AirHMI

LCD

EKRAN

EDİTÖR

KILAVUZU

AİRHMI LCD EKRAN EDITOR KILAVUZU

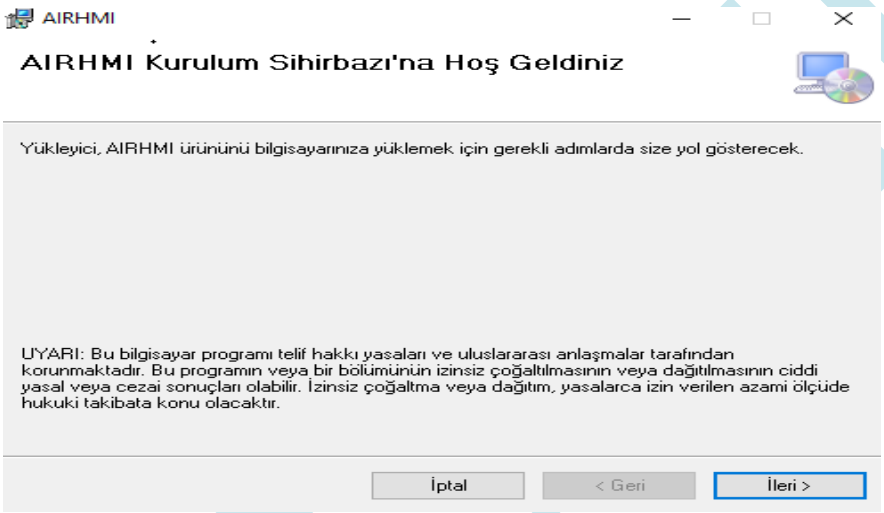
AirHMI Visual Screen Creator, AirHMI LCD ekranları için İnsan Makine Arayüzü GUI'lerini tasarım açışından en üst seviyede memnuniyet ve en verimli sürede oluşturabilmek amacıyla tasarlanmıştır. Editör kullanımında Tasarım ve Programlama dünyasına ait işlevselliklerimiz bulunmaktadır: Görsellik açısından zengin nesne hazinesinden özgün olabileceğiniz ve istekleriniz doğrultusunda rahatlıkla oluşturabileceğiniz ekran tasarımı desteğinin yanı sıra programlama kısmında da kullanıcıya birçok kolaylık sağlamaktadır.

İÇİNDEKİLER

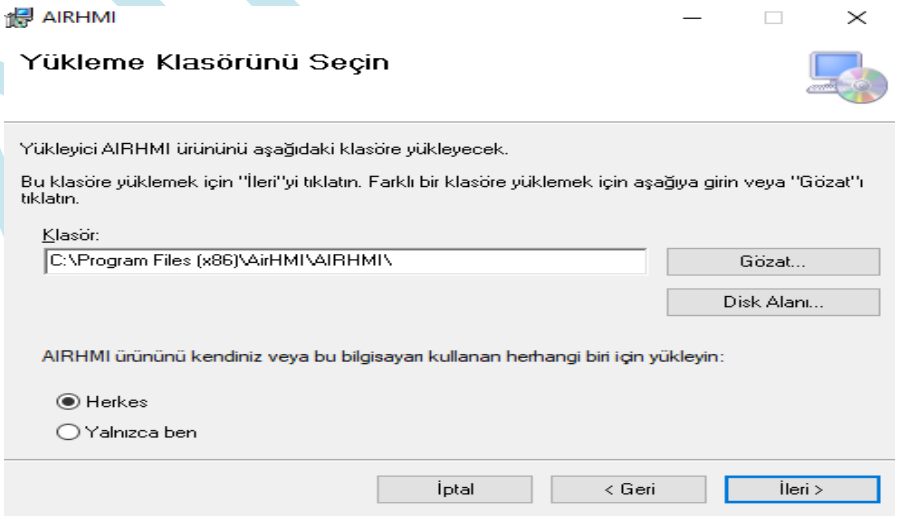
1. AirHMI Visual Screen Creator KURULUMU

İndirme Linki: https://www.airhmi.com/airhmi-visualcreator

AirHMI Editör'ü bilgisayarınıza yüklemek için AIRHMISETUP.msi dosyasına çift tıkayın. Bu işlemden sonra aşağıdaki adımları takip ediniz.



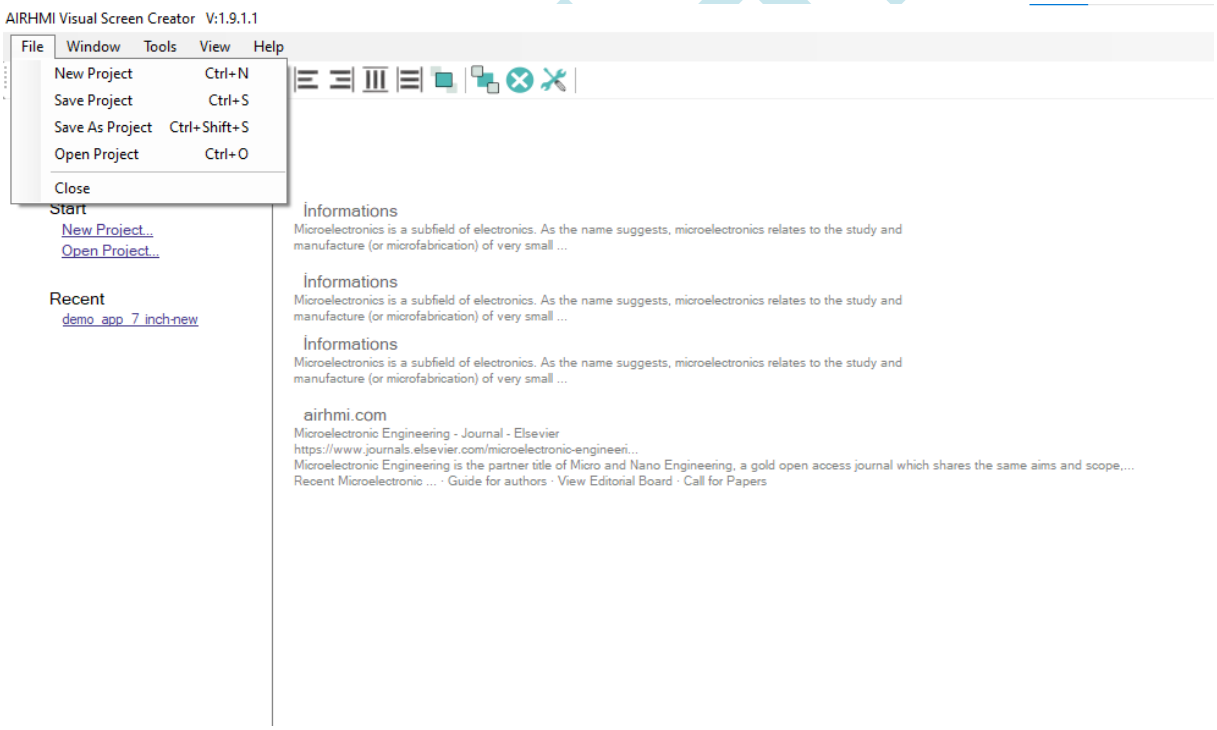
Yükleme klasörünü ve diğer seçenekleri istediğiniz şekilde seçip ileri tuşuna basarak yükleme başlatılır.

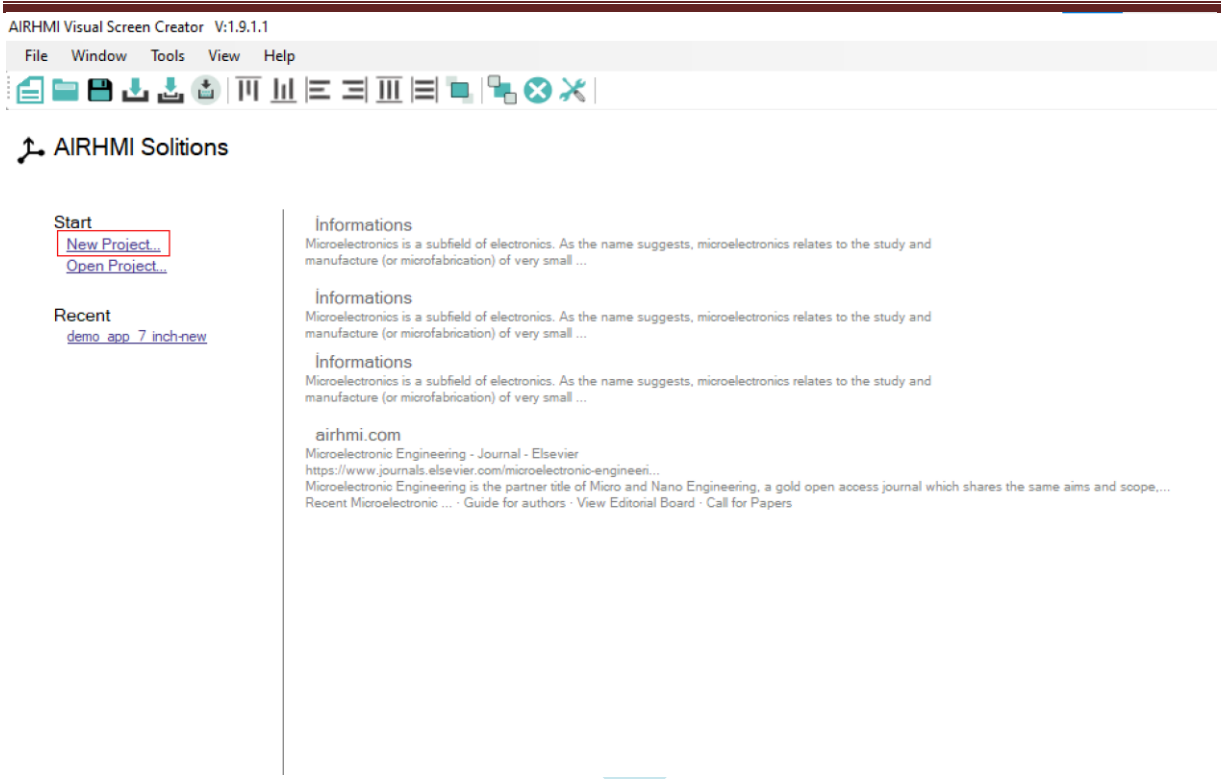


2. PROJE OLUŞTURMA

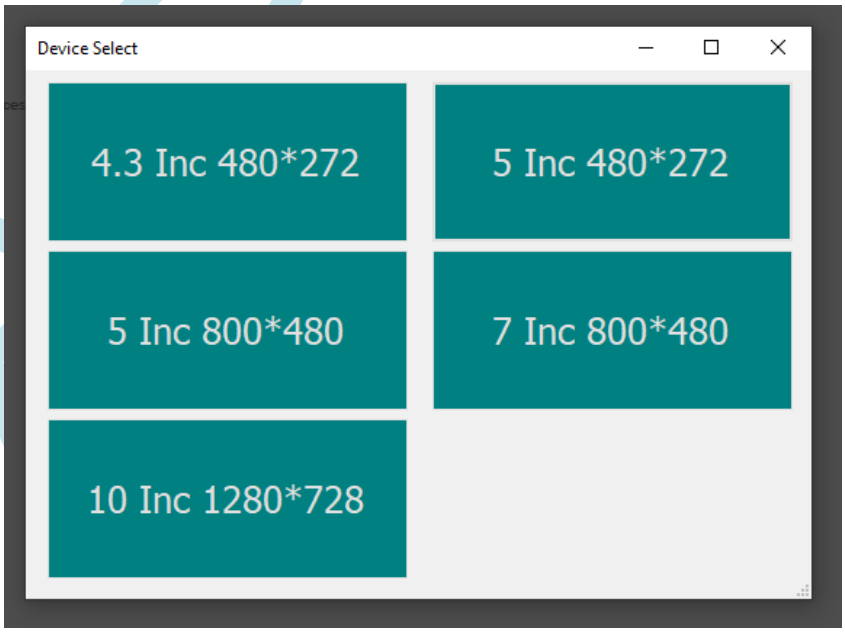
AirHMI ile arayüz oluşturmak için öncelikle AirHMI Editör programını indirip bilgisayarınıza kurmanız gerekmektedir. AirHMI Editör programındaki sürükle-bırak özelliği arayüz geliştirmeyi kolaylaştırmaktadır. AirHMI Editörü ile projelerinize, Buton, Resim, Yazı, İlerleme çubuğu, Gauge, Key, Analog ve Dijital değerleri görmek için sayısal giriş ve çıkışlar gibi birçok bileşen ekleyebilirsiniz.

Programın kurulumu oldukça kolaydır. Kurulumu yaptıktan sonra AirHMI Editör programı çalıştırmalısınız. Karşınıza aşağıdaki resimlerde görüldüğü gibi bir sayfa çıkacaktır. Bu sayfadan sol üst köşede bulunan File – New yolunu izleyerek veya programın ilk açılış sayfasında karşınıza çıkan sekmelerden New Project'e tıklayarak projenizi oluşturuyorsunuz.





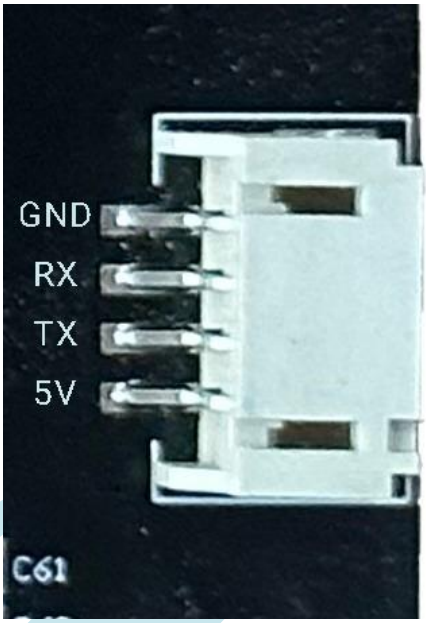
Kayıt işleminden sonra karşınıza aşağıdaki resimde görüldüğü gibi bir sayfa çıkacaktır. Karşınıza çıkan sayfada Ekrana ait boyut ve çözünürlük ile ilgili ayarlar yapılmalıdır.



3. CİHAZ BAĞLANTISI

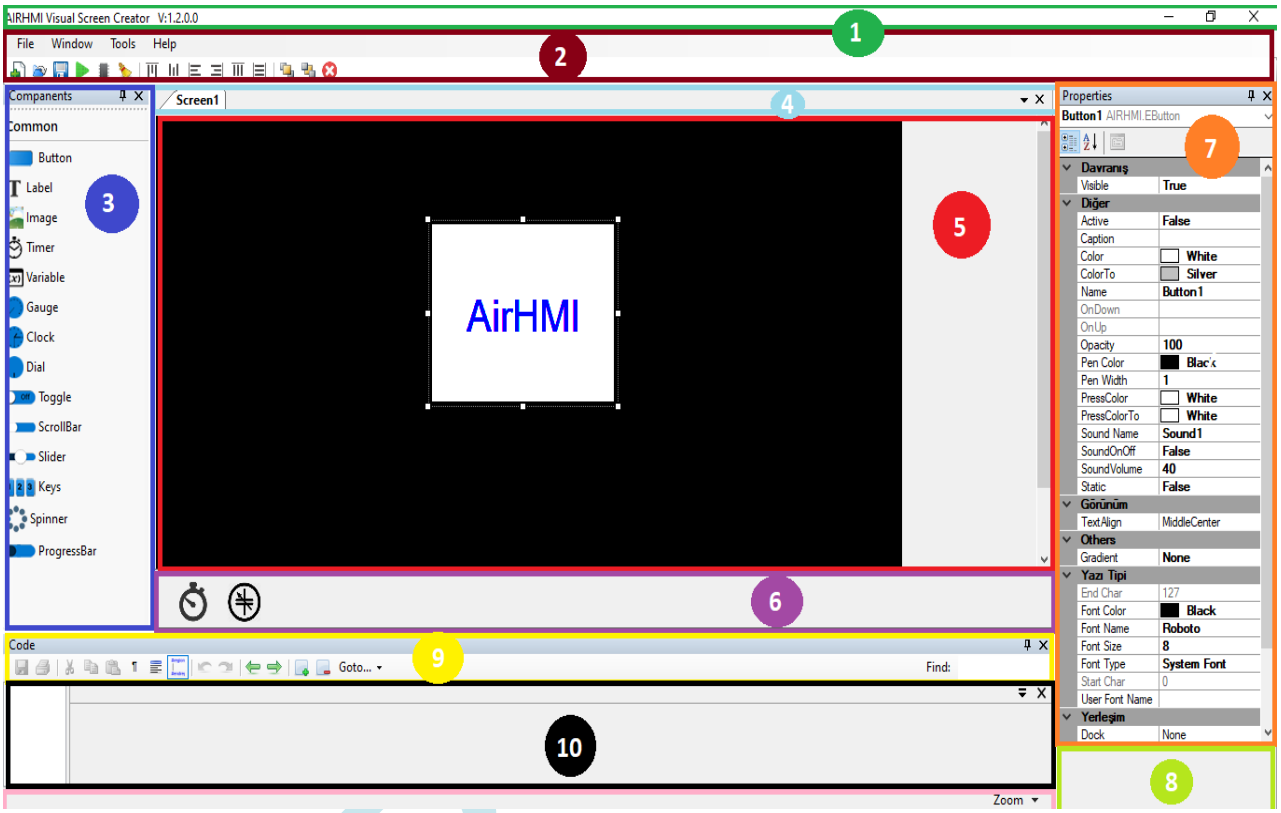
AirHMI ekrana enerji verdiğimiz power konnektör dört pinlidir. 1 ve 4 besleme , orta iki pin ise uart haberleşme pinleridir.

1) POWER konektöre ait pinler şu şekildedir;



Uyarı: 5V beslemeyi ters vermeyiniz. Beslemeyi ters vermeniz durumunda ekranınız zarar görebilir.

4. AirHMI EDİTOR ANA ARAYÜZÜ



4.1 BAŞLIK ÇUBUĞU

Başlık Çubuğu, bir AirHMI projesi açıldığında uygulama ismini ve versiyon numarasını içerir.

4.2 ANA MENÜ ve ARAÇ ÇUBUKLARI

Dosya (File) Menüsü

Kullanıcılar için Yeni Proje Açın, Projeyi Kaydet, Projeyi Farklı Kaydet, Var Olan Bir Projeyi Açın ve Çıkış gibi komutlar bulunmaktadır. Burada önemli olan nokta var olan bir proje

açıkken yeni proje açmak istenildiğinde eski projenin bilgisayarda saklanması ya da yapılan değişikliklerin kaybolmaması isteniyorsa ekrana gelen kaydet mesajına onay verilmelidir.

Pencere (Window)

Pencere alanı içerisinde ;

- Projede kullanılan ana ekrana ek yeni çalışma ekranı oluşturma (Add Screen)
- Tasarlanan arayüz ekranının seçili USB port üzerinden AirHMI LCD Kartına yüklenmesi (Download to Flash)
- Tasarlanan arayüz ekranının harici dosyalar halinde bilgisayar içerisinde istenilen bir dosyaya çıkartılması (Download to SD Kart). USB yüklemenin istenmediği durumlarda SD Kart üzerinden Bootloader yükleme yapmak için kullanılmaktadır. Dosyalar SD karta kopyalanıp proje SD Kart üzerinden çalıştırıldığında dosyalar USB üzerinden yüklenir gibi SD Kart'tan yüklenmektedir.

Araçlar (Tools)

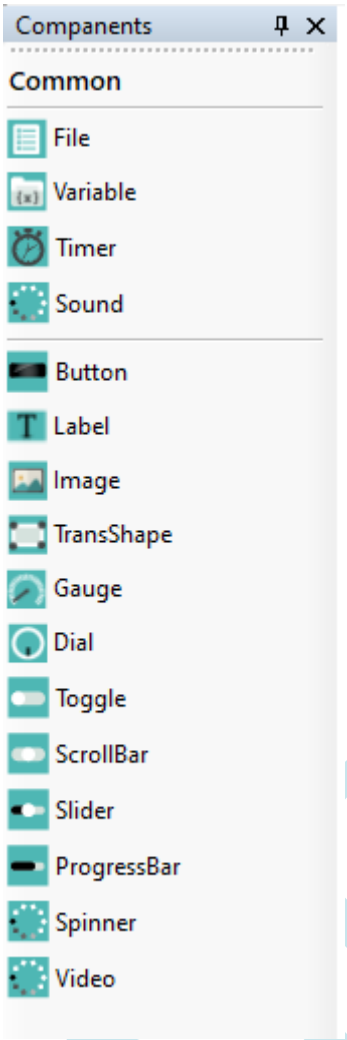
Araçlar içerisinde Options içerisinde USB yükleme için port seçme ve baud rate ayarlama bölümü bulunmaktadır. USB yükleme birçok baud rate değerinde çalıştığı için kullanıcı istediği baud rate ayarını seçerek yüklemesini gerçekleştirebilmektedir.

Hizalama

Sola Hizala, Sağa Hizala, Üst Hizala ve Alta Hizala; dikey ve yatay olarak ortalama özellikleri sayesinde belirlenen nesneler istenen şekilde hizalanmış veya ortalanmış hale getirilir.

Öne Getir ve Arkaya Gönder özellikleri sayesinde iç içe geçen nesnelerin hangisinin önde duracağı belirlenebilir ve arka planda durması istenen nesneler için kullanılır.

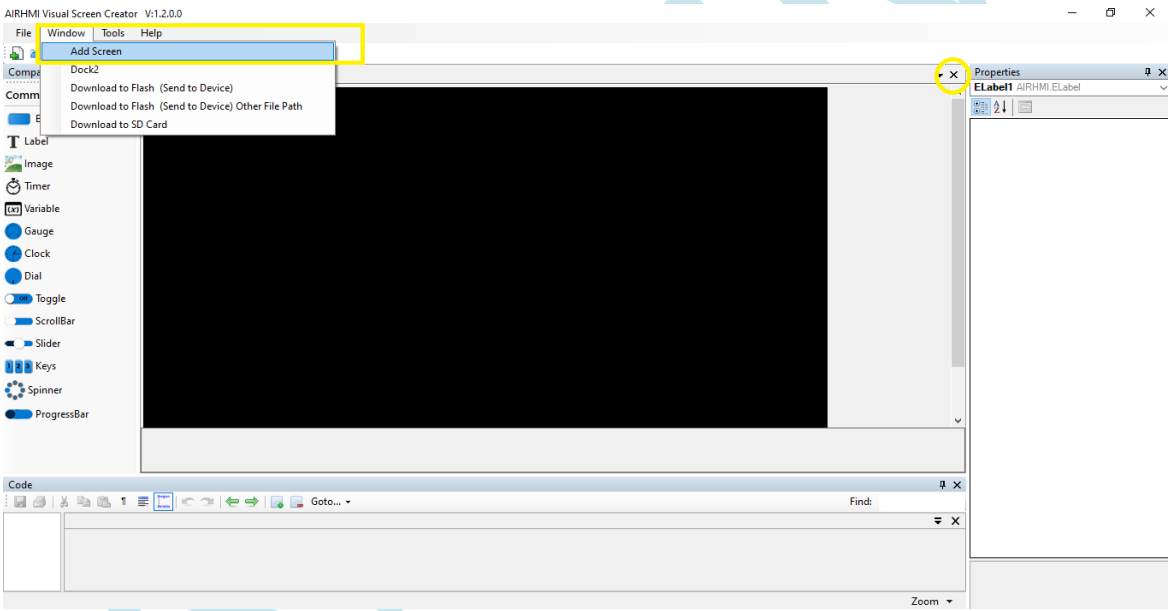
4.3 BİLEŞENLER BÖLMESİ



AirHMI LCD Tasarım Ekranı'nda gösterilecek hazır nesnelerin bulunduğu bölümdür. Kullanılmak istenilen nesne üzerine tıklanıp ekran alanına sürüklenerek projeye eklenmektedir. Ekranda gösterilmeyen harici nesneler de bu bölümde bulunmaktadır: Timer ve Variable. Bu nesneler ekran alanının alt kısmında Görseli Olmayan Bileşenlerin Alanı bölümünde bulunmaktadır. Tasarlanan proje özelinde nesnelerin özelliklerini (konumu, boyutu, ismi, vb…) ayarlama Nesnelerin Öznitelik Alanı adlı bölümde bulunmaktadır.

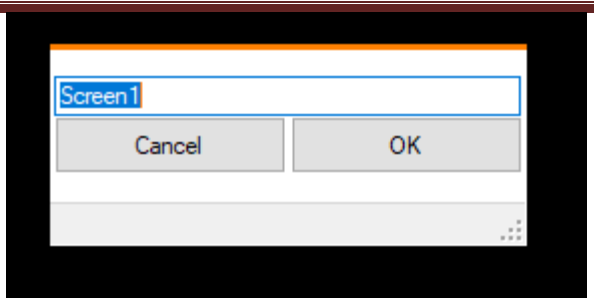
4.4 EKRAN / KOMUT SEKMESİ

Tasarım projeleri genelde tek ekran olarak kullanılmayıp aynı anda farklı ekranlara ihtiyaç duymaktadır. Açılış Genel Gösterim Ekranı, Menü Ayar Ekranı, Detaylı Gösterim Ekranı vs… Bu nedenle AirHMI Editör içerisinde kullanıcı istekleri doğrultusunda birden fazla özgün ve yaratıcı ekran tasarımı yapabilmektedir. Ekran / Komut Sekmesi ile hangi ekranda çalışma yapılacağını seçme işlemi gerçekleştirilmektedir.



Yeni çalışma ekranı eklemek için Window/Add Screen sekmesi kullanılabilir veya çalışma sayfası üzerinde boş bir yerde sağ tıklanarak Add Screen seçilebilir. Açılmış olan çalışma sayfasını silmek için Ekran / Komut Sekmesi satırının sonunda yer alan çarpı(x) işaretine basmak yeterli olacaktır.

Ekranın ismini değiştirmek için ekranda boş bir alanda sağ tıklayarak Rename sekmesine tıklanır. Açılan sekmeden ekranın ismi değiştirilebilir.



4.5 TASARIM ANA EKRAN ALANI

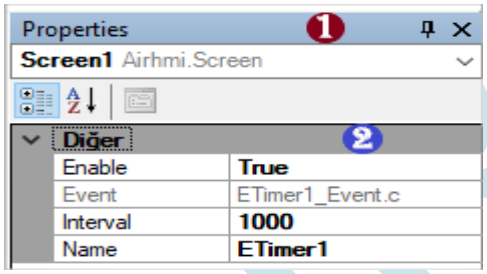
AIR HMI Designer çalışma ekranı tasarım görseli alanıdır. LCD Ekran tasarımında hangi nesnelerin ekranda nerede bulunacağı, boyutları, yazı özellikleri gibi özellikler bu alanda gösterilmektedir.

AHMI SCREEN EDITOR

4.6 GÖRSELİ OLMAYAN BİLEŞENLERİN ALANI

Hazırlanan bir projede bileşenlerin hepsi LCD ekranda gösterilmemektedir. Arka planda çok önemli görevlerde yer alırken LCD ekran üzerinde gösterilmesine gerek olmayan bileşenler de mevcuttur: Timer ve Variable gibi. LCD ekranda gösterilmeyen fakat tasarım esnasında kullanım kolaylığı sağlayabilmesi ve anlaşılabilir olabilmesi için arka planda çalışan bileşenlerin Editör içerisinde gösterilmesi önemlidir. Görseli Olmayan Bileşenlerin Alanı bu doğrultuda projede kullanılan Timer ve Variable gibi bileşenlerin gösterildiği alandır.

4.7 NESNELERİN ÖZNİTELİK ALANI



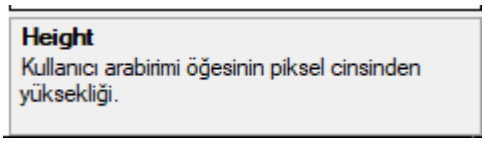
4.8 3.7.1 Projede Kullanılan Nesnelerin Gösterim Alanı

LCD ekran tasarımında birçok nesne kullanımı gerçekleştirebilmektedir. Her nesnenin kendine özgü ayarları yapılmaktadır. Fazla detay istenilen projelerde özellikle ayar yapılmak istenilen nesnenin tasarım ekranından bulunması karmaşık bir hal alabilmektedir. Bu karmaşıklığı önlemek için tasarımda kullanılan bütün nesnelerin listesinin bulunduğu alandır. Bu sayede istenilen nesne seçilip Öznitelik alanında ayarları gerçekleştirilebilmektedir.

4.9 3.7.2 Nesnelerin Öznitelikleri Gösterim / Ayar Alanı

AirHMI Editör'de nesneler projeye dahil edildiklerinde otomatik olarak ilk ayarları ile eklenmektedir. Kullanıcılar kullanım amaçları ve istekleri doğrultusunda ekledikleri nesnelerin isimleri, boyutları, görünümleri, renkleri gibi birçok özelliğini bu alanda düzenleyebilmektedir.

4.10 ÖZNİTELİKLERİN AÇIKLAMA ALANI



Nesnelerin ayarları öznitelik alanında gerçekleştirilmektedir. Fakat orada sadece öznitelik ismi yazmaktadır. Özniteliklerin Açıklama Alanında ise özniteliklerin açıklama kısmı bulunmaktadır. Öznitelik başlıklarının hangi işlevleri yerine getirdiği genel olarak açıklanmıştır.

4.11 KULLANICI PROJE KODU MENÜ ve ARAÇ ÇUBUKLAR

Tasarlanan projede en önemli kısım kod aşamasıdır. Proje temeline göre tasarım ekranında hangi durumlarda nelerin gösterileceği kodlama yapısı ile ayarlanmaktadır. Kod Menüsü kullanıcıya kod yazımında kodu kaydet, kopyala yapıştır, kod içerisinde anahtar kelime ara ve benzeri konularda yardımcı olabilecek bazı temel bileşenleri içermektedir.

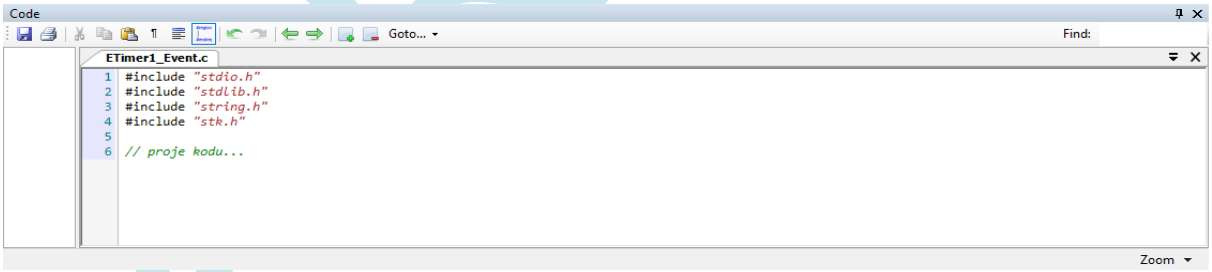
4.12 KULLANICI PROJE KOD ALANI

Kullanıcı Proje Kodu için geçerli bir AirHMI PICOC Kod Talimatını içerir. Bu bölüm programlamayı öğretmeyecek, ancak kullanıcının kod ekleyebilmesi için genel olarak yardımcı olacaktır. Bu alan içerisinde kullanıcılar ister Timer componentinin event'larına isterlerse de ekranda kullandıkları nesnelerin event'larına C tabanlı kodları yazabilecektir. Screen Editor'un desteklediği hazır kütüphane kodları sayesinde yazılım zorluğu minimum seviyeye indirilen bu bölüm için hazır fonksiyonları üçüncü başlık altında (3. Fonksiyonlar) detaylı bir şeklide inceleyebilirsiniz. Orada belirtilen fonksiyonlara ek olarak C tabanlı kodların tamamı bu alana yazılarak programda eş zamanlı olarak çalıştırılabilmektedir.

4.13 KOD ALANI ZOOM ALANI

Proje tasarımında kod alanı yazı boyutunun kullanıcıya kullanımda kolaylık sağlaması için istenilen ölçüde yakınlaştırma ve uzaklaştırma yapabileceği alandır.

4.14 KOD ALANI



AirHMI Editör'ün çözüm odaklı, zaman ve efor konularında en verimli noktada tasarım oluşturmayı hedefleyen yapısının yanında en önemli avantajlarından biri de kolay ve anlaşılabilir kod yapısıdır. Kod yapısı C programlama dilinde hazırlanmıştır. Fakat kullanıcı odaklı olması ve kullanıcıya kullanımda kolaylık sağlayabilmesi için gerekli fonksiyonlar "stk.h" kütüphanesi altında hazırlanmıştır. Temel C kütüphanelerinin ekli olduğu bu düzende C programlama dilini kullanarak kodunuzu oluşturabilir ve gerekli fonksiyonları kodunuzun

başına ekleyebilirsiniz. Hazır C fonksiyonlarına ek olarak nesnelerin kontrol/ayar fonksiyonları, LCD ekran uyku modu, zamanlayıcı kod düzeni gibi önemli birçok konuda hazır fonksiyonları açıklamaları ile birlikte bu kılavuzda bulabilirsiniz. Burada önemli olan nokta bu fonksiyonların aktif olarak çalışabilmesi için "stk.h" kütüphanesinin her kod yapısının başına eklenmesi gerektiğidir.

ETimer1_Event.c

```c
#include "stdio.h"
#include "stk.h"

char uartData[10];
int uartsize;
uartDataGet(uartData, &uartsize);

if(uartsize > 0)
{
    ImageSet ("EImage1" , "Visible" , "1");
    LabelSet ("ELabel1" , "Caption" , "Deneme");
    LocalIntVarSet("Varible1" , 2);

    DrawScreenGet();

}
```

Örnek kod yapısı timer ile hazırlanmıştır. Timer kod yapısı için detaylı anlatım **2.1 TIMER** başlığı altında anlatılmaktadır.

Kod yapısı istenilen duruma göre Timer içerisinde olabileceği gibi Rezistif ekranlar için nesnelere dokunulduğunda çalışmasını istediğimiz kod yapısı da oluşturulabilmektedir. Timer içerisinde Event içerisinde oluşturacağınız kod zamanlayıcı aralığınıza tüm programda aktif olarak çalışırken nesnelerin dokunulduğunda aktif olmasını istediğiniz kod yapısını aynı şekilde öznitelik kısmında bulunan OnUp kısmına eklenmesi gerekmektedir.

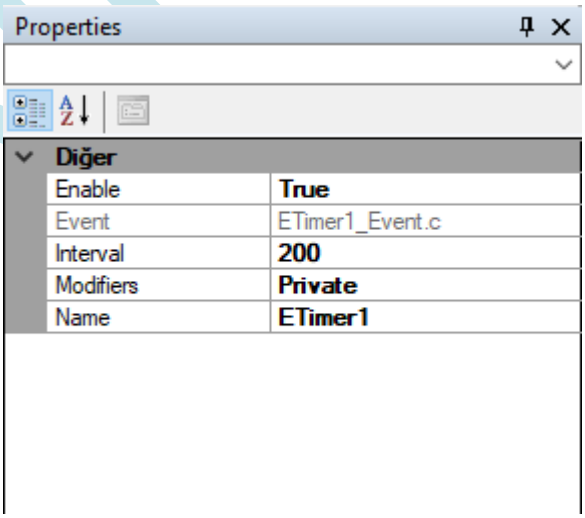
5. AİRHMİ NESNELERİ VE FONKSİYONLAR

5.1 TIMER

Kod yapısı içerisinde belki de en önemli nokta Timer kullanımıdır. Tasarlanan editör ekranının projede gerçek zamanlı çalışmasında oluşacak değişiklikler ve bu değişikliklerin hangi aralıklar ile olacağı Timer Özniteliklerinin içerisinde ayarlanmaktadır. Enable, Timer'ın aktif olup olmayacağını seçmektedir. Interval, milisaniye cinsinden hangi aralıklar ile kodun aktif olacağının seçildiği yerdir. Name, adında da anlaşılacağı gibi Timer'ın ismidir. Event bölümü ise proje tasarımı için oluşturulacak kod kısmını açma bölümüdür. ETimer1_Event.c ise oluşturulan kodun kaydedildiği C dosyasının ismidir.

Timer kullanımında kod yapısı, nesnelerin durumlarından bağımsız olarak Interval içerisinde ayarlanan süreye göre o aralıklarla kod dizinini aktif etmektedir. Kullanıcı eğer projesinde Rezistif bir ekran kullanıyor ve bir nesneye dokunulduğunda işlem yapmak istiyorsa; Dokunulduğunda işlem yapılmasını istediği nesnenin Öznitelikleri ayarlama kısmından OnUp kısmına gelip kodunu bu öznitelik altına eklemesi gerekmektedir. Böylece Timer'dan bağımsız olarak sadece o nesneye dokunulduğunda yazılan kod aktif olacaktır.

Timer Properties Penceresi



Özellik	Seçenek	Açıklama
Enable	True False	Timer nesnesine enable yapar. Timer nesnesine disable yapar.
Name		Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır.
Event		Timer nesnesi yazılım alanıdır.
İnterval		Timer tekrar süresini ayarlar.
Modifiers	Private Public	Sadece bu sayfada çalışan timerdir. Tüm sayfalarda çalışan timerdir.

Fonksiyonlar

1. TimerSet ( )

Açıklama

Buton nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void TimerSet(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	Açıklama
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Enable komutu

TimerSet(Nesne adı , “Enable” , “1 , 0 veya True , False”);

Örnek Kod:
ButtonSet (*"Timer1"* , *"Enable"* , *"True"*);

Interval komutu

TimerSet(Nesne adı , “Interval” , “Milisaniye cinsinden değer.”);

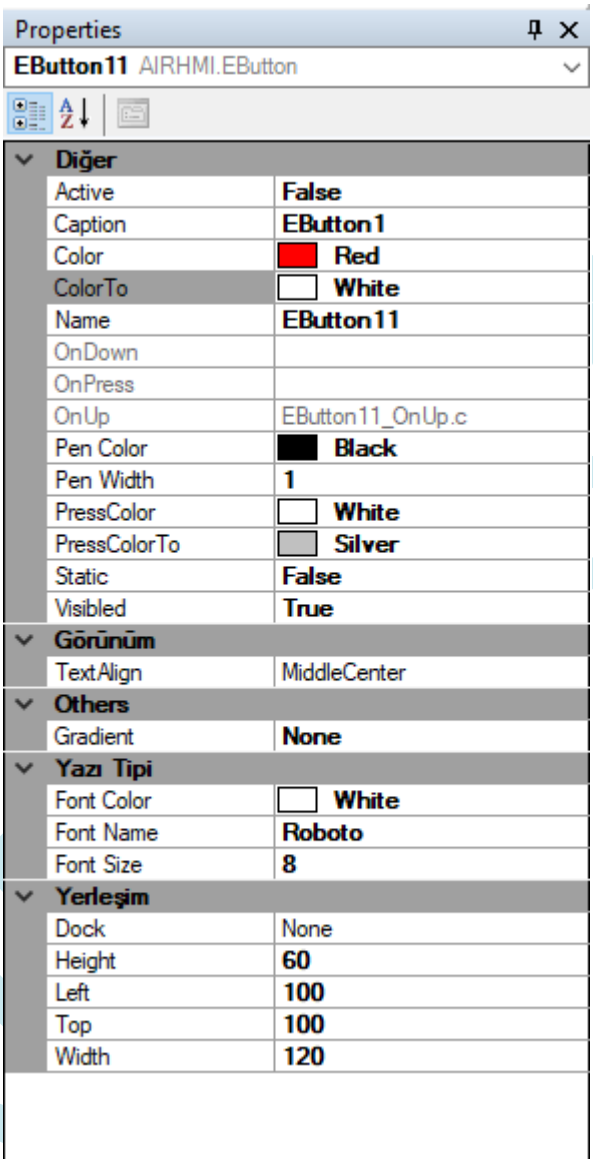
Örnek Kod:
ButtonSet (*"Timer1"* , *"Interval"* , *"1000"*); // interval 1 saniye olarak ayarlar.

5.2 Button

Buton nesnesi basıldığı zaman herhangi bir işlem yaptırmayı sağlayan nesnedir. Örneğin kullanıcıdan alınan veriyi bir yere göndermek, alınan veriyle işlem yapmak veya mesaj verdirmek amacıyla kullanılabilir. Butonun konumunu istediğiniz yere sürükleyebilir ve boyutunu kenarlarından çekerek ayarlayabilirsiniz.

Button Şekilleri

Button Properties Penceresi



Özellik	Seçenek	Açıklama
Active	True False	Buton nesnesine basma işlevine izin verir. Buton nesnesi basma işlevine izin vermez.
Caption,Text		Buton nesnesinin ekranda gözüken adıdır.
Color		Buton nesnesinin ekrandaki rengini belirtir.
ColorTo		Gradient özelliği seçili olur ise, ekranda geçişli bir buton nesnesi oluşur. Bu nesnenin Color dan ColorTo ya geçiş rengini tanımlamak için kullanılır.
Name		Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır.
OnDown		Buton nesnesine basma işlevi sırasında çalışan kod parçası buraya yazılır.
OnPress		Buton nesnesine elimizi basılı tuttuğumuz sürece çalışacak olan kod parçasıdır. Tekrarlı olarak çalışır.
OnUp		Buton nesnesinden elimizi çekme anında çalışan kod parçası buraya yazılır.
Border Color		Buton Nesnesinin etrafının çizgi şeklinde sınırlarını belirtme rengidir.
Border Color		Buton nesnesinin etrafında oluşturulan çizginin kalınlığıdır.
Press Color		Buton nesnesinin basılı durumdaki ekrandaki rengini belirtir.
Press ColorTo		Gradient özelliği seçili olur ise, basılı durumda iken, ekranda geçişli bir buton nesnesi oluşur. Bu nesnenin Press Color dan Press ColorTo ya geçiş rengini tanımlamak için kullanılır.
Static		
Visible	True False	Ekran ilk oluştuğu zaman görünür. Ekran ilk oluştuğu zaman görünmez.
Text Aling		Buton nesnesi üzerindeki yazının butona göre konumlandırılmasıdır.
Gradient	None Top to Buttom Left to Right	Gradient özelliği kapalı olur. ColorTo ve Press ColorTo özelliği devre dışıdır. Gradient renkleri yukarıdan aşağı şeklinde uygulanır. Gradient renkleri soldan sağa doğru uygulanır.
Font Color		Butonun yazı rengidir.
Font Name		Buton nesnesi için farklı font seçenekleri tanımlama yapılır.
Font Size		Nesnenin yazısının fontunun büyüklüğüdür.
Dock		Buton nesnesinin ekrana yaslama şeklidir. Tam ekran şeklinde döşeme işlemi yapabilirsiniz.
Height		Nesnenin yüksekliğidir.
Left		Ekran üzerindeki pozisyonu belirtir. X koordinatı

Top		Ekran üzerindeki pozisyonu belirtir. Y koordinatı
Width		Nesnenin genişliğidir.

Fonksiyonlar

2. ButtonSet ( )

Açıklama

Buton nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void ButtonSet(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	Açıklama
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Visible ayarlama komutu

ButtonSet(Nesne adı , “Visible” , “1 , 0 veya True , False”);

Value özelliği “True” ayarlandığı zaman buton nesnesi gözükür, “False” ayarlandığı zaman ise gözükmez.

Örnek Kod:
ButtonSet (*"EButton1"* , *"Visible"* , *"True"*);

Active ayarlama komutu

ButtonSet(Nesne adı , “Active” , “1 , 0 veya True , False”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Active"* , *"True"*);

Left ayarlama komutu

ButtonSet(Nesne adı , “Left” , “X koordinatı”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Left"* , *"10"*);

Top ayarlama komutu

ButtonSet(Nesne adı , “Top” , “Y koordinatı”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Top"* , *"255"*);

Width ayarlama komutu

ButtonSet(Nesne adı , “Width” , “Size (0 dan Ekran X boyutu kadar)”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Width"* , *"90"*);

Height ayarlama komutu

ButtonSet(Nesne adı , “Height” , “Size (0 dan Ekran Y boyutu kadar)”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Height"* , *"70"*);

Color ayarlama komutu

ButtonSet(Nesne adı , “Color” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Color"* , *"#FFA07A"*);

ColorTo ayarlama komutu

ButtonSet(Nesne adı , “Color To” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"ColorTo"* , *"#FFA07A"*);

Press_Color ayarlama komutu

ButtonSet(Nesne adı , “Press Color” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Press_Color"* , *"#FFA07A"*);

Press_ColorTo ayarlama komutu

ButtonSet(Nesne adı , “Press ColorTo” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Press_ColorTo"* , *"#FFA07A"*);

FontSize ayarlama komutu

ButtonSet(Nesne adı , “FontSize” , “Font size olarak 8-102 arasında ayarlanır.”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"FontSize"* , *"12"*);

Font_Color ayarlama komutu

ButtonSet(Nesne adı , “Font Color” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Font_Color"* , *"#FFA07A"*);

Caption ayarlama komutu

Buton nesnesinin ekranda görünen string ifadesi bu komut ile değiştirilir.

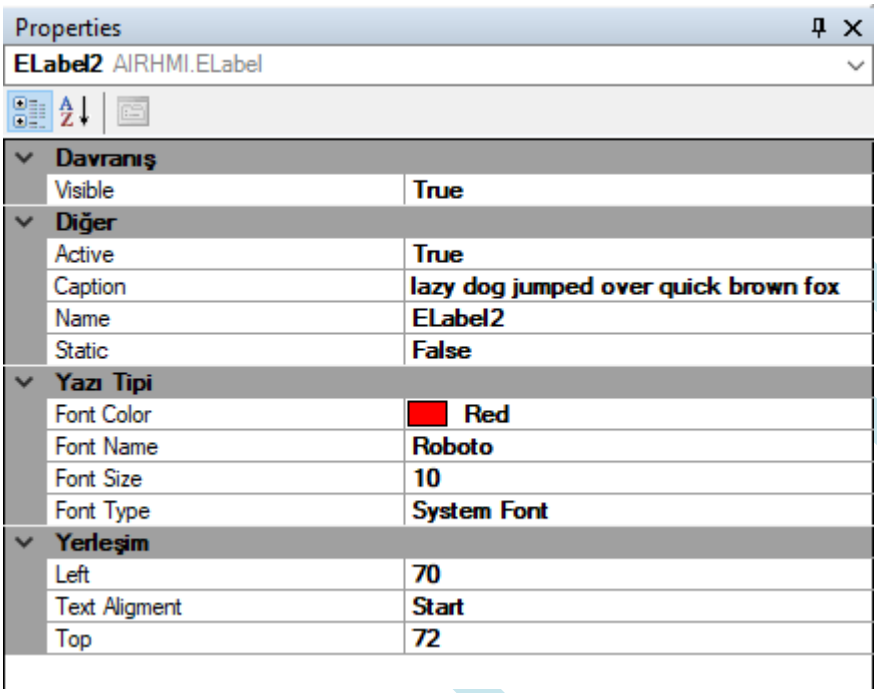
ButtonSet(Nesne adı , “Caption ve Text” , “Hello World!”);

Örnek Kod:
ButtonSet(*"EButton1"* , *"Caption"* , *"Hello World!"*);
ButtonSet(*"EButton1"* , *"Text"* , *"Hello World!"*);

5.3 Label

Ekranda yazı yazma amacı ile kullanılan nesnedir. Font size olarak 8 den 102’ ye kadar desteklemektedir. Default Font “Roboto” dur.

Label Properties Penceresi



Özellik	Seçenek	Açıklama
Active	True False	Açık olması durumunda, label a dokunulduğu zaman klavye otomatik olarak çıkar. Klavye pasif durumdadır.
Caption ,Text		Label nesnesinin ekranda gözüken yazısıdır.
Color		Buton nesnesinin ekrandaki rengini belirtir.
Visible	True False	Ekran ilk oluştuğu zaman görünür. Ekran ilk oluştuğu zaman görünmez.
Name		Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır.
Static		Reserved.
Visible	True False	Ekran ilk oluştuğu zaman görünür. Ekran ilk oluştuğu zaman görünmez.
Text Alingment	Start Center	Label nesnesi sola dayama, Label nesnesi ortalama
Font Color		Labelin yazı rengidir.
Font Name		Label nesnesi için farklı font seçenekleri tanımlama yapılır.
Font Size		Nesnenin yazısının fontunun büyüklüğüdür.
Height		Nesnenin yüksekliğidir.
Left		Ekran üzerindeki pozisyonu belirtir. X koordinatı

Top		Ekran üzerindeki pozisyonu belirtir. Y koordinatı
Width		Nesnenin genişliğidir.

Fonksiyonlar

LabelSet ( )

Açıklama

Label nesnesinin parametre ayarlarını düzenleyen komuttur.

void **LabelSet**(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	**Açıklama**
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Active ayarlama komutu

LabelSet(Nesne adı , “Active” , “1 , 0 veya True , False”);

Örnek Kod:
LabelSet(*"ELabel1"* , *"Active"* , *"True"*);

Visible ayarlama komutu

LabelSet(Nesne adı , “Visible” , “1 , 0 veya True , False”);

Örnek Kod:
LabelSet(*"ELabel1"* , *"Visible"* , *"1"*);

Left ayarlama komutu

LabelSet(Nesne adı , “Left” , “10”);

Örnek Kod:
LabelSet(*"ELabel1"* , *"Left"* , *"10"*);

Top ayarlama komutu

LabelSet(Nesne adı , “Top” , “255”);

Örnek Kod:
LabelSet (*"ELabel1"* , *"Top"* , *"255"*);

FontSize ayarlama komutu

LabelSet(Nesne adı , “FontSize” , “16”);

Örnek Kod:
LabelSet(*"ELabel1"* , *"FontSize"* , *"16"*);

Font_Color ayarlama komutu

LabelSet (Nesne adı , “Font_Color” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
LabelSet(*"ELabel1"* , *"Font_Color"* , *"#FFA07A"*);

Caption, Text ayarlama komutu

Label nesnesinin ekranda görünen string ifadesi bu komut ile değiştirilir.

LabelSet (Nesne adı , “Caption ve Text” , “Hello World!”);

LabelSet (*"ELabel1"* , *"Caption"* , *"Hello World!"*);
LabelSet (*"ELabel1"* , *"Text"* , *"Hello World!"*);

LabelGet()

void **LabelGet**(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	**Açıklama**
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Caption, Text komutu

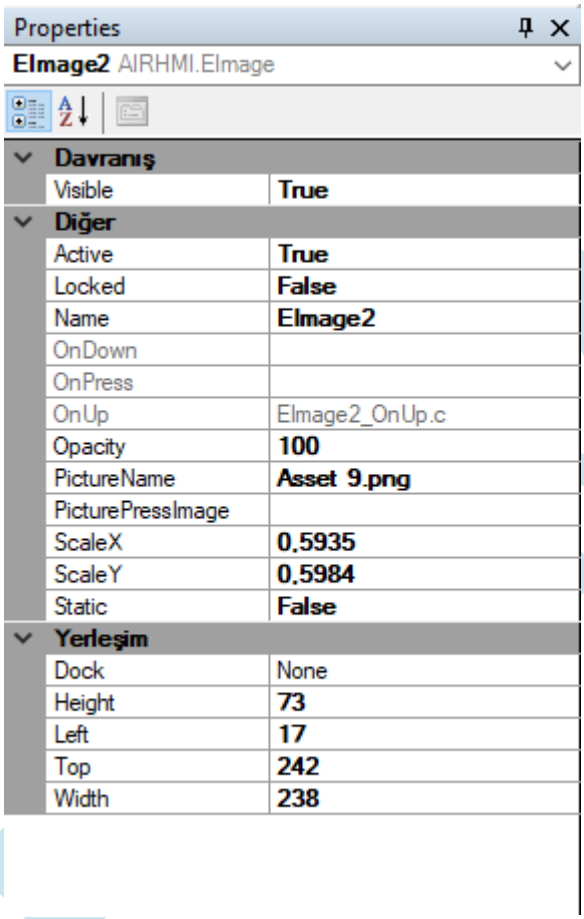
Label nesnesinin ekranda görünen string ifadesi bu komut ile değiştirilir.

LabelGet (Nesne adı , “Caption ve Text” , char * buffer);
Char value[20];
LabelGet(*"ELabel1"* , *"Caption"* , *value*);
LabelGet(*"ELabel1"* , *"Text"* , *value*);

5.4 Image

Image nesnesi resimleri gösterme ve resimleri buton olarak kullanma amacı ile kullanılabilir. Press image özelliği ile bir nesneye iki resim atayarak hiçbir kod yazmadan, normal durumda va press durumundaki resimlerini değiştirebilirsiniz.

Image Properties Penceresi



Özellik	Seçenek	Açıklama
Active	True False	Açık olması durumunda resim buton gibi kullanılabilir. Kapalı olması durumda sadece resim olarak kullanılır.ç
Visible	True False	Ekran ilk oluştuğu zaman görünür. Ekran ilk oluştuğu zaman görünmez.
Name		Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır.
Static		Reserved.
Locked	True False	Ekran a yerleştirilen nesnenin konumu değiştirmeye izin vermez. Resim istediğiniz konuma taşıyabilirsiniz.
Text Alingment	Start Center	Label nesnesi sola dayama, Label nesnesi ortalama
Height		Nesnenin yüksekliğidir.
Left		Ekran üzerindeki pozisyonu belirtir. X koordinatı
Top		Ekran üzerindeki pozisyonu belirtir. Y koordinatı
Width		Nesnenin genişliğidir.
İmage File		Bilgisayardan yüklemeniz gereken resim dosyasıdır.
Press Image File		Image nesnesine basılı tutarken ki resimdir.
ScaleX		İmage nesnesin X boyutundaki büyütme ve küçültme oranıdır.
ScaleY		İmage nesnesin Y boyutundaki büyütme ve küçültme oranıdır.
OnDown		Image nesnesine basma işlevi sırasında çalışan kod parçası buraya yazılır.
OnPress		Image nesnesine elimizi basılı tuttuğumuz sürece çalışacak olan kod parçasıdır. Tekrarlı olarak çalışır.
OnUp		Image nesnesinden elimizi çekme anında çalışan kod parçası buraya yazılır.

Fonksiyonlar

ImageSet ( )

Açıklama

Image nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void ImageSet(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	Açıklama
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Örnek kod

Visible ayarlama komutu

ImageSet(Nesne adı , “Visible” , “1 , 0 veya True , False”);

Örnek Kod:
ImageSet(*"EImage1"* , *"Visible"* , *"True"*);

Left ayarlama komutu

ImageSet(Nesne adı , “Left” , “Left Pozisyonu”);

Örnek Kod:
ImageSet (*"EImage1"* , *"Left"* , *"10"*);

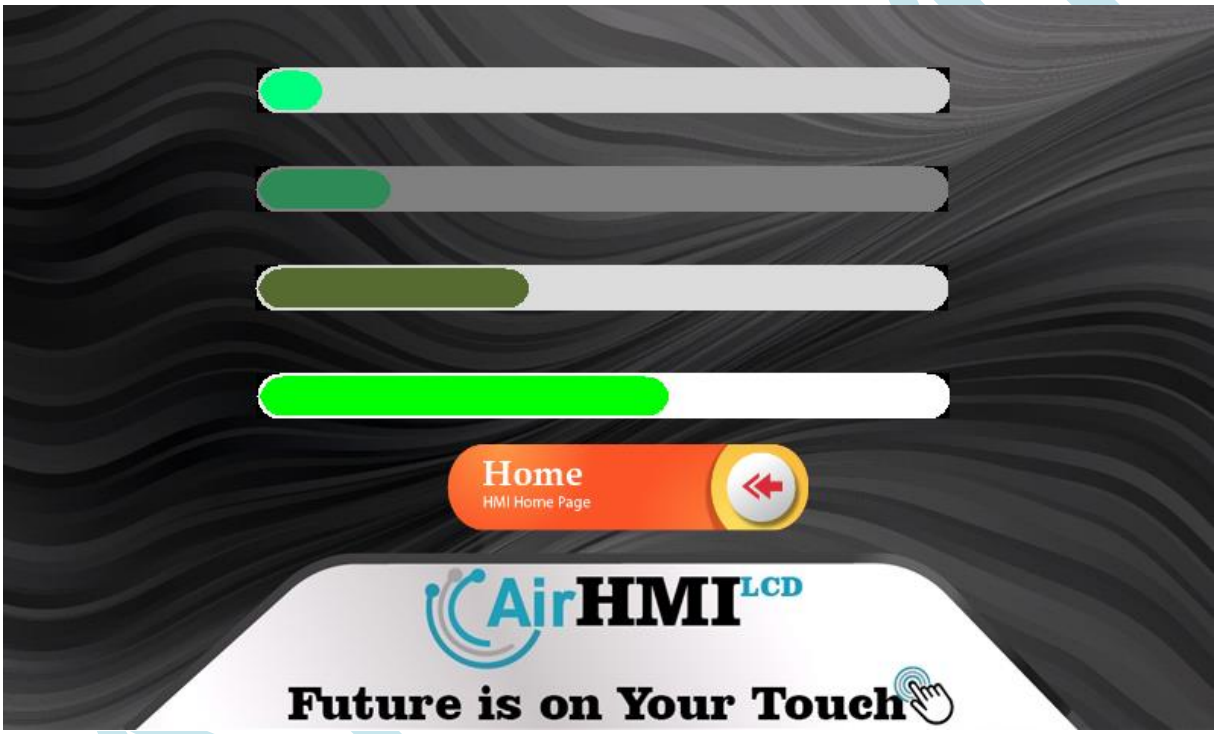
Top ayarlama komutu

ImageSet(Nesne adı , “Top” , “Top Pozisyonu”);

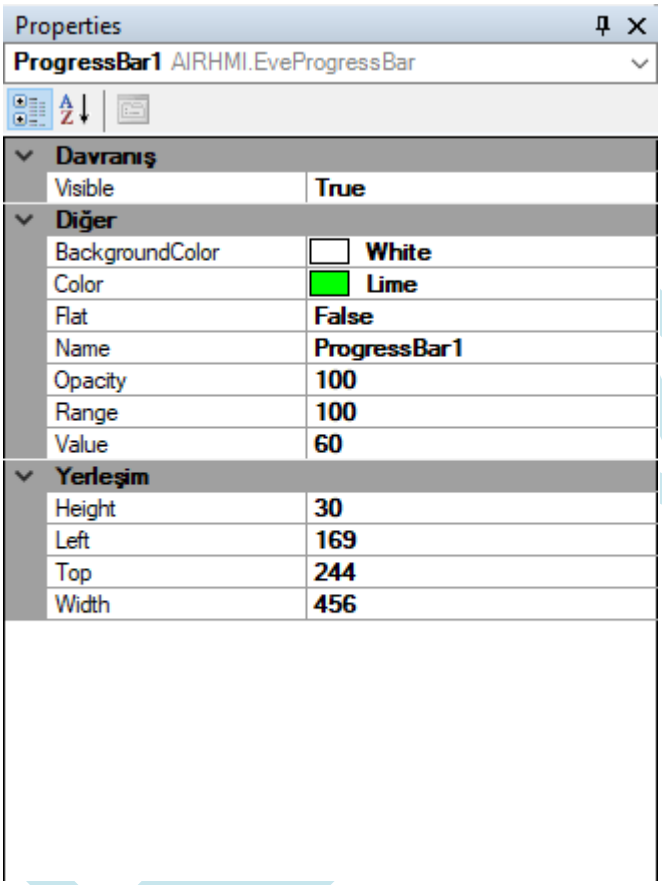
Örnek Kod:
ImageSet (*"EImage1"* , *"Top"* , *"255"*);

5.5 ProgressBar

Progress Bar ifadesi Türkçede “ilerleme çubuğu” anlamına gelmektedir. Uzun bir işlemin yürütülme aşamalarının grafiksel olarak gösterilmesi gerektiği durumlarda kullanılır. Progress Bar kullanımına örnek olarak: yürütülmekte olan bir video ya da ses dosyasının kalan zamanının Progress Bar üzerinde gösterilmesi, bir yakıt deposunun doluluk oranının Progress Bar kullanılarak grafiksel olarak gösterilmesi verilebilir.



ProgressBar Properties Penceresi



Özellik	Seçenek	Açıklama
Visible	True False	Ekran ilk oluştuğu zaman görünür. Ekran ilk oluştuğu zaman görünmez.
Name		Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır.
Color		Progressbar nesnesinin orta kısmında ilerleyen kısmın rengini belirtir.
BackgroundColor		Progressbar nesnesinin arka plan rengini belirtir.
Range		Progress bar toplam kaç değer olacağını belirtir.
Value		Progressbar in ilk ekrana yüklendiğinde yüzde kaçtan başlayacağını belirtir.
Height		Nesnenin yüksekliğidir.
Left		Ekran üzerindeki pozisyonu belirtir. X koordinatı
Top		Ekran üzerindeki pozisyonu belirtir. Y koordinatı
Width		Nesnenin genişliğidir.

Fonksiyonlar

ProgressBarSet ( )

Açıklama

Progress Bar nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void ProgressBarSet(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	Açıklama
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Örnek kod

Visible ayarlama komutu

ProgressBarSet(Nesne adı , “Visible” , “1 , 0 veya True , False”);

Örnek Kod:
ProgressBarSet(*"ProgressBar1"* , *"Visible"* , *"False"*);

Left ayarlama komutu

ProgressBarSet(Nesne adı , “Left” , “Ekrandaki X koordinatı pozisyonu”);

Örnek Kod:
ProgressBarSet(*"ProgressBar1"* , *"Left"* , *"10"*);

Top ayarlama komutu

ProgressBarSet(Nesne adı , “Top” , “Ekrandaki Y koordinatı pozisyonu”);

Örnek Kod:
ProgressBarSet(*"ProgressBar1"* , *"Top"* , *"255"*);

Color ayarlama komutu

ProgressBarSet(Nesne adı , “Color” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
ProgressBarSet(*"ProgressBar1"* , *"Color"* , *"255"*);

BackGround_Color ayarlama komutu

ProgressBarSet(Nesne adı , “BackGround_Color” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
ProgressBarSet(*"ProgressBar1"* , *"BackGround_Color"* , *"1458269"*);

Range ayarlama komutu

ProgressBarSet(Nesne adı , “Range” , “Range (numeric)”);

Örnek Kod:
ProgressBarSet(*"ProgressBar1"* , *"Range"* , *"100"*);

Value ayarlama komutu

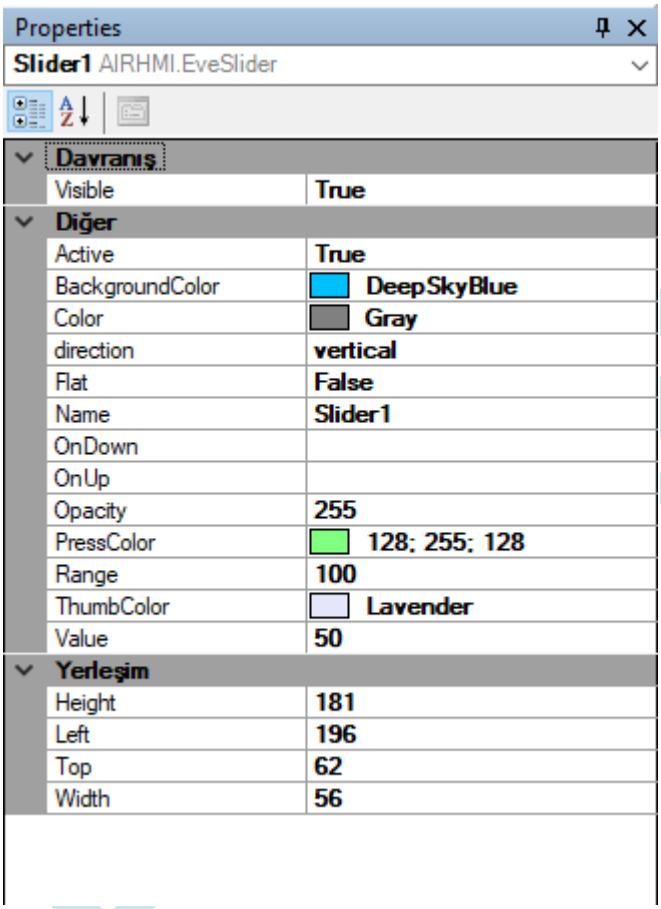
ProgressBarSet(Nesne adı , “Value” , “Value (numeric)”);

Örnek Kod:
ProgressBarSet(*"ProgressBar1"* , *"Value"* , *"50"*);

5.6 Slider

Kaydırıcı veya izleme çubuğu, kullanıcının bir göstergeyi yatay veya dikey olarak hareket ettirerek bir değer ayarlayabildiği grafiksel bir kontrol öğesidir. Bazı durumlarda, kullanıcı ayarı değiştirmek için kaydırıcıdaki bir noktaya da tıklayabilir.

Slider Properties Penceresi



Özellik	Seçenek	Açıklama
Visible	True False	Ekran ilk oluştuğu zaman görünür. Ekran ilk oluştuğu zaman görünmez.
Active	True False	Slider nesnesine basma işlevine izin verir. Slider nesnesi basma işlevine izin vermez.
Name		Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır.
Color		Slider nesnesinin arka kısmında kalan kısmının rengidir.
BackgroundColor		Slider nesnesinin arka plan rengini belirtir.
ThumpColor		Slider nesnesin üzerindeki yuvarlak kısmın rengidir.
PressColor		Slider nesnesine basıldığı zaman üzerindeki yuvarlak kısmın rengini değişir.
Range		Progress bar toplam kaç değer olacağını belirtir.
Value		Progressbar in ilk ekrana yüklendiğinde yüzde kaçtan başlayacağını belirtir.
Direction		Vertical , Horizontal Slider nesnesini ekranda kontrol yönünü belirtir.
Height		Nesnenin yüksekliğidir.
Left		Ekran üzerindeki pozisyonu belirtir. X koordinatı
Top		Ekran üzerindeki pozisyonu belirtir. Y koordinatı
Width		Nesnenin genişliğidir.

SliderSet ( )

Açıklama

Slider nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void SliderSet(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	Açıklama
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Visible ayarlama komutu

SliderSet(Nesne adı , “Visible” , “1 , 0 veya True , False”);

Örnek Kod:
```
SliderSet("Slider1" , "Visible" , "1");
```

Left ayarlama komutu

SliderSet(Nesne adı , “Left” , “Ekrandaki X koordinatı pozisyonu”);

Örnek Kod:
```
SliderSet("Slider1" , "Left" , "10");
```

Top ayarlama komutu

SliderSet(Nesne adı , “Top” , “Ekrandaki Y koordinatı pozisyonu”);

Örnek Kod:
```
SliderSet("Slider1" , "Top" , "255");
```

SliderGet ( )

Açıklama

Slider nesnesinin parametre ayarlarını almaya yarayan komuttur.

Fonksiyon

void SliderGet(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	**Açıklama**
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Value komutu

`SliderGet`(Nesne adı , “Value” , “char * buffer”);

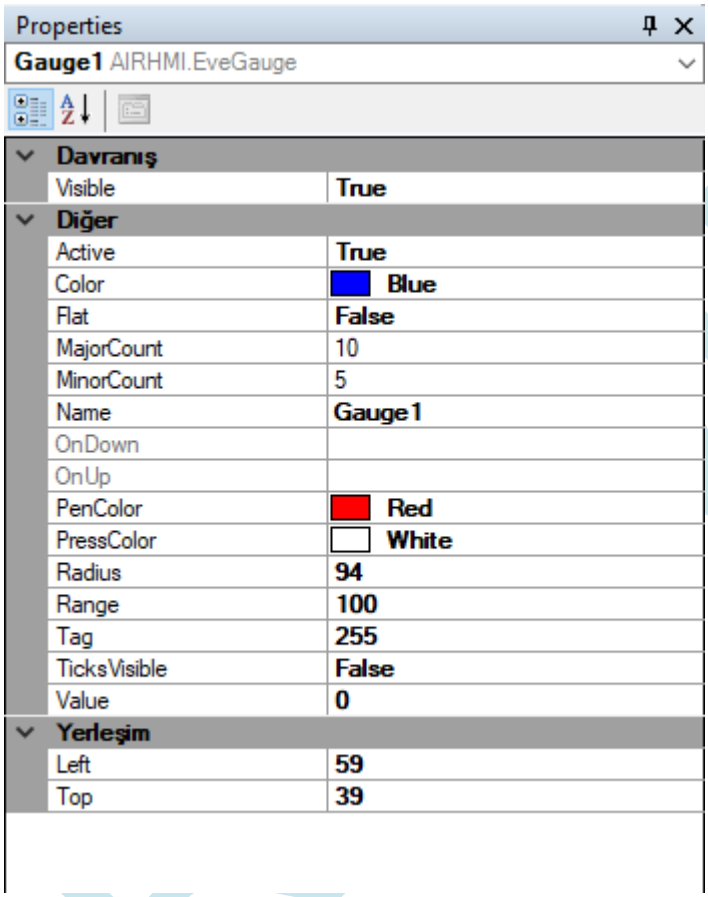
Örnek Kod:
char buffer[20];

```
SliderGet("Slider1" , "Value" , buffer);
```

5.7 Gauge

Gauge nesnesi analog değerleri göstermek için etkili bir nesnedir. Aynı zamanda hız göstergesi olarak da kullanılır.

Gauge Properties Penceresi



Özellik	Seçenek	Açıklama
Visible	True False	Ekran ilk oluştuğu zaman görünür. Ekran ilk oluştuğu zaman görünmez.
Name		Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır.
Color		Gauge nesnesinin arka kısmında kalan kısmının rengidir.
BackgroundColor		Slider nesnesinin arka plan rengini belirtir.
PressColor		Slider nesnesine basıldığı zaman üzerindeki yuvarlak kısmın rengini değişir.
Range		Progress bar toplam kaç değer olacağını belirtir.
Value		Progressbar in ilk ekrana yüklendiğinde yüzde kaçtan başlayacağını belirtir.
Radius		Gauge nesnesinin çapını ayarlar.
TicksVisible		Gauge nesnesinin etrafındaki çizgileri açıp kapatır.
Left		Ekran üzerindeki pozisyonu belirtir. X koordinatı
Top		Ekran üzerindeki pozisyonu belirtir. Y koordinatı

Fonksiyonlar

GaugeSet ( )

Açıklama

Gauge nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void GaugeSet(unsigned char *name , unsigned char *type , unsigned char *value)

Parametre	Açıklama
name	Nesnenin ismi
type	Nesnenin değiştirilecek parametresinin ismi
value	Değiştirilecek parametrenin yeni alacağı değer

Örnek kod

```
Visible ayarlama komutu
GaugeSet( Nesne adı , “Visible” , “1 , 0 veya True , False” );

Örnek Kod:
GaugeSet("Gauge1" , "Visible" , "1");

Left ayarlama komutu

GaugeSet( Nesne adı , “Left” , “Ekrandaki X koordinatı pozisyonu” );

Örnek Kod:
GaugeSet("Gauge1" , "Left" , "10");
```

Top ayarlama komutu

GaugeSet(Nesne adı , “Top” , “Ekrandaki Y koordinatı pozisyonu”);

Örnek Kod:
```
GaugeSet("Gauge1" , "Top" , "255");
```

Color ayarlama komutu

GaugeSet(Nesne adı , “BackGround_Color” , “RGB Color hex formatında #RRGGBB”);

Örnek Kod:
```
GaugeSet("Gauge1" , "Color" , "#ffaa02");
```

Value ayarlama komutu

GaugeSet(Nesne adı , “Value” , “Value (numeric)”);

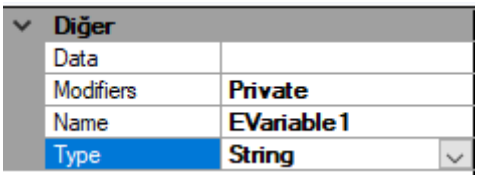
Örnek Kod:
```
GaugeSet("Gauge1" , "Value" , "100");
```

Range ayarlama komutu

GaugeSet(Nesne adı , “Range” , “Value (numeric)”);

Örnek Kod:
```
GaugeSet("Gauge1" , "Range" , "30");
```

5.8 VARIABLE



Değişkenler kod yapısı içerisinde değişkenlerin son değerlerinin veya kod içerisinde her düzenlemede değerinin kaybolmamasının istendiği durumlar için çok önemli bir rol almaktadırlar. Kod yapısı genel itibari ile Timer her aktif olduğunda veya Rezistif ekranlı projelerde dokunmanın aktif olduğu durumlarda derlenip yeniden çalıştığı için içerisinde oluşturulan normal değişkenler kendini sıfırlamaktadır. Bir önceki konumdan veya durumdan veriler kullanılmak istenildiğinde bu durum kullanıcı için büyük sorunlar teşkil etmektedir. Böyle bir sorunun yaşanmasını engellemek için devreye değişkenler girmektedir. Değişkenlerin ismi Öznitelikler bölümünden Name başlığı ile verilmektedir. Kullanılmak istenilen değişkenin tipi ise Type başlığı altından char ise String, sayısal değer ise İnteger olarak seçilmelidir. Bir diğer özelliği olan Modifiers, Öznitelikler kısmından kullanmak istediğimiz değişkenin Private (yerel) ya da Public (global) olacağı seçilmeli. Yerel-global ayrımı birden fazla ekran tasarımı kullanılacak projelerde yapılmaktadır. Tek bir ekranda çalışma gerçekleştirilecek ise Private (yerel) değişken istenilen durumu gerçekleştirebilmektedir. Fakat birden fazla ekran kullanmak istenilen projelerde örneğin ikinci ekranda bulunan bir değer birinci ekrana geçildiğinde de kullanılmak istenilirse burada Public (Global) değişken kullanılmalıdır. Değişkenlerin kod yapısı içerisinde kullanımına dair açıklamalar aşağıda yer almaktadır.

```
GlobalStdVarGet("EVariable1" , "string");
1  2  3        4          5
```

Değişkenin:

1. Global veya Local
2. String veya Integer

3. Değerinin Set ya da Get edileceğini
4. İsmi
5. Yeni değeri veya eski değerinin alınacağı değişken

durumlarına göre istenilen fonksiyon kullanılmalıdır.

```
int value;
LocalStdVarSet("EVariable1" , “string”);  // Local olan String değişkeni Set etme
GlobalIntVarGet("EVariable2", &value);  //  Global olan Integer değişkeni Get etme
```

LocalStdVarGet ( )

Açıklama

Local(yerel) string veri okuma komutudur.

Fonksiyon

void LocalStdVarGet(unsigned char *name , unsigned char *value)

Parametre	**Açıklama**
name	Yerel değişkenin ismi
value	Yerel değişkenin atanacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"
char data[200];
LocalStdVarGet("EVariable1" , data);  // Local olan String değişkeni Get etme
```

LocalStdVarSet ( )

Açıklama

Local(yerel) string değer atama komutudur.

Fonksiyon

void LocalStdVarSet(unsigned char *name , unsigned char *value)

Parametre	Açıklama
name	Yerel değişkenin ismi
value	Yerel değişkenin alacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"

LocalStdVarSet("EVariable1" , “string”);  // Local olan String değişkeni Set etme
```

LocalIntVarGet ( )

Açıklama

Local(yerel) integer veri okuma komutudur.

Fonksiyon

void LocalIntVarGet(unsigned char *name , int *value)

Parametre	**Açıklama**
name	Yerel değişkenin ismi
value	Yerel değişkenin atanacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value;

LocalIntVarGet("EVariable2", &value);  //  Local olan Integer değişkeni Get etme
```

LocalIntVarSet ( )

Açıklama

Local(yerel) integer değer atama komutudur.

Fonksiyon

void LocalIntVarSet(unsigned char *name , int value)

Parametre	Açıklama
name	Yerel değişkenin ismi
value	Yerel değişkenin alacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value = 5;

LocalIntVarSet("EVariable2", value);  //  Local olan Integer değişkeni Set etme
```

GlobalStdVarGet ( )

Açıklama

Global string veri okuma komutudur.

Fonksiyon

void GlobalStdVarGet(unsigned char *name , unsigned char *value)

Parametre	Açıklama
name	Global değişkenin ismi
value	Global değişkenin atanacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"

GlobalStdVarGet("EVariable1" , “string”); // Global olan String değişkeni Get etme
```

GlobalStdVarSet ( )

Açıklama

Global string değer atama komutudur.

Fonksiyon

void GlobalStdVarSet(unsigned char *name , unsigned char *value)

Parametre	Açıklama
name	Global değişkenin ismi
value	Global değişkenin alacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"

GlobalStdVarSet("EVariable1" , “string”); // Global olan String değişkeni Set etme
```

GlobalIntVarGet ( )

Açıklama

Global integer veri okuma komutudur.

Fonksiyon

void GlobalIntVarGet(unsigned char *name , int *value)

Parametre	Açıklama
name	Global değişkenin ismi
value	Global değişkenin atanacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value = 5;

GlobalIntVarGet("EVariable2", &value);  // Global olan Integer değişkeni Get etme
```

GlobalIntVarSet ( )

Açıklama

Global integer değer atama komutudur.

Fonksiyon

void GlobalIntVarSet(unsigned char *name , int value)

Parametre	Açıklama
name	Global değişkenin ismi
value	Global değişkenin alacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value = 10;

GlobalIntVarSet("EVariable2", value);  // Global olan Integer değişkeni Set etme
```

VariableSave( )

Açıklama

Varible'i ekran içerisindeki hafızaya kayıt eder. Bu sayede ekran kapanıp açılsa bile bu variable değeri kalıcı olarak hafızada tutulur. Variable içeriğinde değişiklik yaptıktan sonra tekrar kayıt etmek için aynı fonksiyon tekrar çağırılır. Maksimum 256 adet variable hafızaya kayıt edile bilinir.

Fonksiyon

void VariableSave(unsigned char *name)

Parametre	Açıklama
name	değişkenin ismi

Örnek kod

```
#include "stdio.h"

#include "stk.h"

VariableSave("EVariable1");  //
```

LocalStdVarGet ( )

Açıklama

Local(yerel) string veri okuma komutudur.

Fonksiyon

void VarGet(unsigned char *name , void *value)

Parametre	**Açıklama**
name	Yerel değişkenin ismi
value	Variable tipine göre pointer değeri

Örnek kod

```
#include "stdio.h"

#include "stk.h"
char data[200];
int varint;
double varDouble;
VarGet("EVariable1" , data);
VarGet("EVariable2" , &varint);
VarGet("EVariable3" , &varDouble);
```

*VarGet fonksiyonuna value kısmına NULL verirsek, değişkenin içeriğini seri porttan verir.

```
VarGet("EVariable3" , NULL);
```

Fonksiyon

void VarSet(unsigned char *name , void *value)

Parametre	Açıklama
name	Yerel değişkenin ismi
value	Variable tipine göre pointer değeri

Örnek kod

```
#include "stdio.h"

#include "stk.h"
char data[200];
int varint=5;
double varDouble = 2.15;
VarSet("EVariable1" , data);
VarSet("EVariable2" , &varint);  // EVariable2 nin değeri 5 olur.
VarSet("EVariable3" , &varDouble);  // EVariable3 ün değeri 2.15 olur.
```

Fonksiyon

void VarSeti(unsigned char *name , int value)

Parametre	**Açıklama**
name	Yerel değişkenin ismi
value	İnteger variable değeri

Bu fonksiyon doğrudan variable integer değer atamak için kullanılan bir fonksiyondur. VarSet fonksiyonunda integer değeri adres olarak parametre vermek gerekirken, varSeti fonksiyonunda doğrudan bu değeri verebiliyoruz.

Örnek kod

```
#include "stdio.h"

#include "stk.h"

VarSeti("EVariable1" , 15);

İnt a = 5;

VarSeti("EVariable1" , a);
```

Fonksiyon

void VarSets(char *name , char *value)

Parametre	**Açıklama**
name	Yerel değişkenin ismi
value	String pointer variable

Örnek kod

```
#include "stdio.h"

#include "stk.h"

VarSets("EVariable1" , “Merhaba Dünya!”);

Char *data = “Merhaba Dünya!”;

VarSets("EVariable1" , data);
```

Fonksiyon

void VarSetf(char *name , double value)

Parametre	**Açıklama**
name	Yerel değişkenin ismi
value	double variable değeri

Örnek kod

```
#include "stdio.h"

#include "stk.h"

VarSetf("EVariable1" , 3.14);

double var = 3.14;

VarSets("EVariable1" , var);
```

5.9 Delay( )

Açıklama

Kullanıldığı satırda belirlenen süre kadar beklemeyi sağlayan komuttur.

Fonksiyon

void Delay (int ms)

Parametre	Açıklama
ms	Zaman periyodunu belirtir

Örnek kod

```
#include "stk.h"

Delay(1000);
```

5.10 uartDataGet ( )

Açıklama

UART'tan gelen verilere göre AMHI Editör ekranında işlemler yapılabilmektedir. Kod düzeni içerisinde UART'tan gelen veriyi alma komutudur.

Fonksiyon

void uartDataGet(char *value , int *uartsize)

Parametre	Açıklama
value	UART'tan gelecek verinin depolanacağı string
uartsize	UART'tan gelen verinin boyutu

Örnek kod

```
#include "stdio.h"

#include "stk.h"

char uartData[3000];                    // Uarttan gelecek verinin depolanacağı
string

int uartsize;                            //  Uarttan gelen verinin boyutu

uartDataGet(uartData , &uartsize);   //  Uarttan gelen verinin okunmas
```

5.11 ChangeScreenSet ( )

Açıklama

Kod içerisinde bulunan ekranlar arasında geçiş yapmayı sağlayan komuttur.

Fonksiyon

void ChangeScreenSet(unsigned char *value)

Parametre	Açıklama
value	Geçiş yapılacak ekranın ismi

Örnek kod

```
#include "stk.h"

ChangeScreenSet("Screen1");
```

5.12 dateSet ( )

Açıklama

RTC'de tarih verilerini yenileme/ayarlama komutudur.

Fonksiyon

void dateSet (unsigned char *days , unsigned char *months , unsigned char *years)

Parametre	Açıklama
days	Gün
months	Ay
years	Yıl

Örnek kod

```
#include "stdio.h"

#include "stk.h"

unsigned char day, month, year;     // Kod dizininde örnek Tarih-Saat değişkenleri

day   = 10;
month = 2;
year  = 19;

dateSet(&day, &month , &year);     // RTC den Tarih verilerini ayarlama
```

5.13 timeSet ( )

Açıklama

RTC'de saat verilerini yenileme/ayarlama komutudur.

Fonksiyon

void timeSet(unsigned char *hours , unsigned char *mins)

Parametre	Açıklama
hours	Saat
mins	Dakika

Örnek kod

```
#include "stdio.h"
#include "stk.h"

unsigned char hour, min;          // Kod dizininde örnek Tarih-Saat değişkenleri

hour = 16;
min  = 30;

timeSet(&hour , &min);           // RTC de Saat verilerini yenileme/ayarlama
```

5.14 dateGet ( )

Açıklama

RTC'den tarih verilerini alma komutudur.

Fonksiyon

void dateGet(unsigned char *days , unsigned char *months , unsigned char *years)

Parametre	Açıklama
days	Gün
months	Ay
years	Yıl

Örnek kod

```
#include "stdio.h"
#include "stk.h"
unsigned char day, month, year;      // Kod dizininde örnek Tarih-Saat değişkenleri
dateGet(&day, &month , &year);      // RTC den Tarih verilerini alma
```

5.15 timeGet ( )

Açıklama

RTC'den saat verilerini alma komutudur.

Fonksiyon

void timeGet(unsigned char *hours , unsigned char *mins)

Parametre	Açıklama
hours	Saat
mins	Dakika

Örnek kod

```
#include "stdio.h"
#include "stk.h"

unsigned char hour, min;          // Kod dizininde örnek Tarih-Saat değişkenleri
timeSet(&hour , &min);           // RTC de Saat verilerini okuma
```

5.16 AudioPlay( )

Açıklama

Kullanıcı, çalmak isteği ses dosyasını AirHMI Editör üzerinden projeye ekledikten sonra bu fonksiyon ile çalma işlemini gerçekleştirebilmektedir.

Fonksiyon

void AudioPlay(unsigned char *audioname , unsigned char volume)

Parametre	Açıklama
audioname	Ses dosyasını ismi
volume	Ses düzeyi

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int volume;   // Ses Düzeyi

AudioPlay("SesDosyasınınİsmi" , volume );
```

5.17 AudioStop( )

Açıklama

O anda çalınan ses işlemnin sonlandırmak için kullanılır.

Fonksiyon

void AudioStop ();

Parametre	Açıklama

Örnek kod

```
#include "stdio.h"

#include "stk.h"
```

AudioStop();

5.18 AudioStatusGet( )

Açıklama

Ses dosyasının o anda çalınıp çalınmadığını ayarlar.

Fonksiyon

void AudioStatusGet(int *value)

Parametre	Açıklama
value	Player durumu (1 ses dosyası çalmaya devam ediyor , 0 ses dosyası çalma işlemi bitmiştir.

Durum sorgulama komutu

AudioStatusGet(int *value);

Value özelliği “True” ayarlandığı zaman buton nesnesi gözükür, “False” ayarlandığı zaman ise gözükmez.

Örnek Kod:
int value;
AudioStatusGet(*&value*);

5.19 File_write ( )

Açıklama

Flash'a yazma komutudur.

Fonksiyon

void File_write(unsigned char *name , void *buffer ,int size , int nmemb)

Parametre	Açıklama
name	Kullanılacak .txt dosyasının ismi
buffer	String dizisinin ismi
size	Yazılacak dizinin boyutu
nmemb	1

Örnek kod

```
#include "stdio.h"
#include "stk.h"
char x_file[200];
memset(x_file , 0x00 , sizeof(x_file));
sprintf(x_file , "%s" , "Hello World !!!");

File_write("Message.txt" , x_file , sizeof(x_file), 1);

// Flashta Message.txt isimli bir dosya oluşturuldu ve bu dosya içerisine x_file
verisi sizeof(x_file) boyutu kadar yazıldı.
```

5.20 File_read( )

Açıklama

Flash'tan okuma komutudur.

Fonksiyon

void File_read(unsigned char *name , void *buffer ,int size , int nmemb)

Parametre	Açıklama
name	Kullanılacak .txt dosyasının ismi
buffer	String dizisinin ismi
size	Okuma boyutu
nmemb	1

Örnek kod

```
#include "stdio.h"
#include "stk.h"

char x_file[200];
memset(x_file , 0x00 , sizeof(x_file));

File_write("Message.txt" , x_file , sizeof(x_file), 1);

// Flashta bulunan Message.txt isimli bir dosyanın içerisinde ki verilerden
sizeof(x_file) kadarı x_file değişkenine okundu.
```

5.21 File_size( )

Açıklama

Dosya boyutunu öğrenme komutudur.

Fonksiyon

void File_size(unsigned char *name ,int *size)

Parametre	Açıklama
name	Kullanılacak dosyanın ismi
size	Dosya boyutunun içinde tutulacağı integer bir değişken

Örnek kod

```
#include "stdio.h"
#include "stk.h"

int f_size;

File_size("Message.txt" , &f_size);     // Flashta bulunan Message.txt dosyasının
boyutunu öğrenme.
```

5.22 GPIO_Write( )

Açıklama

Fonksiyon

void GPIO_Write(unsigned char *portName ,int value)

Parametre	**Açıklama**
portName	Gpio port
value	1 veya 0

Örnek kod

GPIO yazma komutu

GPIO_Write(GPIO adi , 1 veya 0);

Örnek Kod:
GPIO_Write(“GPIO_1” , 1);
GPIO_Write(“GPIO_1” , 0);

5.23 GPIO_Read( )

Açıklama

Fonksiyon

void GPIO_ Read(unsigned char *portName ,int *value)

Parametre	**Açıklama**
portName	Gpio port
value	1 veya 0

Örnek kod

GPIO okuma komutu

GPIO_Read(GPIO adi , int *);

Örnek Kod:
int value;
GPIO_Read(“GPIO_1” , &value);

5.24 PWM_Set( )

Açıklama

Airhmi ekran üzerinde 2 adet pwm çıkışı vardır. Bu fonksiyon ile pwm frekans duty ayarlanır.

Fonksiyon

void PWM_Set(int ch , int freq , int duty);

Parametre	Açıklama
ch	Pwm kanalı 1 veya 0
freq	Pwm frekansı
duty	Pwm 1 ,0 yüzdesidir. Değeri 0-100 olarak verilir.

Örnek kod

PWM komutu

PWM_Set(int ch , int freq , int duty);

Örnek Kod:

```
PWM_Set( 0,1000000, 50 ); // Channel 0 , 1Mhz %50 duty.
PWM_Set( 1,2000000, 70 ); // Channel 0 , 2Mhz %70 duty.
```

5.25 BuzzerSet( )

Açıklama

Airhmi ekran dahili buzzer a sahiptir.

Fonksiyon

void BuzzerSet(int interval)

Parametre	Açıklama
interval	Milisaniye cinsinden buzzer çalma süresidir.

Örnek kod

Buzzer komutu

void BuzzerSet(int interval)

Örnek Kod:
BuzzerSet(100); // 100 ms buzzer set edilir.

5.26 I2C_Write( )

Açıklama

Airhmi ekran i2c özelliğine sahiptir.

Fonksiyon

void I2C_Write(int speed , int deviceAddress , char *data , int dataLen)

Parametre	Açıklama
speed	i2c haberleşme hızı
deviceAddress	i2c Slave device adresi
data	data
dataLen	Data uzunluğu

Örnek kod

I2C_Write komutu

void I2C_Write(int speed , int deviceAddress , char *data , int dataLen)

Örnek Kod:

```
Char data[] = {0xaa,0xbb,0xcc};

I2C_Write(10000, 0x55 , data , 3);
```

5.27 I2C_Read ( )

Açıklama

Airhmi ekran i2c özelliğine sahiptir.

Fonksiyon

void I2C_Read(int speed , int deviceAddress , char *data , int dataLen)

Parametre	Açıklama
speed	i2c haberleşme hızı
deviceAddress	i2c Slave device adresi
data	data
dataLen	Data uzunluğu

Örnek kod

I2C_Read komutu

void I2C_Read(int speed , int deviceAddress , char *data , int dataLen)

Örnek Kod:

Char data[3];

I2C_Read(10000, 0x55 , data , 3);

5.28 millis( )

Açıklama

Millis fonksiyonu, programın belirli bir işlevin ne kadar sürede gerçekleştirileceğini takip etmesine olanak tanır. Örneğin, bir sensörden veri okumak ve bir eylem gerçekleştirmek için belirli bir süre geçmesi gerekiyorsa, millis fonksiyonu kullanılarak bu süre takip edilebilir. Millis fonksiyonunun kullanımı basittir. Fonksiyon çağrısı, programın başlangıcından itibaren geçen milisaniye sayısını döndürür. Bu değer, bir değişkene atanarak kullanılabilir veya doğrudan bir karşılaştırma ifadesinde kullanılabilir.

Fonksiyon

void millis(int *value)

Parametre	Açıklama
value	Geçen süreyi verir.

Örnek kod

millis komutu

```
int baslangicZamani;

millis(&baslangicZamani); // Başlangıç zamanı kaydedilir

// İşlemler yapılır

int bitisZamani;

millis(&bitisZamani);

if(bitisZamani - baslangicZamani > 5000) {

// 5 saniye geçti

}
```

5.29 KeypadAlpha( )

Açıklama

Yazılım sırasında kullanıcıdan veri almak için kullanılır. Ekranda tam sayfa bir klevye çıkar. Kullanıcı klavyeyi kullanarak buraya verileri girer ve klavye geri dönüşü ayrı bir pointer a atama yapılır.

Fonksiyon

void KeypadAlpha(char *inData , char *outData)

Parametre	Açıklama
inData	Klavye açıldığı zaman düzenlenecek yazı,
outData	Klavye geri dönüş verisidir.
data	data
dataLen	Data uzunluğu

Örnek kod

millis komutu

```
char data[100];
KeypadAlpha("Merhaba Dunya!",data);

printf("You Wrote %s.\n",data);
```

5.30 Modbus_ReadHoldingRegisters( )

Açıklama

Modbus RTU protokolünde "03" fonksiyon kodu, cihazın holding register'larını okumak için kullanılır. Bu fonksiyon kodu, holding register'ların bir alt kümesini, belirtilen bir cihaz adresindeki belirli bir başlangıç adresinden başlayarak okur. Bu işlem için kullanılan Modbus mesajı, aşağıdaki gibi olabilir:

Örneğin, 1 numaralı cihazın 4000 adresinden itibaren 10 holding register'ını okumak için aşağıdaki Modbus mesajı kullanılabilir:

Adres: 01

Fonksiyon Kodu: 03

Başlangıç Adresi: 4000 (0x0FA0)

Okunacak Holding Register Sayısı: 10 (0x000A)

Bu mesajı gönderdikten sonra cihazın yanıtı, belirtilen holding register'ların değerlerini içeren bir Modbus mesajıdır.

Fonksiyon

void Modbus_ReadHoldingRegisters(unsigned char id, int address,int quantity, unsigned short * data, int timeout_ms);

Parametre	Açıklama
id	Modbus id (0-255)
address	Modbus Slave Register Adresi
quantity	Kaç adet veri okunacağı

data	Modbus datası
timeout_ms	Timeout değeri

Örnek kod

```
#include "stk.h"
#include "stdio.h"

unsigned short data[2];

Modbus_ReadHoldingRegisters(1,4000,2,data,1000);

char resData[200];
sprintf(resData,"%04x - %04x",data[0],data[1]);
LabelSet("ELabel8" ,"Caption"  , resData );
```

5.31 Modbus_WriteSingleRegister( )

Açıklama

Modbus protokolünde "06" fonksiyon kodu, bir cihazdaki tek bir kayıt (register) değerini yazmak için kullanılır. Bu fonksiyon kodu, belirtilen bir cihaz adresindeki belirli bir kayıt adresine tek bir veri değerini yazar.

Modbus RTU protokolünde "06" fonksiyon kodu için kullanılan Modbus mesajı şu şekildedir:

Adres: Cihaz adresi
Fonksiyon Kodu: 06
Kayıt Adresi: yazılacak kayıt adresi
Yazılacak Değer: kayıta yazılacak 16 bit veri

Örneğin, 1 numaralı cihazın 4000 adresine 1234 değerini yazmak için aşağıdaki Modbus mesajı kullanılabilir:

Adres: 01
Fonksiyon Kodu: 06
Kayıt Adresi: 4000 (0x0FA0)
Yazılacak Değer: 1234 (0x04D2)

Bu mesajı gönderdikten sonra cihazın yanıtı bir Modbus mesajı olmayacaktır. Ancak, mesajın gönderildiğinden emin olmak için bir onay mesajı veya hata mesajı alınabilir.

Fonksiyon

void Modbus_WriteSingleRegister(unsigned char id, int address ,unsigned short data, unsigned short *response, int timeout_ms);

Parametre	**Açıklama**
id	Modbus id (0-255)
address	Modbus Slave Register Adresi
data	Modbus datası
response	Modbus Slave device cevabı
timeout_ms	Timeout değeri

Örnek kod

```
#include "stk.h"
#include "stdio.h"

unsigned short data[20];
Modbus_WriteSingleRegister(1,4000,1234,data,1000);
```

5.32 Modbus_WriteMultipleRegisters( )

Açıklama

Modbus protokolünde "16" fonksiyon kodu, birden fazla kayıt değerini yazmak için kullanılır. Bu fonksiyon kodu, belirtilen bir cihaz adresindeki belirli bir kayıt adresinden başlayarak ardışık bir dizi kayıt adresine birden fazla veri değerini yazar.

Modbus RTU protokolünde "16" fonksiyon kodu için kullanılan Modbus mesajı şu şekildedir:

Adres: Cihaz adresi
Fonksiyon Kodu: 16
Başlangıç Adresi: yazılacak kayıt adresi
Yazılacak Kayıt Sayısı: yazılacak toplam kayıt sayısı
Yazılacak Byte Sayısı: yazılacak veri sayısının byte cinsinden boyutu
Veri: yazılacak tüm kayıt değerlerinin ardışık olarak gönderilen byte'larla kodlanmış hali

Örneğin, 1 numaralı cihazın 4000 adresinden itibaren 5 kayıt değerini sırasıyla 1234, 5678, 9101, 1121 ve 3141 olarak yazmak için aşağıdaki Modbus mesajı kullanılabilir:

Adres: 01

Fonksiyon Kodu: 16

Başlangıç Adresi: 4000 (0x0FA0)

Yazılacak Kayıt Sayısı: 5 (0x0005)

Yazılacak Byte Sayısı: 10 (0x0014)

Veri: 04 D2 16 2E 23 29 04 49 0B 71

Fonksiyon

void Modbus_WriteMultipleRegisters(unsigned char id, int address , int quantity, unsigned short *data, unsigned char *response, int timeout_ms);

Parametre	Açıklama
id	Modbus id (0-255)
address	Modbus Slave Register Adresi
qauantity	Modbus'a yazılacak data sayısı
data	Modbus datası
response	Modbus Slave device cevabı
timeout_ms	Timeout değeri

Örnek kod

```
#include "stk.h"
#include "stdio.h"

char data[20];
unsigned short modbusData[2];

modbusData[0] = 10;
modbusData[1] = 11;
Modbus_WriteMultipleRegisters(1,4000,2,modbusData,data,1000);
```

5.33 Modbus_ReadInputRegisters( )

Açıklama

Modbus protokolünde "04" fonksiyon kodu, bir cihazdaki giriş kayıtlarını (input registers) okumak için kullanılır. Bu fonksiyon kodu, belirtilen bir cihaz adresindeki belirli bir giriş kayıt adresinden başlayarak ardışık bir dizi giriş kaydını okur.

Modbus RTU protokolünde "04" fonksiyon kodu için kullanılan Modbus mesajı şu şekildedir:

Adres: Cihaz adresi
Fonksiyon Kodu: 04
Başlangıç Adresi: okunacak giriş kayıt adresi
Okunacak Kayıt Sayısı: okunacak toplam giriş kayıt sayısı

Örneğin, 1 numaralı cihazın 10001 adresinden başlayarak 5 giriş kaydını okumak için aşağıdaki Modbus mesajı kullanılabilir:

Adres: 01
Fonksiyon Kodu: 04
Başlangıç Adresi: 10001 (0x2711)
Okunacak Kayıt Sayısı: 5 (0x0005)

Cihazın yanıtı, istenen giriş kayıtlarının değerlerini içeren bir Modbus mesajı olacaktır. Bu mesajın boyutu, istenen giriş kayıt sayısı ve Modbus RTU protokolünün kullanıldığı özel cihaz özelliklerine bağlı olarak değişebilir.

Fonksiyon

void Modbus_ReadInputRegisters(unsigned char id, int address , int quantity, unsigned short *data, int timeout_ms);

Parametre	Açıklama
id	Modbus id (0-255)
address	Modbus Slave Register Adresi

qauantity	Modbus'a yazılacak data sayısı
data	Modbus datası
timeout_ms	Timeout değeri

Örnek kod

```
#include "stk.h"
#include "stdio.h"

unsigned short data[20];
Modbus_ReadInputRegisters(1,5000,2,data,1000);
```

6. Ethernet

6.1 Dhcp & Statik ip tanımlama

Açıklama

DHCP (Dynamic Host Configuration Protocol) ve statik IP adresleri, bilgisayar ağlarında kullanılan iki farklı IP adresi atanma yöntemidir. İşte her iki yöntemin nasıl çalıştığı ve avantajları:

DHCP (Dynamic Host Configuration Protocol):

DHCP, ağdaki cihazlara otomatik olarak IP adresleri atanmasını sağlayan bir protokoldür. İşleyişi şu şekildedir:

Bir cihaz ağa bağlandığında, DHCP sunucusu tarafından IP adresi, alt ağ maskesi, varsayılan ağ geçidi ve DNS sunucu adresi gibi ağ yapılandırma bilgileri otomatik olarak cihaza atanır.

DHCP sunucusu, ağdaki IP adreslerini yönetir ve cihazların dinamik olarak IP adresleri almasına izin verir.

Bu, büyük ağlarda IP adreslerinin yönetimini kolaylaştırır ve cihazların otomatik olarak ağa katılmasını sağlar.

Statik IP Adresleri:

Statik IP adresleri, her cihaza elle atanmış olan ve değiştirilmeden sabit kalan IP adresleridir. İşleyişi şu şekildedir:

Ağ yöneticisi veya kullanıcılar, her cihaza özel olarak bir IP adresi, alt ağ maskesi, varsayılan ağ geçidi ve DNS sunucu adresleri atarlar.

Bu IP adresleri manuel olarak yapılandırıldığı için, her cihazın sabit bir IP adresi vardır ve değiştirilmez.

Fonksiyon

void EthernetInit_DHCP();

Parametre	Açıklama

Örnek kod

```
#include "stk.h"

EthernetInit_DHCP();
```

Fonksiyon

void EthernetInit_Static(char *ip , char *gw , char *netmask);

Parametre	Açıklama
İp adresi	IP adresi (Internet Protocol Address), bilgisayarlar ve diğer cihazlar arasındaki ağ iletişimi için kullanılan benzersiz bir tanımlayıcıdır.
Gateway	Gateway (Ağ Geçidi), bir ağdaki cihazlar arasında veri iletimini sağlayan önemli bir ağ cihazıdır. Gateway, iki farklı ağ veya iletişim protokolü arasında veri iletişimini kolaylaştırır.
Netmask	Netmask (Ağ Maskesi), bir IP adresinin ağ bölümünü ve cihazın veya alt ağın bölümünü ayırt etmek için kullanılan bir değerdir.

Örnek kod

```
#include "stk.h"

EthernetInit_Static("192.168.1.150","192.168.1.1","255.255.255.0");
```

6.2 IP Adresi Sorgulama

Açıklama

IP adresi (Internet Protocol Address), bilgisayarlar ve diğer ağ cihazları arasında iletişim kurmak için kullanılan bir tanımlayıcıdır. IP adresleri, TCP/IP (Transmission Control Protocol/Internet Protocol) ağ protokolünün bir parçası olarak, cihazların ağda bulunabilirliklerini ve iletişimlerini sağlamak için kullanılır. İnternet ve yerel ağlar gibi ağlarda, her cihazın kendine özgü bir IP adresi vardır. IP adresleri genellikle dört bölümden oluşur ve her bölüm 0 ile 255 arasında bir değere sahip olabilir. Bu dört bölüm, noktalı ondalık biçimde yazılır. Örneğin, "192.168.1.1" bir IPv4 (Internet Protocol version 4) IP adresinin örneğidir.

IP adreslerinin iki temel türü vardır:

IPv4 (Internet Protocol version 4): Bu en yaygın kullanılan IP adresi türüdür. IPv4 adresleri, 32 bitlik bir sayıdır ve 4 ayrı bölümden oluşur (her bölüm 0 ile 255 arasında değer alır). Örneğin, "192.168.1.1" IPv4 bir IP adresidir. Ancak, IPv4 adreslerinin tükenmeye başlaması nedeniyle, IPv6'ya geçiş süreci başlamıştır.

IP adresleri, cihazların ağda benzersiz bir şekilde tanımlanmasını ve verilerin doğru bir şekilde yönlendirilmesini sağlar. Ayrıca, IP adresleri ağda iletişim kurma işlemine yardımcı olur ve internete erişim gibi önemli ağ işlevlerinin gerçekleşmesini sağlar.

Fonksiyon

void EthernetGet_IP(char *ip_adress);

Parametre	Açıklama
ip_adress	Airhmi ekranın sunucudan aldığı ip adresidir.

Örnek kod

```
char data[100];

EthernetGet_IP(data);
```

6.3 MAC Adresi Sorgulama

Açıklama

MAC adresi (Media Access Control Address), ağ cihazlarının donanım kimlik numarasını temsil eden benzersiz bir tanımlayıcıdır. MAC adresi, ağ kartı veya ağ arabirimi üzerinde fiziksel olarak atanır ve genellikle 48 bit (6 bayt) uzunluğundadır. MAC adresi, ağ düzeyinde veri iletimi sırasında cihazların kimliklerini belirlemek için kullanılır.

MAC adresi, genellikle onaltılık tabanla yazılır ve altı çift rakamdan oluşur. Örnek bir MAC adresi şu şekildedir: "00:1A:2B:3C:4D:5E."

Fonksiyon

void EthernetGet_MAC(char *mac_addres);

Parametre	Açıklama
mac_addres	Airhmi Ethernet arabiriminin mac adresi.

Örnek kod

```
char data[100];

EthernetGet_MAC(data);
```

6.4 Ethernet TCP Soket Bağlantısı

Açıklama

Airhmi ekran statik yada DHCP olrak kullanılabilinir.

Fonksiyon

SocketTCP_Create("char * ip, int port);

Parametre	Açıklama
İp adresi	IP adresi (Internet Protocol Address), bilgisayarlar ve diğer cihazlar arasındaki ağ iletişimi için kullanılan benzersiz bir tanımlayıcıdır.
Port Numarası	TCP Soket port numarasıdır.

Örnek kod

```
#include "stk.h"

SocketTCP_Create("192.168.1.49",8000);
```

6.5 Ethernet TCP Soket Gönder Al

Açıklama

TCP soket sunucuya veri gönderme ve alma fonksiyonudur.

Fonksiyon

Void SocketTCP_SendReceive(char *sendData,char *rcvData);

Parametre	Açıklama
sendData	TCP sokete göndermek istediğimiz veri.
rcvData	TCP Soket den alınan veridir.

Örnek kod

```
#include "stk.h"

char DATA[1024];
SocketTCP_SendReceive("GET {path} HTTP/1.1$0d$0aHost: {host}$0d$0a$0d$0a",DATA);
printf("DATA:%s\n",DATA);
```

6.6 Ethernet TCP Soket Gönder

Açıklama

TCP soket sunucuya veri gönderme fonksiyonudur.

Fonksiyon

```
Void SocketTCP_Send(char *SendData,int len);
```

Parametre	Açıklama
sendData	TCP sokete göndermek istediğimiz veri.
len	Veri uzunluğu

Örnek kod

```
#include "stk.h"
#include "stdio.h"

SocketTCP_Send("AIRHMI",6);
```

6.7 Ethernet TCP Soket Al

Açıklama

TCP soket sunucudan veri alma fonksiyonudur.

Fonksiyon

```
Void SocketTCP_Receive(char rcvData);
```

Parametre	Açıklama
rcvData	TCP soketden alınan veri.

Örnek kod

```
#include "stk.h"
#include "stdio.h"

char rcv[100];

SocketTCP_Receive(rcv);
printf("Data:%s\n",rcv);
```

6.8 Ethernet TCP Soket Kapat

Açıklama

TCP soket kapatma fonsiyonudur.

Fonksiyon

```
Void SocketTCP_Close();
```

Parametre	Açıklama

Örnek kod

```
#include "stk.h"
#include "stdio.h"

SocketTCP_Close();
```

6.9 Ethernet TCP Soket Durumu Sorgulama

Açıklama

Airhmi TCP soket durumu sorgulama için kullanılır.

Fonksiyon

```
İnt SocketTCP_GetStatus();
```

Parametre	Açıklama
10	Connected.
Diğerleri	Bağlı değil

Örnek kod

```
#include "stk.h"
#include "stdio.h"

int status = SocketTCP_GetStatus();

if( status == 10 )
    LabelSet("ELabel3" , "Text" , "Connected." );
else
    LabelSet("ELabel3" , "Text" , "Not Connected." );
```

6.10 http post ve get

Açıklama

HTTP (Hypertext Transfer Protocol), web tarayıcıları ve web sunucuları arasındaki iletişim için kullanılan bir protokoldür. HTTP, iki temel metot (method) sunar: GET ve POST. Bu iki metot, web sayfalarının alınması, gönderilmesi ve işlenmesi için kullanılır.

GET Metodu:

GET, sunucuya bir istekte bulunarak belirli bir kaynağın (genellikle bir web sayfası) alınmasını istemek için kullanılır.

GET isteği, URL üzerinden iletilir ve URL'nin sonuna eklenen sorgu parametreleri ile veri aktarımı yapar.

GET isteği, sunucuya bilgi göndermez, yalnızca bilgi almak için kullanılır.

GET isteği, tarayıcıda yeniden yüklenirse veya bir bağlantıya tıklanırsa aynı istek tekrarlanır. Bu nedenle, GET isteği idempotenttir, yani aynı istek tekrarlandığında sonuç değişmez.

GET isteği, tarayıcı geçmişinde görünür, bu nedenle URL'de gönderilen veriler açıkça görülebilir.

Örnek bir GET isteği URL ile şu şekildedir:

GET http://example.com/page.php?param1=value1&param2=value2

POST Metodu:

POST, sunucuya veri göndermek veya bir kaynağı güncellemek için kullanılır.

POST isteği, veriyi HTTP isteğinin gövdesine ekler, bu nedenle URL üzerinden değil, isteğin gövdesinde veri taşır.

POST isteği, veri iletmek için kullanıldığından, genellikle kullanıcı adı, şifre ve diğer hassas bilgiler gibi gizli bilgileri güvenli bir şekilde göndermek için tercih edilir.

POST isteği, tarayıcı geçmişinde görünmez, bu nedenle gönderilen veriler daha güvenli bir şekilde saklanır.

POST isteği idempotent değildir, yani aynı istek tekrarlandığında sonuç değişebilir.

Örnek bir POST isteği şu şekildedir:

POST http://example.com/submit.php

Body:

param1=value1&param2=value2

Her iki HTTP metodu da web uygulamalarında farklı amaçlara hizmet eder. GET genellikle bilgi almak için kullanılırken, POST veri göndermek ve işlem yapmak için kullanılır. Her iki metodun da kullanımı, uygulamanın gereksinimlerine ve güvenlik gereksinimlerine bağlıdır.

7. Kütüphaneler

7.1 stdio.h

"stdio" (Standart Giriş/Çıkış) kütüphanesi, C dilinde yaygın olarak kullanılan bir kütüphanedir. Bu kütüphane, standart giriş/çıkış işlevleri için gereken araçları sağlar. Bu kütüphanede yer alan işlevler, klavye ve fare gibi cihazlardan veri girişi yapmak veya ekrana veya dosyalara veri çıktısı sağlamak için kullanılır. Bunun yanı sıra, standart hata ve bilgi mesajlarının işlenmesi ve yönetimi için de kullanılır.

```
int printf(char *, ...);

int fprintf(FILE *, char *, ...);

int sprintf(char *, char *, ...);

int snprintf(char *, int, char *, ...);
```

7.2 stdlib.h

"stdlib" (Standart Kütüphane) kütüphanesi, C ve C++ programlama dillerinde yaygın olarak kullanılan bir kütüphanedir. Bu kütüphane, çeşitli işlevleri içerir ve özellikle bellek yönetimi, dönüşüm işlemleri, rastgele sayı üretimi, program sonlandırma, dosya işlemleri ve diğer yardımcı işlevler için kullanılır.

"stdlib" kütüphanesi, standart C kütüphanesi ile birlikte kullanılır ve C programlama dilinde standart bir kütüphane olarak kabul edilir. C++ dilinde de kullanılabilir.

Bu kütüphanenin içinde yer alan bazı işlevler şunlardır:

Bellek yönetimi işlevleri (malloc, calloc, realloc, free)

Dönüşüm işlevleri (atoi, atof, itoa)

Rastgele sayı üretme işlevi (rand)

Diğer yardımcı işlevler (abs, exit, qsort)

Bu işlevler, C dilinde sıkça kullanılan işlevlerdir ve birçok programda kullanılırlar. "stdlib" kütüphanesi, kodun okunabilirliğini artırmak ve geliştirme sürecini hızlandırmak için kullanışlı bir araçtır.

```
float atof(char *);

float strtod(char *,char **);

int atoi(char *);

int atol(char *);

int strtol(char *,char **,int);

int strtoul(char *,char **,int);

void *malloc(int);

void *calloc(int,int);

void *realloc(void *,int);

void free(void *);
```

```
int rand();

void srand(int);

int abs(int);

int labs(int);
```

7.3 math.h

"math" (matematik) kütüphanesi, C programlama dilinde yaygın olarak kullanılan bir kütüphanedir. Bu kütüphane, matematiksel işlemler için gereken araçları sağlar.

Bu kütüphanede yer alan işlevler, trigonometrik işlemler, üstel fonksiyonlar, logaritmalar, kök hesaplamaları, aralık kontrolü, sayı yuvarlama işlemleri ve diğer matematiksel işlemler için kullanılır.

math kütüphanesinde bulunan bazı işlevler şunlardır:

sin, cos, tan: Trigonometrik işlemler için kullanılır.

pow: Bir sayının üssünü hesaplamak için kullanılır.

sqrt: Bir sayının karekökünü hesaplamak için kullanılır.

exp: Bir sayının e^x değerini hesaplamak için kullanılır.

log, log10: Bir sayının doğal veya ondalık logaritmasını hesaplamak için kullanılır.

ceil, floor: Bir sayının üst veya alt tam sayıya yuvarlanması için kullanılır.

fabs: Bir sayının mutlak değerini hesaplamak için kullanılır.

Bu işlevler, matematiksel işlemler içeren birçok C programında kullanılırlar. math kütüphanesi, kodun okunabilirliğini artırmak ve geliştirme sürecini hızlandırmak için kullanışlı bir araçtır.

float acos(float);

float asin(float);

float atan(float);

float atan2(float, float);

float ceil(float);

float cos(float);

float cosh(float);

float exp(float);

float fabs(float);

float floor(float);

float fmod(float, float);

float frexp(float, int *);

float ldexp(float, int);

float log(float);

float log10(float);

float modf(float, float *);

float pow(float,float);

float round(float);

float sin(float);

float sinh(float);

float sqrt(float);

float tan(float);

float tanh(float);

7.4 string.h

"string" kütüphanesi, C programlama dilinde yaygın olarak kullanılan bir kütüphanedir. Bu kütüphane, karakter dizileri (string) işlemleri için gereken araçları sağlar.

Bu kütüphanede yer alan işlevler, karakter dizileri ile ilgili işlemler için kullanılır. Bu işlemler arasında, karakter dizilerinin birleştirilmesi, karşılaştırılması, kopyalanması, uzunluklarının hesaplanması ve diğer işlemler yer alır.

string kütüphanesinde bulunan bazı işlevler şunlardır:

strcpy: Bir karakter dizisini başka bir karakter dizisine kopyalamak için kullanılır.
strcat: İki karakter dizisini birleştirmek için kullanılır.
strlen: Bir karakter dizisinin uzunluğunu hesaplamak için kullanılır.
strcmp: İki karakter dizisini karşılaştırmak için kullanılır.
strchr: Bir karakter dizisinde belirli bir karakteri aramak için kullanılır.
strstr: Bir karakter dizisinde belirli bir alt diziyi aramak için kullanılır.
Bu işlevler, C programlama dilinde karakter dizileri ile ilgili işlemler için sıkça kullanılır. string kütüphanesi, kodun okunabilirliğini artırmak ve geliştirme sürecini hızlandırmak için kullanışlı bir araçtır.

Örnek Program:

```c
#include <stdio.h>
#include <string.h>

int main() {
   char string1[20] = "Merhaba";
   char string2[20] = "dünya";
   char string3[40];

   // string1 ve string2 karakter dizilerini birleştirir
   strcat(string1, string2);

   // Birleştirilmiş karakter dizisini string3'e kopyalar
   strcpy(string3, string1);

   printf("Birleştirilmiş karakter dizisi: %s\n", string1);
   printf("Kopyalanan karakter dizisi: %s\n", string3);

   // string1 karakter dizisinde "dünya" alt dizisi aranır
   if (strstr(string1, "dünya") != NULL) {
      printf("string1 karakter dizisinde 'dünya' bulundu.\n");
   } else {
      printf("string1 karakter dizisinde 'dünya' bulunamadı.\n");
   }

   // string1 ve string3 karakter dizileri karşılaştırılır
   if (strcmp(string1, string3) == 0) {
      printf("string1 ve string3 karakter dizileri eşittir.\n");
   } else {
      printf("string1 ve string3 karakter dizileri eşit değildir.\n");
   }

   return 0;
}
```

Program Çıktısı:

```
Birleştirilmiş karakter dizisi: Merhabadünya
Kopyalanan karakter dizisi: Merhabadünya
string1 karakter dizisinde 'dünya' bulundu.
string1 ve string3 karakter dizileri eşittir.
```

void *memcpy(void *,void *,int);

void *memmove(void *,void *,int);

void *memchr(char *,int,int);

int memcmp(void *,void *,int);

void *memset(void *,int,int);

char *strcat(char *,char *);

char *strncat(char *,char *,int);

char *strchr(char *,int);

char *strrchr(char *,int);

int strcmp(char *,char *);

int strncmp(char *,char *,int);

int strcoll(char *,char *);

char *strcpy(char *,char *);

char *strncpy(char *,char *,int);

char *strerror(int);

int strlen(char *);

int strspn(char *,char *);

int strcspn(char *,char *);

char *strpbrk(char *,char *);

char *strstr(char *,char *);

char *strtok(char *,char *);

int strxfrm(char *,char *,int);