

AirHMI LCD EKCRAN EDITÖR KILAVUZU

AIRHMI

AİRHMI LCD EKLAN EDITOR KILAVUZU

AirHMI Visual Screen Creator, AirHMI LCD ekranları için İnsan Makine Arayüzü GUI'lerini tasarım açısından 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.

AIRHMI

AIRHMI LCD EKTRAN EDITOR KILAVUZU

| Tarih | Fonksiyon Adı | Firmware Versiyon |
|--------------|--------------------------------|--------------------------|
| 05.05.2024 | Convert_IntToString | 4.00 |
| 05.05.2024 | Convert_FloatToString | 4.00 |
| 05.05.2024 | Convert_StringToInt | 4.00 |
| 05.05.2024 | Convert_StringToFloat | 4.00 |
| 28.07.2014 | Transhape specs added. | 4.04 |
| 28.07.2014 | Toggle specs added. | 4.04 |
| 07.10.2014 | StructSet and StructGet added. | 4.05 |
| | | |
| | | |
| | | |
| | | |

AIRHMI

AİR HMI LCD EKРАН EDITOR KILAVUZU

İÇİNDEKİLER

| | | |
|------|--|----|
| 1. | AirHMI Visual Screen Creator KURULUMU | 1 |
| 2. | PROJE OLUŞTURMA | 2 |
| 3. | CİHAZ BAĞLANTISI | 4 |
| 4. | AirHMI EDİTOR ANA ARAYÜZÜ | 5 |
| 4.1 | BAŞLIK ÇUBUĞU | 5 |
| 4.2 | ANA MENÜ ve ARAÇ ÇUBUKLARI | 5 |
| 4.3 | BİLEŞENLER BÖLMESİ | 8 |
| 4.4 | EKRAN / KOMUT SEKMESİ | 9 |
| 4.5 | TASARIM ANA EKRAN ALANI | 10 |
| 4.6 | GÖRSELİ OLMAYAN BİLEŞENLERİN ALANI | 11 |
| 4.7 | NESNELERİN ÖZNİTELİK ALANI | 11 |
| 4.8 | 3.7.1 Projede Kullanılan Nesnelerin Gösterim Alanı | 11 |
| 4.9 | 3.7.2 Nesnelerin Öznitelikleri Gösterim / Ayar Alanı | 12 |
| 4.10 | ÖZNİTELİKLERİN AÇIKLAMA ALANI | 12 |
| 4.11 | KULLANICI PROJE KODU MENÜ ve ARAÇ ÇUBUKLAR | 12 |
| 4.12 | KULLANICI PROJE KOD ALANI | 12 |

AİRHMI LCD EKLAN EDITOR KILAVUZU

| | | |
|------|--|----|
| 4.13 | KOD ALANI ZOOM ALANI..... | 13 |
| 4.14 | KOD ALANI..... | 13 |
| 5. | Değişkenlerin Birbirlerine Dönüşümleri..... | 15 |
| 5.1 | İnteger İfadeyi String(Char Dizi)'e Dönüştürme | 15 |
| 5.2 | sprintf kullanımı | 17 |
| 5.3 | atoi..... | 18 |
| 5.4 | atof..... | 19 |
| 6. | AİRHMI NESNELERİ VE FONKSİYONLAR..... | 20 |
| 6.1 | TIMER | 20 |
| 6.2 | Button | 23 |
| 6.3 | Label..... | 29 |
| 6.4 | Image..... | 34 |
| 6.5 | ProgressBar | 39 |
| 6.6 | Slider | 44 |
| 6.7 | Gauge | 49 |
| 6.8 | ListView | 54 |
| 6.9 | ListWheel | 63 |

AIRHMI LCD EKTRAN EDITOR KILAVUZU

| | | |
|------|--------------------------|-----|
| 6.10 | TransShape | 68 |
| 6.11 | Toggle..... | 70 |
| 6.12 | Graph..... | 76 |
| 6.13 | Variable | 81 |
| 6.14 | Delay()..... | 91 |
| 6.15 | uartDataGet()..... | 92 |
| 6.16 | ChangeScreenSet ()..... | 93 |
| 6.17 | dateSet () | 94 |
| 6.18 | timeSet ()..... | 95 |
| 6.19 | dateGet ()..... | 96 |
| 6.20 | timeGet () | 97 |
| 6.21 | AudioPlay()..... | 98 |
| 6.22 | AudioStop()..... | 99 |
| 6.23 | AudioStatusGet()..... | 100 |
| 6.24 | VideoPlay() | 101 |
| 6.25 | Video_Play_XY()..... | 102 |
| 6.26 | File_write ()..... | 104 |

AİRHMI LCD EKLAN EDITOR KILAVUZU

| | | |
|------|--|-----|
| 6.27 | File_read() | 105 |
| 6.28 | File_size() | 106 |
| 6.29 | GPIO_Write() | 107 |
| 6.30 | GPIO_Read() | 108 |
| 6.31 | PWM_Set() | 109 |
| 6.32 | BuzzerSet() | 110 |
| 6.33 | I2C_Write() | 111 |
| 6.34 | I2C_Read () | 112 |
| 6.35 | millis() | 113 |
| 6.36 | KeypadAlpha() | 114 |
| 6.37 | Modbus_ReadHoldingRegisters() | 115 |
| 6.38 | Modbus_WriteSingleRegister() | 117 |
| 6.39 | Modbus_WriteMultipleRegisters() | 119 |
| 6.40 | Modbus_ReadInputRegisters() | 121 |
| 7. | Ethernet | 123 |
| 7.1 | Dhcp & Statik ip tanımlama | 123 |
| 7.2 | IP Adresi Sorgulama | 125 |

AİRHMI LCD EKLAN EDITOR KILAVUZU

| | | |
|------|--|-----|
| 7.3 | MAC Adresi Sorgulama..... | 126 |
| 7.4 | Ethernet TCP Soket Bağlantısı..... | 127 |
| 7.5 | Ethernet TCP Soket Gönder Al..... | 128 |
| 7.6 | Ethernet TCP Soket Gönder..... | 129 |
| 7.7 | Ethernet TCP Soket Al..... | 130 |
| 7.8 | Ethernet TCP Soket Kapat..... | 131 |
| 7.9 | Ethernet TCP Soket Durumu Sorgulama..... | 132 |
| 7.10 | http post ve get..... | 133 |
| 8. | Kütüphaneler..... | 135 |
| 8.1 | stdio.h..... | 135 |
| 8.2 | stdlib.h..... | 136 |
| 8.3 | math.h..... | 138 |
| 8.4 | string.h..... | 141 |

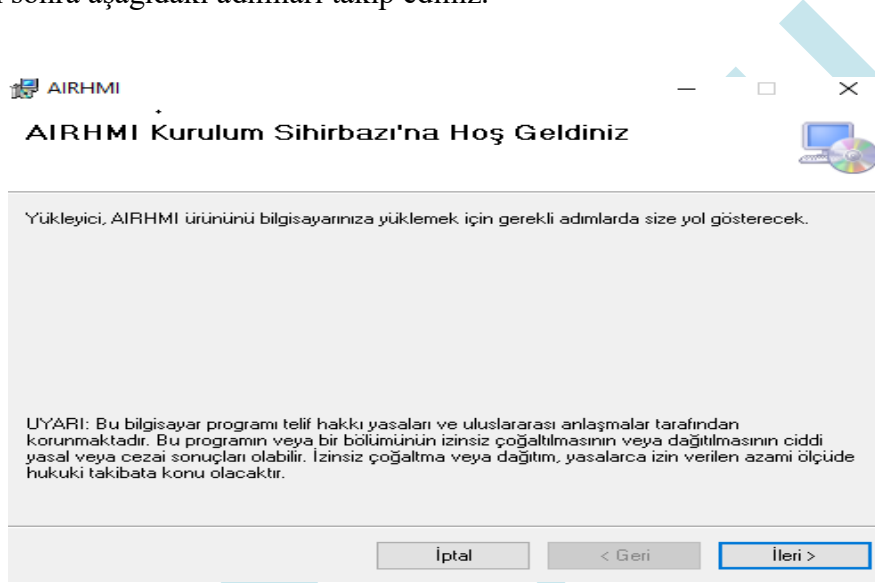
AIRHMI LCD EKРАН EDITOR KILAVUZU

1. AirHMI Visual Screen Creator KURULUMU

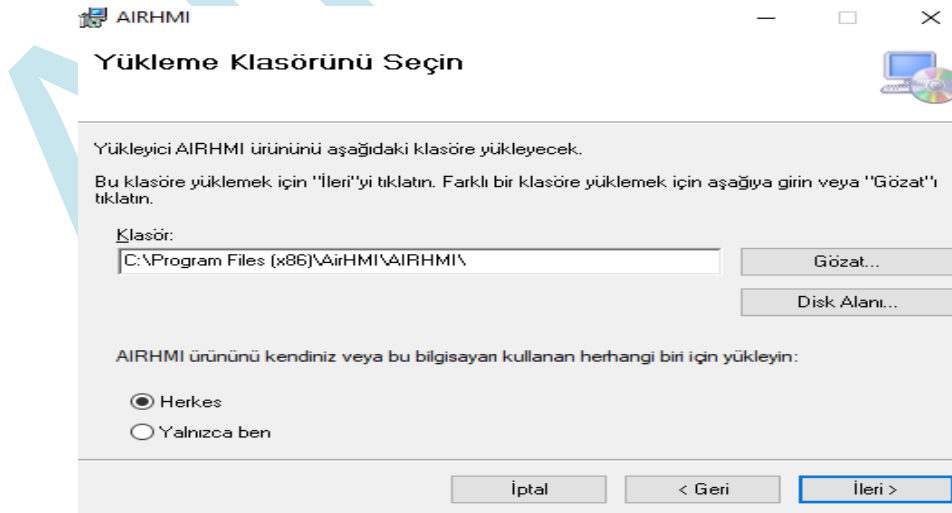
İndirme Linki: <https://www.airhmi.com/airhmi-visualcreator>

AirHMI Editör'ü bilgisayarınıza yüklemek için AIRHMISETUP.msi dosyasına çift tıklayın.

Bu işlemten sonra aşağıdaki adımları takip ediniz.



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

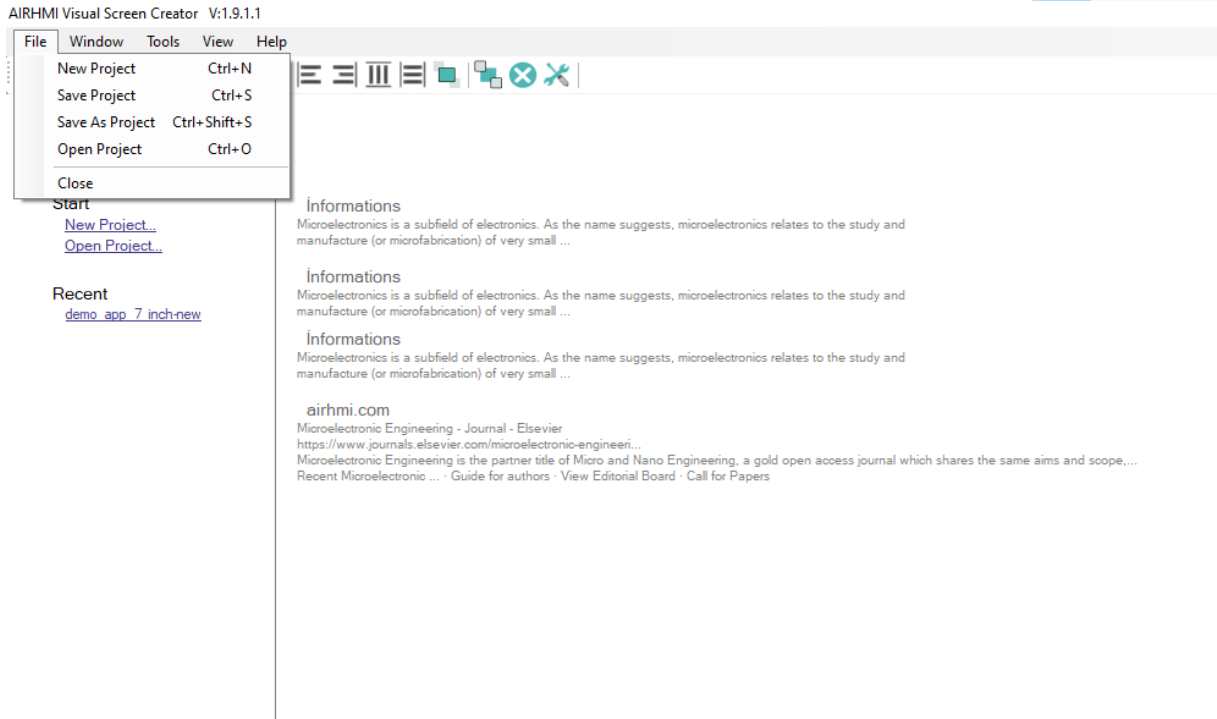


AIRHMI LCD EKРАН EDITOR KILAVUZU

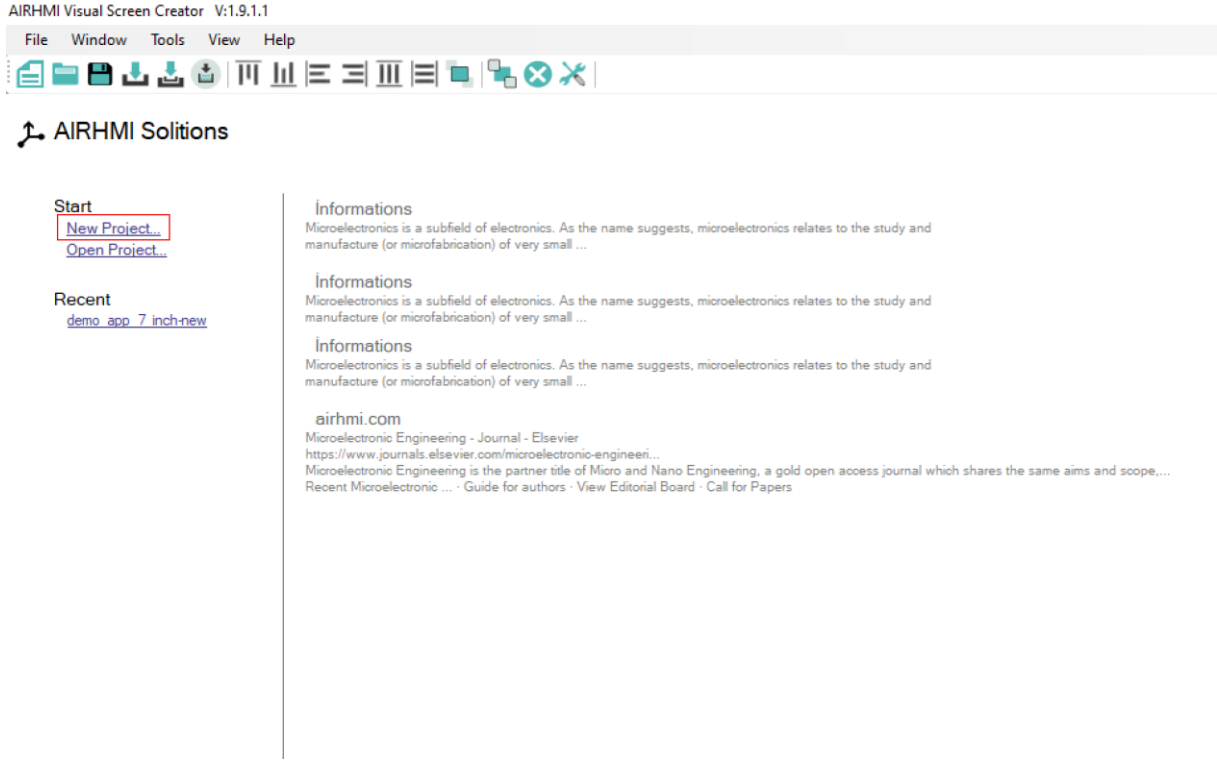
2. PROJE OLUŞTURMA

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 çalıştırılmalıdır. 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.



AIRHMI LCD EKРАН EDITOR KILAVUZU



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

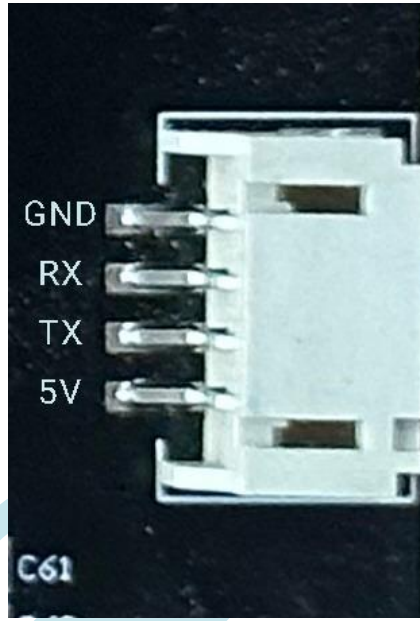


AIRHMI LCD EKРАН EDITOR KILAVUZU

3. CİHAZ BAĞLANTISI

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

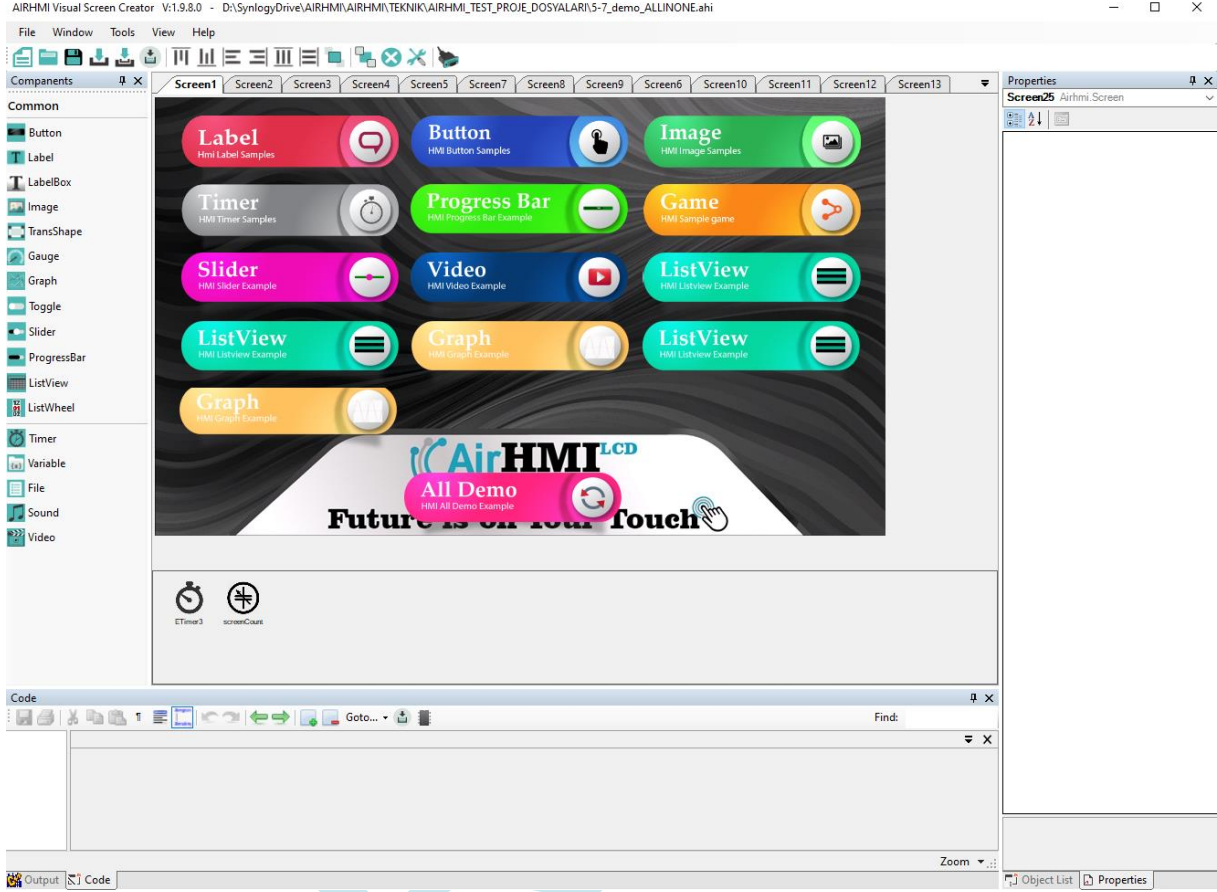
1) POWER konektöre ait pinler şu şekildedir;



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

AIRHMI LCD EKРАН EDITOR KILAVUZU

4. AirHMI EDITOR 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



AIRHMI LCD EKРАН EDITOR KILAVUZU

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



AİRHMI LCD EKРАН EDITOR KILAVUZU

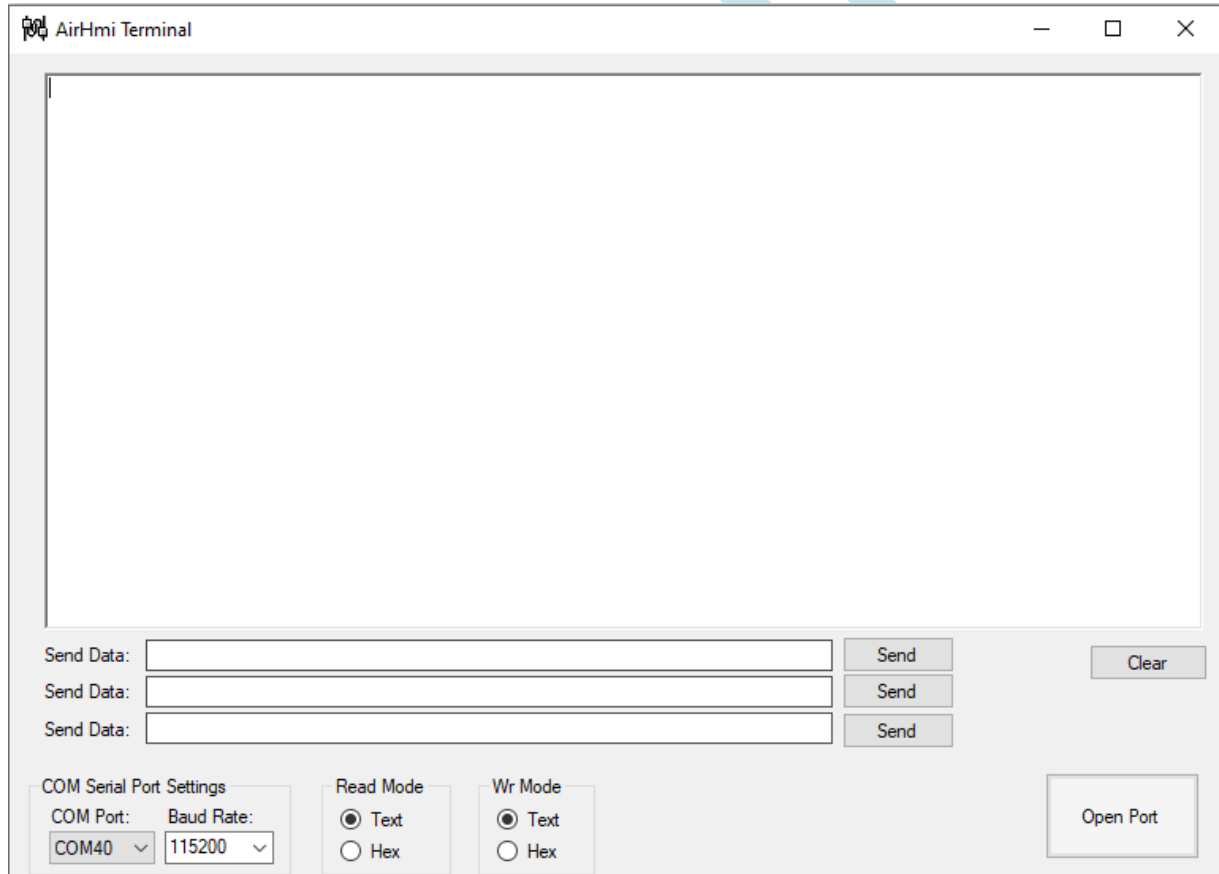
Sola Hizala, Sağa Hizala, Üst Hizala ve Alta Hizala; dikey ve yatay olarak ortalama özellikleri sayesinde belirlenen nesnelere istenen şekilde hizalanmış veya ortalanmış hale getirilir.

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

Seri Port Terminal Ekranı

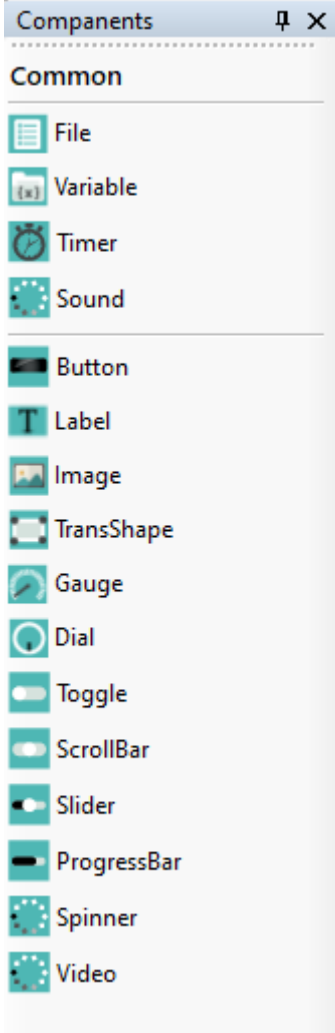


Araçlar penceresinden seri port simgesine tıkladığımız zaman,



Bu pencere açılır. Airhmi ekranınız ile

4.3 BİLEŞENLER BÖLMESİ



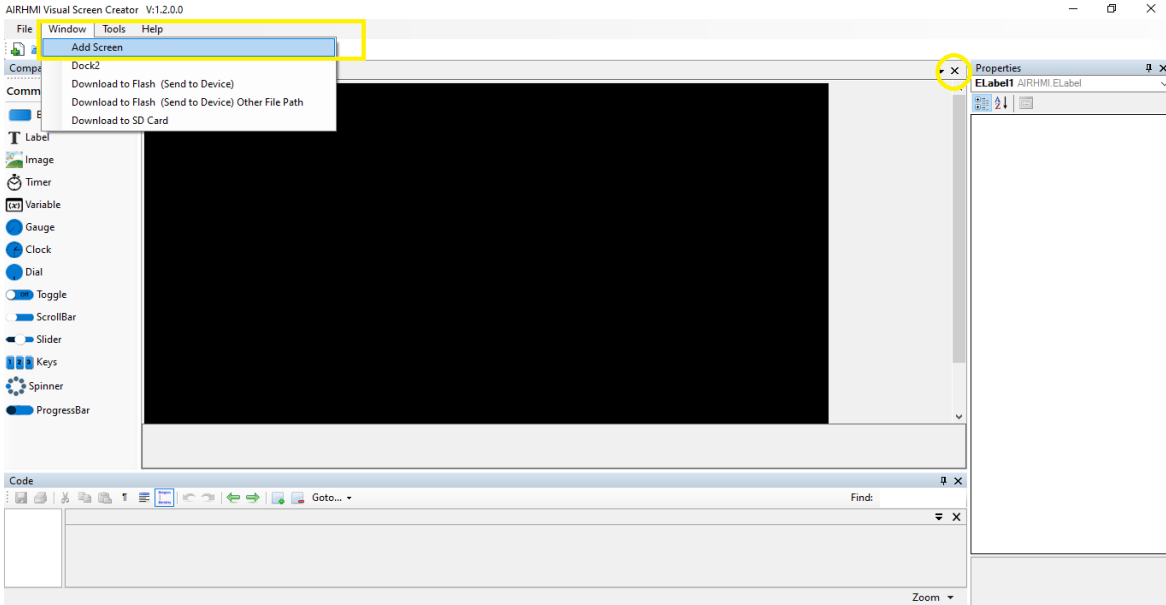
AirHMI LCD Tasarım Ekranı'nda gösterilecek hazır nesnelerin bulunduğu bölümdür. Kullanılmak istenilen nesne üzerine tıklanıp ekran alanına sürüklenerek projeye eklenmektedir. Ekranda gösterilmeyen harici nesneler de bu bölümde bulunmaktadır: Timer ve Variable. Bu nesneler ekran alanının alt kısmında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

AIRHMI LCD EKРАН EDITOR KILAVUZU

4.4 EKРАН / KOMUT SEKМESİ



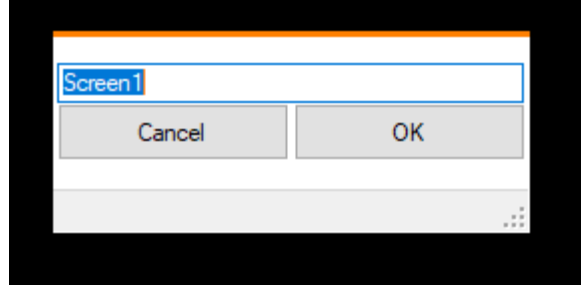
Tasarım projeleri genelde tek ekran olarak kullanılmayıp aynı anda farklı ekranlara ihtiyaç duymaktadır. Açılış Genel Gösterim Ekranı, Menü Ayar Ekranı, Detaylı Gösterim Ekranı vs... Bu nedenle AirHMI Editör içerisinde kullanıcı istekleri doğrultusunda birden fazla özgün ve yaratıcı ekran tasarımı yapabilmektedir. 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.

AIRHMI LCD EKLAN EDITOR KILAVUZU



4.5 TASARIM ANA EKLAN 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

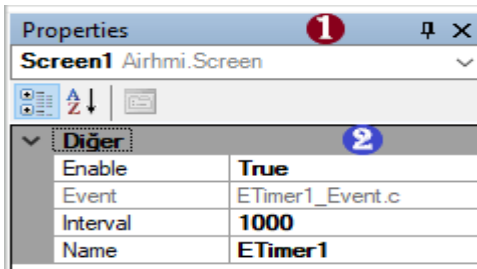
AİRHMI LCD EKРАН EDITOR KILAVUZU

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



4.83.7.1 Projede Kullanılan Nesnelerin Gösterim Alanı

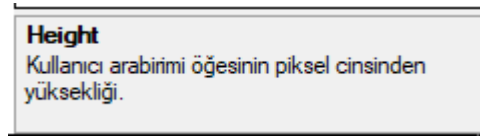
LCD ekran tasarımında birçok nesne kullanımı gerçekleştirebilmektedir. Her nesnenin kendine özgü ayarları yapılmaktadır. Fazla detay istenilen projelerde özellikle ayar yapılmak istenilen nesnenin tasarım ekranından bulunması karmaşık bir hal alabilmektedir. Bu karmaşıklığı önlemek için tasarımda kullanılan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

AİRHMI LCD EKРАН EDITOR KILAVUZU

4.93.7.2 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 ÖZİNTELİKLERİN 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



AIRHMI LCD EKРАН EDITOR KILAVUZU

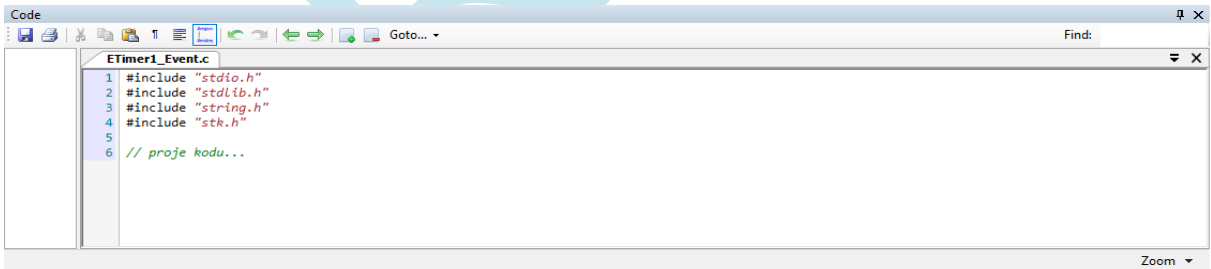
Kullanıcı Proje Kodu için geçerli bir AirHMI PICOC Kod Talimatını içerir. Bu bölüm programlamayı öğretmeyecek, ancak kullanıcının kod ekleyebilmesi için genel olarak yardımcı olacaktır. Bu alan içerisinde kullanıcılar ister Timer componentinin event'larına isterlerse de ekranda kullandıkları nesnelere event'larına C tabanlı kodları yazabileceklerdir. 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 şekilde 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

AİRHMI LCD EKРАН EDITOR KILAVUZU

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
1 #include "stdio.h"
2 #include "stk.h"
3
4 char uartData[10];
5 int uartsiz;
6 uartDataGet(uartData, &uartsiz);
7
8 if(uartsiz > 0)
9 {
10     ImageSet ("EImage1" , "Visible" , "1");
11     LabelSet ("ELabel1" , "Caption" , "Deneme");
12     LocalIntVarSet("Variable1" , 2);
13
14     DrawScreenGet();
15
16 }
```

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. Değişkenlerin Birbirlerine Dönüşümleri

Airhmi ekranlar C programlama alt yapısını kullanmaktadır. Aşağıdaki fonksiyonlarda işlemler yapabilmek için çoğu zaman tip dönüşümlerine ihtiyacımız olacaktır. Tip dönüşümleri matematiksel ifadeler ve veriler arasında işlemler için gereklidir. Ayrıca verilerin variable nesnelere kayıt edilmesi içinde tanımlanan variable ile aynı tipe dönüşmüş olması gerekir. Airhmi ekranı etkin bir şekilde kullanabilmek için bu konu oldukça önemlidir. Detaylı örnekler ile gerekli tüm dönüşümler hakkında detaylı bilgi verilecektir. Değişkenlerin dönüşümü için standart C kodları olan `sprintf`, `atoi` ve `aof` kullanılabilir. Bunun yanında işlemleri daha kolay yapabilmek adına airhmi size hazır fonksiyonlar sunmaktadır. Aşağıdaki fonksiyonları kullanarak tip dönüşümleri yapabilirsiniz.

5.1 İnteger İfadeyi String(Char Dizi)'e Dönüştürme

Airhmi de integer ifadeler tam sayı değer alan 4 byte yer kaplayan değişkenlerdir.

```
int i; i=5;
```

buradaki integer i ifadesini airhmi de bir label da göstermek için veya char * olan bir alana veri olarak göndermek için dönüşüm yapmak ihtiyacı olabilir. Bu durumda `Convert_IntToString` fonksiyonunu kullanabilirsiniz.

```
void Convert_IntToString(int,char *)
```

İnteger değişkenleri char * dizisine dönüştürmek için kullanılır.

Örnek Kullanım:

```
#include "stk.h"  
int i = 5;  
char data[200];
```

```
Convert_IntToString(i,data);  
LabelSet("Label1","Text",data);
```

Bu örnekte `LabelSet` fonksiyonu 3. Parametre olarak char * tipinde bir değişken kabul eder.

AIRHMI LCD EKРАН EDITOR KILAVUZU

i deęişkenini doğrudan bu fonksiyona veremeyiz. Bundan dolayı Convert_IntToString fonksiyonu ile LabelSet fonksiyonu için gerekli olan char * tipine dönüştürmüş olduk.

void Convert_FloatToString(float,char *)

Airhmi float ifadeler kullanımı oldukça yaygındır. Float ifadeleri Labelde göstermek veya başka bir fonksiyonda char * olarak kullanmak için dönüşüm yapmak gerekir.

Örnek Kullanımı:

```
#include "stk.h"
float i = 5.2;
char data[200];

Convert_FloatToString(i,data);
LabelSet("Label1","Text",data);
```

void Convert_StringToInt(char *,int *)

String(char *) ifadeleri integere dönüştürme işlemi, genelde bir yazı ile matematiksel işlemlere tabi tutmak için kullanılır. Örneğin bir label değerini alıp 2 ile çarpıp başka bir label a yazalım.

```
#include "stk.h"
int i;
char data[200];

LabelGet("Label1","Text",data);
Convert_StringToInt(data,&i);
i = i * 2;
Convert_IntToString(i,data);
LabelSet("Label2","Text",data);
```


AIRHMI LCD EKLAN EDITOR KILAVUZU

```
void Convert_StringToFloat(char *,float *)
```

String virgüllü bir ifadeyi matematiksel işleme tabi tutma ihtiyacımız olabilir. Klavyeden girilen bir değeri çarpma bölme gibi işlemlere tabi tutmak için float ifadeye çevirmek gerekir.

```
#include "stk.h"
float i;
char data[200];

LabelGet("Label1","Text",data);
Convert_StringToFloat(data,&i);
i = i * 2.5;
Convert_FloatToInt(i,data);
LabelSet("Label2","Text",data);
```

* Convert_StringToFloat fonksiyonunda 2. Parametre olarak float * olmasının sebebi şudur, float değişken olan i nin değerini fonksiyon doldurup bize geri verir. Aslında burada pointer kullanılmıştır. Pointer ler konusunu bilmiyorsanız kısaca internette araştırma yapabilirsiniz.

5.2sprintf kullanımı

sprintf c de tip dönüşümleri veya ifadeleri birleştirme gibi birçok işlev için kullanılır.

Örnek 1:

```
char data[20];
int i = 5;
sprintf(data,"%d",i); // integer olan i değerini char * a dönüştürmüş olduk.
```

Örnek 2:

```
char data[20];
int i = 5;
int k = 6;
sprintf(data,"%d%d",i,k); // i ve k değerini tek bir değişkende char * olarak yazdık.
Data nın içeriği = 56 değeridir.
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

Örnek 3:

```
char data[20];
float i = 5.2;
int k = 6;
sprintf(data,"%f%d",i,k); // i ve k değerini tek bir değişkende char * olarak yazdık.
data nın içeriği = 5.26 değeridir.
```

printf in diğer kullanımları için internete bolca örneği vardır.

5.3atoi

atoi C programlama dilinde bir fonksiyondur ve bir karakter dizisini (string) tam sayıya dönüştürmek için kullanılır. Genellikle, karakter dizileri kullanıcı girişi veya dosya okuma gibi yerlerden alınan verileri işlerken kullanılır.

atoi fonksiyonunun basit bir kullanım örneği:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    char str[] = "12345";
    int number = atoi(str);
    return 0;
}
```

Bu örnekte, "12345" karakter dizisi atoi fonksiyonuna geçirilir ve bu dizeyi tam sayıya dönüştürür. Sonuç olarak, number değişkeni 12345 olur. Artık integer olarak number değişkeni üzerinden işlemler yapabilirsiniz.

5.4atof

atof fonksiyonu C programlama dilinde bir diğerk sık kullanılan fonksiyondur. Bu fonksiyon, bir karakter dizisini (string) ondalıklı sayıya dönüştürmek için kullanılır.

atof fonksiyonunun basit bir kullanım örneđi:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    char str[] = "3.14";
    double number = atof(str);
    return 0;
}
```

Bu örnekte, "3.14" karakter dizisi atof fonksiyonuna geçirilir ve bu dizeyi ondalıklı sayıya dönüştürür. Sonuç olarak, number deđişkeni 3.14 olur.

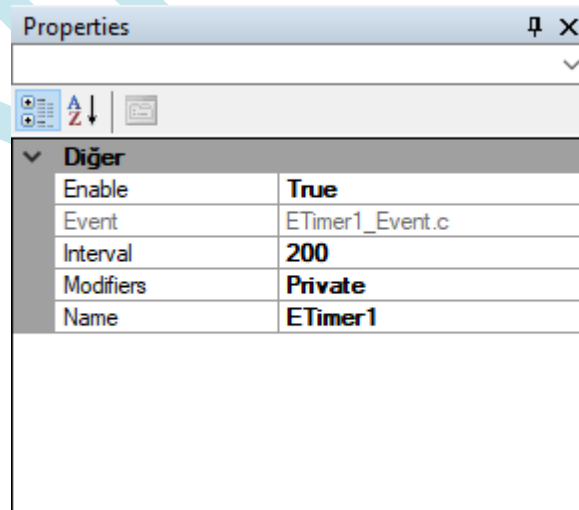
6. AİRHMI NESNELERİ VE FONKSİYONLAR

6.1 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 olacağını 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



| Properties | |
|------------|-----------------|
| ▼ | |
| A Z ↓ | |
| ▼ Diğer | |
| Enable | True |
| Event | ETimer1_Event.c |
| Interval | 200 |
| Modifiers | Private |
| Name | ETimer1 |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| 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 |

AIRHMI LCD EKРАН EDITOR KILAVUZU

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.
```

AIRHMI

AIRHMI LCD EKLAN EDITOR KILAVUZU

6.2Button

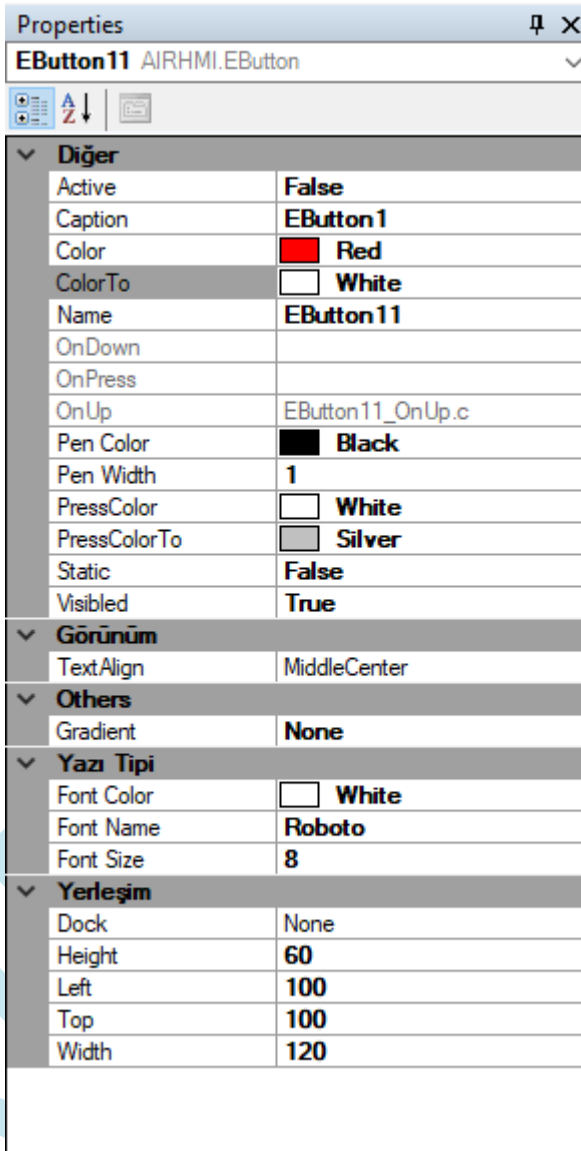
Buton nesnesi basıldıđı zaman herhangi bir iřlem yaptırmayı sađlayan nesnedir. Örneđin kullanıcıdan alınan veriyi bir yere göndermek, alınan veriyle iřlem yapmak veya mesaj verdirmek amacıyla kullanılabilir. Butonun konumunu istediđiniz yere sürükleyebilir ve boyutunu kenarlarından çekerek ayarlayabilirsiniz.







Button Şekilleri



AIRHMI LCD EKРАН EDITOR KILAVUZU

Button Properties Penceresi



| Properties | |
|--------------------------|---|
| EButton11 AIRHMI.EButton | |
| A Z ↓ | |
| ▼ Diğer | |
| Active | False |
| Caption | EButton 1 |
| Color |  Red |
| ColorTo |  White |
| Name | EButton 11 |
| OnDown | |
| OnPress | |
| OnUp | EButton11_OnUp.c |
| Pen Color |  Black |
| Pen Width | 1 |
| PressColor |  White |
| PressColorTo |  Silver |
| Static | False |
| Visibled | True |
| ▼ Görünüm | |
| TextAlign | MiddleCenter |
| ▼ Others | |
| Gradient | None |
| ▼ Yazı Tipi | |
| Font Color |  White |
| Font Name | Roboto |
| Font Size | 8 |
| ▼ Yerleşim | |
| Dock | None |
| Height | 60 |
| Left | 100 |
| Top | 100 |
| Width | 120 |

AIRHMI LCD EKРАН EDITOR KILAVUZU

| 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 Bottom 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 |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| | | |
|-------|--|---|
| 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 gözüktür, “False” ayarlandığı zaman ise gözükmeyiz.

Örnek Kod:

```
ButtonSet ("EButton1" , "Visible" , "True");
```

Active ayarlama komutu

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

AİRHMI LCD EKРАН EDITOR KILAVUZU

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");
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

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!");
```

AIRHMI LCD EKTRAN EDITOR KILAVUZU

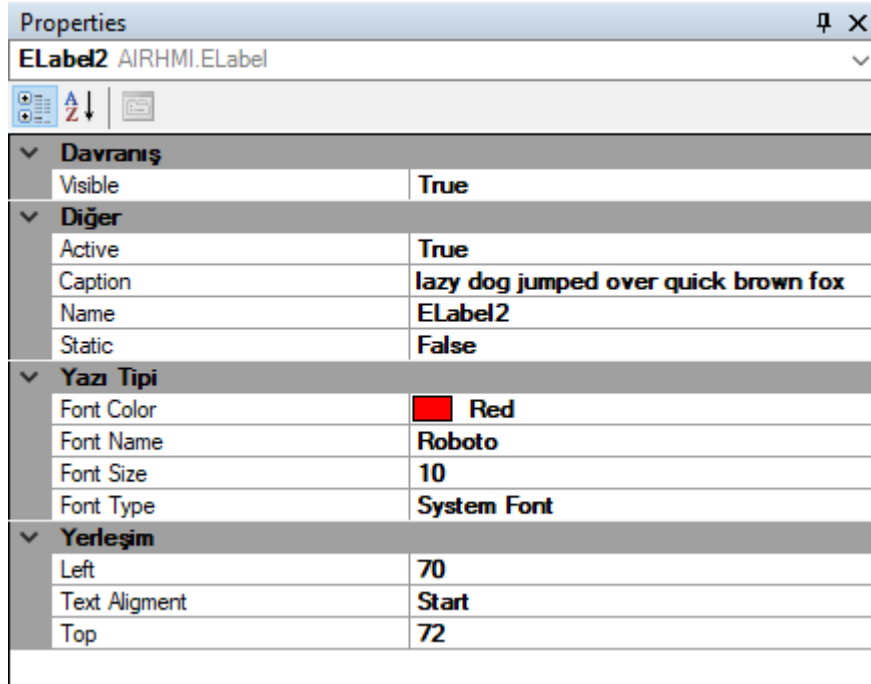
6.3Label

Ekranı yazı yazma amacı ile kullanılan nesnedir. Font size olarak 8 den 102' ye kadar desteklemektedir. Default Font "Roboto" dur.



AIRHMI LCD EKРАН EDITOR KILAVUZU

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 |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| | | |
|-------|--|---|
| 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("ELabell" , "Active" , "True");
```

Visible ayarlama komutu

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

Örnek Kod:

```
LabelSet("ELabell" , "Visible" , "1");
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

Left ayarlama komutu

LabelSet(Nesne adı , “Left” , “10”);

Örnek Kod:

LabelSet(*"ELabell"* , *"Left"* , *"10"*);

Top ayarlama komutu

LabelSet(Nesne adı , “Top” , “255”);

Örnek Kod:

LabelSet (*"ELabell"* , *"Top"* , *"255"*);

FontSize ayarlama komutu

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

Örnek Kod:

LabelSet(*"ELabell"* , *"FontSize"* , *"16"*);

Font_Color ayarlama komutu

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

Örnek Kod:

LabelSet(*"ELabell"* , *"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 (*"ELabell"* , *"Caption"* , *"Hello World!"*);

LabelSet (*"ELabell"* , *"Text"* , *"Hello World!"*);

AİRHMI LCD EKРАН EDITOR KILAVUZU

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);
```

AIRHMI LCD EKTRAN EDITOR KILAVUZU

6.4Image

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.



AIRHMI LCD EKРАН EDITOR KILAVUZU

Image Properties Penceresi

| Properties | |
|-----------------------|----------------|
| Elmage2 AIRHMI.Elmage | |
| Z ↓ | |
| Davranış | |
| Visible | True |
| Diğer | |
| Active | True |
| Locked | False |
| Name | Elmage2 |
| OnDown | |
| OnPress | |
| OnUp | Elmage2_OnUp.c |
| Opacity | 100 |
| PictureName | Asset 9.png |
| PicturePressImage | |
| ScaleX | 0,5935 |
| ScaleY | 0,5984 |
| Static | False |
| Yerleşim | |
| Dock | None |
| Height | 73 |
| Left | 17 |
| Top | 242 |
| Width | 238 |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| Özellik | Seçenek | Açıklama |
|------------------|-----------------|--|
| Active | True False | Açık olması durumunda resim buton gibi kullanılabilir. Kapalı olması durumunda 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. |
| Image File | | Bilgisayardan yüklemeniz gereken resim dosyasıdır. |
| Press Image File | | Image nesnesine basılı tutarken ki resimdir. |
| ScaleX | | Image nesnesin X boyutundaki büyütme ve küçültme oranıdır. |
| ScaleY | | Image 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");
```

AİRHMI LCD EKRAM EDITOR KILAVUZU

Top ayarlama komutu

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

Örnek Kod:

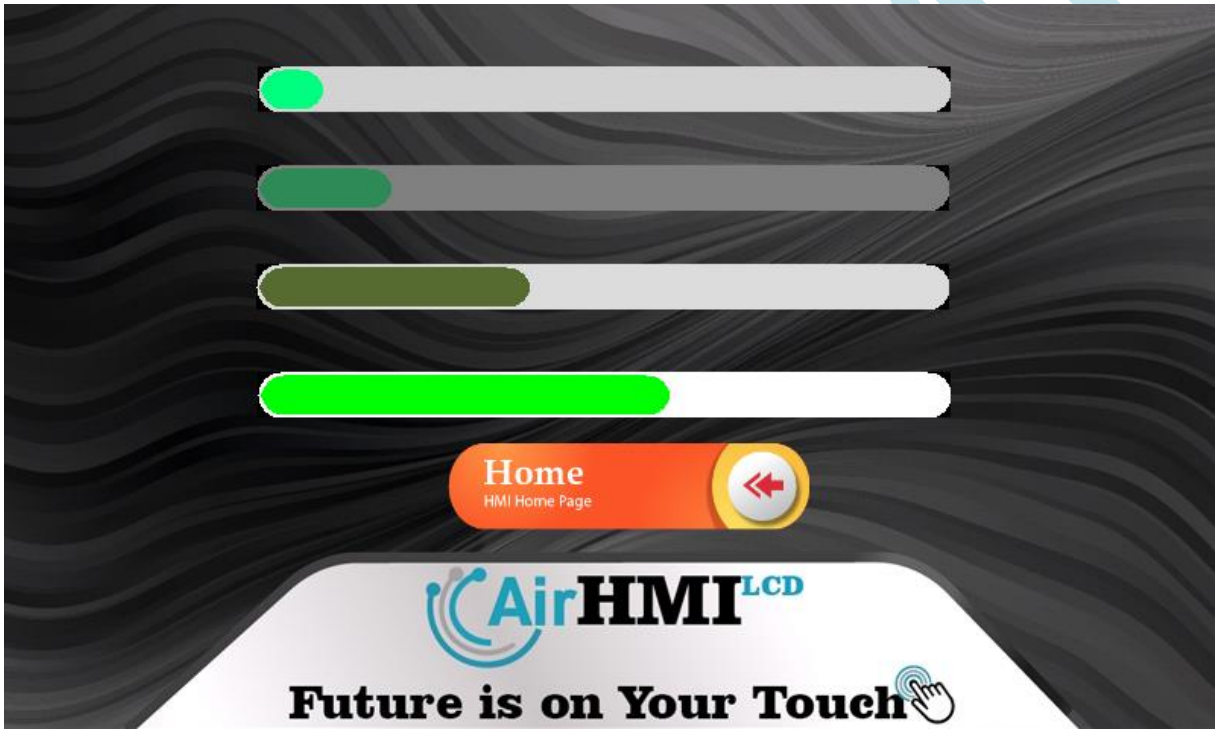
ImageSet ("*EImage1*" , "*Top*" , "*255*");

AİRHMI

AIRHMI LCD EKLAN EDITOR KILAVUZU

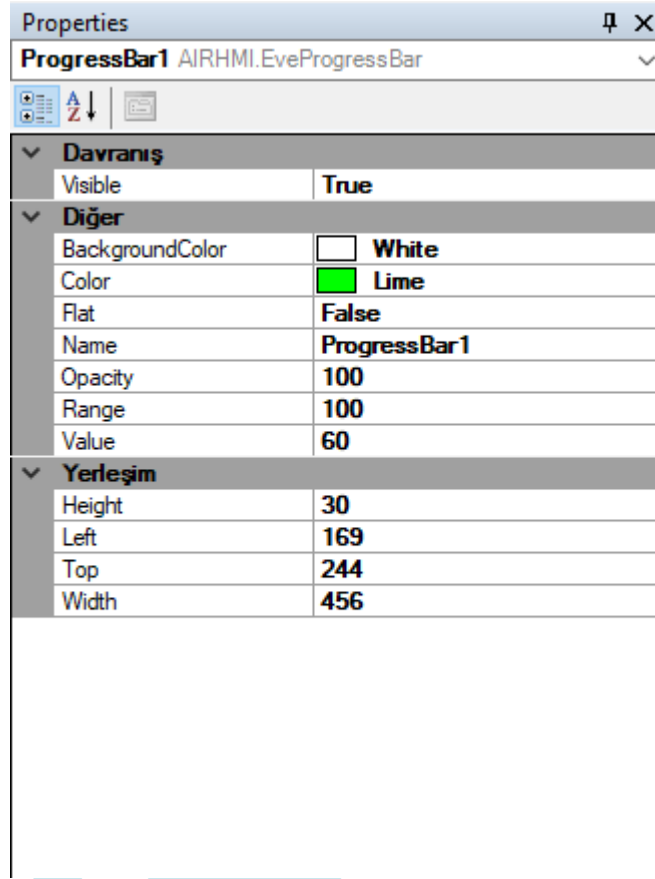
6.5ProgressBar

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.



AIRHMI LCD EKРАН EDITOR KILAVUZU

ProgressBar Properties Penceresi



| Properties | |
|------------------------------------|--------------|
| ProgressBar1 AIRHMI.EveProgressBar | |
| Z ↓ | |
| ▼ Davranış | |
| Visible | True |
| ▼ Diğer | |
| BackgroundColor | White |
| Color | Lime |
| Flat | False |
| Name | ProgressBar1 |
| Opacity | 100 |
| Range | 100 |
| Value | 60 |
| ▼ Yerleşim | |
| Height | 30 |
| Left | 169 |
| Top | 244 |
| Width | 456 |

AİRHMI LCD EKLAN EDITOR KILAVUZU

| 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");
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

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");
```

AIRHMI LCD EKРАН EDITOR KILAVUZU

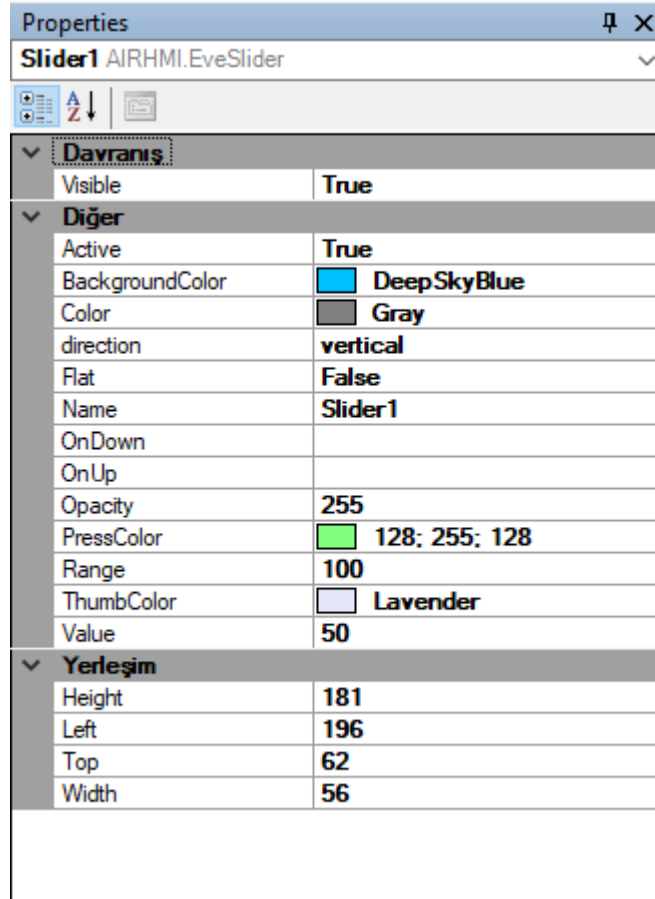
6.6Slider

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



AIRHMI LCD EKCRAN EDITOR KILAVUZU

Slider Properties Penceresi



| Properties | |
|--------------------------|---------------|
| Slider1 AIRHMI.EveSlider | |
| A Z | |
| ▼ Davranis | |
| Visible | True |
| ▼ Diđer | |
| Active | True |
| BackgroundColor | DeepSkyBlue |
| Color | Gray |
| direction | vertical |
| Flat | False |
| Name | Slider1 |
| OnDown | |
| OnUp | |
| Opacity | 255 |
| PressColor | 128; 255; 128 |
| Range | 100 |
| ThumbColor | Lavender |
| Value | 50 |
| ▼ Yerleşim | |
| Height | 181 |
| Left | 196 |
| Top | 62 |
| Width | 56 |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| 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 değiştirir. |
| 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. |
| | | |

AİRHMI LCD EKTRAN EDITOR KILAVUZU

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");
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

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);
```


AIRHMI LCD EKTRAN EDITOR KILAVUZU

6.7Gauge

Gauge nesnesi analog deęerleri gstermek iin etkili bir nesnedir. Aynı zamanda hız gstergesi olarak da kullanılır.



AIRHMI LCD EKLAN EDITOR KILAVUZU

Gauge Properties Penceresi

| Davranış | |
|--------------|--|
| Visible | True |
| Diğer | |
| Active | True |
| Color | ■ Blue |
| Flat | False |
| MajorCount | 10 |
| MinorCount | 5 |
| Name | Gauge1 |
| OnDown | |
| OnUp | |
| PenColor | ■ Red |
| PressColor | ■ White |
| Radius | 94 |
| Range | 100 |
| Tag | 255 |
| TicksVisible | False |
| Value | 0 |
| Yerleşim | |
| Left | 59 |
| Top | 39 |

AIRHMI LCD EKРАН EDITOR KILAVUZU

| 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 değiştirir. |
| 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" );
```

AİRHMI LCD EKCRAN EDITOR KILAVUZU

Top ayarlama komutu

GaugeSet(Nesne adı , "Top" , "Ekcrandaki 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:

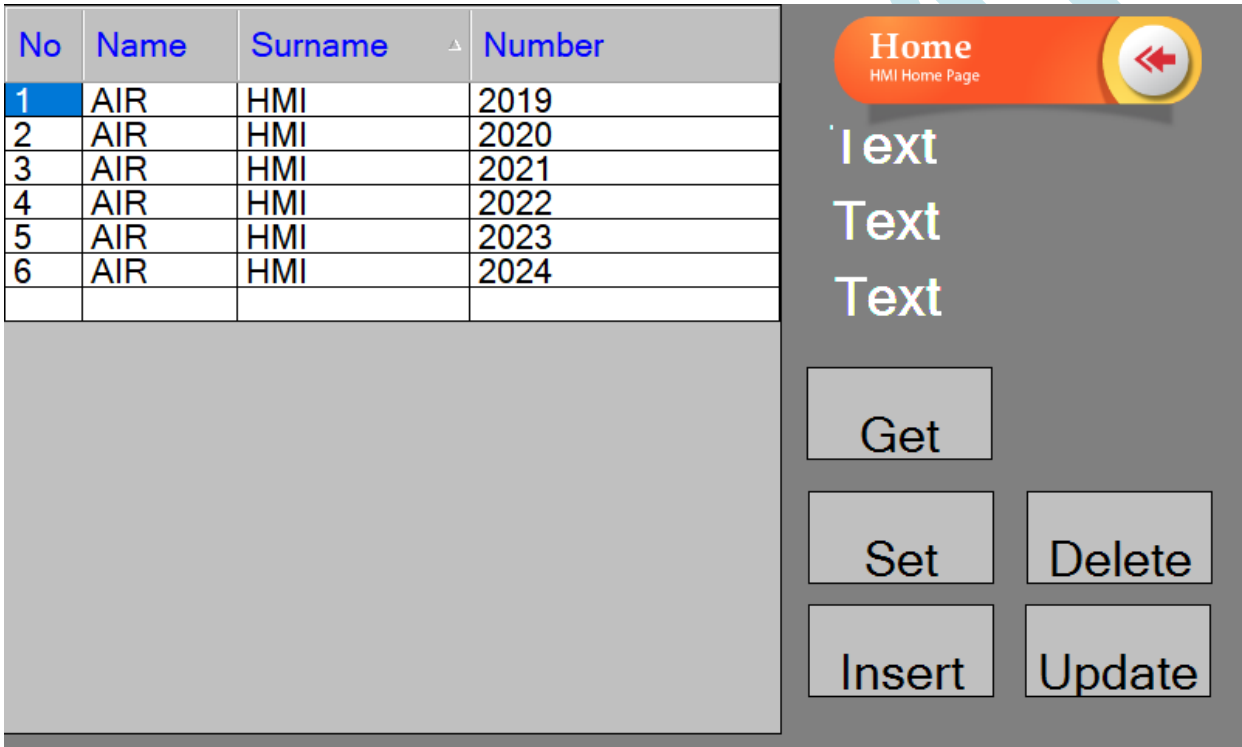
```
GaugeSet("Gauge1" , "Range" , "30");
```

AIRHMI LCD EKTRAN EDITOR KILAVUZU

6.8 ListView

ListView nesnesi, airhmi ekran üzerinde kullanıcılara tablo oluşturma imkânı sağlar. Tabloya girilen veriler çok yönlü olarak kullanılabilir. Örneğin bir dil dosyası olarak veya sistem ayarlarını ayrı ayrı tutabileceğiniz yapı veya sistem loglarını tutabileceğiniz bir yapı olarak kullanabilirsiniz. Listeye veri ekleme, güncelleme, silme verileri okuma gibi birçok fonksiyonları aşağıda bulabilirsiniz.






| No | Name | Surname | Number |
|----|------|---------|--------|
| 1 | AIR | HMI | 2019 |
| 2 | AIR | HMI | 2020 |
| 3 | AIR | HMI | 2021 |
| 4 | AIR | HMI | 2022 |
| 5 | AIR | HMI | 2023 |
| 6 | AIR | HMI | 2024 |
| | | | |



The screenshot displays the AIRHMI LCD Editor interface. On the left, a ListView table is shown with columns for 'No', 'Name', 'Surname', and 'Number'. The table contains six rows of data, with the first row highlighted in blue. On the right, a control panel is visible, featuring a 'Home' button with a back arrow, three 'Text' labels, and five action buttons: 'Get', 'Set', 'Delete', 'Insert', and 'Update'.

AIRHMI LCD EKLAN EDITOR KILAVUZU

ListView Properties Penceresi

| List1 AIRHMI.EListView | |
|-------------------------|--|
| ▼ Davranış | |
| Enabled | False |
| ▼ Diğer | |
| BackGround Color |  Silver |
| Enable Motion | True |
| Enable Sort | 1 |
| Grid Visible | True |
| List File | List 1_List File.txt |
| Locked | False |
| Name | List 1 |
| Visibled | True |
| ▼ Grid | |
| Grid Font | Roboto |
| Grid Font Color |  Black |
| Grid Font Size | 16 |
| selected Item Color |  Blue |
| ▼ Header | |
| Header BackGround Color |  Silver |
| Header Font | 0 |
| Header Font Color |  Blue |
| Header Font Size | 16 |
| Header Height | 50 |
| Header List | No^Name^Surname^Number |
| Header List Size | 50^100^150^200 |
| Header Type | int^str^str^str |
| ▼ Yerleşim | |
| Height | 471 |
| Left | 0 |
| Top | 0 |
| Width | 503 |

AIRHMI LCD EKРАН EDITOR KILAVUZU

| 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. |
| Left | | Ekran üzerindeki pozisyonu belirtir. X koordinatı |
| Top | | Ekran üzerindeki pozisyonu belirtir. Y koordinatı |
| Enable Motion | | Liste içerisinde sağa sola sürüklenmeye izin verir. |
| Grid Visisble | | Liste grid çizgileri gözükmesi ile ilgilidir. |
| List File | | Editör içerisinde listeye ilk defa veri atmak istediğimizde kullanabiliriz. |
| Grid Font | | Grid içerisindeki yazının fontudur. |
| Grid Font Color | | Grid içerisindeki yazının rengidir. |
| Grid Font Size | | Grid içerisindeki yazının font büyüklüğüdür. |
| Header Font Color | | Header Font Rengidir. |
| Header_BackGround_Color | | Header Arkaplan Font Rengidir. |
| Header_Font_Size | | Header Font Size dir. |
| Header_List | | Başlıklar kısmı buraya yazılmalıdır. |
| Header_List_Size | | Herbir başlığın kapladığı alanı belirler. |
| Header_Type | | Verilerin tipini belirtir. |
| BackGround_Color | | Liste arka planıdır. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

AİRHMI LCD EKРАН EDITOR KILAVUZU

ListViewSet()

Açıklama

ListView nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

```
void ListViewSet(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

Row değeri değiştirme komutu

Seçilen satırdaki yerin değerini değiştirmek için kullanılır. Dokunmatik ekrandan o satır seçilmiş olması gerekir. Row0 dan başlar Rown e kadar gider. Buradaki n değeri sütun sayısıdır.

```
ListViewSet( Nesne adı , “Rown” , yeni değer String olarak );
```

Örnek Kod:

```
ListViewSet( Nesne adı , “Row0” , “1” );  
ListViewSet( Nesne adı , “Row1” , “AIR” );  
ListViewSet( Nesne adı , “Row2” , “HMI” );
```

Delete_Selected komutu

AİRHMI LCD EKLAN EDITOR KILAVUZU

Seçilen satırı silmek için kullanılır. 3. Parametrenin önemi yok. Bundan dolayı 0 veriyoruz.

```
ListViewSet( Nesne adı , “Delete_Selected” , 0 );
```

Örnek Kod:

```
ListViewSet( “List1” , “Delete_Selected” , 0 );
```

update komutu

Listeyi kalıcı olarak hafızada kaydetmek için kullanılır. 3. Parametrenin önemi yok. Bundan dolayı 0 veriyoruz.

```
ListViewSet( Nesne adı , “update” , 0 );
```

Örnek Kod:

```
ListViewSet( “List1” , “update” , 0 );
```

insert komutu

Listeye yeni veri girmek için kullanılır. Bu komut listenin en sonuna veri ekler.

```
ListViewSet( Nesne adı , “insert” , “data1^data2^data3” );
```

Örnek Kod:

```
ListViewSet( “List1” , “insert” , “7^air^hmi^2025” );
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

ListViewGet()

Açıklama

ListView nesnesinden veri okuma komuttur.

Fonksiyon

void ListViewGet(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

Row değerini okuma komutu

Seçilen satırdaki yerin değerini okumak için kullanılır. Dokunmatik ekrandan o satır seçilmiş olması gerekir. Row0 dan başlar Rown e kadar gider. Buradaki n değeri sütun sayısıdır.

ListViewGet(Nesne adı , "Rown" , değer String olarak);

Örnek Kod:

```
char row[200];
```

```
ListviewGet("List1", "Row1", row);
```

```
ListviewGet("List1", "Row2", row);
```

```
ListviewGet("List1", "Row3", row);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

ListViewSetXY()

Açıklama

ListView nesnesinin içeriğini değiştirmek için kullanılır. Koordinat sistemi gibi kullanılır. Bu fonksiyon kullanımı için nesneye dokunmuş olmanıza gerek yoktur. İstedığınız gibi herhangi bir alanı güncelleme yapabilirsiniz.

Fonksiyon

```
void ListViewSetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

| Parametre | Açıklama |
|-----------|--------------------|
| name | Nesnenin ismi |
| X | Sutun numarasıdır. |
| Y | Satır numarasıdır. |
| Value | Sutun değeridir. |

Örnek kod

```
ListViewSetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

Örnek Kod:

```
ListViewSetXY("List1" , 1 , 2 , "yeni deger"); // 2. Satır 1. Sutunun değerini değiştir.
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

ListViewGetXY()

Açıklama

ListView nesnesinin içeriğini okumak için kullanılır. Koordinat sistemi gibi kullanılır. Bu fonksiyon kullanımı için nesneye dokunmuş olmanıza gerek yoktur. İstedığınız gibi herhangi bir alandan veri okuması yapabilirsiniz.

Fonksiyon

```
void ListViewGetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

| Parametre | Açıklama |
|-----------|--------------------|
| name | Nesnenin ismi |
| X | Sutun numarasıdır. |
| Y | Satır numarasıdır. |
| Value | Sutun değeridir. |

Örnek kod

```
ListViewGetXY(unsigned char *name , int X , int Y , unsigned char *value);
```

Örnek Kod:

```
char data[100];
```

```
ListViewGetXY("List1" , 1 , 2 , data); // 2. Satır 1. Sutunun değerini aldık.
```

```
LabelSet("label1","Text",data);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

ListViewSetSort()

Açıklama

ListView nesnesinin ilk sütununa göre alfabetik sıralama yapar. Bu fonksiyon yazılım aşamasında listenin sıralı mı, normal mi başlayacağını belirlemek veya belirli durumlarda bunu gerçekleştirmek için kullanılabilir. Fakat kullanabilmek için liste özelliklerinde “Enable Sort” bölümünün aktif olması gerekir (Enable Sort = 1 olmalı). Enable Sort özelliğini hem editör hemde yazılım tarafından aktif edilebilir. Bu özelliği listeyi tam tersine çevirmek istediğiniz durumlarda da kullanabilirsiniz fakat tersleme işlemi yapmak için ilk sütunda sıralı şekilde rakamlar olması gerekir (örneğin: 1,2,3,4,5).

Fonksiyon

```
void ListViewSetSort (unsigned char *name ,int value);
```

| Parametre | Açıklama |
|-----------|--|
| name | Nesnenin ismi |
| Value | Nesneyi reverse etmek için 1 aksi durum için 0 kullanılır. |

Örnek kod

```
ListViewSetSort(unsigned char *name, int value);
```

Örnek Kod:

```
ListViewSetSort (“ListView1”, 1); // liste ters formda.
```

```
ListViewSetSort (“ListView1”, 0); // liste normal formda.
```

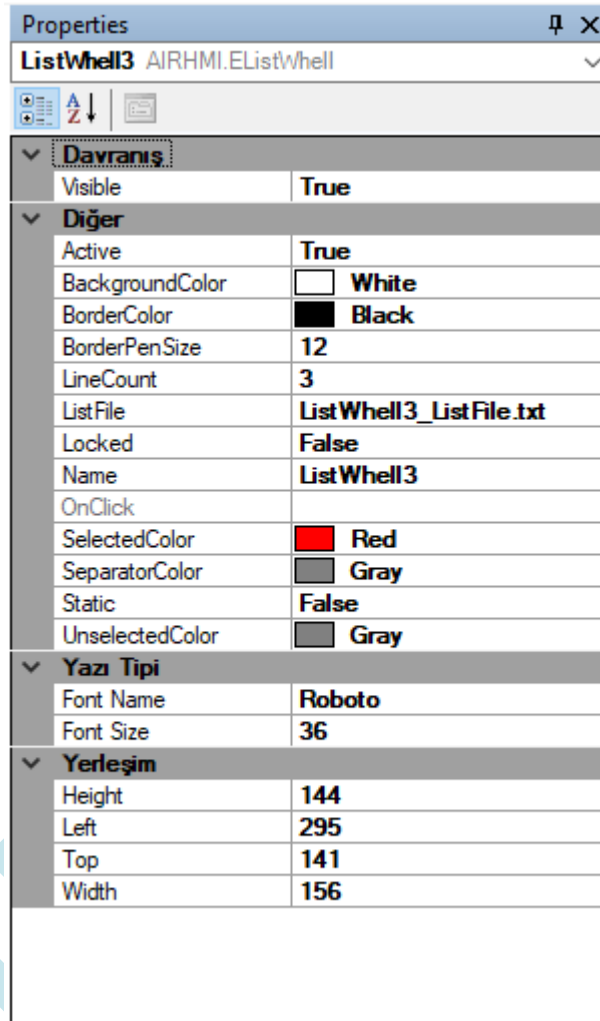
6.9ListWheel



Kullanıcıların bir liste içerisindeki öğeler arasında dikey olarak kaydırma yapmasını sağlar. Bu widget, her bir öğeyi bir silindir üzerinde dönen bir çark gibi gösterir ve bu da kullanıcıya 3D benzeri bir kaydırma deneyimi sunar. Özellikle tarih seçiciler, saat seçiciler gibi alanlarda kullanılarak kullanıcıya estetik ve işlevsel bir deneyim sunar. ListWheel genellikle listenin öğeleri merkeze geldikçe büyür ve merkezden uzaklaştıkça küçülür, böylece odak noktasını vurgular.

AİRHMI LCD EKРАН EDITOR KILAVUZU

ListWheel Properties Penceresi



| Properties | |
|------------------------------|-------------------------|
| ListWheel3 AIRHMI.EListWheel | |
| A Z ↓ | |
| Davranis | |
| Visible | True |
| Diğer | |
| Active | True |
| BackgroundColor | White |
| BorderColor | Black |
| BorderPenSize | 12 |
| LineCount | 3 |
| ListFile | ListWheel3_ListFile.txt |
| Locked | False |
| Name | ListWheel3 |
| OnClick | |
| SelectedColor | Red |
| SeparatorColor | Gray |
| Static | False |
| UnselectedColor | Gray |
| Yazı Tipi | |
| Font Name | Roboto |
| Font Size | 36 |
| Yerleşim | |
| Height | 144 |
| Left | 295 |
| Top | 141 |
| Width | 156 |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| 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. |
| Left | | Ekran üzerindeki pozisyonu belirtir. X koordinatı |
| Top | | Ekran üzerindeki pozisyonu belirtir. Y koordinatı |
| Line Count | | Ekrandaki liste gösterim sayıdır. |
| List File | | Editör içerisinde listeye ekranda gözükmeye istediğimiz sayıları alt alta yazılır. |
| Grid Font | | Grid içerisindeki yazının fontudur. |
| Selected Color | | Seçilmiş olan yazının rengidir. |
| Unselected Color | | Seçili olmayan diğer sayıların rengidir. |
| Sperator Color | | Sayılar arasındaki çizgi rengidir. |
| BackGround_Color | | Arkaplan rengidir. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

AİRHMI LCD EKCRAN EDITOR KILAVUZU

ListWheelSet()

Açıklama

ListWheel nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

```
void ListWheelSet(unsigned char *name , unsigned char *type , unsigned char *value)
```

| Parametre | Açıklama |
|-----------|--|
| name | Nesnenin ismi |
| type | Nesnenin deęiştirilecek parametresinin ismi |
| value | Deęiştirilecek parametrenin yeni alacağı deęer |

Örnek kod

```
ListWheelSet( Nesne adı , "Value" , yeni deęer String olarak );
```

Örnek Kod:

```
#include "stk.h"  
char data[20];  
LabelGet("ELabel1" , "Text" , data);  
ListWheelSet("ListWhell1", "Value", data);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

ListWheelGet()

Açıklama

ListWheel nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

```
void ListWheelGet(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

```
ListWheelSet( Nesne adı , "Value" , yeni değer String olarak );
```

Örnek Kod:

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

```
char data[20];
```

```
ListWheelGet("ListWhell1","Value",data);
```

```
LabelSet("ELabel1" ,"Text" , data);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.10 TransShape

TransShape nesnesi ekranda görünmez fakat bir buton gibi çalışır. Tasarımı yaparken istenilen konuma, istenilen boyutta yerleştirilir. Bunun sonucunda görünmez bir butona sahip olursunuz. Tıklanıldığında ilgili kod bloğunu çalıştırır.

| | |
|---------|---------|
| Layout | |
| Height | 100 |
| Left | 177 |
| Top | 241 |
| Width | 100 |
| Misc | |
| Active | True |
| Locked | False |
| Name | EShape1 |
| OnDown | |
| OnPress | |
| OnUp | |

| Özellik | Seçenek | Açıklama |
|---------|---------------|--|
| Height | | Nesnenin tamamına ait yükseklik parametresi (pixel cinsinden). |
| Left | | Nesnenin yatay eksenindeki konumunu ifade eden parametresi (pixel cinsinden). |
| Top | | Nesnenin dikey eksenindeki konumunu ifade eden parametre (pixel cinsinden). |
| Width | | Nesnenin tamamına ait genişlik parametresi (pixel cinsinden). |
| Active | True False | TransShape nesnesine basma işlemine izin verir. TransShape nesnesine basma işlemine izin vermez. |
| Locked | True False | Nesneyi ekranda istediğiniz yere getirdikten sonra özellik True yapılırsa orada sabit kalacaktır. Özellik False olduğu durumda istediğiniz konuma getirebilirsiniz. Ekran üzerinde birden fazla nesne ile çalıştığınızda kullanmanız işinizi oldukça kolaylaştıracaktır. |
| Name | | Nesneye ait isim parametresi. Bu parametre her nesne için farklı olmalıdır çünkü yazılım yaparken ancak burada tanımlanan isim ile nesneye ulaşılabilir. |
| OnDown | | Nesneye basıldığında çalışacak yazılım buraya tıkladığınızda açılacak sayfada yazılır. |
| OnPress | | Nesneye basılı tutulduğunda çalışacak yazılım buraya tıkladığınızda açılacak sayfada yazılır. |
| OnUp | | Nesneye basılıp bırakıldığında çalışacak yazılım buraya tıkladığınızda açılacak sayfada yazılır. |

Fonksiyonlar

ShapeSet(" Nesne_ismi ", " Deęistirilecek_ozellik ", " Yeni_deęer ");

Açıklama

TransShape nesnesine ait parametreleri guncelleyen fonksiyon.

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

| Parametre | Açıklama |
|-----------|--|
| name | Nesnenin ismi |
| type | Nesnenin deęistirilecek parametresinin ismi |
| value | Deęistirilecek parametrenin yeni alacaęı deęer |

Active ayarlama komutu

ShapeSet (Nesne adı , "Active" , "1 , 0 veya True , False");

Örnek Kod:

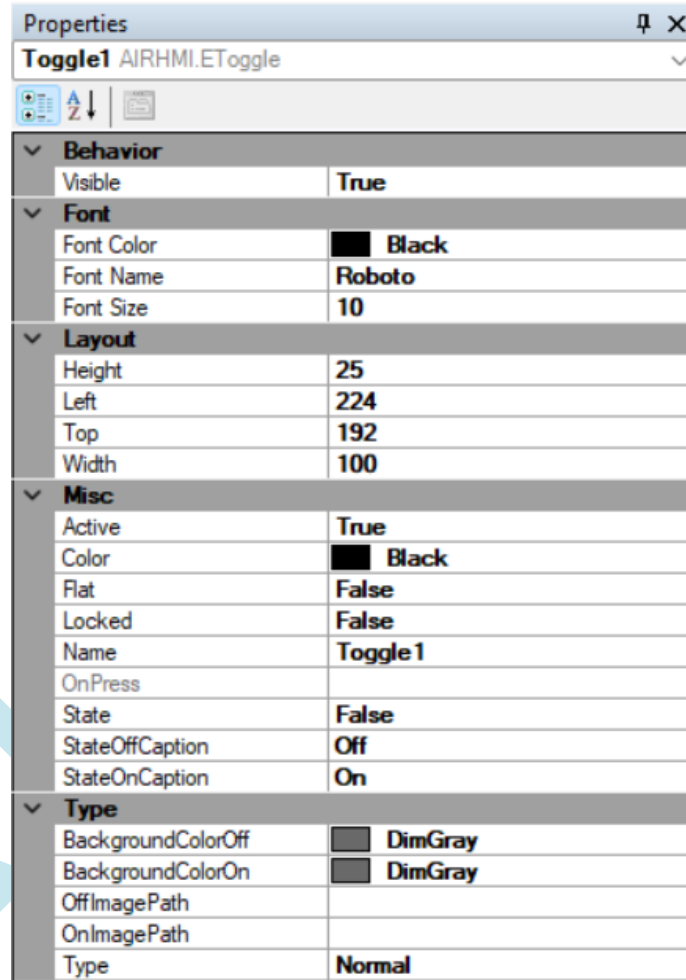
ShapeSet ("Eshape1" , "Active" , "True");

AIRHMI LCD EKРАН EDITOR KILAVUZU

6.11 Toggle

Toggle buton, bir arayüzde iki farklı durumu (örneğin açık/kapalı, etkin/pasif, var/yok) değiştirmek için kullanılan bir AIR HMI kullanıcı arayüzü öğesidir. Kullanıcı, toggle butonunu tıklayarak bu iki durum arasında geçiş yapabilir. Toggle butonlar, özellikle ayarların etkinleştirilmesi veya devre dışı bırakılması gibi işlemler için yaygın olarak kullanılır.

Toggle Properties Penceresi



| Properties | |
|------------------------|----------|
| Toggle1 AIRHMI.EToggle | |
| ▼ Behavior | |
| Visible | True |
| ▼ Font | |
| Font Color | Black |
| Font Name | Roboto |
| Font Size | 10 |
| ▼ Layout | |
| Height | 25 |
| Left | 224 |
| Top | 192 |
| Width | 100 |
| ▼ Misc | |
| Active | True |
| Color | Black |
| Flat | False |
| Locked | False |
| Name | Toggle 1 |
| OnPress | |
| State | False |
| StateOffCaption | Off |
| StateOnCaption | On |
| ▼ Type | |
| BackgroundColorOff | DimGray |
| BackgroundColorOn | DimGray |
| OffImagePath | |
| OnImagePath | |
| Type | Normal |

| Özellik | Seçenek | Açıklama |
|------------|---------|--|
| Visible | True | Ekran ilk başlatıldığı anda görünür. |
| | False | Ekran ilk başlatıldığı anda görünmez. |
| Font Color | | Toggle nesnesi üzerindeki metne ait renk seçeneği. |
| Font Name | | Toggle nesnesi üzerindeki metne ait font seçeneği. |
| Font Size | | Toggle nesnesi üzerindeki metne ait boyut seçeneği. |
| Height | | Nesnenin tamamına ait yükseklik parametresi (pixel cinsinden). |

AIRHMI LCD EKРАН EDITOR KILAVUZU

| | | |
|---------------------------|---------------|---|
| Left | | Nesnenin yatay eksenindeki konumunu ifade eden parametresi (pixel cinsinden). |
| Top | | Nesnenin dikey eksenindeki konumunu ifade eden parametre (pixel cinsinden). |
| With | | Nesnenin tamamına ait genişlik parametresi (pixel cinsinden). |
| Active | True False | Toggle nesnesine basma işlevine izin verir. Toggle nesnesine basma işlevine izin vermez. |
| Color | | Nesneye ait renk parametresi (Hex cinsinden ifade edilir <u>örneğin</u> : “#ffffff”). |
| Locked | True False | Nesneyi ekranda istediğiniz yere getirdikten sonra özellik <u>True</u> yapılırsa orada sabit kalacaktır. Özellik <u>False</u> olduğu durumda istediğiniz konuma getirebilirsiniz. Ekran üzerinde birden fazla nesne ile çalıştığınızda kullanmanız işinizi oldukça kolaylaştıracaktır. |
| Name | | Nesneye ait isim parametresi. Bu parametre her nesne için farklı olmalıdır çünkü yazılım yaparken ancak burada tanımlanan isim ile nesneye ulaşılabilir. |
| OnPress | | Nesneye basılı tutulduğunda çalışacak yazılım buraya tıkladığınızda açılacak sayfada yazılır. |
| State | True False | Ekran ilk oluşturulduğunda toggle nesnesinin hangi durumda başlayacağını (ON veya OFF) ifade eden parametre. |
| StateOffCaption | | Toggle nesnesi OFF durumunda iken üzerinde yazacak metin parametresi. |
| StateOnCaption | | Toggle nesnesi ON durumunda iken üzerinde yazacak metin parametresi. |
| BackgroundColorOff | | Toggle nesnesi OFF durumunda iken arka plan rengi (Hex formatında <u>örneğin</u> : “#ffffff”). |
| BackgroundColorOn | | Toggle nesnesi OFF durumunda iken arka plan rengi (Hex formatında <u>örneğin</u> : “#0000ff”). |
| OffImagePath | | Toggle nesnesini kendi görselleriniz ile yapmak istediğinizde buradaki alandan görseli yükleyebilirsiniz. Nesne OFF durumunda iken bu görsel aktif olacaktır. |
| OnImagePath | | Toggle nesnesini kendi görselleriniz ile yapmak istediğinizde buradaki alandan görseli yükleyebilirsiniz. Nesne ON durumunda iken bu görsel aktif olacaktır. |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| | | |
|-------------|--------|---|
| Type | Normal | Toggle nesnesi varsayılan haliyle kullanıldığında seçilecek. |
| | Image | Toggle nesnesi kendi görselleriniz ile kullanıldığında seçilecek. |

Fonksiyonlar

ToggleSet(" Nesne_ismi ", " Deđistirilecek_özelliđ ", " Yeni_deđer ");

Açıklama

Toggle nesnesine ait parametreleri güncelleyen fonksiyon.

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

| Parametre | Açıklama |
|-----------|--|
| name | Nesnenin ismi |
| type | Nesnenin deđiştirilecek parametresinin ismi |
| value | Deđiştirilecek parametrenin yeni alacađı deđer |

Visible ayarlama komutu

ToggleSet(Nesne adı , "Visible" , "1 , 0 veya True , False");

Örnek Kod:

```
ToggleSet("Toggle1" , "Visible" , "True");  
ToggleSet("Toggle1" , "Visible" , "False");
```

```
ToggleSet("Toggle1" , "Visible" , "1");  
ToggleSet("Toggle1" , "Visible" , "0");
```


AİRHMI LCD EKРАН EDITOR KILAVUZU

Active ayarlama komutu

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

Örnek Kod:

```
ButtonSet("Toggle1" , "Active" , "True");
```

Font color ayarlama komutu

ToggleSet(Nesne adı , “Font_Color” , “Hex formatında renk kodu örneğin: #0000ff”);

Örnek Kod:

```
ToggleSet("Toggle1" , "Font_Color" , "#0000ff");
```

Font size ayarlama komutu

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

Örnek Kod:

```
ToggleSet("Toggle1" , "Font_Size" , "12");
```

Height ayarlama komutu

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

Örnek Kod:

```
ToggleSet("Toggle1" , "Height" , "25");
```

Width ayarlama komutu

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

Örnek Kod:

```
ToggleSet("Toggle1" , "Width" , "100");
```

Top ayarlama komutu

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

AİRHMI LCD EKLAN EDITOR KILAVUZU

Örnek Kod:

```
ToggleSet ("Toggle1" , "Top" , "255");
```

Left ayarlama komutu

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

Örnek Kod:

```
ToggleSet ("Toggle1" , "Left" , "10");
```

Color ayarlama komutu

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

Örnek Kod:

```
ToggleSet ("Toggle1" , "Color" , "#FFA07A");
```

State ayarlama komutu

ToggleSet (Nesne adı , “State” , “1 , 0 veya True , False”);

Örnek Kod:

```
ToggleSet ("Toggle1" , "State" , "True");
```

ToggleGet(“ Nesne_ismi ” , “ Getirilecek_özellik ” , “ Değerin_atanacağı_değişken ”);

Açıklama

Toggle nesnesine ait parametrelere erişen ve getiren fonksiyon.

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

| Parametre | Açıklama |
|-----------|---------------|
| name | Nesnenin ismi |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| | |
|-------|---|
| type | Nesnenin getirilecek parametresinin ismi |
| Value | Getirilecek parametrenin atanacağı deęişken |

State Durumunu Getirme Komutu

ToggleGet (Nesne adı , "State" , "atanacak_deęişken");

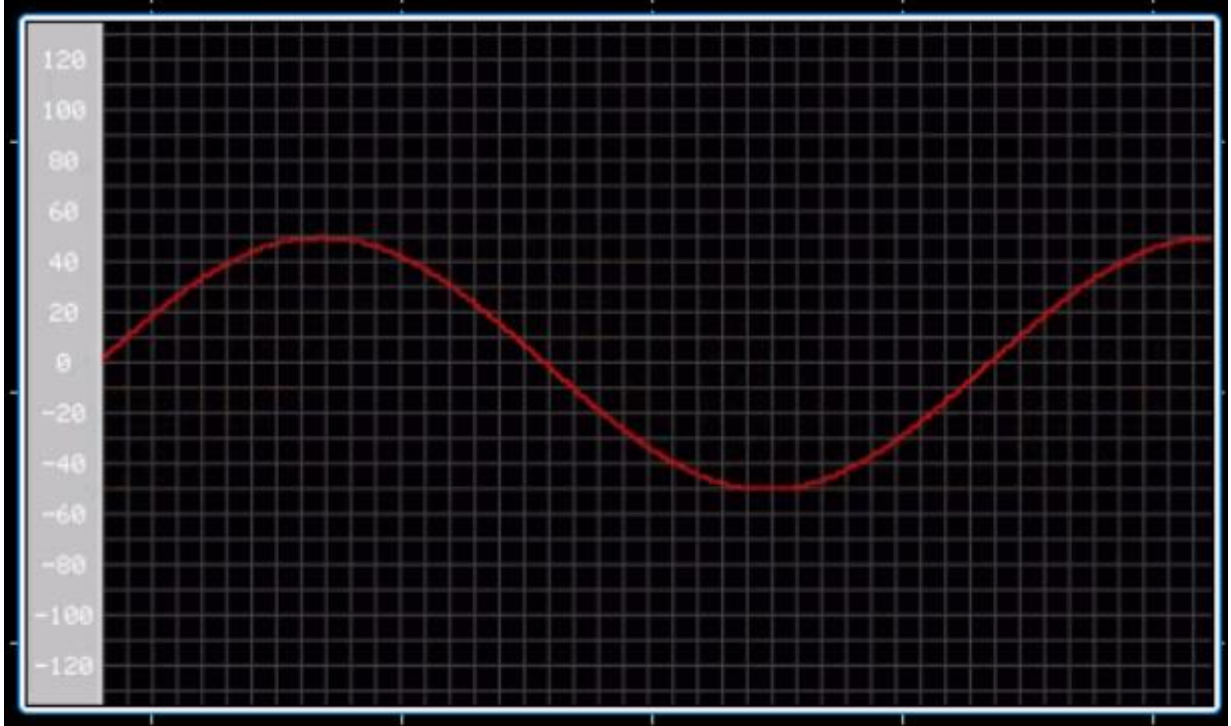
Örnek kod:

```
int get_state;  
ToggleGet ("Toggle1" , "State" , get_state);
```

AİRHMI LCD EKTRAN EDITOR KILAVUZU

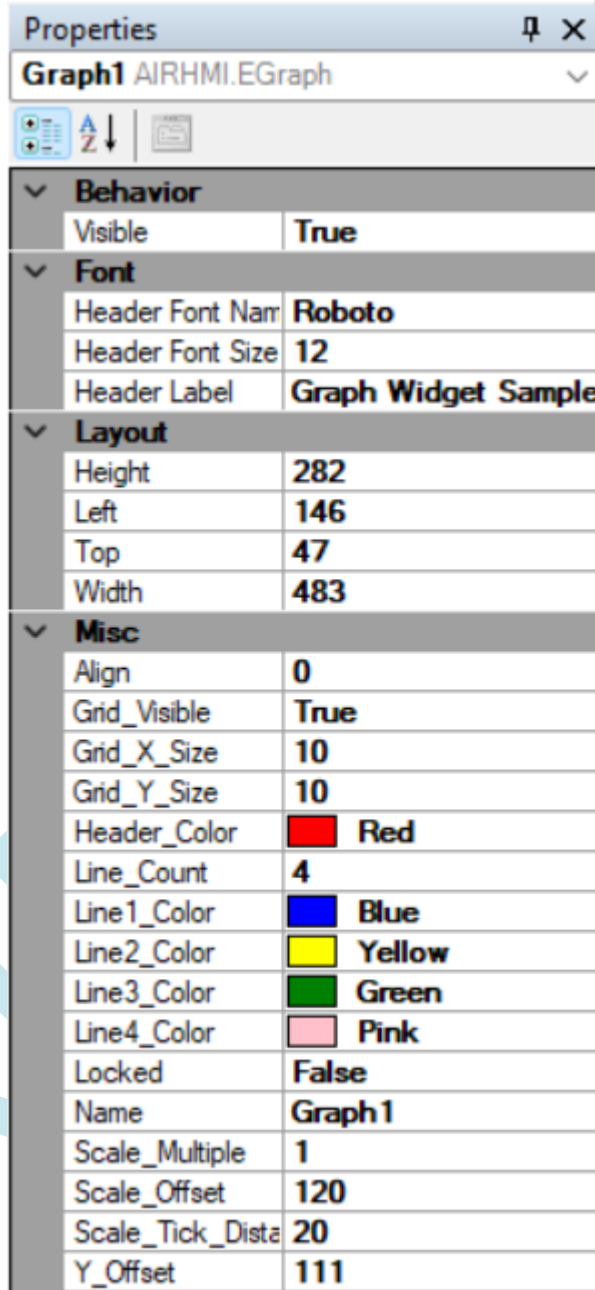
6.12 Graph

Ekranında grafik çizmek için kullanılır. Bu nesne sayesinde verilerinizi anlamlandırıp görselleştirebilirsiniz.



AIRHMI LCD EKРАН EDITOR KILAVUZU

Graph Properties Penceresi



| Özellik | Seçenek | Açıklama |
|-------------------------|---------|---|
| Visible | True | Ekran ilk oluştuğu zaman görünür. |
| | False | Ekran ilk oluştuğu zaman görünmez. |
| Header Font Name | | Graph nesnesi için farklı font seçenekleri tanımlama yapılır. |
| Header Font Size | | Nesnenin yazısının fontunun büyüklüğüdür. |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| | | |
|----------------------------|-------|---|
| Header Label | | Graph nesnesinin başlığında yazılı olan metin. |
| Name | | Nesnenin tasarım için kullanılan adıdır. Kod kısmındaki nesne adı bölümünde bu isim kullanılır. |
| Height | | Nesnenin yüksekliğidir. |
| Left | | Ekran üzerindeki pozisyonu belirtir. X koordinatı |
| Top | | Ekran üzerindeki pozisyonu belirtir. Y koordinatı |
| Width | | Nesnenin genişliğidir. |
| Align | | Graph nesnesini hizalamak için kullanılan parametre. |
| Grid_Visible | True | Gridler gözüktür. |
| | False | Gridler gözükmez |
| Grid_X_Size | | Grid içerisindeki hücrelerin X eksenindeki piksel boyutu |
| Grid_Y_Size | | Grid içerisindeki hücrelerin Y eksenindeki piksel boyutu |
| Header_Color | | Başlığın rengine ait parametre. |
| Line_Count | | Graph nesnesi üzerine kaç tane line olduğunu ayarladığınız parametre. |
| Line1_Color | | İlk line'a ait renk parametresi. |
| Line2_Color | | İkinci line'a ait renk parametresi. |
| Line3_Color | | Üçüncü line'a ait renk parametresi. |
| Line4_Color | | Dördüncü line'a ait renk parametresi. |
| Locked | True | Ekran a yerleştirilen nesnenin konumu değiştirmeye izin vermez. |
| | False | Grafığı istediğiniz konuma taşıyabilirsiniz. |
| Scale_Multiple | | Y eksenindeki hücre adım miktarını ayarlamak için kullanılır. |
| Scale_Offset | | Grafığın Y eksenindeki merkezinin offset değerini ayarlar. |
| Scale_Tick_Distance | | Y eksenindeki değerlerin hangi sıklıkla yazılacağını belirler. |
| Y_Offset | | Tüm datanın Y eksenindeki offset değeridir. |

Fonksiyonlar

GRAPH_AddValue();

Açıklama

Grafik nesnesine veri eklemek için kullanılan fonksiyon.

void **GRAPH_AddValue** (unsigned char *name , int channel, int value)

AIRHMI LCD EKLAN EDITOR KILAVUZU

| Parametre | Açıklama |
|-----------|---|
| name | Nesnenin ismi |
| channel | Nesnenin kaçınıcı line parametresini güncellemek istediđiniz. |
| value | Eklenecek yeni deđer. |

GRAPH_AddValue(Nesne adı , Kaçınıcı kanal olduđu , deđer);

Örnek Kod:

```
int data=10;
```

```
GRAPH_AddValue ("Graph1" , 1 , data); // grafiđe sürekli 10 deđerini ekler.
```

Örnek Kod:

```
int data;
```

```
VarGet("EVariable1",&data);
```

```
GRAPH_AddValue ("Graph1" , 1 , data); // grafiđe deđiřkenden gelen veriyi ekler.
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

GRAPH_Clear()

void **GRAPH_Clear** (unsigned char *name)

Açıklama

Grafik nesnesini temizlemek için kullanılan fonksiyon. Grafik üzerindeki çizilen çizgilerin hepsini temizler.

| Parametre | Açıklama |
|-----------|---------------|
| name | Nesnenin ismi |

GRAPH_Clear (Nesne adı);

GRAPH_Clear ("*Graph1*");

6.13 Variable

| Dğer | |
|-----------|------------|
| Data | |
| Modifiers | Private |
| Name | EVariable1 |
| Type | String |

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.

VariableSave()

Açıklama

AİRHMI LCD EKРАН EDITOR KILAVUZU

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

Fonksiyon

```
void VariableSave(unsigned char *name )
```

| Parametre | Açıklama |
|-----------|-----------------|
| name | değişkenin ismi |
| | |

Örnek kod

```
#include "stdio.h"
#include "stk.h"
VariableSave("EVariable1"); //
```

VarGet ()

Açıklama

Veri okuma komutudur. Private ve Public variable i okur.

Variable string ise deęişkenin direk adı yazılır.

Örneęin: char dataStr[20]; VarGet("var1",dataStr);

AİRHMI LCD EKLAN EDITOR KILAVUZU

Variable integer yada double olma durumunda ise deęişkenin adı ile birlikte önüne & işareti eklenir.

Örneęin: `int dataInt; VarGet("var2",&dataInt);`

Fonksiyon

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

| Parametre | Açıklama |
|-----------|-------------------------------------|
| name | deęişkenin ismi |
| value | Variable tipine göre pointer deęeri |

Örnek kod

```
#include "stdio.h"

#include "stk.h"
char dataStr[200];
int dataInt;
double dataDouble;
VarGet("EVariable1",dataStr);
VarGet("EVariable2",&dataInt);
VarGet("EVariable3",&dataDouble);
```

*Uart tarafından variablein içerięini okumak istersek, VarGet fonksiyonuna value kısmına NULL veririz bu durumda deęişkenin içerięini seri porttan verir.

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

AİRHMI LCD EKTRAN EDITOR KILAVUZU

VarSet ()

Açıklama

Veri yazma komutudur. Private ve Public variable i yazabilir.

Variable string ise değişkenin direk adı yazılır.

Örneğin: char dataStr[20]; VarSet("var1",dataStr);

Variable integer yada double olma durumunda ise değişkenin adı ile birlikte önüne & işareti eklenir.

Örneğin: int dataInt; VarSet("var2",&dataInt);

Fonksiyon

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

| Parametre | Açıklama |
|-----------|-------------------------------------|
| name | 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.
```

AIRHMI LCD EKTRAN EDITOR KILAVUZU

VarSeti()

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);
```

AİRHMI LCD EKLAN EDITOR KILAVUZU

VarSets()

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);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

VarSetf()

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);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

StructGet ()

Açıklama

C dilinde struct (yapı), birden fazla veri türünü tek bir çatı altında toplamak için kullanılır. Bir yapının içerisinde farklı türde değişkenler bulunabilir ve bu değişkenlere yapı elemanları denir. struct, nesneye yönelik programlamada sınıflara benzer, ancak fonksiyon içermez.

Bir `struct` tanımlaması şu şekilde yapılır:

```
c Kodu kopyala  
  
#include <stdio.h>  
  
// Yapı tanımı  
struct Kisi {  
    char isim[50];  
    int yas;  
    float boy;  
};  
  
int main() {  
    // Yapıdan bir değişken oluşturma  
    struct Kisi kisi1;  
  
    // Değişkenlere değer atama  
    strcpy(kisi1.isim, "Ahmet");  
    kisi1.yas = 25;  
    kisi1.boy = 1.75;  
  
    // Yapı elemanlarına erişim ve ekrana yazdırma  
    printf("İsim: %s\n", kisi1.isim);  
    printf("Yas: %d\n", kisi1.yas);  
    printf("Boy: %.2f\n", kisi1.boy);  
  
    return 0;  
}
```


AİRHMI LCD EKLAN EDITOR KILAVUZU

Fonksiyon

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

| Parametre | Açıklama |
|-----------|-----------------------------------|
| name | değişkenin ismi |
| value | struct tipine göre pointer değeri |

Örnek kod

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

typedef struct
{
    int data1;
    int data2;
    char data3[100];
} data_t;

data_t data;
StructGet("data" , &data);
```

AİRHMI LCD EKLAN EDITOR KILAVUZU

StructSet ()

Açıklama

Structe veriyi okumak için kullanılır.

Fonksiyon

```
void VarSet(unsigned char *name , void *value , int size)
```

| Parametre | Açıklama |
|-----------|-------------------------------------|
| name | değişkenin ismi |
| value | Variable tipine göre pointer değeri |
| Size | Structure nin boyutudur. |

Örnek kod

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

typedef struct
{
    int data1;
    int data2;
    char data3[100];
} data_t;

data_t data;

data.data1 = 1;
data.data2 = 2;

printf(data.data3, "%s", "1234");

StructSet("data" , &data , sizeof(data_t));
```

6.14 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);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.15 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 *uartsizе)
```

| Parametre | Açıklama |
|-----------|--|
| value | UART'tan gelecek verinin depolanacağı string |
| uartsizе | UART'tan gelen verinin boyutu |

Örnek kod

```
#include "stdio.h"  
#include "stk.h"  
  
char uartData[3000]; // Uarttan gelecek verinin depolanacağı  
string  
int uartsizе; // Uarttan gelen verinin boyutu  
  
uartDataGet(uartData , &uartsizе); // Uarttan gelen verinin okunması
```

**Uart üzerinden ekrana veri gönderirken VarSet , VarGet gibi fonksiyonlar ile ekran ile haberleşme yapabilirsiniz.*

Konu ile ilgili eğitim videosunu izleyebilirsiniz.



6.16 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");
```

AİRHMI LCD EKLAN EDITOR KILAVUZU

6.17 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
```

AİRHMI LCD EKLAN EDITOR KILAVUZU

6.18 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
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.19 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
```


AİRHMI LCD EKLAN EDITOR KILAVUZU

6.20 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
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.21 AudioPlay()

Açıklama

Kullanıcı, çalmak isteđi 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 );
```

6.22 AudioStop()

Açıklama

O anda çalınan ses işleminin sonlandırmak için kullanılır.

Fonksiyon

```
void AudioStop ();
```

| Parametre | Açıklama |
|-----------|----------|
| | |
| | |

Örnek kod

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

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

```
AudioStop();
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.23 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 gözüktür, “False” ayarlandığı zaman ise gözükmaz.

Örnek Kod:

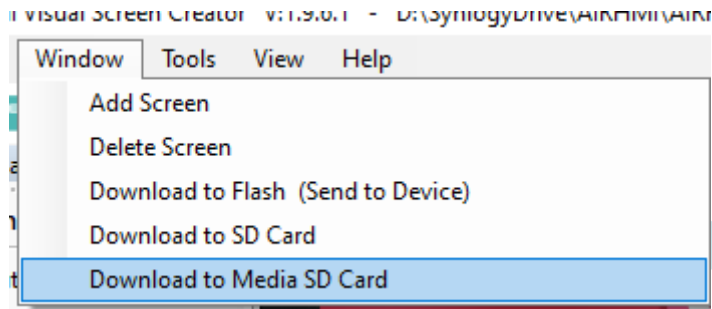
```
int value;  
AudioStatusGet(&value);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.24 VideoPlay()

Açıklama

Video dosyası oynatmak için kullanılır. Video yu tam ekran olarak oynatır. Video dosyasını sd kart içerisine editör ile aktarmanız gerekir. Bunun için editör içerisinden



Seçeneğini seçerek sd kartınızı göstermeniz gerekmektedir. Sd kartın formatı FAT32 olmalıdır.

Fonksiyon

```
void VideoPlay(unsigned char *name , int volume);
```

| Parametre | Açıklama |
|-----------|------------------------|
| Name | Video dosyası adı. |
| volume | Video ses seviyesidir. |

Örnek kod

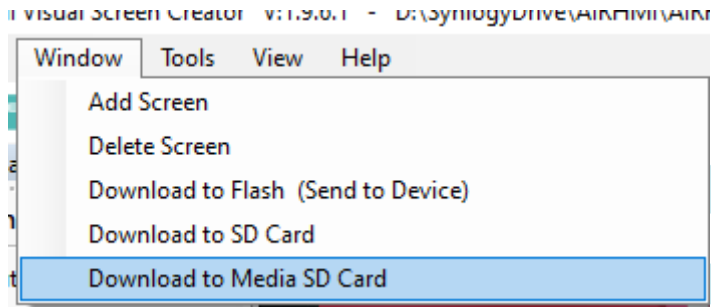
```
VideoPlay("EVideo1",100);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.25 Video_Play_XY()

Açıklama

Video dosyası oynatmak için kullanılır. Video yu ekranın herhangi bir konumunda oynatmak için kullanılır. Video dosyasını sd kart içerisine editör ile aktarmanız gerekir. Bunun için editör içerisinden



Seçeneğini seçerek sd kartınızı göstermeniz gerekmektedir. Sd kartın formatı FAT32 olmalıdır.

Fonksiyon

void Video_Play_XY(char *name , int volume , int x , int y)

| Parametre | Açıklama |
|-----------|------------------------|
| Name | Video dosyası adı. |
| volume | Video ses seviysidir. |
| x | Ekran x koordinatıdır. |
| y | Ekran y koordinatıdır |

AIRHMI LCD EKРАН EDITOR KILAVUZU

Örnek kod

```
Video_Play_XY("EVideo1",100 , 10 , 20); // ekranın 10,20 koordinatında videoyu  
oynatır.
```

AIRHMI

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.26 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
verişi sizeof(x_file) boyutu kadar yazıldı.
```


AİRHMI LCD EKРАН EDITOR KILAVUZU

6.27 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.
```

AIRHMI LCD EKРАН EDITOR KILAVUZU

6.28 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.
```

6.29 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 );
```

6.30 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 );
```

AIRHMI LCD EKРАН EDITOR KILAVUZU

6.31 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.
```

6.32 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.

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.33 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);
```

AIRHMI LCD EKРАН EDITOR KILAVUZU

6.34 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);
```


6.35 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 çağırısı, 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  
  
}
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.36 KeypadAlpha()

Açıklama

Yazılım sırasında kullanıcıdan veri almak için kullanılır. Ekranda tam sayfa bir klavye 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 Dünya!",data);  
  
printf("You Wrote %s.\n",data);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.37 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 |

AİRHMI LCD EKLAN EDITOR KILAVUZU

| | |
|------------|----------------|
| 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 );
```

AİRHMI LCD EKTRAN EDITOR KILAVUZU

6.38 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 |

AİRHMI LCD EKLAN EDITOR KILAVUZU

| | |
|------------|----------------------------|
| 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);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

6.39 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

AİRHMI LCD EKРАН EDITOR KILAVUZU

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);
```


AİRHMI LCD EKTRAN EDITOR KILAVUZU

6.40 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 |

AİRHMI LCD EKРАН EDITOR KILAVUZU

| | |
|------------|--------------------------------|
| 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);
```

7. Ethernet

7.1 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();
```

AİRHMI LCD EKLAN EDITOR KILAVUZU

| 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");
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

7.2 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);
```

AİRHMI LCD EKTRAN EDITOR KILAVUZU

7.3MAC 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_adres);
```

| Parametre | Açıklama |
|-----------|--|
| mac_adres | Airhmi Ethernet arabiriminin mac adresi. |

Örnek kod

```
char data[100];
```

```
EthernetGet_MAC(data);
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

7.4 Ethernet TCP Soket Bağlantısı

Açıklama

Airhmi ekran statik yada DHCP olarak kullanılabilir.

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);
```

AIRHMI LCD EKLAN EDITOR KILAVUZU

7.5 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);
```


AİRHMI LCD EKРАН EDITOR KILAVUZU

7.6 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);
```

7.7 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);
```

7.8 Ethernet TCP Soket Kapat

Açıklama

TCP soket kapatma fonksiyonudur.

Fonksiyon

```
Void SocketTCP_Close();
```

| Parametre | Açıklama |
|-----------|----------|
| | |
| | |
| | |

Örnek kod

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

```
SocketTCP_Close();
```

AİRHMI LCD EKРАН EDITOR KILAVUZU

7.9 Ethernet TCP Soket Durumu Sorgulama

Açıklama

Airhmi TCP soket durumu sorgulama için kullanılır.

Fonksiyon

```
int 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." );
```

7.10 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¶m2=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.

AIRHMI LCD EKLAN EDITOR KILAVUZU

POST isteđi idempotent deđildir, yani aynı istek tekrarlandıđında sonu deđiřebilir.

Örnek bir POST isteđi řu řekildedir:

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

Body:

param1=value1¶m2=value2

Her iki HTTP metodu da web uygulamalarında farklı amalara 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 bađlıdır.

8. Kütüphaneler

8.1stdio.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 *, ...);
```

AIRHMI LCD EKРАН EDITOR KILAVUZU

8.2stdlib.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 *);
```


AIRHMI LCD EKLAN EDITOR KILAVUZU

`int rand();`

`void srand(int);`

`int abs(int);`

`int labs(int);`

AIRHMI

8.3math.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);

AİRHMI LCD EKTRAN EDITOR KILAVUZU

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);

AİRHMI LCD EKLAN EDITOR KILAVUZU

`float sin(float);`

`float sinh(float);`

`float sqrt(float);`

`float tan(float);`

`float tanh(float);`

AİRHMI

8.4string.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.

AİRHMI LCD EKРАН EDITOR KILAVUZU

Örnek Program:

```
#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;
}
```

AİRHMI LCD EKLAN EDITOR KILAVUZU

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);
```