1. LabVIEW ile Programlama

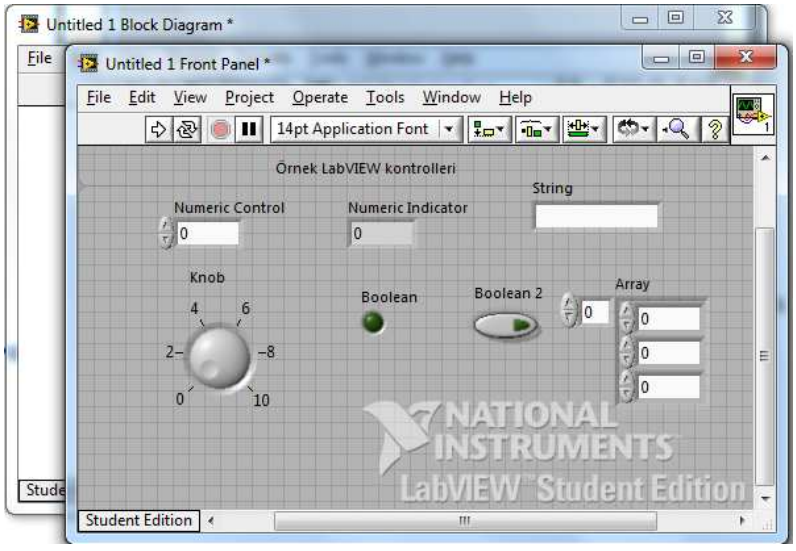
LabVIEW ile programlama mantığı, program kodu yazılan programlama mantığına benzemekle birlikte, kontrol adı verilen nesneler arasında veri yolu bağlantısı ile program akışı sağlanır. Kontroller içindeki veri elektrik kablosuna benzer çizgi üzerinden bağlı olduğu kontrole geçer. Labview'de içinde veri bulunan üç çeşit nesne bulunur.

Control : Kullanıcının program çalışırken veri girdiği nesnelerdir. Çeşitli şekilleri bulunur.

Indicator : Kullanıcı program çalışırken veri giremez, hesaplama sonucu bulunur. Çeşitli şekilleri vardır.

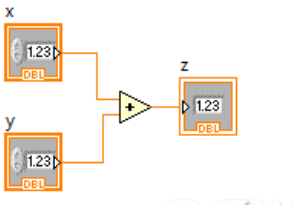
Constant : Sabit değerdir. Program çalışırken değiştirilemez.

Control ve indicator birbirine benzerdir. Control isimli nesnelerde bulunan veri veri yolları üzerinden indicator isimli nesneye veya matematiksel sembollere doğru akar. Elektrikte iki farklı fazın birbirine bağlanması hata meydana getirir, benzer şekilde farklı controllara bağlı veri yolu bir birine bağlanmamalıdır. Bir kontrole ve indikatöre sadece bir veri yolu bağlanabilir. Elektrikte bir kablonun ortasına başka bir kablo bağlasak bu kabloda aynı voltta elektrik olur, aynı şekilde bir veri yolu kablosunun herhangi bir yerine başka bir veri yolu kablosu bağlasak, bu kabloda da aynı veri bulunur.



Resim 1 LabVIEW genel görünümü

LabVIEW'de iki pencere vardır. Görsel nesnelerin bulunduğu Front Panel, program akışının bulunduğu Block Diagram penceresi. Programlama block diagram penceresinden yapılır. Şekil 1'da örnek bir Labview programı bulunur. Burada x ve y control, z ise indicatordür. x ve y controllerindeki veri  sembolü üzerine akar, bu sembolden çıkan veri z indicatörüne girer.



Şekil 1 Örnek LabVIEW diyagramı

Normal programlama mantığında, bir başlangıç vardır ve bir de sonu vardır yani bir yerde program akışı durur. LabVIEW'de ise program akışı, en uzaktaki indicatöre gittiğinde yine başa döner yani tüm akış bir sonsuz döngü içindedir. Döngü sonsuz olduğu için, control içindeki verinin işlem görmesi için bir olay tanımına ihtiyaç yoktur. Bir controle veri girildiğinde, o veri doğrudan kablo üzerinden akar ve kablonun bağlı olduğu yere gider.

Çok bilinen programlamada, programlar tek görevlidir. Yani bu programlar örneğin aynı anda iki sayıyı toplayamaz. Bu toplama işlemini sıralı olarak yaparlar. LabVIEW ise çok görevlidir yani aynı anda iki toplama işlemini yapar. Bütün işlemler paralel yürür. Her an veri yolu kablolarında veri vardır. LabVIEW'de paralel çalışan birden fazla döngü bulunabilir.

1.1. LabVIEW veri tipleri

Blok diyagramında nesnelerin iki görünümü vardır.  ve  görünümleridir. Labview sıfıra bölme hatası vermez. +inf : $+\infty$ ve –inf : $-\infty$ 'dur. Tablo 1'de çok kullanılan veri tipleri görülmektedir.

1.2. LabVIEW fonksiyonları

LabVIEW'de fonksiyonların sembolleri vardır. Bu sembollerin giriş ve çıkış terminalleri bulunur. Örneğin,  sembolü bir toplama fonksiyonudur. İki girişi, bir de çıkışı vardır. Blok diyagramında fare sağ tuşuna tıklandığında, fonksiyonların listesi görünür. Bu listede fonksiyonlar görevlerine göre alt gruplara ayrılmıştır.

Tablo 1 LabVIEW veri tipleri

Control	Indicator	Veri Tipi	Control	Indicator	Veri tipi
		Tek duyarlı ondalık sayı			Tek duyarlı kompleks sayı
		Çift duyarlı ondalık sayı			Çift duyarlı kompleks sayı
		Ekstra duyarlı ondalıklı sayı			Ekstra duyarlı kompleks sayı
		8 bit işaretli tamsayı			8 bit işaretsiz tamsayı
		16 bit işaretli tamsayı			16 bit işaretsiz tamsayı
		32 bit işaretli tamsayı			32 bit işaretsiz tamsayı
		64 bit işaretli tamsayı			64 bit işaretsiz tamsayı
		Boolean (mantıksal)			Enumerated tip
		String (metin)			Path
		Dizi (veri tipi belirlenmemiş)			Matris (çift duyarlı ondalıklı sayı)

Kullanmak istediğimiz fonksiyon tıklanarak blok diyagramına yerleştirilir ve kablo bağlantısı yapılır.

Yandaki şekilde fonksiyon grupları görülmektedir. Fonksiyon gruplarının sayısı yüklü modüllerin sayısına bağlıdır.

Programming, bir VI için temel fonksiyonlar bulunur.

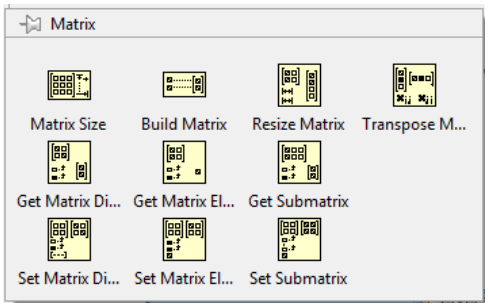
Mathematics, matematiksel analiz için çok çeşitli platformlar sunar. Matematik algoritmalar, çeşitli pratik çözümler vardır.

Resim 2

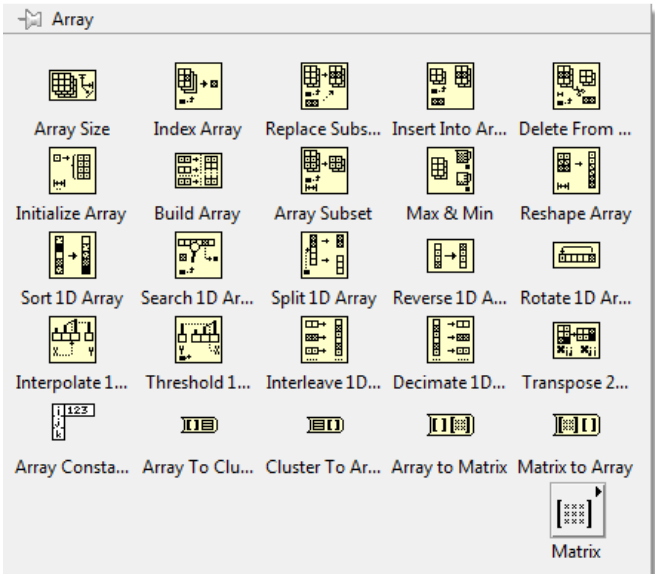
1.3. LabVIEW'de diziler

Front Panel'de fare sağ tuşuna basıldığında kontrol listesi görülür. Bu kontrol listesinde  sembolüne tıklanır. İçi boş bir dizi nesnesi oluşur. LabVIEW'de dizi bir kontrol dizisidir. Diyelim ki biz sayı dizisi oluşturmak istiyoruz, yine sağ tuşa tıklayarak sayı kontrolünü seçeriz ve bu kontrolü boş olan dizinin içine yerleştiririz.

Diziler varsayılan olarak tek boyutludur. Dizi nesnesi üzerinde fare sağ tuşuna basılarak, "add dimension" tıklanarak diziye yeni boyut eklenmiş olur. Diziler bir boyutlu ise vektör, iki boyutlu ise matris adını alır. Vektör ve matrisler üzerinde matematiksel işlem yapmak için özel fonksiyon yazmaya gerek yoktur çünkü toplama, çıkartma, ters matris bulma, devrik matris bulma gibi her türlü matematik fonksiyon mevcuttur.



Resim 3 Matris foksiyonları



Resim 4 Dizi fonksiyonları

1.4. Döngüler ve yapılar

1.4.1. For Döngüsü

For döngüsü, görünümü aşağıdadır. Döngü içindeki alt diyagram, girilen değer kez yani N kez çalıştırılır.

Dışarıdan değer alan bir sayı ucu vardır.

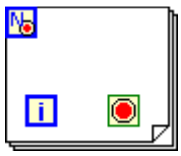
N: Sayı ucudur. Bu uçtan döngünün kaç kez döneceğinin sayısı girilir. i sayısı N sayısına ulaştığında döngü sonlanır.

i : İterasyon ucudur. Döngünün iterasyon numarasıdır. Her zaman sıfırdan başlar.

Sayı ve iterasyon uçları 32 bit işaretli tamsayıdır.

1.4.2. Mantıksal uç olan for döngüsü

Yukarıdaki for döngüsü ile aynıdır. Döngüden erken çıkmak için kullanılır. Blok diyagramında, for döngüsü seçilerek fare sağ tuşuna basılır ve “Conditional terminal” tıklanır. Bu yapıda, bir koşula bağlı olarak döngüden çıkılabilir.



: Bu uca true değer gönderildiğinde döngü durur. True değerin döngüyü devam ettirmesi istendiğinde, yani false değeri ile döngü durdurulmak istendiğinde bu sembol üzerinde fare sağ tuşuna basılır “Continue if True” tıklanır.

: Bu uca false değer gönderildiğinde döngü durur.

1.4.3. While döngüsü

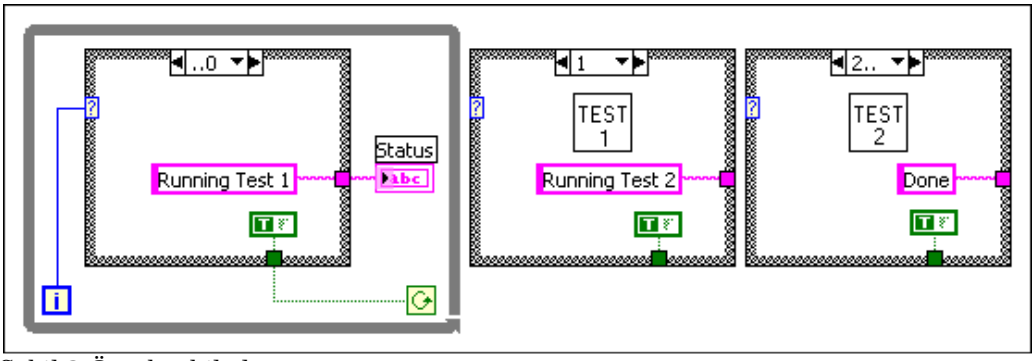
Metin tabanlı programlama dillerindeki while döngüsüne benzerdir. Koşul gerçekleştiğinde döngüden çıkılır. Aşağıdaki şekle benzerdir. Kontrol ucu yapısı for döngüsü ile aynıdır. veya uçları kontrol uçlarıdır. True ile veya false ile çıkılması isteğine göre biri seçilir. ise her zaman sıfırdan başlayan, iterasyon numarasını tutan nesnedir.

1.4.4. Case yapısı

Duruma bağlı olarak çalışacak diyagram tanımı yapılır. Görünümü aşağıdadır.

True : kullanılan durumlar, buraya yazılır. Burada yazılı durum geldiğinde, case yapısı içindeki blok şeması çalışır.

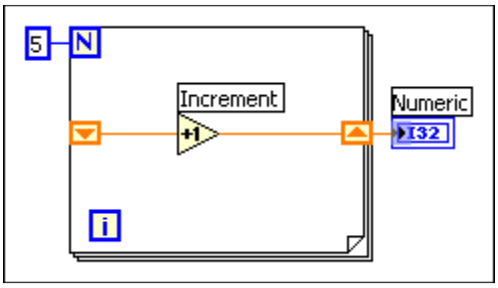
?: Bu uca, sayı, boolean, string ve enumerated tipi değerin koblosu bağlanabilir. Buradan gelen değere bağlı olarak, hangi bloğun çalışacağına karar verir.



Şekil 2 Örnek while loop ve case yapısı

1.4.5. Shift Register

Gelecek döngü iterasyonunda, iterasyonun kaldığı yerden devam etmesi istenebilir. Shift register döngü çerçevesinin yanına konur. sembolü shift registerdir.

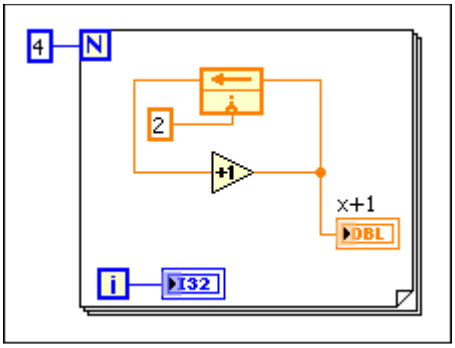


Şekil 3 Shift register örneği

Yukarıdaki şekilde shift registerin arasında Increment fonksiyonu vardır. Shift registere Numeric isimli nesne bağlıdır. Bu nesnenin değeri, increment fonksiyonu sayesinde sürekli bir artar aynı zamanda bu nesnenin değeri hiç başa dönmez. Program çalıştığı sürece sürekli artar.

1.4.6. Feedback Node

Shift registerin benzeri işleve sahiptir. Farkı döngü yeniden başlayınca nesne ilk değerini alır. Görünümü aşağıdadır.



Şekil 4 Feedback node örneği

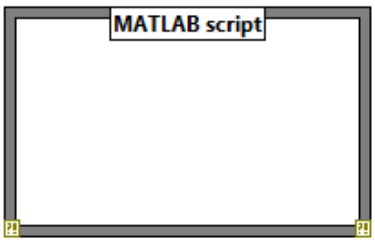
1.4.7. MathScript Yapısı

Görünümü aşağıdadır. Buradaki pencereye metin tabanlı script kodları yazılır. Matlab'a benzer komutlar yazılabilir. Bu yapı LabVIEWin ek bir modülüdür. Script'in çerçevesi üzerinde fare sağ tuşuna basıldığında gelen kısayol menüsünde "Add Input" tıklayarak giriş ucu, "Add Output" u tıklayarak çıkış ucu tanımlanır. Bu uçlara kablo bağlayarak veri gönderilir.

Şekil 5 MathScript penceresi

1.4.8. Matlab Script Yapısı

MathScript'e benzer yapıdadır. Matlab scriptini kullanabilmek için bilgisayara Matlab programı yüklü olmalıdır. Doğrudan matlabı kullanır. Bu nedenle matlaba ait tüm fonksiyonları, kendi fonksiyonu gibi kullanır. Görünümü aşağıdadır.



Şekil 6 MATLAB Script penceresi

1.5. SubVI (Alt VI belgesi)

Bir VI belgesi içinden başka bir VI çağrılabilir. Bir VI dosyası bir fonksiyon olarak kullanılabilir. Bu VI'den yapılan fonksiyona parametre göndermek için aşağıdaki işlemler yapılır.

- Front Panel'de sağ üst köşede VI dosyasının sembolü vardır. Bu sembol üzerinde fare sağ tuşuna basınız.
- Açılan kısayol menüsünden "Show connector" ü tıklayınız.
- Connector'deki her bir hücre'ye parametre ataması yapılır.
- Connector'ün hücre yapısı değiştirilebilir. Değiştirmek için connector üzerinde fare sağ tuşuna basınız. Kısayol menüsünde "Patterns"i tıklayarak uygun hücre şeklini seçiniz.
- Connector hücresine, control ataması yapılır.
    - Önce atama yapılacak hücreye tıklayınız,

- Sonra bu hücreye atanacak controlü tıklayınız. Seçili hücreye control atanmıştır.

- Dosyayı kaydediniz.

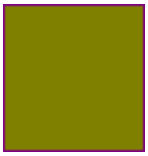
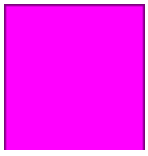
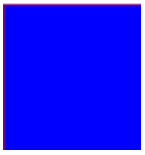
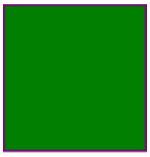
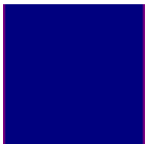
Yukarıdaki şekilde tanımlanmış bir VI dosyasını, başka bir VI dosyasına eklemek için aşağıdaki adımlar uygulanır.

- VI dosyasının Block Diagram sayfasına geçiniz.
- Fare sağ tuşuna tıklayınız. Fonksiyonlar kısa yolu açılacaktır.
- "Select a VI.." tıklayınız. Ekleyeceğiniz VI dosyasını seçiniz. OK tıklayınız.
- Terminal uçlarına kablo bağlantılarını yapınız.
- Dosyayı kaydedip çalıştırınız.

1.6. LabVIEW Kablo Renkleri

LabVIEW'de taşıdığı veriye göre kablolar renklenmiştir.

Tablo 2 LabVIEW kablo renkleri

				
RefNum	Ondalıklı Sayı	Variant	Hata hattı	String
				
Tamsayı	Sayısal Cluster/ Analog Dalga	Boolean Sayısal Dalga	Express Dinamik veri	Void Bozuk hat