Örneğin bir yap-boz yapacaksınız ancak sonunda neye benzeyeceğine dair elinizde hiçbir şey yok, tamamen hayal gücü ile yapmak zorundasınız, işte bu projede tasarlanan platform da bu şekilde tasarlandı.

Sıkı ve tutarlı bir zaman yönetimi sonucunda dördüncü ve kararlı prototip tasarlandı. (*Resim 2.7, Resim 2.8*) Prototipin detaylı olarak fotoğraflarını **Ek-B** içerisinde görebilirsiniz

Resim 2.7-3B kesim makinesi prototip ön kısım

Resim 2.8-3B kesim makinesi prototip arka kısım

3.2. 3B Nesnelerin Tasarlanması
3B nesnelerin Blender yazılımıyla tasarlanması

3B kesim platformunun tasarımından sonra, bu platform üzerinde kesim yapabilmek için 3B nesne tasarımına ihtiyaç duyuldu. Araştırmalarımız sonucunda nesne tasarımı için en uygun programın Blender olduğuna karar verildi.

Blender seçim nedenleri:

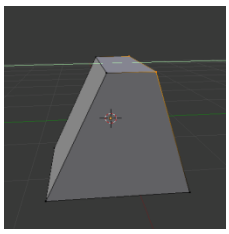
1. Özgür yazılım.
2. Birden fazla işletim sistemi desteği mevcut.(Linux, Mac Os X, Windows)
3. Birçok eğitim videosu ve kaynağı mevcut.
4. 3B nesneleri Wavefront(.obj) dosya türünde çıktı verebiliyor. (Wavefront(.obj) dosya türü C# üzerinde koordinat okumak için kullanılan bir dosya türüdür.)

3.2.1. İlk Tasarım – Piramit
Neden Piramit Tasarımı?

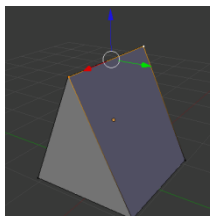
Blender uygulamasına giriş sürecinde, küp ve küreye kıyasıyla daha detaylı bir örneklemeye ihtiyaç duyuldu. Ayrıca 3B kesim platformu tasarım detayı doğrultusunda yukarıdan aşağıya doğru bir kesim için, Piramit örneği tasarlandı.

Tasarım Aşaması:

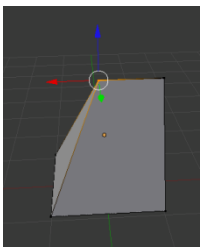
1. X ekseni tepesinde karşılıklı iki nota seçilip, Y-ekseni üzerinde kaydırıldıktan sonra üç açılı prizmalar elde edildi. *(Resim 3.1, Resim 3.2)*
2. Tümleşik prizmalar y-ekseni üzerinde kaydırıldı. *(Resim 3.3)*
3. Piramit hazır hale geldi. *(Resim 3.4, Resim 3.5)*



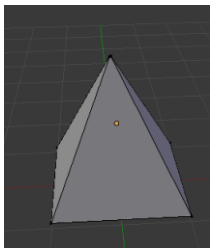
Resim 3.1-Piramit_1



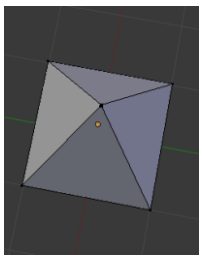
Resim 3.2-Piramit_2



Resim 3.3-Piramit_3



Resim 3.4-Piramit_4



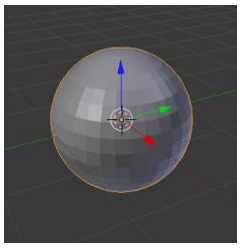
Resim 3.5-Piramit_5

3.2.2. İkinci Tasarım – Yarım Küre
Neden Yarım Küre?

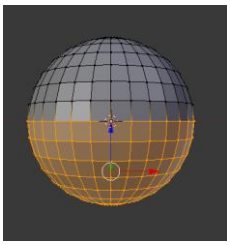
Yarım küre örneği Blender uygulamasında tasarlaması kolay ama 3B kesim platformunda kesimi zor olduğu için tasarlandı. Bununla birlikte 3B kesim platform dizaynı gereği üst yarı düzlemde kalan kısmı kesilmek üzere tasarlandı.

Tasarım Aşaması:

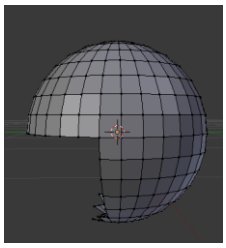
1. UV Küre seçeneği düzleme eklendi. (*Resim 3.6*)
2. Düzenleme modu içerisinde kürenin alt yarı düzlemde kalan bütün yüzleri seçilerek kesildi.(*Resim 3.7, Resim 3.8*)
3. Yarım Küre hazır hale geldi. (*Resim 3.9*)



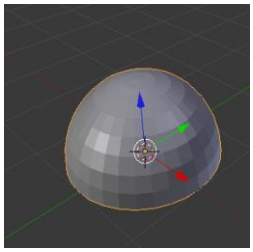
Resim 3.6-Küre_1



Resim 3.7-Küre_2



Resim 3.8-Küre_3



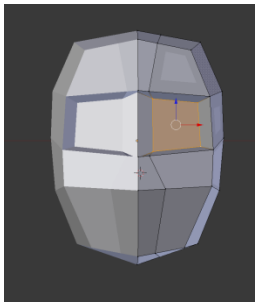
Resim 3.9-Küre_4

3.2.3. Üçüncü Tasarım – Karakter Yüz
Neden Karakter Yüz?

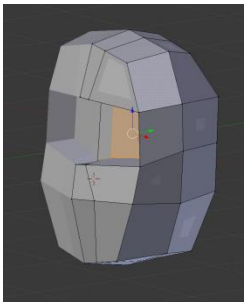
Bu nesneyi Blender uygulamasında tasarlamak ve 3B kesim platformunda kesmek zor olduğu için tasarlandı.

Tasarım Aşaması:

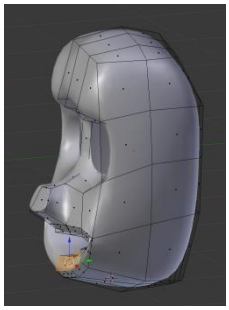
1. Örnek bir küpe kıvrımlar verilip, kafa şekline benzetilmeye çalışıldı. Z ekseni üzerindeki yüzün yarısı kesilip ayna düzenleyicisi ile daha iyi şekil verdirildi.
2. Gözler modelin içinde kesildi. (*Resim 3.10, Resim 3.11*)
3. Burun modelin içinde kesildi.
4. Dudaklar modelin içinde kesildi.
5. Kafanın şekli yumuşak hatlara sahip olmadığı için sol menüden smooth seçeneğiyle kafa şekli düzeltildi.(*Resim 3.12, Resim 3.13*)
6. Projedeki sınırlama yüzünden nesne y-ekseni üzerinde yatırılıp merkez y noktasında iki parçaya bölündü alt kısım atıldı. (*Resim 3.14, Resim 3.15, Resim 3.16)*



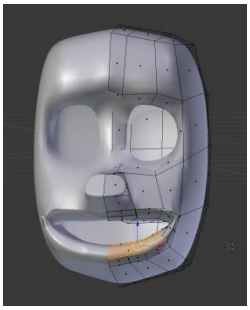
Resim 3.10-Yüz_1



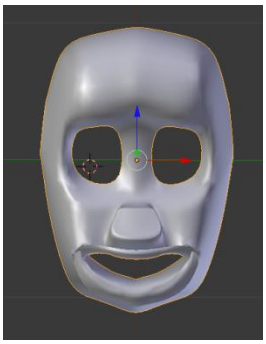
Resim 3.11-Yüz_2



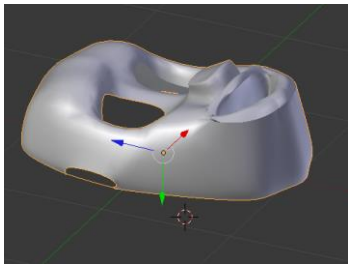
Resim 3.12-Yüz_3



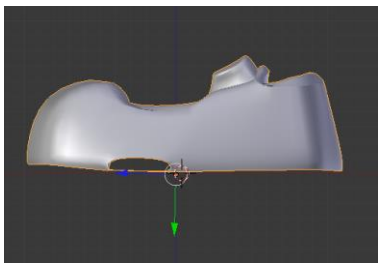
Resim 3.13-Yüz_4



Resim 3.14-Yüz_5



Resim 3.15-Yüz_6



Resim 3.16-Yüz_7

3.3. 3B Kesme Platformunu Görsel Olarak Kontrol Etmek

3.3.1. NXT-G kesim blokları tasarlanması

NXT-G yazılımı kendi içinde basit programlama bloklarıyla birlikte geliyor. Bu bölümde basit nesne kesimi için motor hareketlerini içiren programlama blokları hakkında bilgi verilecektir. Kesme işlemi için örnek olarak görsel programlamada kurulan bir motor eylemi beş kere tekrarlandı, bununla birlikte aynı işlem daha karmaşık bir problem için karmaşık tekrarlama işlevleri gerçeklenerek tasarlanabilir. İki tane (farklı motorlar) programlama bloğu ara yüzün basit olmasını sağlamak için tasarlandı.

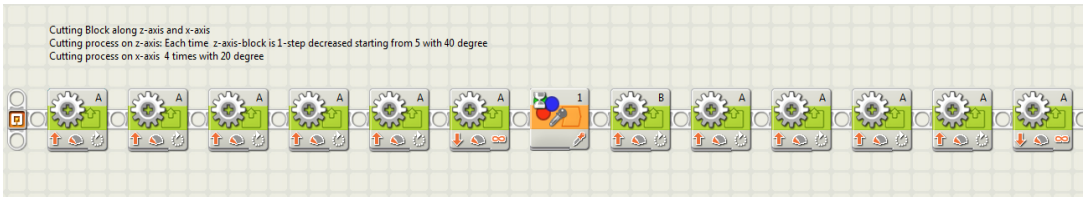
NXT-G yazılımının çok fazla hata ile birlikte karmaşık işlevler program bloğuna verildiğinde çökmeler olduğu gözlemlendi. Bir programlama bloğunun içerisindeki fazla motor hareketleri çoğunlukla çökmelere neden oluyor. Bu yüzden API kullanımı projemiz için daha kararlı olacaktır. Görsel programlamada, problem bloğu oluşturulduktan sonra, beyin içerisinde derlenmeden önce beyin hafızasına indiriliyor. Bu yüzen beyine karmaşık programlama blokları göndermek söz konusu olduğunda hafızamız doluyor ve kullanılamaz hale geliyor.

Programlama bloklarının detayları aşağıdadır:

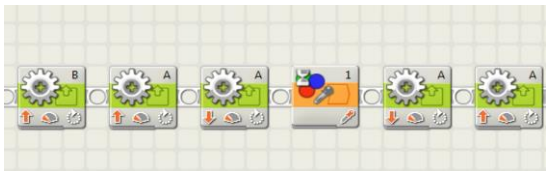
1. **Cut_Process bloğu:** Bu blok z-ekseni ve x-ekseni üzerinde beş defa kesim gerçekliyor. Her bir kesim sürecinde z-ekseni üzerinde bir adım azaltma yapılıp yarım üçgen bloklar elde ediliyor. Detaylı olarak *Resim 3.17'de* çalışan motorlar ve Renk sensörünün bir bloktaki uyumu gözlenebilir.
    - Birinci A-Motorunun detayı Resim 3.18,
    - İkinci A-Motorunun detayı Resim 3.19,
    - B-Motorunun detayı Resim 3.20,
    - Renk Sensörünün detayı Resim 3.21'de görülebilir.

Not! : NXT-G programında çalışırken önemli bir yazılım detayı gözlendi. Bir blok durduktan sonra, devam edebilmek için ikinci bir parametreye ihtiyaç duyuyor, eğer farklı bir blokla devam ediyorsa sorun oluyor. Bu yüzden *Cut_Process* bloğunun son işlevinden sonra, bazı motor hareketleri bloğun sonuna eklenmeli (*Resim 3.22*)

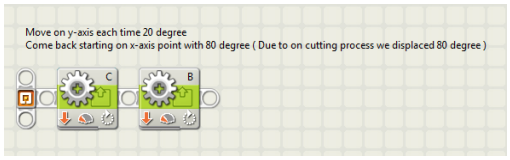
2. **Come_0:** Bu blok y-ekseninde bir hamle ileri adım atmayı ve sonra x-ekseninde başlangıç noktasına gelmeyi sağlıyor*(Resim 3.23)*
    - C-motorunun detayları Resim 3.24
    - B-motorunun detayları Resim 3.25'de görülebilir.
3. **Simple_ex:** Bu blok Cut_process bloğunun ve Come_0 bloğunu bir döngü içerisinde beş sefer gerçekliyor.



Resim 3.17- Cut_Process programlama bloğu



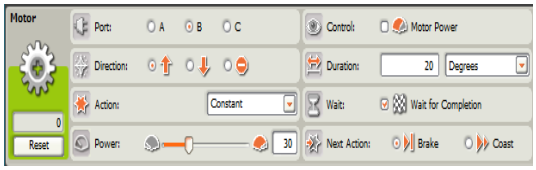
Resim 3.22- Son 2 komut önemli!



Resim 3.23- Come_0 programlama bloğu

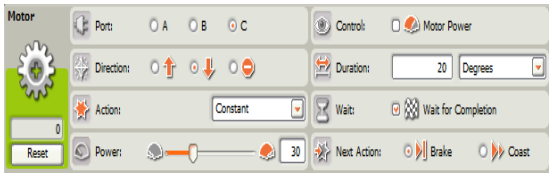
Resim 3.18- Birinci A Motoru

Resim 3.19- İkinci A Motoru

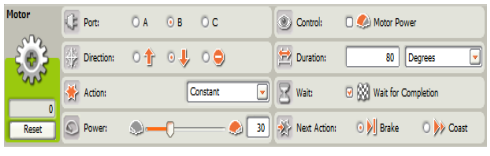


Resim 3.20 B Motoru

Resim 3.21- Renk Sensörü



Resim 3.24-Come_0 C motoru



Resim 3.25-Come_0 B motoru

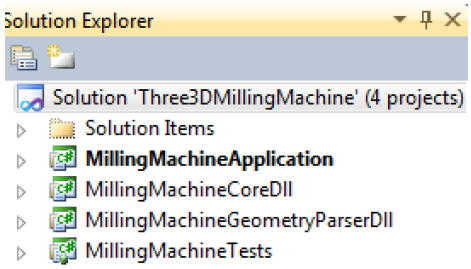
Resimlerde gösterilen motor detayları aşağıda açıklanmıştır:

1) **Port**: Bu kısım NXT-Brick parçasının hangi portuna hangi motorun bağlandığını belirtir.
2) **Direction**: Motorun yönünü belirtir
3) **Action**: Motorun nasıl hareket etmesi gerektiğini belirtir. Üç farklı modu bulunur. Bunlar *C*onstant(Sabit), Ramp Up(Rampa çık), Ramp Down(Rampa in). Bu modlar *Duration* kısmı *unlimited(sınırsız)* olmadığında çalışır.
4) **Power**: Bu bölüm Motorun gücünü ayarlamak için kullanılır.
5) **Control**: Bu kısım serve motor kontrolünü güçle kontrol etmek için kullanılır.
6) **Duration**: Bu kısım farklı motor hareketleri için geçerlidir. Dört farklı seçenek mevcuttur: *Unlimited*(sınırsız), *Degrees*(açılarla), *Second*(ikinci motora bağlı olarak) ve *Default*(öntanımlı).
7) **Wait**: Bu seçenek bir sonraki adımın bitişinin kontrolü için kullanılır.
8) **Next Action:** Bu kısım bir sonraki motor hareketine göre bir eylem yapmayı belirtir. Eğer *Brake*(Dur) seçeneği seçilmişse motor duraklar, eğer *Coast*(Devam Et) seçeneği aktifse motor herhangi bir duraklama süresi olmadan hareketine devam eder.

3.4. API kullanarak 3B Kesim Platformunu kontrol etmek

3.4.1. 3B Kesim platform Uygulaması

3B kesim platformunu uygulamak için Microsoft WPF(Windows Presentation Foundation) forum oluşturma seçeneği kullanıldı. Projemiz bir adet Windows Uygulaması, üç adet Dinamik Bağlantı Kütüphanesi (DLL) içeriyor. *Şekil-2*



Şekil-2 – 3B Kesim Platformu Çözüm Tablosu

3.4.1.1 MillingMachineGeometryParselDLL

Bu dinamik kütüphane “Wavefront (.obj)” dosyasını okumak için oluşturuldu. Bu dosya Blender yazılımının dışarı aktar seçenekleri arasında bulunan bir dosya türüdür. “Wavefront (.obj)” dosyası projede satır satır ayrıştırma yapmak için tasarlanmış bir düz metin dosyasıdır. Ayrıştırma işlemi dosyadaki geometri türünü temel alarak bu işlemi gerçekleştirir. “Wavefront (.obj)” dosyası aşağıda belirtilen yapıya sahiptir:

```
# List of Vertices, with (x,y,z[,w]) coordinates, w is optional and defaults to 1.0.
  v 0.123 0.234 0.345 1.0
  v ...
  ...

  # Texture coordinates, in (u[,v][,w]) coordinates, v and w are optional and default
to 0.
  vt 0.500 -1.352 [0.234]
  vt ...
  ...

  # Normals in (x,y,z) form; normals might not be unit.
  vn 0.707 0.000 0.707
  vn ...
  ...

  # Face Definitions
  f 1 2 3
  f 3/1 4/2 5/3
  f 6/4/1 3/5/3 7/6/5
  f ...
  ...
```

Projeyi gerçekleştirebilmek için, *vertex*(koordinat) ve *face*(yüz) bilgileri bütün işlevleri çalıştırmak için yeterli olacaktır. DLL içerisinde "*Vertex*" ve "Face" adlı iki sınıf oluşturuldu. Bu sınıfların örnekleri, "*Wavefront .obj*" dosyasının ayrıştırılması sonucu, satır başı harfleri dikkate alınarak (*Vertex* için "v", *Face* için "f") oluşturuldu. Bu işlemin nasıl gerçeklendiği "*GeometryParser*" kodunda anlatılacaktır.

"GeometryParser" sınıfı: Bu sınıf kütüphanedeki temel yönetim sınıfıdır. Nesnenin geometrik yoluna bakılarak, "*Wavefront .obj*" dosyasının okunmasından ve gerekli *vertex* ve *face* listelerini oluşturmaktan sorumludur.

```
    public class GeometryParser
    {

        public List<Vertex> VertexList { get; set; }
        public List<Face> FaceList { get; set; }

        public GeometryParser()
        {
            VertexList = new List<Vertex>(); /// Create new vertex list
            FaceList = new List<Face>(); /// Create new face list
        }

        public void Parse3DGeometry(string folderdir)
        {

            string line;

            double firstV, secondV, thirdV;
            int firstF, secondF, thirdF;
            StreamReader sr = new StreamReader(folderdir); ///  This function is
getting values from file
            while (sr.Peek() > -1) /// Read until file ends
            {
                line = sr.ReadLine();
                if (line[0] == 'v') /// This condition parses the vertices of  the
geometry
                {
                    string[] parsedVertices = line.Split('  ');
                    firstV = Convert.ToDouble(parsedVertices[1]);
                    secondV = Convert.ToDouble(parcedVertices[2]);
                    thirdV = Convert.ToDouble(parcedVertices[3]);

                    Vertex myVertex = new Vertex { x = firstV, y = secondV, z = thirdV
};
                    VertexList.Add(myVertex); /// Parsed vertices added vertex list
```

```
                }
                else if (line[0] == 'f') /// This condition parses the faces of the
geometry
                {
                    string[] parsedFaces = line.Split('  ');
                    firstF = Convert.ToInt32(parsedFaces[1]);
                    secondF = Convert.ToInt32(parsedFaces[2]);
                    thirdF = Convert.ToInt32(parsedFaces[3]);

                    Face myFace = new Face { firstIndex = firstF, secondIndex =
secondF, thirdIndex = thirdF };
                    FaceList.Add(myFace); /// Parsed faces added face list

                }
            }
            sr.Close();
        }
    }
```

"Vertex" Sınıfı: Bu sınıf 3B nesnelerin 3B noktalarının *double* veri tipinde koordinatlarını tutan sınıftır.

```
    /// <summary>
    /// Vertex: This contains the Vertex coordinates as 3 dimensional Cartesian
coordinate
    /// </summary>
    public class Vertex
    {
        public double x { get; set; }
        public double y { get; set; }
        public double z { get; set; }

        public string Serialize()
        {
            return String.Format("Vertex:({0}, {1}, {2})", x, y, z);
        }

        public Vertex Clone()
        {
            return new Vertex { x = this.x, y = this.y, z = this.z};
        }
    }
```

"Face" Sınıfı: Bu sınıf 3B nesnelerin, her dört *vertex* bilgisinin bir *face*(yüz) oluşturduğu bilgilerini tutan bir sınıftır. *Face* bilgisi sadece ekrana 3B nesne çizmek için kullanıldı. Kesim operasyonu için herhangi bir işlevi kullanılmadı.

```
    /// <summary>
    /// Face: This class represents the index of vertices for a triangle
    /// </summary>
    public class Face
    {
        public int firstIndex;
        public int secondIndex;
        public int thirdIndex;

        public string Serialize()
        {
            return String.Format("Face:({0}, {1}, {2})", firstIndex, secondIndex,
thirdIndex);
        }
    }
```

"Displacement" sınıfı: Motor dönüşlerini hesaplamak buna ek olarak üç boyutta iki *vertex* bilgisi arasında yer değişikliği için gerekli *vertex* bilgilerini alan sınıftır. Bunun için DLL içerisinde yeni bir sınıf tasarlandı.

```
public class Displacement /// This class is responsible for moving motors one direction to another
 {
   public double x { get; set; }
        public double y { get; set; }
        public double z { get; set; }
        public Vertex SourceVertex { get; set; }
        public Vertex TargetVertex { get; set; }

        public Displacement()/// Constructor method
        {

        }

        public string Serialize()
        {
            return String.Format("Dsp:({0}, {1}, {2})", x, y, z);
        }

        public Displacement(Vertex sourceVertex, Vertex targetVertex) /// This method
declares target and source destinations
        {
            x = targetVertex.x - sourceVertex.x;
            y = targetVertex.y - sourceVertex.y;
            z = targetVertex.z - sourceVertex.z;

            SourceVertex = sourceVertex;
            TargetVertex = targetVertex;
        }
    }
```

3.4.1.2. MillingMachineApplication

Bu uygulama üç adet "XAML" dosyasını; yani "*App.xaml*", "*MainWindows.xaml*" ve "*GeometryWindow.xaml*" dosyalarını içerir. "*App.xaml*", Microsoft firmasının .NET frameworkü içerisinde gelen çekirdek WPF kütüphanesi "*PresentationFramework.dll*" (*System.Windows* isim uzayı) içindeki "*Application*" sınıfını extend eder. Uygulama çalıştığında "*MainWindows.xaml*" örneğini oluşturur. Bu örnek programın ana kullanıcı ara yüzünü, hareket kontrollerini ve ana "*StartCutting*"(kesme) işlevini içerir. "*MainWindows.xaml*" sınıfı "PresentationFramework.dll" (*System.Windows* isim uzayı) içindeki "*Windows*" sınıfından extend edilmiş bir sınıftır.

NXT beyin ile ilgili bütün bağlantılar *"NXTBrick"* sınıfının bir örneği olan *"nxt"* olarak aşağıda tanımlanmıştır.

```
// NXT brick
        private NXTBrick nxt = new NXTBrick();
```

Sınıftaki farklı işlevler de ayrıca aşağıda tanımlanmıştır;

```
// regulation modes
        private NXTBrick.MotorRegulationMode[] regulationModes = new
NXTBrick.MotorRegulationMode[] {
            NXTBrick.MotorRegulationMode.Idle,
            NXTBrick.MotorRegulationMode.Speed,
            NXTBrick.MotorRegulationMode.Sync };
        // run states
        private NXTBrick.MotorRunState[] runStates = new NXTBrick.MotorRunState[] {
            NXTBrick.MotorRunState.Idle,
            NXTBrick.MotorRunState.RampUp,
            NXTBrick.MotorRunState.Running,
            NXTBrick.MotorRunState.RampDown };
        // sensor types
        private NXTBrick.SensorType[] sensorTypes = new NXTBrick.SensorType[] {
            NXTBrick.SensorType.NoSensor, NXTBrick.SensorType.Switch,
```

```
            NXTBrick.SensorType.Temperature, NXTBrick.SensorType.Reflection,
            NXTBrick.SensorType.Angle, NXTBrick.SensorType.LightActive,
            NXTBrick.SensorType.LightInactive, NXTBrick.SensorType.SoundDB,
            NXTBrick.SensorType.SoundDBA, NXTBrick.SensorType.Custom,
            NXTBrick.SensorType.Lowspeed, NXTBrick.SensorType.Lowspeed9V,
            NXTBrick.SensorType.Highspeed, NXTBrick.SensorType.ColorFull,
NXTBrick.SensorType.ColorRed,
            NXTBrick.SensorType.ColorGreen, NXTBrick.SensorType.ColorBlue,
NXTBrick.SensorType.ColorNone, NXTBrick.SensorType.ColorExit};
        // sensor modes
        private NXTBrick.SensorMode[] sensorModes = new NXTBrick.SensorMode[] {
            NXTBrick.SensorMode.Raw, NXTBrick.SensorMode.Boolean,
            NXTBrick.SensorMode.TransitionCounter,
NXTBrick.SensorMode.PeriodicCounter,
            NXTBrick.SensorMode.PCTFullScale, NXTBrick.SensorMode.Celsius,
            NXTBrick.SensorMode.Fahrenheit, NXTBrick.SensorMode.AngleSteps };
```

3B kesim platformunda, NXT beyin üç boyut için üç farklı motoru kontrol etmektedir. "*MainWindows.xaml*" sınıfı içerisinde bu motor setleri aşağıda tanımlamıştır ;

```
        // Returns selected motor
        private NXTBrick.Motor GetSelectedMotorZ()
        {
            return (NXTBrick.Motor)1;
        }

        // Returns selected motor
        private NXTBrick.Motor GetSelectedMotorY()
        {
            return (NXTBrick.Motor)0;
        }

        // Returns selected motor
        private NXTBrick.Motor GetSelectedMotorX()
        {
            return (NXTBrick.Motor)2;
        }

        private NXTBrick.Motor GetThreeDSelectedMotor(int axis)
        {
            return (NXTBrick.Motor)axis;
        }
```

Aynı sınıf içerisinde, bir önceki bölümde anlatılan *AForge.NET'in* içinde "*NXTBrick*" sınıfı içerisinde tanımlanan "*SetMotorState*" işlevinin benzeri görevi yapan, her bir NXT motoruna üst seviye komutlar gönderen "*RunMotorInAnAxis*" işlevi tasarlanmıştır.

```
private NXTBrick.MotorState threeDMotorState;
        /// <summary>
        ///
        /// </summary>
        /// <param name="axis"> 0: Y axis (cutting motor), 1: Z axis, 2: X
axis</param>
        /// <param name="displacement"></param>
        private bool RunMotorInAnAxis(int axis, int displacement) /// This method
includes basic operations on motors
        {
            if (displacement != 0)
            {
                if (displacement < 0)
                {
                    threeDMotorState.Power = -1 * Convert.ToByte(powerUpDown.Text);
                }
                else
                {
                    threeDMotorState.Power = Convert.ToByte(powerUpDown.Text);
                }

                try
```

```
                {
                    threeDMotorState.TachoLimit = Math.Abs(displacement);
                }
                catch
                {
                    threeDMotorState.TachoLimit = 1000;
                    tachoLimitBox.Text = threeDMotorState.TachoLimit.ToString();
                }

                // set motor's state
                if (nxt.SetMotorState(GetThreeDSelectedMotor(axis), threeDMotorState,
false) != true)
                {
                    System.Diagnostics.Debug.WriteLine("Failed setting motor state");
                }

                return true;
            }

            return false;
        }
```

"*MainWindows.xaml*" içerisinde tanımlanan "*StartCutting*" işlevi temel kesim operasyonunu içermektedir. Bu metot bütün alt seviye tanımları, 3B nesnenin geometri bilgisini, *motor/sensor* kontrolünü ve kesme mantığını bir araya getiren metottur. Metodun temel yapısı aşağıdaki kod parçasında tanımlanmıştır;

```
/// <summary>
        /// StartCutting: Main Cutting Method. Before calling this method, be sure
that the 3d machine is ready.
        /// </summary>
        /// <param name="calibrationOnly"></param>
        private void StartCutting(bool calibrationOnly)
        {
             MillingMachineManager myMillingMachineManager = new
MillingMachineManager(AppDomain.CurrentDomain.BaseDirectory + @"\head-vise.obj");

             threeDMotorState = new NXTBrick.MotorState();


            // prepare motor's state to set
             threeDMotorState.Power = Convert.ToByte(powerUpDown.Text);
             threeDMotorState.TurnRatio = 0;
             threeDMotorState.Mode = ((modeOnCheck.IsChecked == true) ?
NXTBrick.MotorMode.On : NXTBrick.MotorMode.None) |
                 ((modeBrakeCheck.IsChecked == true) ? NXTBrick.MotorMode.Brake :
NXTBrick.MotorMode.None) |
                 ((modeRegulatedBox.IsChecked == true) ? NXTBrick.MotorMode.Regulated
: NXTBrick.MotorMode.None);
             threeDMotorState.Regulation =
regulationModes[regulationModeCombo.SelectedIndex];
             threeDMotorState.RunState = runStates[2];

            // Prepare sensor state
            if (nxt.SetSensorMode(GetSelectedSensor(),
              sensorTypes[BLUE COLOR SENSOR STATE INDEX],
              sensorModes[0], false) != true)
            {
                System.Diagnostics.Debug.WriteLine("Failed setting input mode");
            }

            List<Displacement> displacementList;

            if (calibrationOnly)
            {
                displacementList =
myMillingMachineManager.GetVertexCalibrationListForMachineCoordinatesAsDisplacement();
            }
            else
            {
                displacementList =
myMillingMachineManager.GetVertexListForMachineCoordinatesAsDisplacement();
            }
```

```
            foreach (Displacement v in displacementList)
            {
                // Run X axis motor

                int xDisplacement = Convert.ToInt32(v.x);

                if (RunMotorInAnAxis(2, xDisplacement))
                {
                    Thread.Sleep(Math.Abs(xDisplacement * 100));
                }


                //displayTxt.Text += xDisplacement + ",";

                // Run Z axis motor

                int zDisplacement = Convert.ToInt32(v.z);

                if (RunMotorInAnAxis(1, zDisplacement))
                {
                    Thread.Sleep(Math.Abs(zDisplacement * 100));
                }


                // This is the main rotating/cutting motor. We use actual vertex
information rather than displacement information and
                // use color detector for returning back

                int yPosition;
                if (calibrationOnly)
                {
                    yPosition = Convert.ToInt32(v.y);
                }
                else
                {
                    yPosition = Convert.ToInt32(v.TargetVertex.y);
                }


                if (RunMotorInAnAxis(0, yPosition))
                {
                    Thread.Sleep(Math.Abs(yPosition * 10));


                    NXTBrick.SensorValues sensorValues;

                    do
                    {
                        // get input values for Sensor
                        if (nxt.GetSensorValue(GetSelectedSensor(), out sensorValues))
                        {
                            validCheck.IsChecked = sensorValues.IsValid;
                            calibratedCheck.IsChecked = sensorValues.IsCalibrated;
                            sensorTypeBox.Text = sensorValues.SensorType.ToString();
                            sensorModeBox.Text = sensorValues.SensorMode.ToString();
                            rawInputBox.Text = sensorValues.Raw.ToString();
                            normalizedInputBox.Text =
sensorValues.Normalized.ToString();
                            scaledInputBox.Text = sensorValues.Scaled.ToString();
                            calibratedInputBox.Text =
sensorValues.Calibrated.ToString();
                        }
                        else
                        {
                            System.Diagnostics.Debug.WriteLine("Failed getting input
values");
                        }


                        // Now go up until blue color is reached
                        threeDMotorState.Power = -1 *
Convert.ToByte(powerUpDown.Text);
                        threeDMotorState.TachoLimit = 5;
```

```
                        // set motor's state
                        if (nxt.SetMotorState(GetSelectedMotorY(), threeDMotorState,
false) != true)
                        {
                            System.Diagnostics.Debug.WriteLine("Failed setting motor
state");
                        }

                        Thread.Sleep(Math.Abs(50));

                    } while (sensorValues.Raw < 250);
                }
            }
        }
```

"GeometryWindow.xaml" sınıfı da "PresentationFramework.dll"(System.Windows isim uzayı) kütüphanesi içerisindeki "Windows" sınıfından genişletilmiştir. Bu sınıf 3B nesneyi uygulama içerisinde görsel olarak görmek için tasarlanmıştır.

```
namespace MillingMachineApplication
{
    /// <summary>
    /// Interaction logic for GeometryWindow.xaml
    /// </summary>
    public partial class GeometryWindow : Window
    {
        System.Windows.Threading.DispatcherTimer dispatcherTimer;

        private GeometryModel3D geometryModel;
        private int degree = 90;

        public GeometryWindow()
        {
            InitializeComponent();

            dispatcherTimer = new System.Windows.Threading.DispatcherTimer();
            dispatcherTimer.Tick += new EventHandler(dispatcherTimer_Tick);
            dispatcherTimer.Interval = new TimeSpan(0, 0, 1);
        }

        private void rotateHead()
        {
            degree += 30;
            RotateTransform3D myRotateTransform3D = new RotateTransform3D();
            AxisAngleRotation3D myAxisAngleRotation3d = new AxisAngleRotation3D();
            myAxisAngleRotation3d.Axis = new Vector3D(0, 3, 0);
            myAxisAngleRotation3d.Angle = degree;
            myRotateTransform3D.Rotation = myAxisAngleRotation3d;
            geometryModel.Transform = myRotateTransform3D;
        }

        private void dispatcherTimer_Tick(object sender, EventArgs e)
        {
            rotateHead();
        }

        public void Basic3DShapeExample()
        {
            // Declare scene objects.
            Viewport3D myViewport3D = new Viewport3D();
            Model3DGroup myModel3DGroup = new Model3DGroup();
            GeometryModel3D myGeometryModel = new GeometryModel3D();
            ModelVisual3D myModelVisual3D = new ModelVisual3D();
            // Defines the camera used to view the 3D object. In order to view the 3D
object,
            // the camera must be positioned and pointed such that the object is
within view
            // of the camera.
            PerspectiveCamera myPCamera = new PerspectiveCamera();

            // Specify where in the 3D scene the camera is.
            myPCamera.Position = new Point3D(0, 0, 2);

            // Specify the direction that the camera is pointing.
```

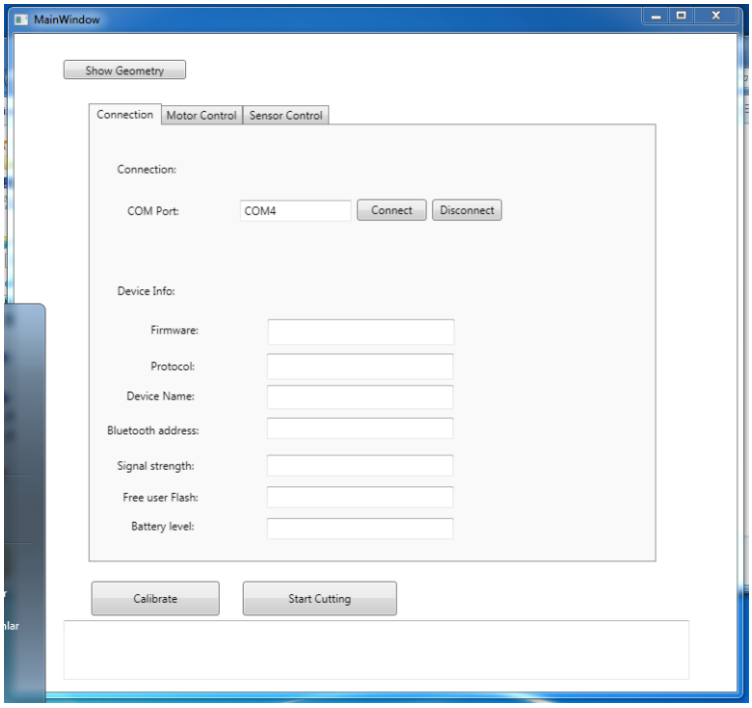
```
            myPCamera.LookDirection = new Vector3D(0, 0, -1);

            // Define camera's horizontal field of view in degrees.
            myPCamera.FieldOfView = 60;

            // Asign the camera to the viewport
            myViewport3D.Camera = myPCamera;
            // Define the lights cast in the scene. Without light, the 3D object
cannot
            // be seen. Note: to illuminate an object from additional directions,
create
            // additional lights.
            DirectionalLight myDirectionalLight = new DirectionalLight();
            myDirectionalLight.Color = Colors.White;
            myDirectionalLight.Direction = new Vector3D(-0.61, -0.5, -0.61);

            myModel3DGroup.Children.Add(myDirectionalLight);

            // The geometry specifes the shape of the 3D plane. In this sample, a flat
sheet
            // is created.
            MeshGeometry3D myMeshGeometry3D = new MeshGeometry3D();

            // Create a collection of vertex positions for the MeshGeometry3D.
            Point3DCollection myPositionCollection = new Point3DCollection();

            MillingMachineManager myMillingMachineManager = new
MillingMachineManager(AppDomain.CurrentDomain.BaseDirectory + @"\head-vise.obj");

            foreach (Vertex v in myMillingMachineManager.GetNormalisedVertexList())
            {
                myPositionCollection.Add(new Point3D(v.x, v.y, v.z));
            }

            myMeshGeometry3D.Positions = myPositionCollection;

            // Create a collection of triangle indices for the MeshGeometry3D.
            Int32Collection myTriangleIndicesCollection = new Int32Collection();

            foreach (Face f in myMillingMachineManager.ObjectGeomertyParser.FaceList)
            {
                myTriangleIndicesCollection.Add(f.firstIndex - 1);
                myTriangleIndicesCollection.Add(f.secondIndex - 1);
                myTriangleIndicesCollection.Add(f.thirdIndex - 1);
            }

            myMeshGeometry3D.TriangleIndices = myTriangleIndicesCollection;

            // Apply the mesh to the geometry model.
            myGeometryModel.Geometry = myMeshGeometry3D;

            // The material specifies the material applied to the 3D object. In this
sample a
            // linear gradient covers the surface of the 3D object.
            // Create a horizontal linear gradient with four stops.
            LinearGradientBrush myHorizontalGradient = new LinearGradientBrush();
            myHorizontalGradient.StartPoint = new Point(0, 0.5);
            myHorizontalGradient.EndPoint = new Point(1, 0.5);
            myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.Yellow,
0.0));
            myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.Red,
0.25));
            myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.Blue,
0.75));
            myHorizontalGradient.GradientStops.Add(new GradientStop(Colors.LimeGreen,
1.0));

            // Define material and apply to the mesh geometries.
            DiffuseMaterial myMaterial = new DiffuseMaterial(myHorizontalGradient);
            myGeometryModel.Material = myMaterial;

            // Apply a transform to the object. In this sample, a rotation transform
is applied,
            // rendering the 3D object rotated.
            RotateTransform3D myRotateTransform3D = new RotateTransform3D();
            AxisAngleRotation3D myAxisAngleRotation3d = new AxisAngleRotation3D();
            myAxisAngleRotation3d.Axis = new Vector3D(0, 3, 0);
```

```
            myAxisAngleRotation3d.Angle = 90;
            myRotateTransform3D.Rotation = myAxisAngleRotation3d;
            myGeometryModel.Transform = myRotateTransform3D;

            // Add the geometry model to the model group.
            myModel3DGroup.Children.Add(myGeometryModel);

            geometryModel = myGeometryModel;

            // Add the group of models to the ModelVisual3d.
            myModelVisual3D.Content = myModel3DGroup;
            myViewport3D.Children.Add(myModelVisual3D);

            // Apply the viewport to the page so it will be rendered.
            this.Content = myViewport3D;

            dispatcherTimer.Start();
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            Basic3DShapeExample();
        }
    }
}
```

3.4.1.3. MillingMachineCoreDLL

3.4.2. Uygulama Arayüzleri

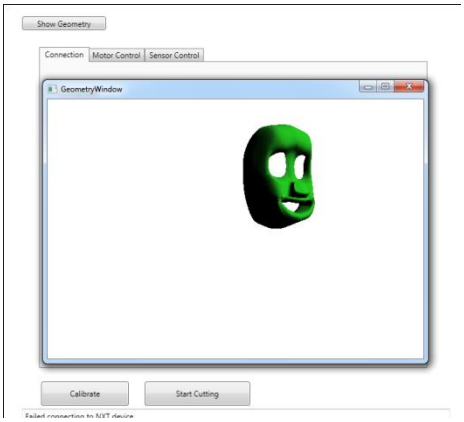
Program başlatıldığında *Resim 3.25* ara yüzü ekranda görünecektir.



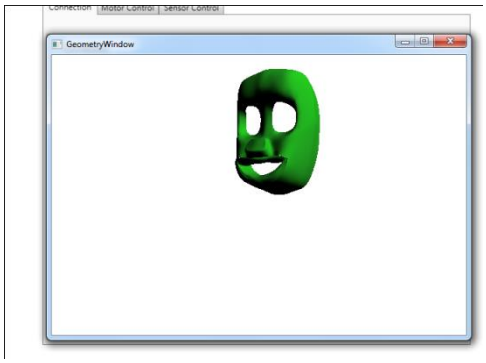
Resim 3.25- 3B kesim platformu arayüzü

Ara yüzde bir kaç *buton* ve *sekme* bulunuyor. ***Show Geometry*** butonundan başlayacak olursak bu buton Blender uygulamasında tasarlanan kesilecek olan

geometrik şekli ekrana basmak için kullanıldı. Butona basıldığında nesne ekranda görünüyor ve 360 derece dönmeye başlıyor. (*Resim 3.26, Resim 3.27*)

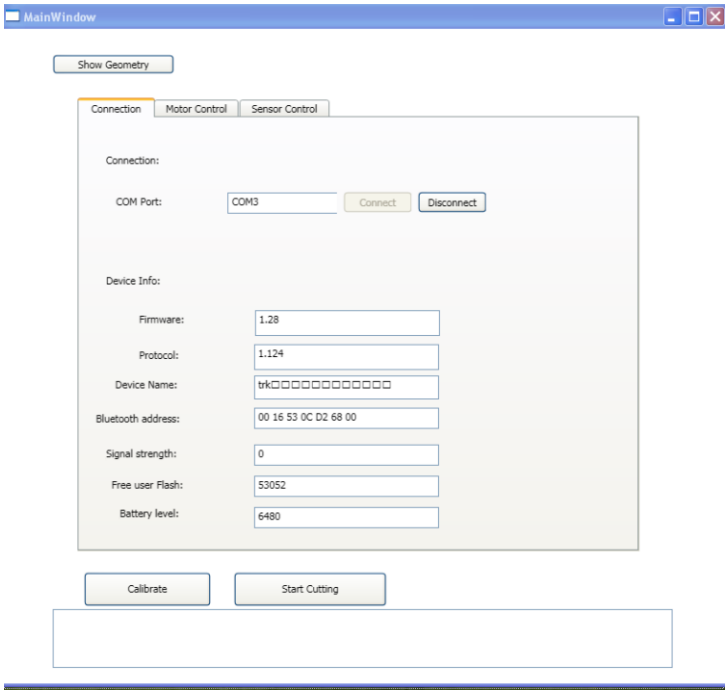


Resim 3.26 – 3B Nesne_Göster_1



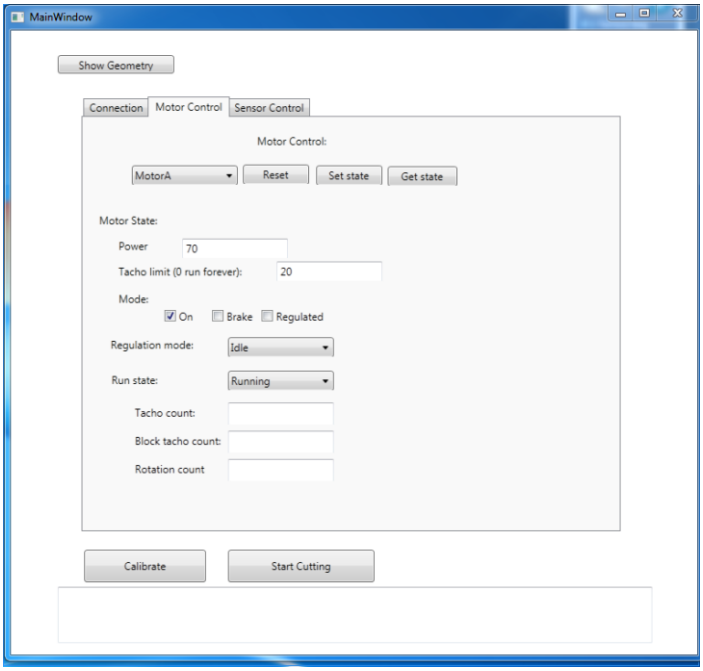
Resim 3.27 – 3B Nesne_Göster_2

Bu buton "*MillingMachineGeometryParsell.dll*" kütüphanesi içerisindeki geometri sınıfının örneğini oluşturur. Bu butonun altında, üç farklı sekme mevcuttur. İlk sekme NXT Brick ile bağlantı kurmak için kullanılıyor. Örnek bir kablosuz bağlantı Resim 3.28 ile gösterilmiştir.

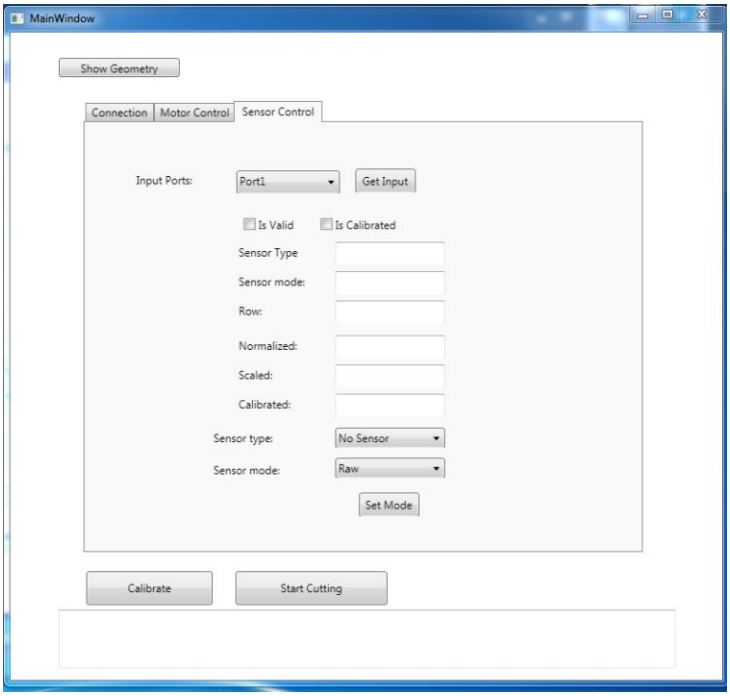


Resim 3.28- Kurulmuş NXT Brick Bağlantısı

Motor Control sekmesi ise motoru elle kontrol etmek için kullanılıyor. *(Resim 3.29)* **Sensor Control** Sekmesi de adından anlaşıldığı üzere sensör kontrolü, sensörlerden girdi/çıktı almak için kullanılıyor. *(Resim 3.30)*



Resim 3.29 – Motor Control sekmesi



Resim 3.30-Sensor Control sekmesi

Bu üç sekmenin altında, iki tane butonu bulunmaktadır. Birincisi, Calibrate butonu 3B objeyi kesmeden önce bir kalibrasyon ayarı yapmaktadır. Start Cutting butonu temel işlevi yani 3B kesim işlemini gerçekleştirmekten sorumludur.

3.5. 3B kesim makinasını test etme

Programı tamamen çalıştırmadan önceki dönüşümler ve hesaplamaları test etmek için tasarlanmış bir test sınıfı tasarlanmıştır. 3B kesme makinasının motorlarını kontrol etmek ve olası bir hasarı engellemek için böyle bir sınıf tasarlanmıştır.

3.5.1. MillingMachineTestDLL

UnitTestMachineManager: Bu sınırları kontrol eden bir test sınıfıdır. "*sampleVertexList*" içerisindeki değerler kalibrasyon gerektiği için değişmişlerdir. Bu test sınıfının kod gövdesi aşağıda verilmiştir.

```
[TestClass]
public class UnitTestMachineManager
{
    [TestMethod]
    public void FindMaximumAndMiniumValuesTest()
    {

        // Calibration sample
        List<Vertex> sampleVertexList = new List<Vertex>();
        sampleVertexList.Add(new Vertex { x = 3, y = 3, z = 91 });
        sampleVertexList.Add(new Vertex { x = 9, y = 4, z = 12 });
        sampleVertexList.Add(new Vertex { x = 2, y = 12, z = -3 });
        sampleVertexList.Add(new Vertex { x = -8, y = -5, z = 8 });
        sampleVertexList.Add(new Vertex { x = 4, y = -4, z = 12 });
        sampleVertexList.Add(new Vertex { x = 2, y = 0, z = 7 });


        // Act
        double maxXValue = Utility.FindMaxValue(sampleVertexList, m => m.x);
        double minXValue = Utility.FindMinValue(sampleVertexList, m => m.x);

        double maxYValue = Utility.FindMaxValue(sampleVertexList, m => m.y);
        double minYValue = Utility.FindMinValue(sampleVertexList, m => m.y);

        double maxZValue = Utility.FindMaxValue(sampleVertexList, m => m.z);
        double minZValue = Utility.FindMinValue(sampleVertexList, m => m.z);


        //Assert functions

        Assert.AreEqual(maxXValue, 9);
        Assert.AreEqual(minXValue, -8);
        Assert.AreEqual(maxYValue, 12);
        Assert.AreEqual(minYValue, -5);
        Assert.AreEqual(maxZValue, 91);
        Assert.AreEqual(minZValue, -3);
    }

}
```

4. ÖZET VE GELECEK ÇALIŞMALAR

4.1.Özet

Bu projede, örnek bir Lego Mindstorms® 3B kesim platformu başarıyla tasarlandı. Platformu kontrol etmek için ayrı bir Windows uygulaması geliştirildi. Bu proje boyunca Lego Mindstorms® kitinin içerdiği parçalarla birlikte, Lego Technic® ve basit bir matkap motoru kullanıldı. Birinci bölümdeki beklentiler doğrultusunda bütün bu kısımlar bir araya getirildi.

Proje boyunca, bir sürü zorlukla karşılaşıldı. İlk olarak prototip alışma sürecinde birden fazla örnek tasarlandı. Bu alışma sürecinde edinilen deneyim sonucunda nihai 3B kesim platformunun ortaya çıkması beklendiğinden fazla zaman aldı. 3B kesim platformunun kararlılık sıkıntısı bu zorlukların en başında idi. Blender yazılımında tasarlanan, 3B nesnelerin ölçeklenmesi ciddi programlama ve matematik bilgisi gerektirdi. Lego Mindstorms® NXT-G yazılımı büyük projeler için ideal bir yazılım değildi ve bir sürü çökme sıkıntısıyla karşılaşıldı. Üçüncü parti iletişim API'leri projenin gerçeklenmesinde yazılım mimarisine fikir vermesi açısında çok yardımcı oldu.

Bu projede, platformun kalibrasyonunu yapmak oldukça zorlu bir işti. İlk olarak platformu kalibre etmek için kullanılan NXT-G programında birden fazla blok tasarlandı. Yine de kalibre edilmiş değer ölçümü çok zaman aldı. API ile ise bu durum daha kolay ve hassas oldu. Kalibrasyon işlevi belirli bir bölgeyi kesmek için geliştirildi. Özellikle Unit Test projesi makinesinin hasar almasını önlemek için kullanıldı.

Bu projede nesne dosyalarından (.obj) okunan koordinatlar doğrultusunda, geliştirilen C# programında koordinat sistemi y ekseni x ve z eksenine dik şekilde tasarlandı. Y eksenin hareketi diğer eksenlere göre farklı tasarlandı. X ve z eksenlerinde alınan yer değiştirme değerlerini göz önüne alarak motorlar doğru yönde hareket etti. Y ekseninde ise, motor kesim yapan parçayı (matkap ucu) referans noktasından y-eksenindeki vertex değerine kadar hareket ettirdi. Başlangıç noktasına dönüş için ise renk sensörü mavi rengi kesme sinyali alacak şekilde tasarlandı. Bu eksenlerin koordinat düzlemindeki uyumu ve bununla birlikte renk sensörünüyle olan senkronizasyonu ayrıca zor bir görev oldu.

Sonuç olarak, kararlı bir 3B kesim platformu, projenin amacı doğrultusunda Lego® parçalarıyla başarıyla tasarlandı. Blender uygulamasındaki 3B nesne tasarımı deneyimi ayrıca bu proje için bir artı oldu.

Projenin yazarları proje boyunca Lego®'nun kendi programlama arabirimini tanıyarak Visual Studio'da C# programı tasarlarken ciddi bir programlama deneyimi kazandılar. Hem donanım hem de yazılım unsurları bu projeyi dikkate değer bir proje haline getirdi.

4.2. Gelecek Çalışmalar

Bu proje boyunca görsel programlamanın üst seviye problemler karşısındaki soyutlama aşamasında karmaşıklığa ve bir çözümünün olmaması sorununa yol açtığı fark edildi. Karmaşık bir proje üzerinde çalışırken, robotik kit bilgi eksikliği, robotun motorları, sensörleri, girdi/çıktı karakteristiği arasındaki temel ilişkiyi anlamak ciddi hayal kırıklıklarına yol açacaktır. Görsel programlanın eksiklikleri, API programlamaya kıyasla eksik kaldığı noktaları içeren bir karşılaştırma tezi ileriye dönük bir makale olarak düşünülebilir.

Temel gelecek çalışma 3B kesim platformunun kapasitesinin geliştirilmesi üzerine olacaktır. Şimdiki tasarım 4,8 cm * 6,8 cm'yi geçmeyecek nesneler için tasarlanmıştır. Bu kısıtlama ileride daha büyük bir platform tasarlanarak aşılabilir. Ayrıca bu proje sadece wavefront (.obj) dosya türünü tanımaktadır. Yeni ayrıştırma metodları bu projeye eklenebilir. Eklenecek bu ayrıştırma işlevleri daha fazla nesne dosyasını destekleyebilir. Ayrıca örneğin farklı bir platform tasarlanarak 360^{o} boyunca nesneler kesilebilir.

Son olarak bu proje boyunca programlama dili olarak C# dili seçildi. Bunun dışında başka diller ve platformlarda (örnek; Java API) kesme işlevini gerçekleyecek ara yüzler tasarlanabilir.

REFERANSLAR

1. Akin, H. L., Meriçli, Ç, Meriçli T. and Doğrultan, E. (2009). Introduction to Autonomous Robotics with Lego Mindstorms*, WSPC - Proceedings, clawar09-Education.*
2. AForge.NET (2008). <*http://www.aforgenet.com/* >
3. Arduino (2005). <*http://arduino.cc/en/* >
4. Blender Foundation (2012). < *http://www.blender.org/* >
5. Jonathan Knudsen (2000). *Lego MindStorms: An Introduction.* O'Reilly Network, <*http://www.oreillynet.com/pub/a/network/2000/01/31/mindstorms/index.html*>
6. Jörgen Lindh, Thomas Holgersson (2007). Does lego training stimulate pupils ability to solve logical problems? J*ournal Computers & Education archive*, Volume 49 Issue 4, Pages 1097-1111.
7. Kim, S. H. and Jeon, J.W. (2007). Programming LEGO Mindstorms NXT with visual programming*, Control, Automation and Systems, 2007. ICCAS '07. International Conference on 17-20 Oct. 2007, Pages 2468 - 2472.*
8. LEGO MINDSTORMS:What is NXT? (2002). <*http://mindstorms.lego.com/en-us/whatisnxt/default.aspx*>
9. LEJOS, Java for Lego Mindstorms (2009). <*http://lejos.sourceforge.net/*>
10. Microsoft Robotics Developer Studio (2012). <*http://www.microsoft.com/robotics/*>
11. National Instruments' LabVIEW (2013). < *http://www.ni.com/labview/* >
12. NXC and NBC (2012).< *http://bricxcc.sourceforge.net/nbc/*>
13. Nxtprograms (2007). < *http://www.nxtprograms.com/* >
14. Perdue D., J. and Valk, L. (*2011). THE UNOFFICIAL LEGO MINDSTORMS NXT 2.0 INVENTOR'S GUIDE.* Publisher: William Pollock, No Starch Press, Inc. 38 Ringold Street, San Fransisco, CA 94103.

EKLER

Ek A – Lego Mindstorms NXT 2.0 Robotik Kiti
The NXT Brick

Resim 1

Resimdeki parça Mindstorms robotunun beyin parçasıdır. Bilgisayardan kontrol edilebilir robotu canlandıran temel parçadır.

Motor girişleri
Brick parçasının üç tane motor girişi vardır –A,B ve C girişleri
Sensör girişleri
Brick parçasının dört dane sensör girişi vardır – 1,2,3 ve 4 girişleri
USB girişi
Bu giriş Brick parçasını bilgisayara USB kablo ile bağlanması için tasarlanmış giriştir. Ayrıca bu girişten bağımsız kablosuz iletişim için robotun içerisinde Bluetooth bağlantısı mevcuttur.
Sesli hoparlör
Gerçek sesler üretebilen Brick'e entegre olan bir hoparlör sistemidir.
Brick Tuşları
Turuncu tuş: Aç/Çalıştır
Açık gri yön tuşları: NXT menüsünü sağa yola sola hareket ettirmek için kullanılır.
Kapalı gri tuşu: Temizle/Geri dön
Teknik detayları:
*32-bit ARM7 microcontroller
*256 KByte FLASH,64 Kbyte RAM
*8-bit AVR microcontroller
*4 KByes FLASH, 512 Byte RAM
*Bluetooth kablosun iletişim (Bluetooth Class II V2.0 comliant)
*USB portu(12 MBit/s)
*4 girdi girişi, 6-kablo platformu (Gelecekte kullanım için bir port IEC 61158 Tipi 4/EN 50 170 teknolojisi kullanıyor)

Renk Sensörü

Resim 2

Robota renk ayrımı katan bir sensördür. (Resim 2)

Renk sensörü üç fonksiyonu bir arada çalıştıran bir sensördü. Renk sensörü renkler arasındaki farklılıkları ve açık koyu renkleri ayırt edebilir. Açık renkli oda ve yüzeyde 6 farklı rengi tespit edebilir. Ayrıca renk lambası olarak kullanılabilir.

Dokuma Sensörü

Resim 3

Dokunma sensörü (Resim 3) robota dokunma hissi veren sensördü. Kendisine basıldığında ve bırakıldığında tepki verecek şekilde tasarlanmıştır.

Ultrasonik Sensör

Resim 4

Robota görüş veren bir sensördür.(Resim 4) Robotun çevresindeki nesneleri tespit etmesine yarar. Ayrıca robotun herhangi bir yüzeye çarpmasını mesafe ölçerek engeller.

Ultrasonik sensör cm ve inç cinsinden mesafe ölçer. 0-255 cm arasını +/- 3 cm hata ile ölçebilir.

Ultrasonik sensör yarasalardaki bilimsel kuram ile çalışır. Gönderilen ses dalgasının nesneye ulaşma ve geri dönüşü arasındaki süreyi hesaplar – (örnek eko etkisi)

En güzel okumalar en büyük nesnelerde olur. Yumuşak dokudan yapılmış ya da oyulmuş nesneler (örnek top) veya çok küçük nesneleri okumak zordur.

Serve Motorlar

Resim 5

Kitler birlikte gelen üç adet Serve motor(Resim 5) robotun hareketini sağlar. NXT-G yazılımı kullanarak motorları kontrol etmek için *Move* bloğunu kullanılırsa iki motor senkronize şekilde, düz bir doğrultuda hareket eder.

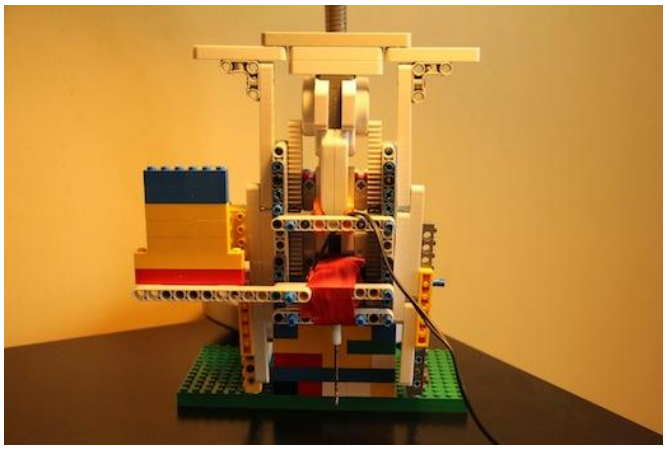
Gömülü Dönüş Sensörü

Bütün motorlar içerisinde gömülü dönüş sensörlerini içerirler. Motor hareketlerini daha kesin değerler üretmesi için tasarlanmıştır. Dönüş sensörü motor dönüşlerini derece(+/- bir derece keskinlik ile) ya da tam dönüşleri ölçer. Bir tam dönüş 360 derecedir, 180 derecelik bir dönüş motorun yarım tur dönmesi anlamına gelir.

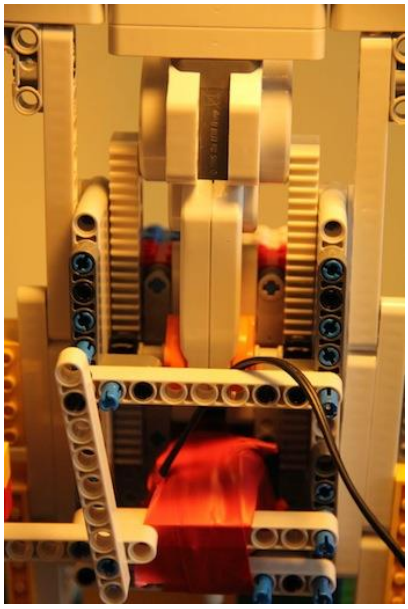
Gömülü dönüş sensörü farklı motorlarda farklı hızlardadır (bu hızlar yazılım içerisinde güç bölümünden ayarlanabilir)

Ek B- Platformun Detaylı Fotoğrafları

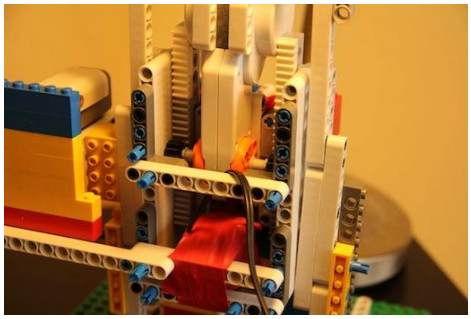
1.Bölüm: Y-ekseni



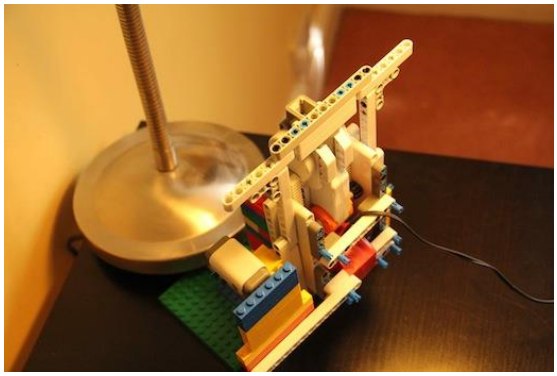
Resim B.1-Ön Görünüm



Resim B.2- Yakın Ön Görünüm



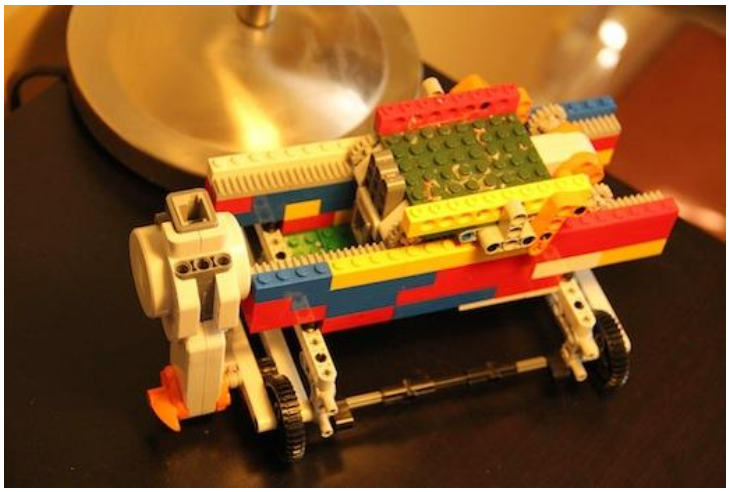
Resim B.3- Yan Üst Görünüm



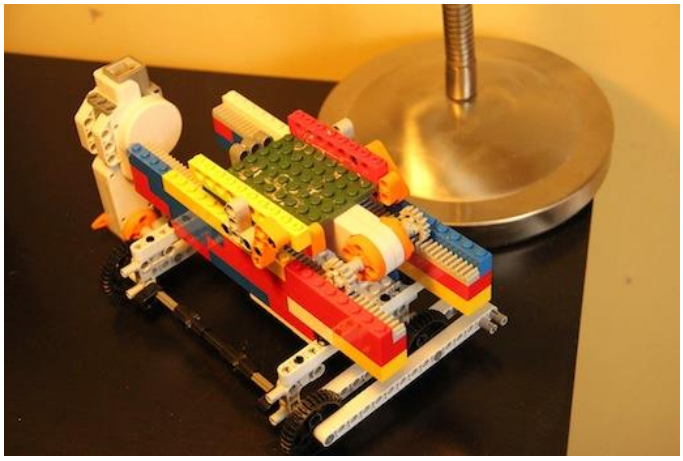
Resim B.4- Üst Görünüm

Resim B.5- Arka Görünüm

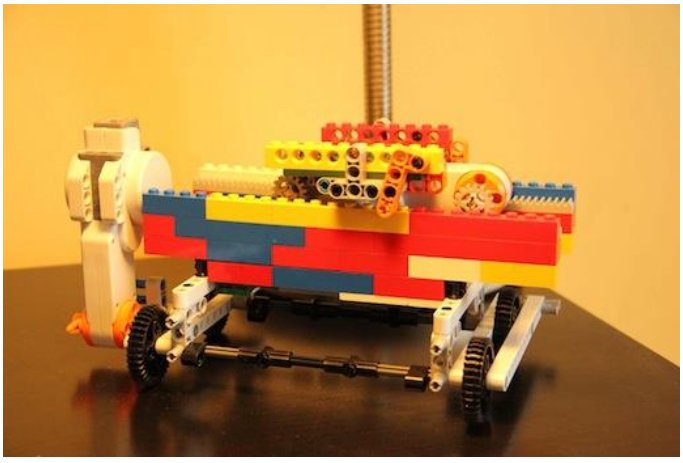
2.Bölüm: X ekseni ve Z ekseni birlikte



Resim B.6- Sol Üst Görünüm

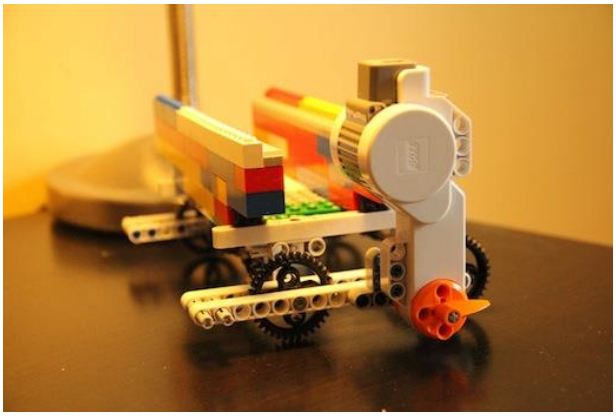


Resim B.7- Sağ Üst Görünüm

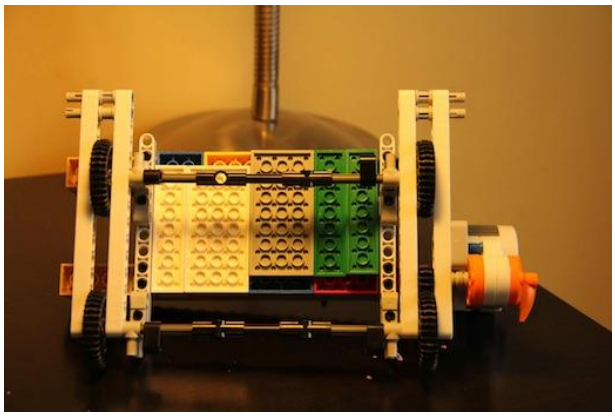


Resim B.8- Yan Görünüm

3.Bölüm: Z-ekseni

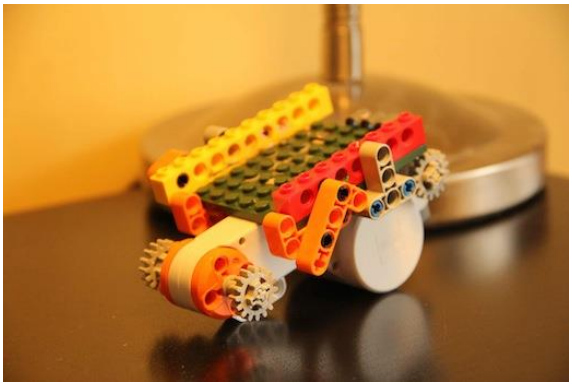


Resim B.9- Yan Görünüm

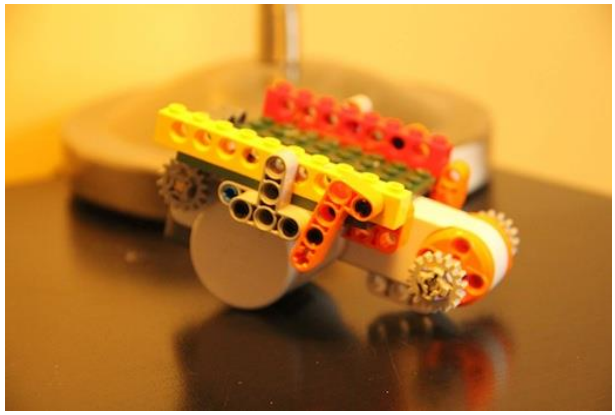


Resim B.10- Alt Görünüm

4.Bölüm: X-ekseni

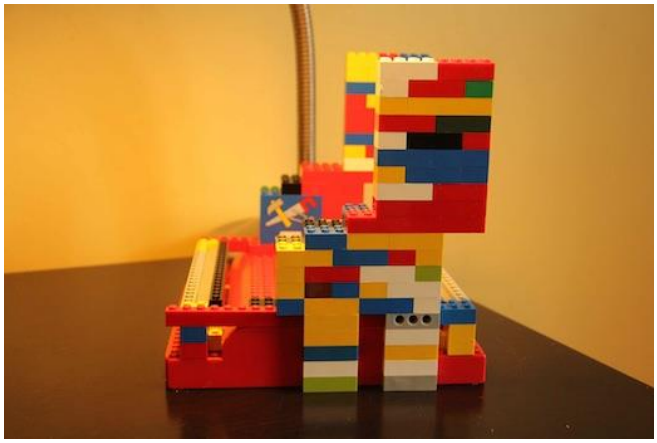


Resim B.11- Sol Yan Görünüm



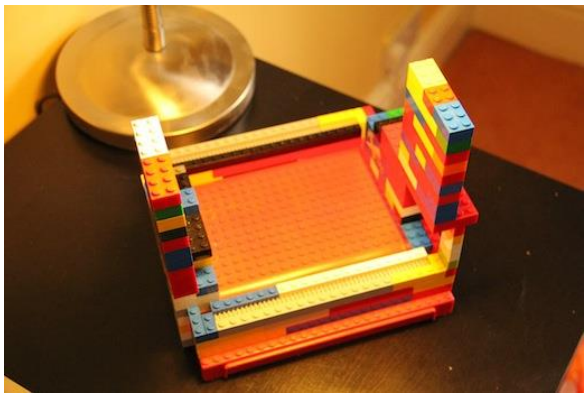
Resim B.12- Sağ Yan Görünüm

Bölüm 5: Platform



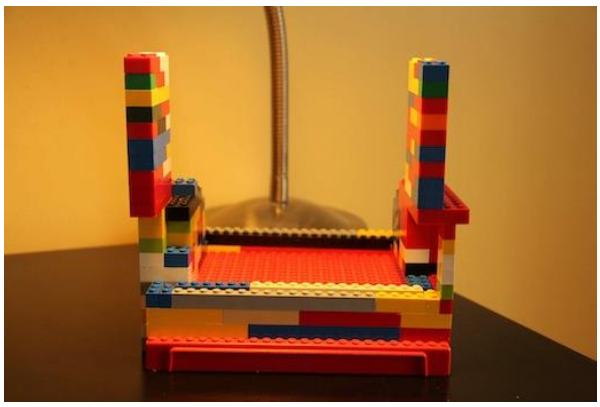
Resim B.13- Sol Yan Görünüm

Resim B.14- Sağ Yan Görünüm



Resim B.15- Üst Görünüm

Resim B.16- Ön Görünüm



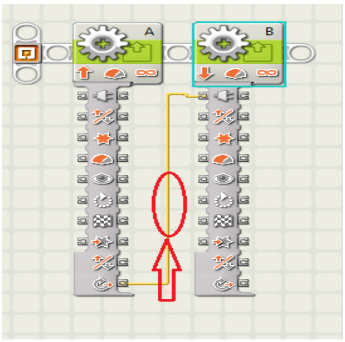
Resim B.17- Arka Görünüm

Ek C- NXT-G programlama

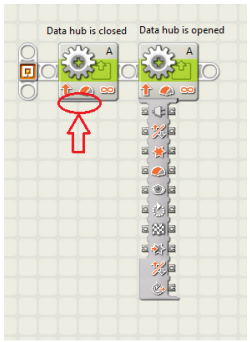
NXT 2.0 kiti istediğiniz projeleri belirli çerçevelerde tasarlamak için size bir görsel program arayüzü sunar. Basit programlama bilgisi NXT-G programında temel seviye görevleri gerçeklemek için yeterli olacaktır. Ama daha detaylı programlama gerektiğinde farklı programlar kullanılmalıdır.

Veri Kabloları

Birbirinden farklı programlama blokları arasında bilgi akışı için kullanılır.

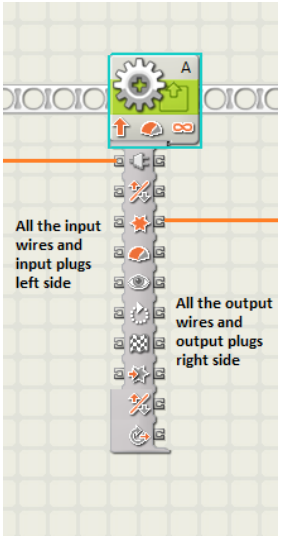


Resim 1- Farklı programlama blokları arasındaki data kablosu



Resim 2- Blokun veri girişine altındaki küçük sekmeye tuşlayarak ulaşılabilir

Veri kablosu içerisindeki detaylı veriler Resim 2'de görülebilir.
İki farklı çeşidi vardır girdi fişleri ve çıktı fişleri (Resim 3)



Resim 3- Girdi/Çıktı fişleri

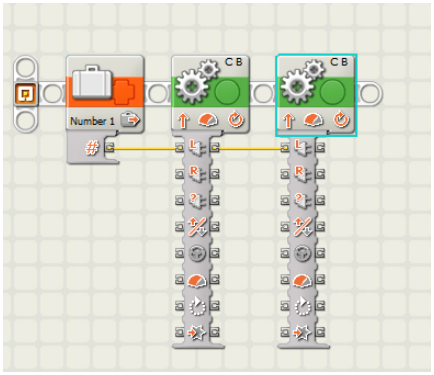
Veri Fişi Karakteristikleri

İlk olarak veri fişi bloktaki belirli işleri gerçekleyecek, simgelerle ifade edilmiş işlevleri gerçekler
İkinci olarak, veri fişi özel data tipi kullanır: *numara, mantık, yazı.*
Üçüncü olarak veri fişi sadece belirli aralıktaki değerler kabul eder.

Note!: Bekle, Döngü ve değiştir blokları veri fişlerine sahip değildir ama Döngü bloğu ve Değiştir bloğu belirli ayarları gerçekleyen veri fişlerine sahiptir.

Kablo Yolu

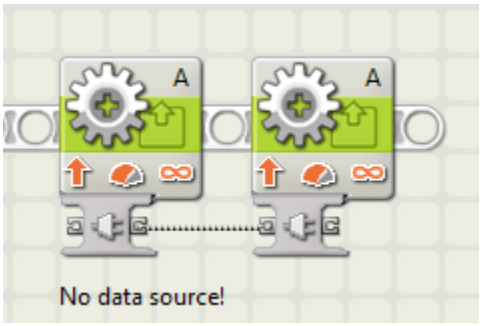
Herhangi bir veri, veri fişinden çıktığı ve diğer bir bloğa girişi boyunca izlediği yola kablo yolu denir. (Resim 4)



Resim 4- Örnek kablo yolu

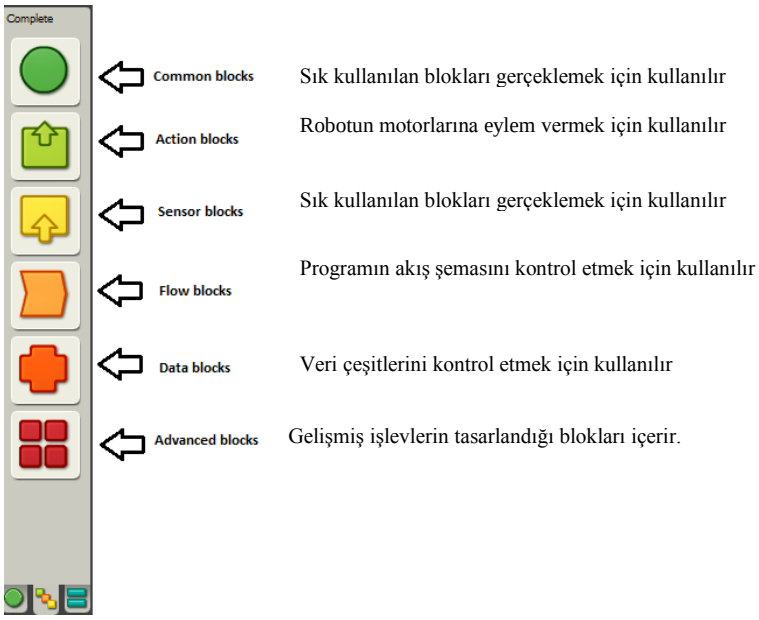
Kırık veri kabloları

Eğer veri kablosuna gerçekli olmayani eksik girdi verildiğinde(Resim 5),veya birden çok girdi verildiğinde veri yolu "kırılır".



Resim 5-Girdi eksiği olan kırık bir veri yolu

Tüm Görsel Palet



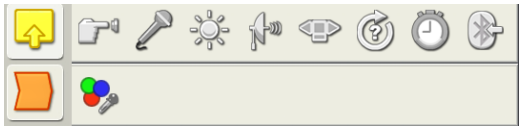
Resim 6-Görsel palet içerisinde altı programlama blok kategorisi bulunur.

Genel Bloklar

Resim 7-Genel bloklar altı adet alt-bloktan oluşur. Bunlar Hareket Bloğu, Kaydet/Oynat Bloğu, ses bloğu, görüntü bloğu, bekleme bloğu, döngü bloğu ve değiştirme bloğu

Eylem Blokları

Resim 8-Eylem bloğu içerisinde motor bloğu, ses bloğu, görüntü bloğu, mesaj gönderme blou ve ışık lambası bloğundan oluşur



Resim 9- Sensör bloğu içerisinde dokuz farklı kategoriyi içerir. Bunlar dokunma sensörü bloğu, ses sensörü bloğu, ışık sensörü bloğu, ultrasonik sensör bloğu, NXT tuşları sensör bloğu, döndürme sensörü bloğu, mesaj alma sensörü bloğu ve renk sensörü bloğudur

Not: Lego Mindstorms NXT 2.0 kiti(#8547) kiti içerisinde ses ya da ışık sensörü bulundurmaz bu özellikleri kullanabilmek için bu ürünleri satın almak gerekir.

Akış Şeması Bloğu

Resim 10- Akış şeması bloğu bekleme bloğu, döngü bloğu, değiştirme bloğu ve bekleme bloğundan oluşur

Veri Blokları

Resim 11- Veri blokları yedi farklı veri bloğunu içerisinde bulundurur. Bunlar Mantık bloğu, Matematik bloğu, Karşılaştırma blığu, Değer aralığı bloğu, Rastgele bloğu, Değişken bloğu ve Sabit bloktur

Gelişmiş Bloklar

Resim 12- Gelişmiş Blok içerisinde yedi farklı alt blok bulundurur. Bunlar Rakamı harfe dönüştür, Yazı bloğu, Canlı tut bloğu, Dosya erişim bloğu, Kalibrasyon bloğu, Motor sıfırlama bloğu, Bluetooth bağlantı bloğudur.

Benim bloklarım, standart bir programlama bloğudur. (Birden fazla bloğu bir araya alan bir blokmuş gibi gösteren programlama bloğudur)

Web bloğu başkasının internette tasarlamış olduğu "*Benim bloğum*" örneklemesini programda kullanmak için tasarlanmıştır.