AirHMI

LCD

EKRAN

EDİTÖR

KILAVUZU

AİRHMI LCD EKRAN EDITOR KILAVUZU

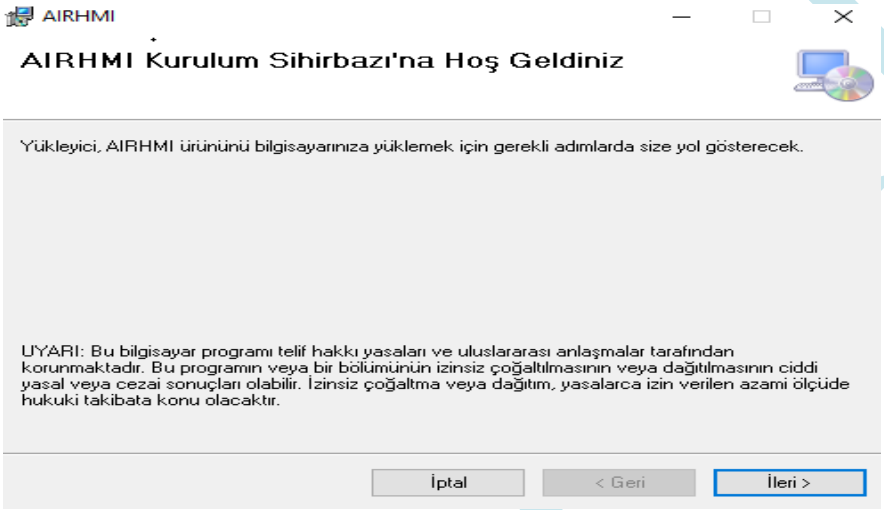
AirHMI Visual Screen Creator, AirHMI LCD ekranları için İnsan Makine Arayüzü GUI'lerini tasarım açışından en üst seviyede memnuniyet ve en verimli sürede oluşturabilmek amacıyla tasarlanmıştır. Editör kullanımında Tasarım ve Programlama dünyasına ait işlevselliklerimiz bulunmaktadır: Görsellik açısından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

İÇİNDEKİLER

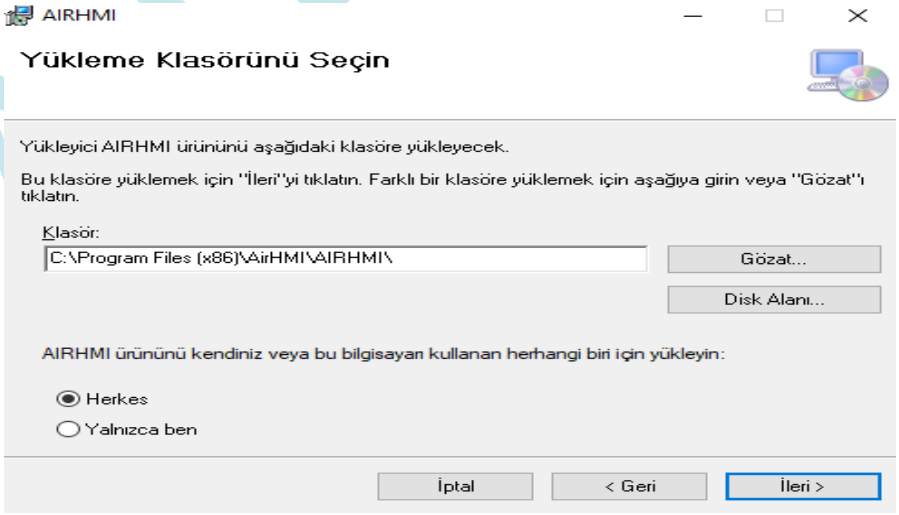
1. AirHMI Visual Screen Creator KURULUMU

İndirme Linki: http://www.airhmi.com/Setup/AIRHMISETUP.msi

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



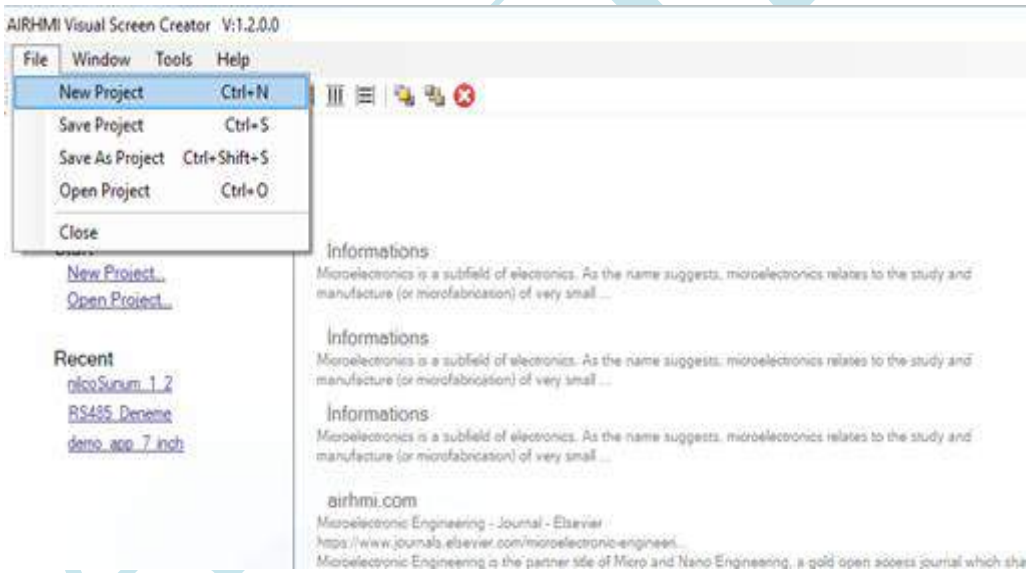
Yükleme klasörünü ve diğer seçenekleri istediğiniz şekilde seçip ileri tuşuna basarak yükleme başlatılır.

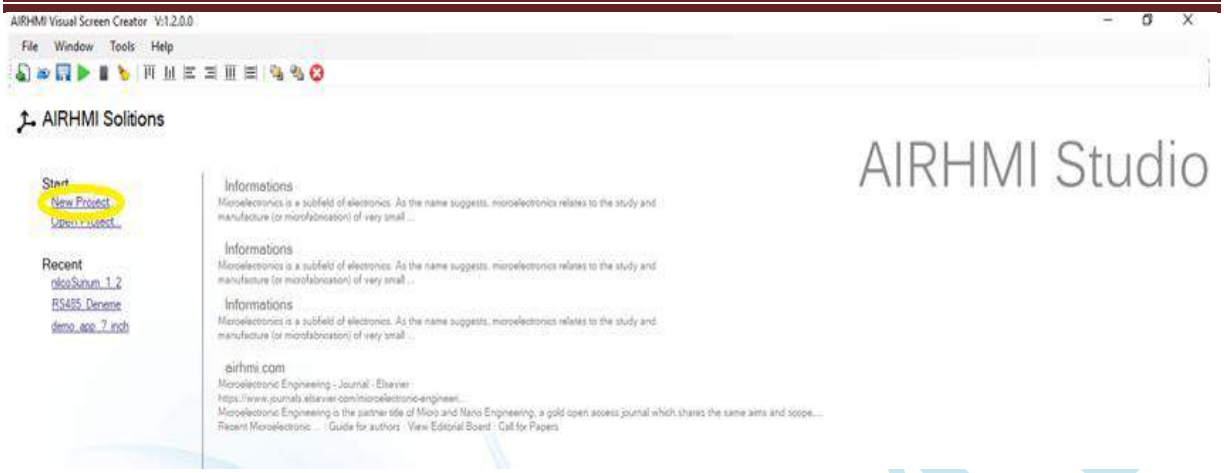


2. PROJE OLUŞTURMA

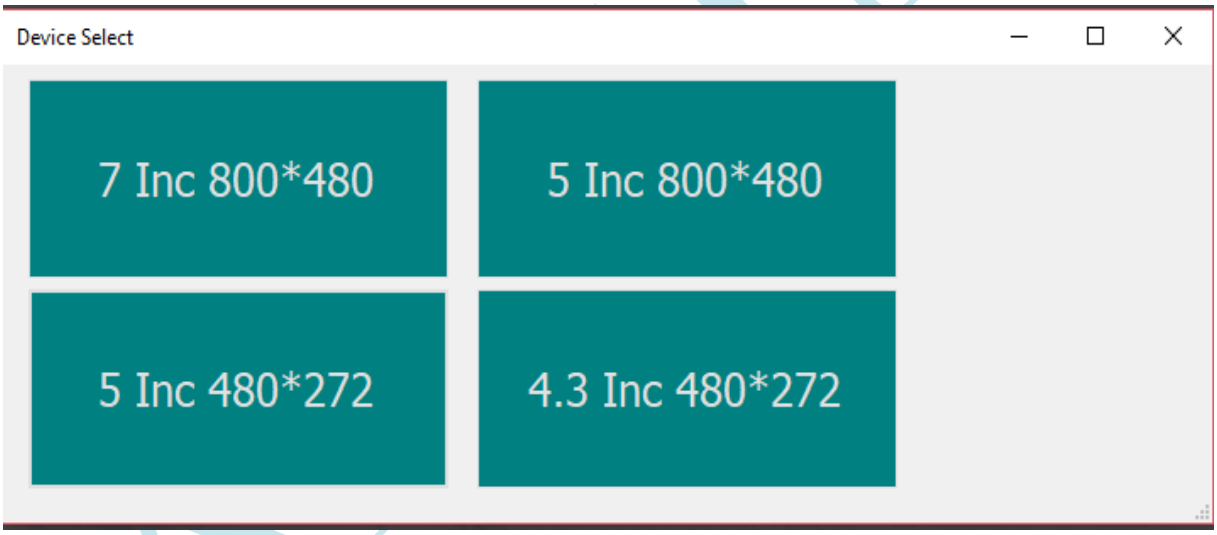
AirHMI ile arayüz oluşturmak için öncelikle AirHMI Editör programını indirip bilgisayarınıza kurmanız gerekmektedir. AirHMI Editör programındaki sürükle-bırak özelliği arayüz geliştirmeyi kolaylaştırmaktadır. AirHMI Editörü ile projelerinize, Buton, Resim, Yazı, İlerleme çubuğu, Gauge, Key, Analog ve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

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





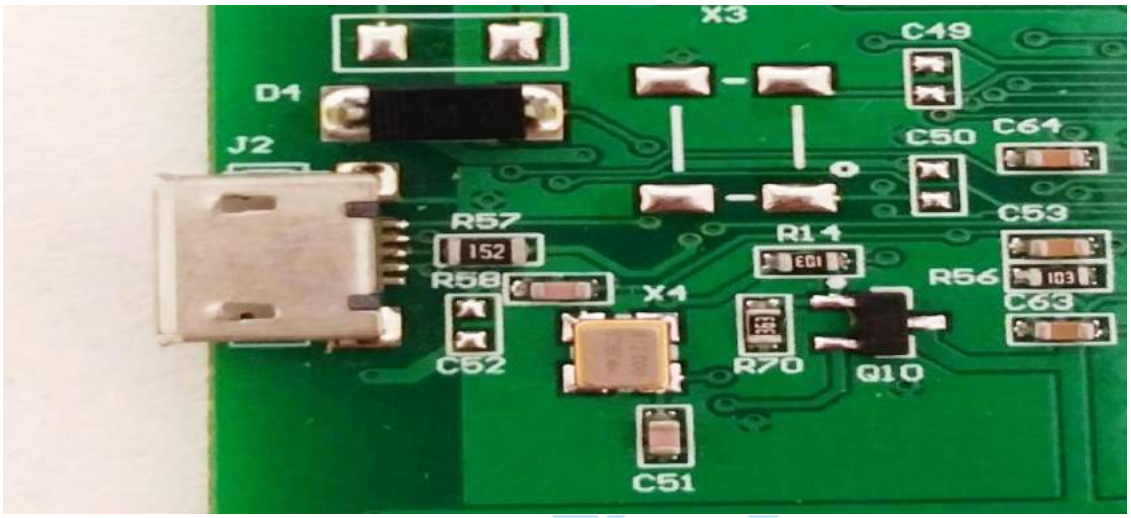
Kayıt işleminden sonra karşınıza aşağıdaki resimde görüldüğü gibi bir sayfa çıkacaktır. Karşınıza çıkan sayfada Ekrana ait boyut ve çözünürlük ile ilgili ayarlar yapılmalıdır.



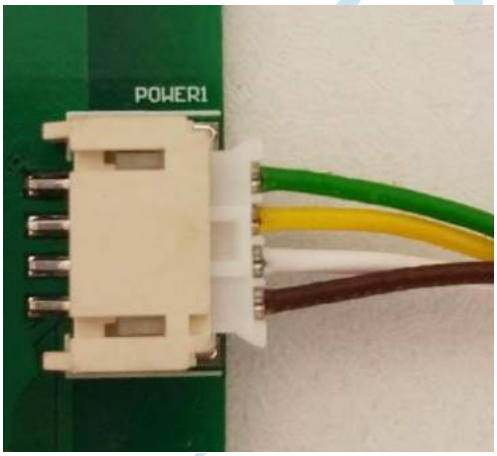
3. CİHAZ BAĞLANTISI

AirHMI ekrana enerji vermek iki yöntem kullanılabilir; POWER konektörü veya USB konektörü. Cihaz için standart baudrate 115200 8N1'dir.

1) USB konektörü;



2) Eğer POWER konnektörü kullanılacaksa konektöre ait pinler şu şekildedir;



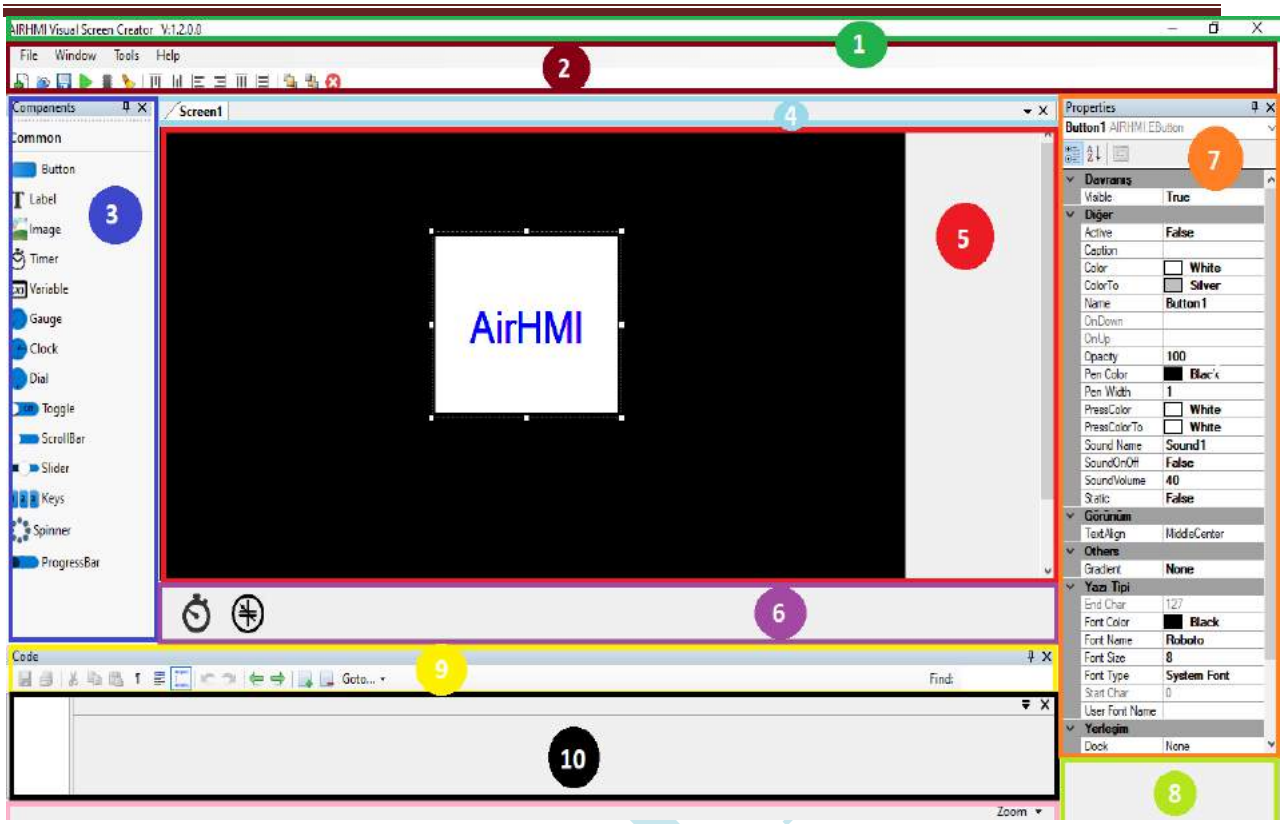
YEŞİL = GND

SARI= RX

BEYAZ=TX

KAHVERENGİ=BESLEME

4. AirHMI EDİTOR ANA ARAYÜZÜ



4.1BAŞLIK ÇUBUĞU

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

4.2ANA MENÜ ve ARAÇ ÇUBUKLARI

Dosya (File) Menüsü

Kullanıcılar için Yeni Proje Açın, Projeyi Kaydet, Projeyi Farklı Kaydet, Var Olan Bir Projeyi Açın ve Çıkış gibi komutlar bulunmaktadır. Burada önemli olan nokta var olan bir proje açıkken yeni proje açmak istenildiğinde eski projenin bilgisayarda saklanması ya da

yapılan değişikliklerin kaybolmaması isteniyorsa ekrana gelen kaydet mesajına onay verilmelidir.

Pencere (Window)

Pencere alanı içerisinde ;

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

Araçlar (Tools)

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

Hizalama

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

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

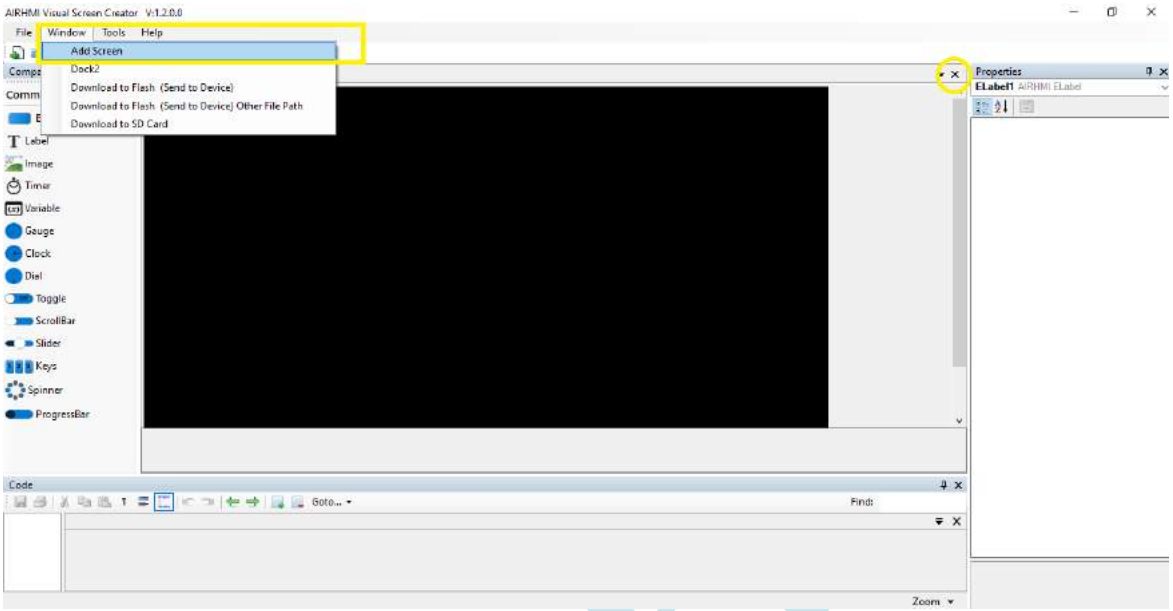
4.3 BİLEŞENLER BÖLMESİ

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

4.4 EKRAN / KOMUT SEKMESİ

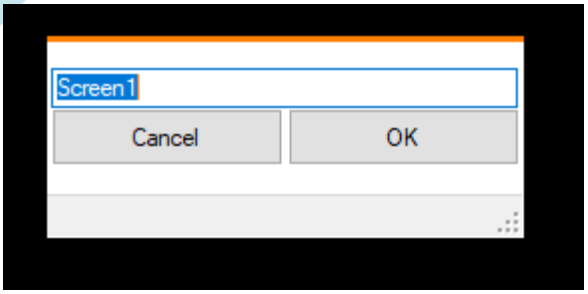
Tasarım projeleri genelde tek ekran olarak kullanılmayıp aynı anda farklı ekranlara ihtiyaç duymaktadır. Açılış Genel Gösterim Ekranı, Menü Ayar Ekranı, Detaylı Gösterim Ekranı vs… Bu nedenle AirHMI Editör içerisinde kullanıcı istekleri doğrultusunda birden

fazla özgün ve yaratıcı ekran tasarımı yapabilmektedir. Ekran / Komut Sekmesi ile hangi ekranda çalışma yapılacağını seçme işlemi gerçekleştirilmektedir.



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

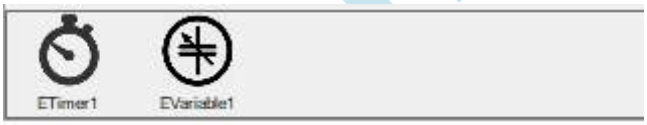
Ekranın ismini değiştirmek için ekranda boş bir alanda sağ tıklayarak Rename sekmesine tıklanır. Açılan sekmeden ekranın ismi değiştirilebilir.



4.5TASARIM ANA EKRAN ALANI

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

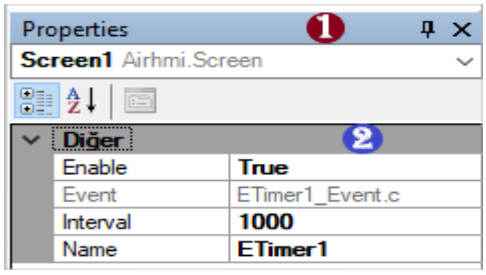
4.6 GÖRSELİ OLMAYAN BİLEŞENLERİN ALANI



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



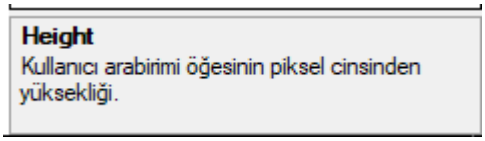
3.7.1 Projede Kullanılan Nesnelerin Gösterim Alanı

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

3.7.2 Nesnelerin Öznitelikleri Gösterim / Ayar Alanı

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

4.8 ÖZNİTELİKLERİN AÇIKLAMA ALANI



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

4.9 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.

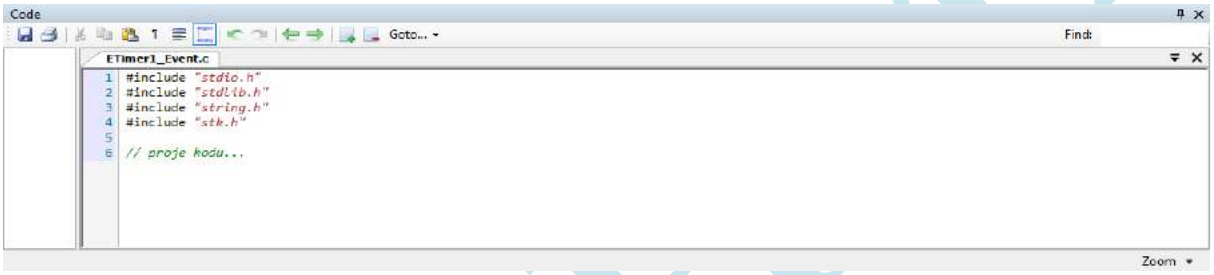
4.10 KULLANICI PROJE KOD ALANI

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

4.11 KOD ALANI ZOOM ALANI

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

5. KULLANICI KOD YAPISI



AirHMI Editör'ün çözüm odaklı, zaman ve efor konularında en verimli noktada tasarım oluşturmayı hedefleyen yapısının yanında en önemli avantajlarından biri de kolay ve anlaşılabilir kod yapısıdır. Kod yapısı C programlama dilinde hazırlanmıştır. Fakat kullanıcı odaklı olması ve kullanıcıya kullanımda kolaylık sağlayabilmesi için gerekli fonksiyonlar "stk.h" kütüphanesi altında hazırlanmıştır. Temel C kütüphanelerinin ekli olduğu bu düzende C programlama dilini kullanarak kodunuzu oluşturabilir ve gerekli fonksiyonları kodunuzun başına ekleyebilirsiniz. Hazır C fonksiyonlarına ek olarak nesnelerin kontrol/ayar fonksiyonları, LCD ekran uyku modu, zamanlayıcı kod düzeni gibi önemli birçok konuda hazır fonksiyonları açıklamaları ile birlikte bu kılavuzda bulabilirsiniz. Burada önemli olan nokta bu fonksiyonların aktif olarak çalışabilmesi için "stk.h" kütüphanesinin her kod yapısının başına eklenmesi gerektiğidir.

ETimer1_Event.c

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

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

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

    DrawScreenGet();

}
```

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

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

5.1 TIMER

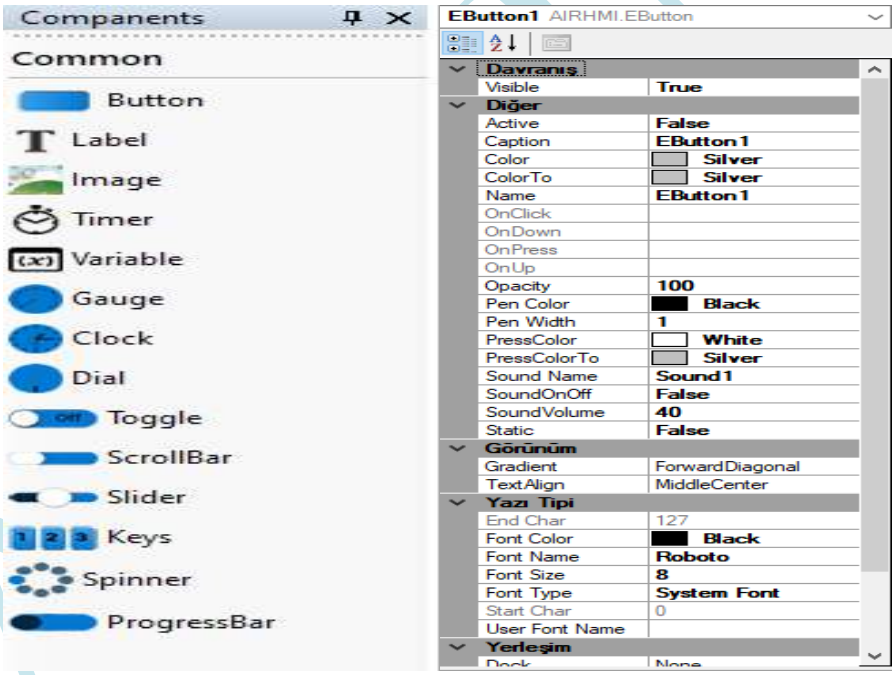
Diğer	
Enable	True
Event	ETimer1_Event.c
Interval	1000
Name	ETimer1

Kod yapısı içerisinde belki de en önemli nokta Timer kullanımıdır. Tasarlanan editör ekranının projede gerçek zamanlı çalışmasında oluşacak değişiklikler ve bu değişikliklerin hangi aralıklar ile olacağı Timer Özniteliklerinin içerisinde ayarlanmaktadır. Enable, Timer'ın aktif olup olmayacağını seçmektedir. Interval, milisaniye cinsinden hangi aralıklar ile kodun aktif olacağının seçildiği yerdir. Name, adında da anlaşılacağı gibi Timer'ın ismidir. Event

bölümü ise proje tasarımı için oluşturulacak kod kısmını açma bölümüdür. ETimer1_Event.c ise oluşturulan kodun kaydedildiği C dosyasının ismidir.

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

5.2 AirHMI EKRAN NESNELERİ



5.2.1 Button

Buton nesnesi basıldığı zaman herhangi bir işlem yaptırmayı sağlayan nesnedir. Örneğin kullanıcıdan alınan veriyi bir yere göndermek, alınan veriyle işlem yapmak veya

mesaj verdirmek amacıyla kullanılabilir. Butonun konumunu istediğiniz yere sürükleyebilir ve boyutunu kenarlarından çekerek ayarlayabilirsiniz.

5.2.2 Label

Ekranda bulunan herhangi bir nesne için tanıtıcı etiketlerin yazılmasını sağlayan, kullanıcıya bilgi vermek amacıyla kullanılan ekran nesnesidir.

5.2.3 Image

Ekleyeceğiniz resimlerin ekran boyutundan büyük olmamasına dikkat etmelisiniz. Eğer eklemiş olduğunuz resim ekran boyutundan büyük ise editör içi yeniden boyutlandırma olmadığından resminizin tamamı görüntülenemeyecektir.

Dikkat edilmesi gereken bir diğer önemli nokta da eklenecek resimlerin formatı PNG resim formatında olmalıdır.

5.2.4 ProgressBar

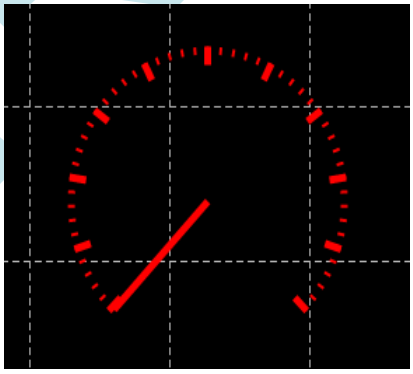
Progress Bar ifadesi Türkçede “ilerleme çubuğu” anlamına gelmektedir. Uzun bir işlemin yürütülme aşamalarının grafiksel olarak gösterilmesi gerektiği durumlarda kullanılır. Progress Bar kullanımına örnek olarak: yürütülmekte olan bir video ya da ses dosyasının

kalan zamanının Progress Bar üzerinde gösterilmesi, bir yakıt deposunun doluluk oranının Progress Bar kullanılarak grafiksel olarak gösterilmesi verilebilir.

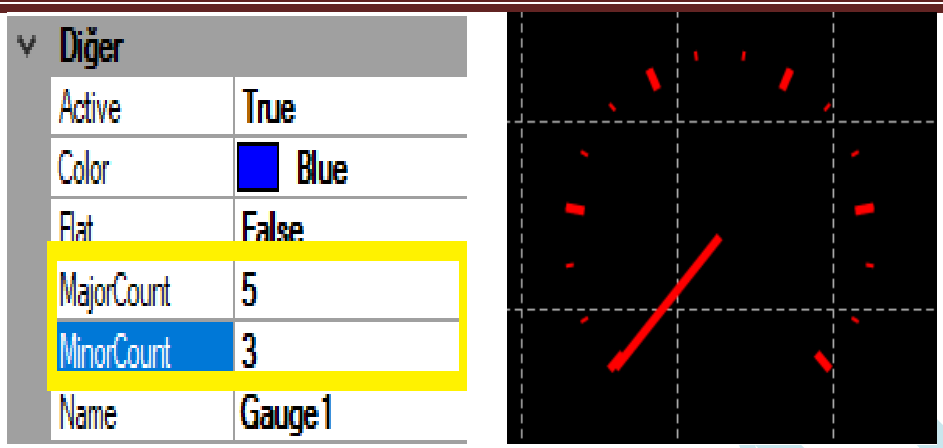
5.2.5 ScrollBar

Tasarım ekrana sığmadığı zaman veya tasarım ekranının kaydırılması istendiği durumlarda ekranın kaydırılmasını sağlayan nesnedir.

5.2.6 Gauge

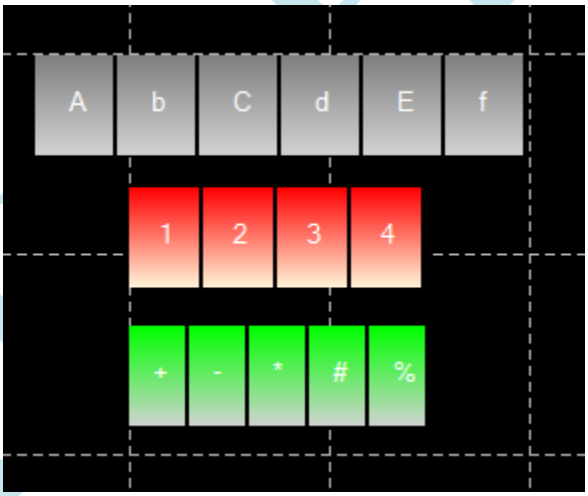


Gauge nesnesi için öznitelikler bölümünden skaladaki periyot aralığı düzenlenebilir.

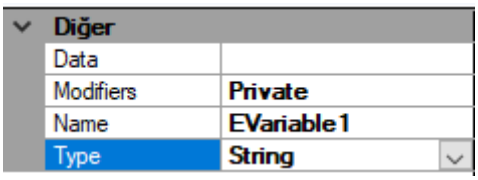


5.2.7 Keys

Ekran klavyesi olarak kullanabileceğiniz nesnedir. Öznitelik bölümünden Caption özelliği ile istediğiniz şekilde harf, rakam, karakter girerek klavye için tasarım yapabilirsiniz.



5.3 VARIABLE



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

Değişkenin:

1. Global veya Local
2. String veya Integer
3. Değerinin Set ya da Get edileceğini
4. İsmi
5. Yeni değeri veya eski değerinin alınacağı değişken

durumlarına göre istenilen fonksiyon kullanılmalıdır.

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

5.4 GPIO

AirHMI Ekran kartı üzerinde kullanıcıların istekleri doğrultusunda kullanabilmesi için 4 farklı GPIO pini bulunmaktadır. Bu pinler projelerde isteğe bağlı olarak INPUT veya OUTPUT olarak ayarlanabilmektedir. Kod yapısı;

void halGpioSet(unsigned char *pin, unsigned char *value)

halGpioSet(* pinismi , *okuma/yazma)

```
halGpioSet("GPIO_OUT_1", "Write");  // Gerçek zamanlı kullanılan örnek kod yapısı
halGpioSet("GPIO_OUT_2", "Read");  // Gerçek zamanlı kullanılan örnek kod yapısı
```

şeklinde olmaktadır.

5.5 UART

Tasarlanan projenin gerçek hayatta kullanımında proje özelinde geliştirilmiş bir işlemci kartı veya hazır geliştirme kitleri kullanılmaktadır. Bu kartların uart çıkışından veri göndererek UART’tan gelen verilere göre AirHMI Editör ekranında işlemler yapılabilmektedir. Kod düzeni içerisinde UART’tan gelen veriyi alma işlemi aşağıda gösterildiği gibi gerçekleştirilmektedir.

- UART’tan **veri almak** için uartDataGet() komutu kullanılır.

void uartDataGet(char *value , int *uartsize)

```
char uartData[30];           //  Uarttan gelecek verinin depolanacağı string
int uartsize;               //  Uarttan gelen verinin boyutu
uartDataGet(uartData , &uartsize); //Uarttan gelen verinin kod içerisinde alınması
```

- UART'tan **veri göndermek** için printf komutu kullanılır.

```
printf("Hello world");
```

5.6 FLASH'A DOSYA OKUMA YAZMA

AirHMI LCD Kartı donanımında projelerde dahili hafıza sorunu olmaması ve kullanıcının yükleyeceği dosyalar için yeterli alan olması amacı ile 32Mbyte Flash bulunmaktadır. Editör üzerinden oluşturulan projeler bu flasha yüklenmektedir. Ayrıca kod dizininde ani elektrik kesintilerinde veya güç kaybının meydana geldiği durumlarda kullanıcının önemli gördüğü ve her durumda kaydedilmesini istediği veriler için PICOC kod yapısı içerisinde flasha yükleme yapma ve flashtan okuma gibi temel işlemler tasarlanmıştır. Bu sayede kullanıcı istediği verileri ileride kullanmak için dahili flasha ekleyip güvenli bir şekilde saklayabilmektedir. Kod düzeni içerisinden flasha veri yazma, okuma ve flashta bulunan bir dosyanın boyutunu alma işlemleri aşağıda bulunan fonksiyonlar ile gerçekleşmektedir.

- **Flashtan okuma**

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

File_read(.txt dosyasının ismi , string dizisinin adı , okuma boyutu , 1);

- **Flasha yazma**

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

File_write(.txt dosyasının ismi , string dizisinin adı , yazılacak dizinin boyutu , 1);

Örnek;

```
Char deneme[200];

File_write("Message.txt" , deneme, sizeof(deneme), 1);
```

- **Dosya Boyutunu Öğrenme**

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

File_size(.txt dosyasının ismi, integer bir değişken);

5.7 RTC İLE TARİH SAAT GÖSTERİMİ

AirHMI LCD Kartı donanımında, tasarımlarda kullanılmak üzere harici RTC entegresi bulunmaktadır. Bu sayede saat tarih ayarlarını harici bir entegre yada veriye ihtiyaç duymadan yapabilir ve LCD ekran üzerinde bir Text ile tarih saat bilgisi gösterilebilir. Kullanıcıya kolaylık sağlaması amacı ile gerçekleştirilmiş bu sistemde saat verilerini almak ve ayarlamak için yapılması gerekenler oldukça basit ve anlaşılırdır. RTC ile tarih saat ayarları ve kullanımı aşağıda bulunan kodlar ile yapılmaktadır.

```
unsigned char day, month, year, hour, min; // Kod dizininde örnek Tarih-Saat
                                              değişkenleri
dateGet(&day , &month , &year);         // RTC den Tarih verilerini alma
timeGet(&hour , &min);                 // RTC den Saat verilerini alma
dateSet(&day , &month , &year);     // RTC de Tarih verilerini yenileme/ayarlama
timeSet(&hour , &min);            // RTC de Saat verilerini yenileme/ayarlama
```

5.8 SES DOSYASI OYNATMA

Endüstriyel alanda gerçekleştirilen projelerde belirli sınırlamaların, değerlerin aşımında projenin uyarı bildirmesi önemli bir durum teşkil etmektedir. AirHMI LCD Kartının tasarımının temelinde birçok ihtiyacı karşılamayı hedefleyen yapısı ile harici ses dosyası oynatma ve kendi içerisinde bulunan uyarı seslerini oynatma imkanı sunmaktadır. Ses çıkışı PWM sinyali şeklinde olmaktadır. Kullanıcı, çalmak isteği ses dosyasını AirHMI Editör üzerinden projeye ekledikten sonra aşağı da bulunan 1 numaralı kod ile çalma işlemini gerçekleştirebilmektedir. Kendi içerisinde bulunan ses dosyalarını çalmak için ise 2 numaralı kod yapısı kullanılmaktadır. Kod yapıları:

- ```
  int volume;              // Ses Düzeyi
  AudioPlay("SesDosyasınınİsmi" , &volume );
  ```

AudioPlay fonksiyonu içerisinde ilk değişkene çalmak istenilen ses dosyasının isminin, ikinci değişkene ise 0-255 arası istenilen ses düzeyinin verilmesi gerekmektedir.

- ```
  PlaySound(integerDeger);
  ```

Hazır uyarı sesleri proje içerisinde yüklü bulunmaktadır. PlaySound fonksiyonu ile daha önce hazırlanmış 5 farklı uyarı sesinden istenilen integerDeger bölümüne 1,2,..,5 gibi yazılarak seçilebilir.

5.9 LCD GÖSTERİM EKRANI

Tasarlanan projelerde kod yapısı içerisinde yapılan değişikliklerin ekranda görünebilmesi için olmazsa olmaz nokta LCD ekranın yenilenmesi işlemidir. Hazırlanan proje içerisinde kod yapısı üzerinden yenilenen nesnelerin yeni özelliklerinin ekran üzerinden de ayarlanabilmesi için ekran yenileme işlemi gerçekleştirilmelidir. Kullanıcı kod içerisinde sayfa yenilemesinin gerçekleştirilmesini istediği noktaya aşağı da bulunan kodu ekleyerek bu işlemi gerçekleştirebilir.

- `DrawScreenGet();` `// Sayfa yenileme komutu`

Kullanıcı hazırlanmak istediği projede birden çok ekran tasarımı gerçekleştirebilmektedir. Böyle tasarımlarda sayfalar arası geçiş önemli bir durum teşkil etmektedir. Kod dizini içerisinden kullanıcı isteği doğrunda aşağıda bulunan kod içerisine geçiş yapmak istediği ekranın ismini ekleyerek ekran geçişini yapabilmektedir.

void ChangeScreenSet(unsigned char *value)

- `ChangeScreenSet ("GeçişYapılacakEkranınİsmi");`

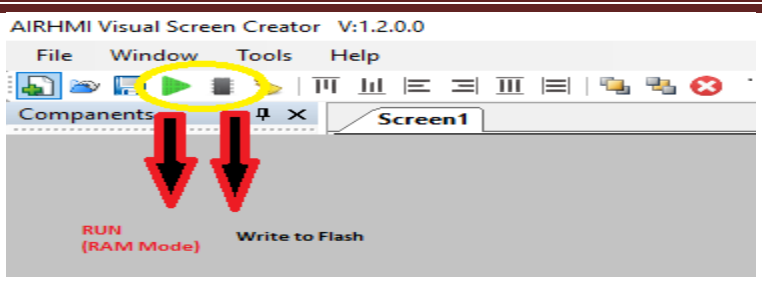
5.10 PASİF UYKU MODU

Endüstriyel projelerde temel amaç, zaman ve iş gücü açısından olabilecek en verimli değerlerde proje tasarımını gerçekleştirmektir. Bu amaç doğrultusunda projelerde kullanılan entegreler ve kodlama sisteminde dikkatli davranmak gerekmektedir. AirHMI LCD Ekran tasarımında bu hususlar dikkate alınarak tasarlanmış olmakla birlikte kullanıcıya da enerjiden tasarruf sağlamak amacı ile uyku modu desteği sunmaktadır. Proje aktif olarak çalışırken LCD ekranda görüntünün gerekli olmadığı durumlarda LCDsleep fonksiyonu kullanılarak LCD ekran kendini uyku moduna almaktadır. Bu sayede proje veri kaybı yaşamadan aktif olarak çalışırken LCD ekranın fazladan güç çekmesi de engellenmektedir. Kullanıcı aynı şekilde LCD ekranda görüntünün yeniden gelmesini istediği durumlarda ise LCDwakeup fonksiyonunu kullanarak yeniden görüntülü ekrana sahip olabilmektedir.

- `LCDsleep();` `// LCD uyku modu aktif`
- `LCDwakeup();` `// LCD uyku modu kapalı`

6. TASARIM YÜKLEME

AirHMI Editör'de oluşturulan tasarımı cihaza yüklemek için iki seçenek mevcuttur: Run (RAM Mode) ve Write to Flash.

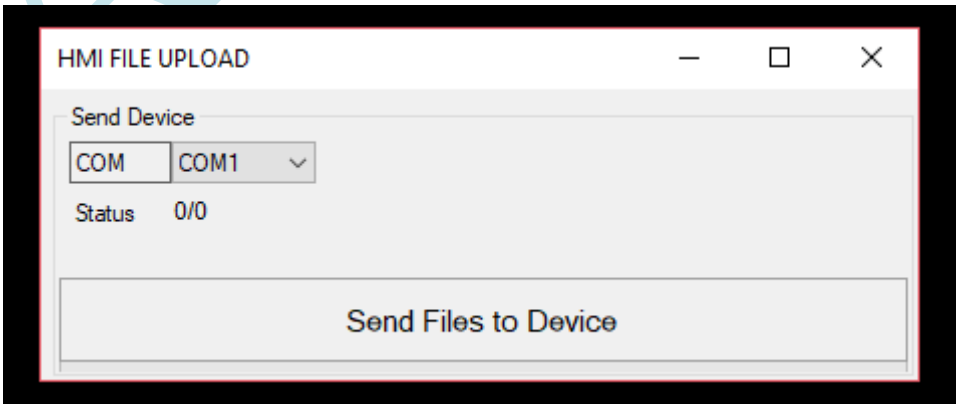


6.1 Run(RAM Mode)

Tasarım geliştirme sırasında editörde hazırlanmış olan tasarımı cihaza hızlı bir şekilde yükleyip ekranda görmek amacıyla kullanılır. Ancak cihazın gücünü kestiğiniz zaman yüklemiş olduğunuz tasarım da silinir. Flash modda yüklemeye göre avantajı tasarımınızı hızlı bir şekilde cihaza yüklemesidir.

6.2 Write to Flash

AirHMI LCD Kartı donanımında projelerde dahili hafıza sorunu olmaması ve kullanıcının yükleyeceği dosyalar için yeterli alan olması amacı ile 32Mbyte Flash bulunmaktadır. AirHMI Editör üzerinden oluşturulan projeler bu flasha yüklenmektedir. Ayrıca kod dizininde ani elektrik kesintilerinde veya güç kaybının meydana geldiği durumlarda kullanıcının önemli gördüğü ve her durumda kaydedilmesini istediği veriler için flasha yükleme özelliği eklenmiştir. Bu sayede kullanıcı istediği verileri ileride kullanmak için dahili flasha ekleyip güvenli bir şekilde saklayabilmektedir.



Tasarımı cihaza yükleme aşamasına geçtiğinizde her iki yükleme yöntemi için de karşınıza şekildeki gibi bir ekran açılacaktır. Açılan ekrandan cihazınızın bağlı olduğu COM portunu seçerek yüklemeyi başlatabilirsiniz. Yüklemeye ait ilerlemeyi de bu ekrandan takip edebilirsiniz.

7. FONKSİYONLAR

AirHMI Screen Editor içerisinde kod yapısında kullanılan temel fonksiyonların açıklamaları ve örnek kod yapısı bu dizin altında bulunmaktadır. Nesneler ile ilgili fonksiyonlarda değiştirilebilecek olan parametreler AirHMI Screen Editor üzerinden nesne özelliklerine bakarak kolayca anlaşılabilmektedir.

7.1delay( )

Açıklama

Kullanıldığı satırda belirlenen süre kadar beklemeyi sağlayan komuttur.

Fonksiyon

void delay (int ms)

Parametre	Açıklama
ms	Zaman periyodunu belirtir

Örnek kod

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

delay(1000);
```

7.2KeyGet ( )

Açıklama

Key nesnesine dokunulduğunda hangi harfe/sayıya basıldığını kullanıcıya iletmek için kullanılan komuttur. Klavye vb. işlemlerde basılan değeri kod dizini içerisinde başka yerde kullanmak için sabit bir değişkenin içerisinde saklamaktadır.

Fonksiyon

void KeyGet(unsigned char *value)

Parametre	**Açıklama**
value	Değerin saklandığı değişken

Örnek kod

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

unsigned char value;

KeyGet(&value);
```

7.3 ClockSet ( )

Açıklama

Clock nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void ClockSet(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
        ClockSet ("EImage1" , "Visible" , "1");

Left ayarlama komutu
        ClockSet ("Clock1" , "Left" , "10");


Top ayarlama komutu
        ClockSet ("Clock1" , "Top" , "255");

Radius ayarlama komutu
        ClockSet ("Clock1" , "Radius" , "35");

Color ayarlama komutu
        ClockSet ("Clock1" , "Color" , "255");

Press_Color ayarlama komutu
        ClockSet ("Clock1" , "Press_Color" , "1745238");

Hour ayarlama komutu
        ClockSet ("Clock1" , "Hour" , "23");

Minute ayarlama komutu
        ClockSet ("Clock1" , "Minute" , "30");

Flat ayarlama komutu
```

```
        ClockSet ("Clock1" , "Flat" , "1");

Center ayarlama komutu
        ClockSet ("Clock1" , "Center" , "50");

NoSecond ayarlama komutu
        ClockSet ("Clock1" , "NoSecond" , "0");

NoBackGround ayarlama komutu
        ClockSet ("Clock1" , "NoBackGround" , "1");

NoHands ayarlama komutu
        ClockSet ("Clock1" , "NoHands" , "0");

NoTicks ayarlama komutu
        ClockSet ("Clock1" , "NoTicks" , "1");

OnDown ayarlama komutu
        ClockSet ("Clock1" , "OnDown" , "Clock2_OnDown.c");

OnUp ayarlama komutu
        ClockSet ("Clock1" , "OnUp" , "Clock 1_OnUp.c");
```

7.4GaugeSet ( )

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 ("Gauge1" , "Visible" , "1");

Left ayarlama komutu
        GaugeSet ("Gauge1" , "Left" , "10");

Top ayarlama komutu
        GaugeSet ("Gauge1" , "Top" , "255");

Radius ayarlama komutu
        GaugeSet ("Gauge1" , "Radius" , "35");

Color ayarlama komutu
        GaugeSet ("Gauge1" , "Color" , "255");

Press_Color ayarlama komutu
        GaugeSet ("Gauge1" , "Press_Color" , "1745238");

Value ayarlama komutu
        GaugeSet ("Gauge1" , "Value" , "100");

Range ayarlama komutu
        GaugeSet ("Gauge1" , "Range" , "30");

Major_Count ayarlama komutu
        GaugeSet ("Gauge1" , "Major_Count" , "10");

Minor_Count ayarlama komutu
        GaugeSet ("Gauge1" , "Minor_Count" , "5");

Flat ayarlama komutu
        GaugeSet ("Gauge1" , "Flat" , "1");

Center ayarlama komutu
        GaugeSet ("Gauge1" , "Center" , "50");

NoBackGround ayarlama komutu
        GaugeSet ("Gauge1" , "NoBackGround" , "1");

NoHands ayarlama komutu
        GaugeSet ("Gauge1" , "NoHands" , "0");

NoTicks ayarlama komutu
        GaugeSet ("Gauge1" , "NoTicks" , "1");

OnDown ayarlama komutu
        GaugeSet ("Gauge1" , "OnDown" , "Gauge2_OnDown.c");

OnUp ayarlama komutu
        GaugeSet ("Gauge1" , "OnUp" , "Gauge1_OnUp.c");
```

7.5DialSet ( )

Açıklama

Dial nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void DialSet(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
        GaugeSet ("Gauge1" , "Visible" , "1");

Left ayarlama komutu
        GaugeSet ("Gauge1" , "Left" , "10");

Top ayarlama komutu
        GaugeSet ("Gauge1" , "Top" , "255");

Radius ayarlama komutu
        GaugeSet ("Gauge1" , "Radius" , "35");

Color ayarlama komutu
        GaugeSet ("Gauge1" , "Color" , "255");

Press_Color ayarlama komutu
        GaugeSet ("Gauge1" , "Press_Color" , "1745238");

Value ayarlama komutu
        GaugeSet ("Gauge1" , "Value" , "100");

BackGround_Color ayarlama komutu
        GaugeSet ("Gauge1" , "BackGround_Color" , "65280");
```

```
Flat ayarlama komutu
        GaugeSet ("Gauge1" , "Flat" , "1");

Center ayarlama komutu
        GaugeSet ("Gauge1" , "Center" , "50");

OnDown ayarlama komutu
        GaugeSet ("Gauge1" , "OnDown" , "Gauge2_OnDown.c");

OnUp ayarlama komutu
        GaugeSet ("Gauge1" , "OnUp" , "Gauge1_OnUp.c");
```

7.6SliderSet ( )

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  ("Slider1" , "Visible" , "1");

Left ayarlama komutu
        SliderSet  ("Slider1" , "Left" , "10");

Top ayarlama komutu
        SliderSet  ("Slider1" , "Top" , "255");
```

```
Width ayarlama komutu
        SliderSet  ("Slider1" , "Width" , "90");


Height ayarlama komutu
        SliderSet  ("Slider1" , "Height" , "70");

Color ayarlama komutu
        SliderSet  ("Slider1" , "Color" , "255");

Press_Color ayarlama komutu
        SliderSet  ("Slider1" , "Press_Color" , "1745238");

Thumb_Color ayarlama komutu
        SliderSet  ("Slider1" , "Thumb_Color" , "65280");

BackGround_Color ayarlama komutu
        SliderSet  ("Slider1" , "BackGround_Color" , "1458269");

Range ayarlama komutu
        SliderSet  ("Slider1" , "Range" , "100");

Value ayarlama komutu
        SliderSet  ("Slider1" , "Value" , "10");

Flat ayarlama komutu
        SliderSet  ("Slider1" , "Flat" , "1");

OnDown ayarlama komutu
        SliderSet  ("Slider1" , "OnDown" , "Slider2_OnDown.c");

OnUp ayarlama komutu
        SliderSet  ("Slider1" , "OnUp" , "Slider1_OnUp.c");
```

7.7LabelSet ( )

Açıklama

Label nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

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

```
Visible ayarlama komutu
        LabelSet ("ELabel1" , "Visible" , "1");

Left ayarlama komutu
        LabelSet ("ELabel1" , "Left" , "10");

Top ayarlama komutu
        LabelSet ("ELabel1" , "Top" , "255");

FontHandle ayarlama komutu
        LabelSet ("ELabel1" , "FontHandle" , "23");

FontName ayarlama komutu
        LabelSet ("ELabel1" , "FontName" , "Roboto");

Font_Color ayarlama komutu
        LabelSet ("ELabel1" , "Font_Color" , "16777215");

Caption ayarlama komutu
        LabelSet ("ELabel1" , "Caption" , "Button");

Center ayarlama komutu
        LabelSet ("ELabel1" , "Caption" , "Center");
```

7.8 ToggleSet ( )

Açıklama

Toggle nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void ToggleSet(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

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

Left ayarlama komutu

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

Top ayarlama komutu

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

Width ayarlama komutu

```
ToggleSet ("Toggle1" , "Width" , "90");
```

State ayarlama komutu

```
ToggleSet ("Toggle1" , "State " , "1");
```

StateOffCaption ayarlama komutu

```
ToggleSet ("Toggle1" , "StateOffCaption " , "No");
```

StateOnCaption ayarlama komutu

```
ToggleSet ("Toggle1" , "StateOnCaption " , "Yes");
```

Color ayarlama komutu

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

Press_Color ayarlama komutu

```
ToggleSet ("Toggle1" , "Press_Color" , "1745238");
```

BackGround_Color ayarlama komutu

```
ToggleSet ("Toggle1" , "BackGround_Color " , "65280");
```

Flat ayarlama komutu

```
ToggleSet ("Toggle1" , "Flat" , "1");
```

OnDown ayarlama komutu

```
        ToggleSet ("Toggle1" , "OnDown" , "Toggle2_OnDown.c");

OnUp ayarlama komutu
        ToggleSet ("Toggle1" , "OnUp" , "Toggle1_OnUp.c");
```

7.9ScrollBarSet ( )

Açıklama

Scroll Bar nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void ScrollBarSet(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
        ScrollBarSet ("ScrollBar1" , "Visible" , "1");

Left ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Left" , "10");

Top ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Top" , "255");

Width ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Width" , "90");

Height ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Height" , "70");

Color ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Color" , "255");
```

```
Press_Color ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Press_Color" , "1745238");

Range ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Range" , "100");

Size ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Size" , "20");

Value ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Value" , "50");

BackGround_Color ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "BackGround_Color" , "1458269");

Flat ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "Flat" , "1");

OnDown ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "OnDown" ,
      "ScrollBar_OnDown.c");

OnUp ayarlama komutu
        ScrollBarSet ("ScrollBar1" , "OnUp" ,
       "ScrollBar_OnUp.c");
```

7.10 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  ("ProgressBar1" , "Visible" , "1");

Left ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Left" , "10");

Top ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Top" , "255");

Width ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Width" , "90");

Height ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Height" , "70");

Color ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Color" , "255");

Press_Color ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Press_Color" , "1745238");

Range ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Range" , "100");

Value ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Value" , "50");

BackGround_Color ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "BackGround_Color" , "1458269");

Flat ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "Flat" , "1");

OnDown ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "OnDown" ,
      "ProgressBar_OnDown.c");

OnUp ayarlama komutu
        ProgressBarSet  ("ProgressBar1" , "OnUp" ,
       "ProgressBar_OnUp.c");
```

7.11 KeySet ( )

Açıklama

Key nesnesinin parametre ayarlarını düzenleyen komuttur.

Fonksiyon

void KeySet(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
        KeySet  ("Key1" , "Visible" , "1");

Left ayarlama komutu
        KeySet  ("Key4" , "Left" , "10");

Top ayarlama komutu
        KeySet  ("Key7" , "Top" , "255");

Width ayarlama komutu
        KeySet  ("Key1" , "Width" , "90");

Height ayarlama komutu
        KeySet  ("Key4" , "Height" , "70");

Color ayarlama komutu
```

```
        KeySet  ("Key7" , "Color" , "255");

ColorTo ayarlama komutu
        KeySet  ("Key1" , "ColorTo" , "65280");

Press_Color ayarlama komutu
        KeySet  ("Key4" , "Press_Color" , "1745238");

Max_Lenght ayarlama komutu
        KeySet  ("Key7" , "Max_Lenght " , "1458269");

FontHandle ayarlama komutu
        KeySet  ("Key1" , "FontHandle" , "23");

FontName ayarlama komutu
        KeySet  ("Key4" , "FontName" , "Roboto");

Font_Color ayarlama komutu
        KeySet  ("Key7" , "Font_Color" , "16777215");

Caption ayarlama komutu
        KeySet  ("Key4" , "Caption" , "Button");

OnDown ayarlama komutu
        KeySet  ("Key7" , "OnDown" , "Key7_OnDown.c");

OnUp ayarlama komutu
        KeySet  ("Key7" , "OnUp" , "Key7_OnUp.c");
```

7.12 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

Örnek kod

```
Visible ayarlama komutu
        ButtonSet ("EButton1" , "Visible" , "1");

Left ayarlama komutu
        ButtonSet ("EButton4" , "Left" , "10");

Top ayarlama komutu
        ButtonSet ("EButton7" , "Top" , "255");

Width ayarlama komutu
        ButtonSet ("EButton1" , "Width" , "90");

Height ayarlama komutu
        ButtonSet ("EButton4" , "Height" , "70");

Color ayarlama komutu
        ButtonSet ("EButton7" , "Color" , "255");

ColorTo ayarlama komutu
        ButtonSet ("EButton1" , "ColorTo" , "65280");
Press_Color ayarlama komutu
        ButtonSet ("EButton4" , "Press_Color" , "1745238");

Press_ColorTo ayarlama komutu
        ButtonSet ("EButton7" , "Press_ColorTo" , "1458269");

FontHandle ayarlama komutu
        ButtonSet ("EButton1" , "FontHandle" , "23");

FontName ayarlama komutu
        ButtonSet ("EButton4" , "FontName" , "Roboto");

Font_Color ayarlama komutu
        ButtonSet ("EButton7" , "Font_Color" , "16777215");

Gradient ayarlama komutu
        ButtonSet ("EButton1" , "Gradient" , "3");

Caption ayarlama komutu
        ButtonSet ("EButton4" , "Caption" , "Button");

OnDown ayarlama komutu
```

```
        ButtonSet ("EButton7" , "OnDown" , "EButton2_OnDown.c");

OnUp ayarlama komutu
        ButtonSet ("EButton7" , "OnUp" , "EButton1_OnUp.c");
```

7.13 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 ("EImage1" , "Visible" , "1");

Left ayarlama komutu
        ImageSet ("EImage1" , "Left" , "10");

Top ayarlama komutu
        ImageSet ("EImage1" , "Top" , "255");

RotationAngle ayarlama komutu
        ImageSet ("EImage1" , "RotationAngle" , "90");

RotationCenterLeft ayarlama komutu
        ImageSet ("EImage1" , "RotationCenterLeft" , "400");
```

```
RotationCenterTop ayarlama komutu
        ImageSet ("EImage1" , "RotationCenterTop" , "240");

ColorTo ayarlama komutu
        ImageSet ("EImage1" , "ScaleX" , "65280");

ScaleY ayarlama komutu
        ImageSet ("EImage1" , "ScaleY" , "1745238");

OnDown ayarlama komutu
        ImageSet ("EImage1" , "OnDown" , "EImage2_OnDown.c");

OnUp ayarlama komutu
        ImageSet ("EImage1" , "OnUp" , "EImage1_OnUp.c");
```

7.14 LocalStdVarGet ( )

Açıklama

Local(yerel) string veri okuma komutudur.

Fonksiyon

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

Parametre	Açıklama
name	Yerel değişkenin ismi
value	Yerel değişkenin atanacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"

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

7.15 LocalStdVarSet ( )

Açıklama

Local(yerel) string değer atama komutudur.

Fonksiyon

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

Parametre	Açıklama
name	Yerel değişkenin ismi
value	Yerel değişkenin alacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"

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

7.16 LocalIntVarGet ( )

Açıklama

Local(yerel) integer veri okuma komutudur.

Fonksiyon

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

Parametre	Açıklama
name	Yerel değişkenin ismi
value	Yerel değişkenin atanacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value;

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

7.17 LocalIntVarSet ( )

Açıklama

Local(yerel) integer değer atama komutudur.

Fonksiyon

void LocalIntVarSet(unsigned char *name , int value)

Parametre	Açıklama
name	Yerel değişkenin ismi
value	Yerel değişkenin alacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value;

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

7.18 GlobalStdVarGet ( )

Açıklama

Global string veri okuma komutudur.

Fonksiyon

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

Parametre	Açıklama
name	Global değişkenin ismi
value	Global değişkenin atanacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"

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

7.19 GlobalStdVarSet ( )

Açıklama

Global string değer atama komutudur.

Fonksiyon

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

Parametre	Açıklama
name	Global değişkenin ismi
value	Global değişkenin alacağı string

Örnek kod

```
#include "stdio.h"

#include "stk.h"

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

7.20 GlobalIntVarGet ( )

Açıklama

Global integer veri okuma komutudur.

Fonksiyon

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

Parametre	Açıklama
name	Global değişkenin ismi
value	Global değişkenin atanacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value;

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

7.21 GlobalIntVarSet ( )

Açıklama

Global integer değer atama komutudur.

Fonksiyon

void GlobalIntVarSet(unsigned char *name , int value)

Parametre	Açıklama

name	Global değişkenin ismi
value	Global değişkenin alacağı integer

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int value;

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

7.22 halGpioSet ( )

Açıklama

Kullanıcıların istekleri doğrultusunda kullanabilmesi için 4 farklı GPIO pini bulunmaktadır. Bu pinleri isteğe bağlı olarak INPUT veya OUTPUT olarak ayarlayabilen komuttur.

Fonksiyon

void halGpioSet(unsigned char *pin, unsigned char *value)

Parametre	**Açıklama**
pin	Pin ismini verir
value	Write/ Read (Okuma/yazma)

Örnek kod

```
#include "stdio.h"

#include "stk.h"

halGpioSet("GPIO_OUT_1" , "Write");      // 1 numaralı pini input olarak ayarlar

halGpioSet("GPIO_OUT_2" , "Read");      // 2 numaralı pini output olarak ayarlar
```

7.23 uartDataGet ( )

Açıklama

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

Fonksiyon

void uartDataGet(char *value , int *uartsize)

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

Örnek kod

```
#include "stdio.h"

#include "stk.h"

char uartData[30];                  // Uarttan gelecek verinin depolanacağı string

int uartsize;                          //  Uarttan gelen verinin boyutu

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

7.24 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 "stdio.h"

#include "stk.h"

ChangeScreenSet ("GeçişYapılacakEkranınİsmi");
```

7.25 DrawScreen( )

Açıklama

Sayfa yenileme komutudur.

Fonksiyon

void DrawScreen()

Örnek kod

```
#include "stdio.h"

#include "stk.h"

DrawScreen();        // Sayfa yenileme komutu
```

7.26 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
```

7.27 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
```

7.28 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
```

7.29 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
```

7.30 AudioPlay( )

Açıklama

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

Fonksiyon

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

Parametre	Açıklama
audioname	Ses dosyasını ismi
volume	Ses düzeyi

Örnek kod

```
#include "stdio.h"

#include "stk.h"

int volume;   // Ses Düzeyi

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

7.31 PlaySound ( )

Açıklama

Cihazın içinde hazır uyarı sesleri yüklü olarak bulunmaktadır. PlaySound fonksiyonu ile daha önce hazırlanmış 5 farklı uyarı sesinden istenilen int option bölümüne 1,2,..,5 gibi yazılarak seçilebilir.

Fonksiyon

void PlaySound(int option)

Parametre	Açıklama
option	1-5 arası seçim yapılabilir.

Örnek kod

```
#include "stdio.h"

#include "stk.h"

PlaySound(1)
```

7.32 ScreenColorSet ( )

Açıklama

Sayfanın arka plan rengini ayarlamayı sağlayan komuttur.

Fonksiyon

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

Parametre	Açıklama
name	Arka plan rengi değiştirilecek olan sayfanın ismi
value	Sayfanın arka plan renk kodu (int)

Örnek kod

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

ScreenColorSet("Screen1", "0");    //Screen1 ekranının arka plan rengi siyah ayarlandı
```

7.33 LCDsleep ( )

Açıklama

Proje aktif olarak çalışırken LCD ekranda görüntünün gerekli olmadığı durumlarda LCD ekranı uyku moduna alan komuttur.

Fonksiyon

void LCDsleep()

Örnek kod

```
#include "stdio.h"

#include "stk.h"

LCDsleep();              //  LCD uyku modu aktif
```

7.34 LCDwakeup ( )

Açıklama

Uyku durumundaki LCD ekranda görüntünün yeniden gelmesini sağlayan komuttur.

Fonksiyon

void LCDwakeup()

Örnek kod

```
#include "stdio.h"

#include "stk.h"

LCDwakeup();            //  LCD uyku modu kapalı
```

7.35 SoundFileState( )

Açıklama

Ses dosyasının ses düzeyi hakkında bilgi veren komuttur.

Fonksiyon

void SoundFileState(int *sound_state)

Parametre	Açıklama
sound_state	Ses dosyanın önceden ayarlanmış ses düzeyi

Örnek kod

```
#include "stdio.h"
#include "stk.h"
int volume;

SoundFileState(&volume);        // Ayarlı Ses Seviyesini Öğrenme
```

7.36 File_write ( )

Açıklama

Flash'a yazma komutudur.

Fonksiyon

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

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

Örnek kod

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

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

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

7.37 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.
```

7.38 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.
```

7.39 RotateScreen ( )

Açıklama

Ekranın yatay veya dikey olmasını sağlayan komuttur.

Fonksiyon

void RotateScreen(int option)

Parametre	Açıklama
option	Ekranın yatay veya dikey olması

Örnek kod

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

RotateScreen(1);    // LCD ekran çalışma alanını dikey konuma ayarlama komutudur

RotateScreen(0);    // LCD ekran çalışma alanını yatay konuma ayarlama komutudur
```

7.40 RoleSet ( )

Açıklama

Rölelerin açık-kapalı olma durumlarını kontrol eden komuttur.

Fonksiyon

void RoleSet(char *name , char *value)

Parametre	Açıklama
name	Rölenin ismi
value	Rölenin açık-kapalı olma durumu

Örnek kod

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

RoleSet("Role1" , "1");     // Role1 isimli role aktif hale getirildi

RoleSet("Role2" , "0");     // Role2 isimli role kapatıldı
```

7.41 RS485Set ( )

Açıklama

RS485 haberleşme protokolü ile veri alıp gönderme komutudur.

Fonksiyon

void RS485Set(char *name, char *value)

Parametre	Açıklama
name	Verinin gönderileceğini veya alınacağını belirler
value	Gönderilecek veya alınacak veri

Örnek kod - 1

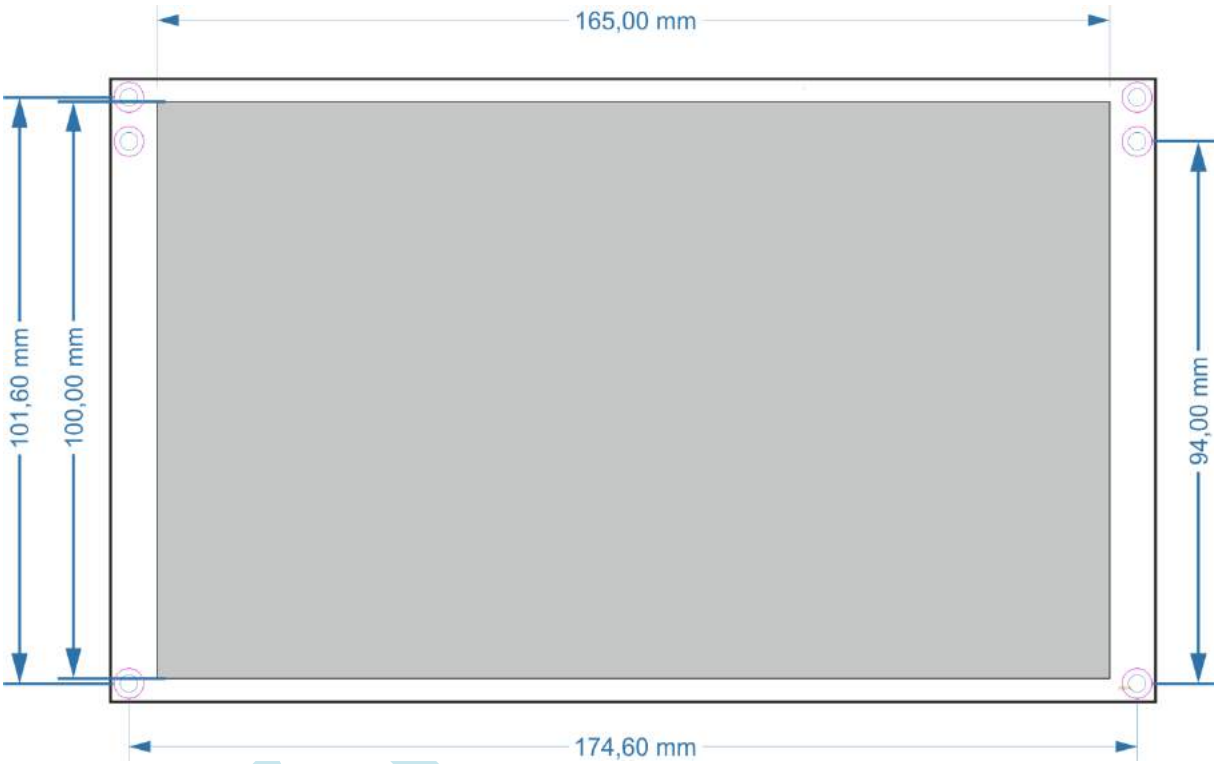
```
// RS485 üzerinden gelen veriyi reveice değişkenine atama

#include "string.h"
#include "stdio.h"
#include "stdlib.h"
#include "stk.h"

char receive[100];

memset(receive , 0x00 , sizeof(receive));

RS485Set("Receive" , receive);
```

Örnek kod – 2

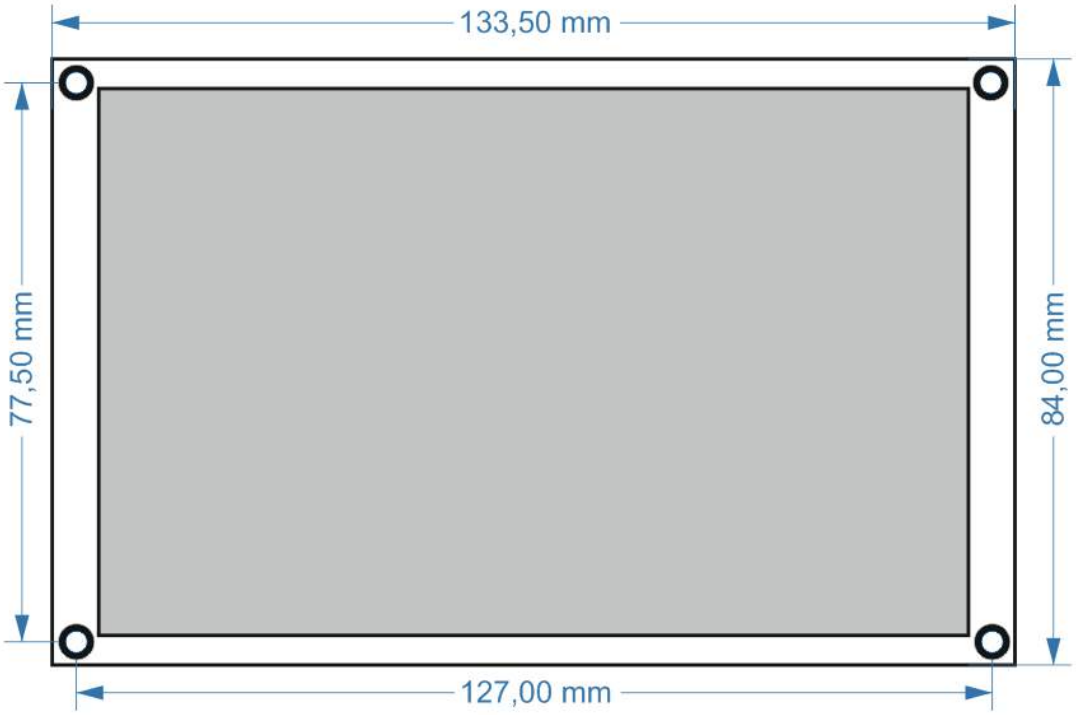
```
// RS485 üzerinden send değişkeninde veriyi görderme

#include "string.h"
#include "stdio.h"
#include "stdlib.h"
#include "stk.h"

char send[100];

memset(send, 0x00 , sizeof(send));

sprintf(send , "%s" , "Hello World !!!");

RS485Set("Send" , send);
```

8. Ürün Boyutları

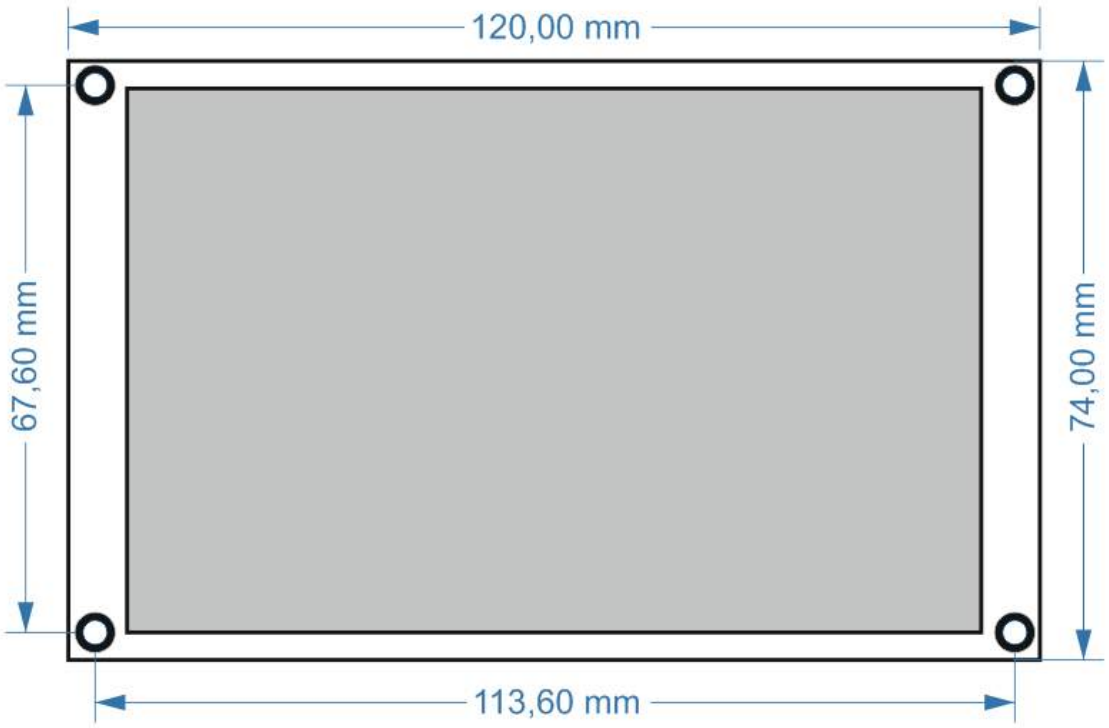
8.1 BT800X480S70_RT



8.2BT800X480S50_RT



8.3BT480X272S43_RT



AirHMI

LCD

SCREEN EDITOR GUIDE

AIRHMI LCD SCREEN EDITOR GUIDE

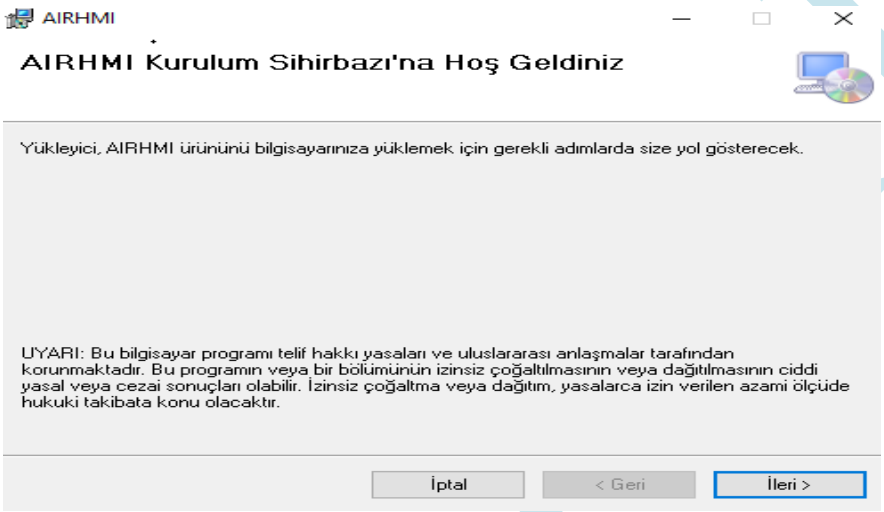
AirHMI Visual Screen Creator is designed to create the highest level of satisfaction and the most efficient time to design Human Machine Interface GUIs for AirHMI LCD displays. In the use of the editor, we have the functionalities of the world of Design and Programming: In addition to the support of the screen design that you can be original from the richness of object in visuality and you can easily create according to your wishes, it also provides many convenience to the programming part.

Table of Contents

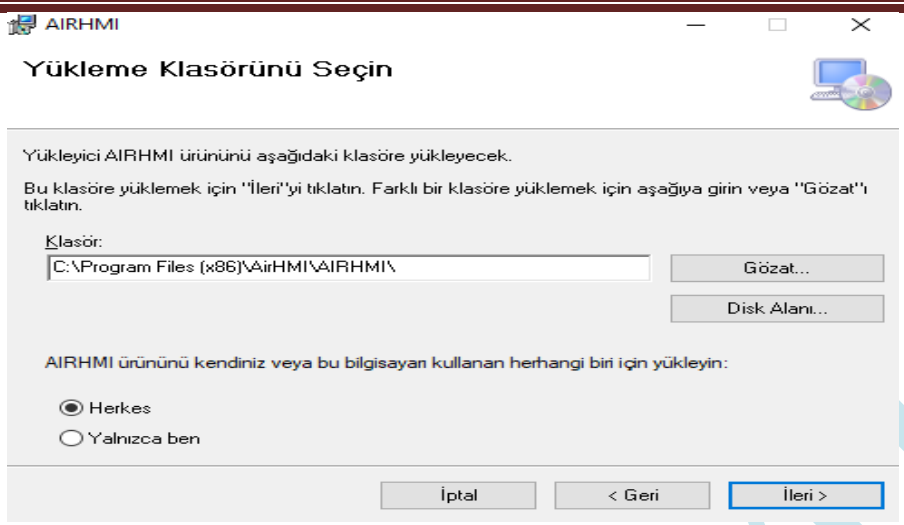
1. AirHMI Visual Screen Creator INSTALLATION

Download Link: http://www.airhmi.com/Setup/AIRHMISETUP.msi

Double-click the AIRHMISETUP.msi file to install the AirHMI Editor on your computer. Then follow the steps below.



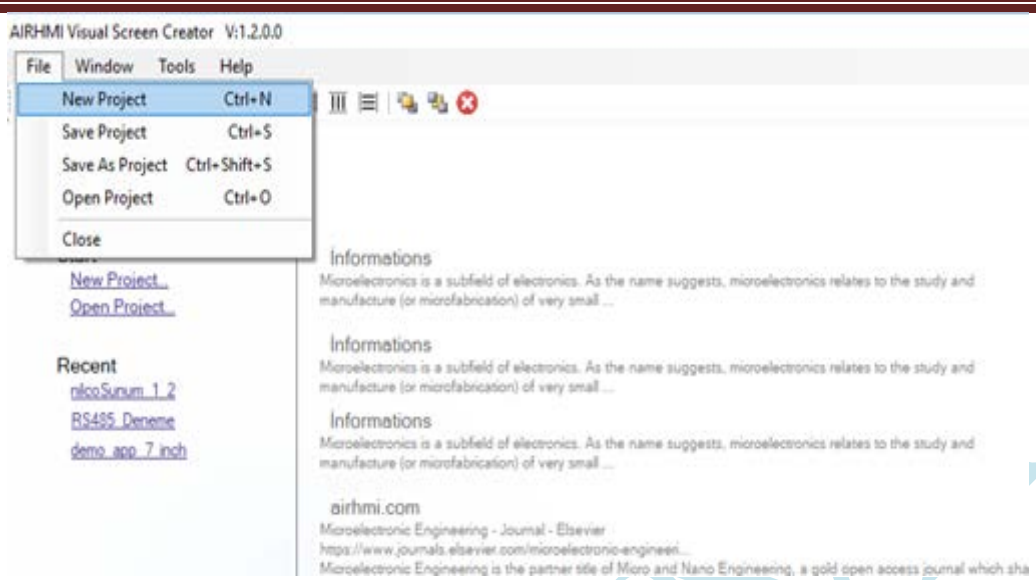
Select the installation folder and other options as desired and press next to start the installation.

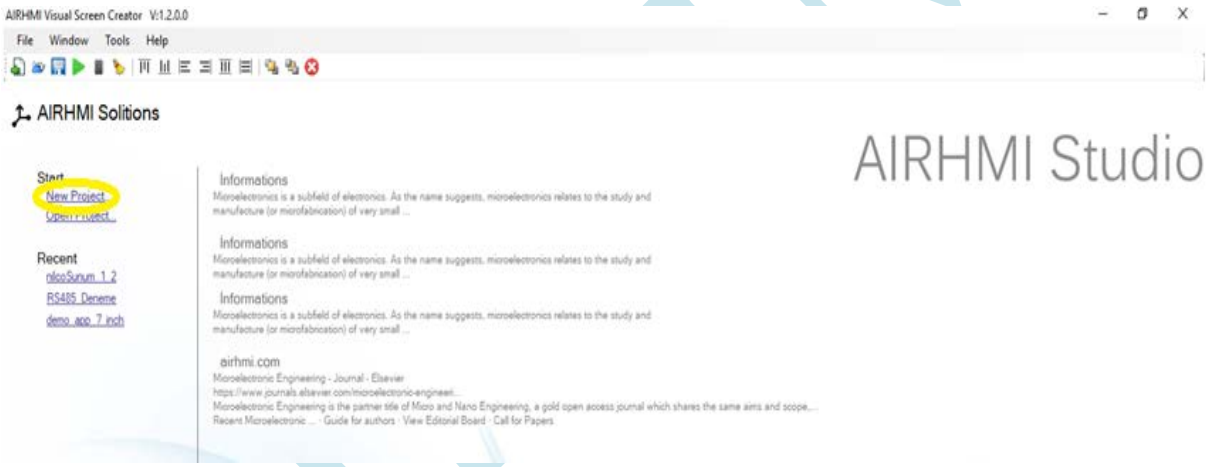


2. Creating Project

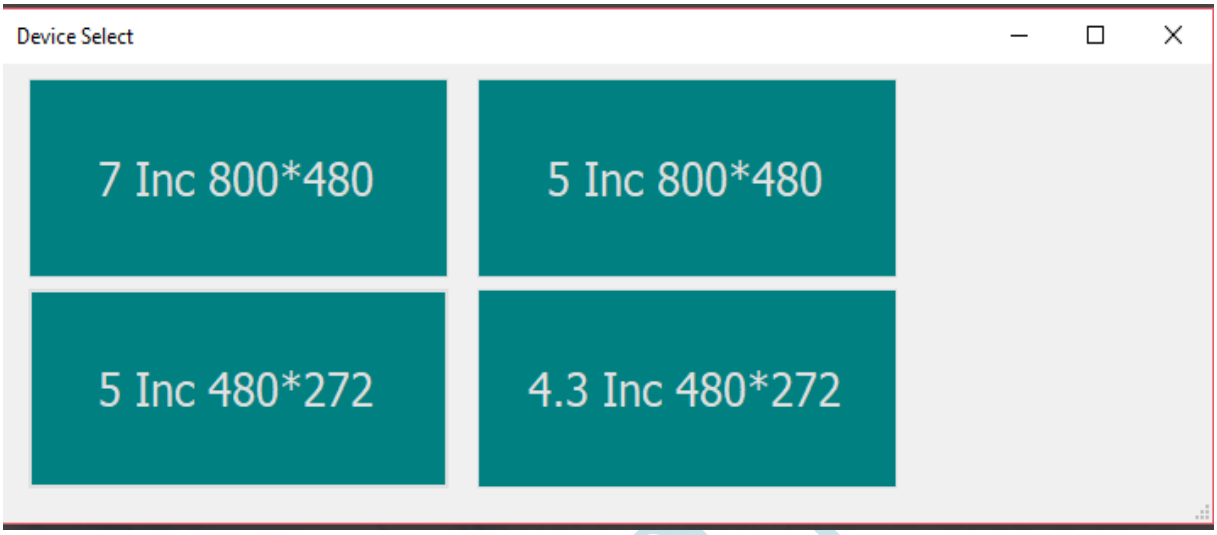
To interface with AirHMI, you first need to download and install AirHMI Editor on your computer. The drag-and-drop feature in the AirHMI Editor program facilitates interface development. With AirHMI Editor, you can add many components to your projects, such as Buttons, Images, Text, Progress bar, Gauge, Key, Analog and Digital values, such as digital inputs and outputs.

The program is very easy to install. After installation, you should run the AirHMI Editor program. You will see a page as shown in the pictures below. You can create your project by following File - New path in the upper left corner of this page or by clicking New Project on the first landing page of the program.


AIRHMI Visual Screen Creator V:1.2.0.0
File
Window
Tools
Help
New Project Ctrl+N
Save Project Ctrl+S
Save As Project Ctrl+Shift+S
Open Project Ctrl+O
Close
New Project...
Open Project...
Recent
nlcoSunum_1_2
RS485_Deneme
demo_app_7 inch
Informations
Microelectronics is a subfield of electronics. As the name suggests, microelectronics relates to the study and manufacture (or microfabrication) of very small ...
Informations
Microelectronics is a subfield of electronics. As the name suggests, microelectronics relates to the study and manufacture (or microfabrication) of very small ...
Informations
Microelectronics is a subfield of electronics. As the name suggests, microelectronics relates to the study and manufacture (or microfabrication) of very small ...
airhmi.com
Microelectronic Engineering - Journal - Elsevier
https://www.journals.elsevier.com/microelectronic-engineeri...
Microelectronic Engineering is the partner title of Micro and Nano Engineering, a gold open access journal which sha


AIRHMI Visual Screen Creator V:1.2.0.0
File
Window
Tools
Help
AIRHMI Solitions
AIRHMI Studio
Start
New Project...
Open Project...
Recent
nlcoSunum_1_2
RS485_Deneme
demo_app_7 inch
Informations
Microelectronics is a subfield of electronics. As the name suggests, microelectronics relates to the study and manufacture (or microfabrication) of very small ...
Informations
Microelectronics is a subfield of electronics. As the name suggests, microelectronics relates to the study and manufacture (or microfabrication) of very small ...
Informations
Microelectronics is a subfield of electronics. As the name suggests, microelectronics relates to the study and manufacture (or microfabrication) of very small ...
airhmi.com
Microelectronic Engineering - Journal - Elsevier
https://www.journals.elsevier.com/microelectronic-engineeri...
Microelectronic Engineering is the partner title of Micro and Nano Engineering, a gold open access journal which shares the same aims and scope,...
Recent Microelectronic ... · Guide for authors · View Editorial Board · Call for Papers

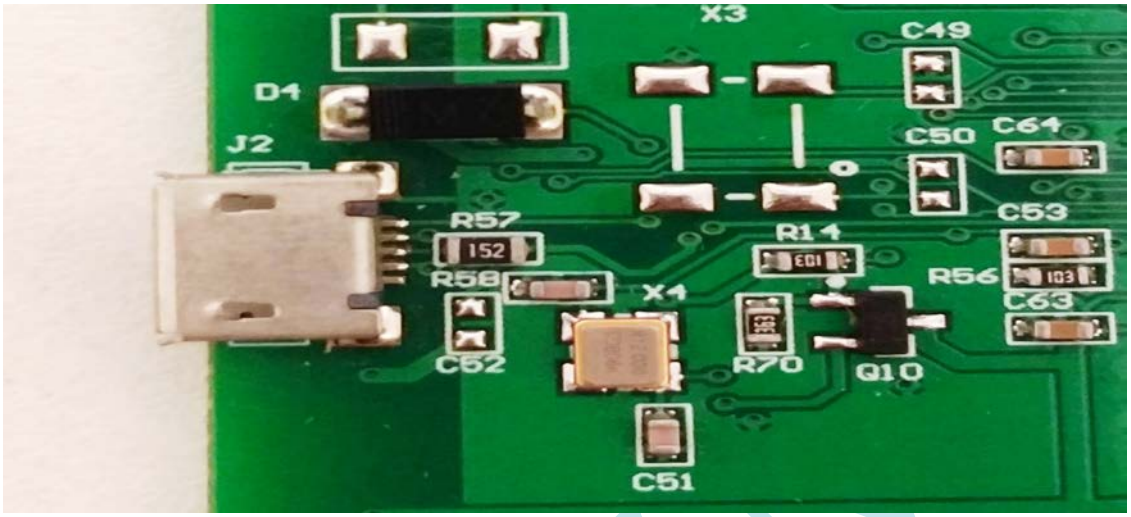
After registration, you will see a page as shown in the image below. On the page that appears you should make settings related to the size and resolution of the screen.



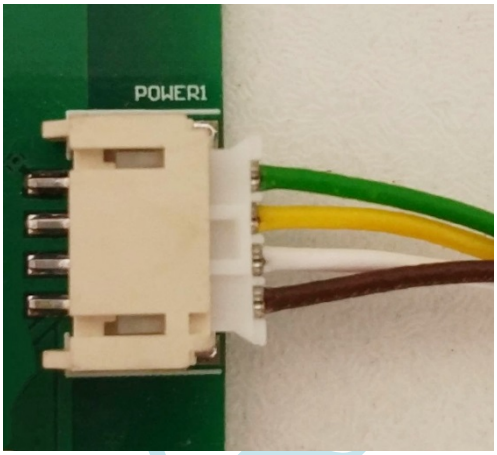
3. Device Connection

Two methods can be used to energize the AirHMI display; POWER connector or USB connector. The standard baudrate for the device is 115200 8N1.

1) USB Connector;



2) If the POWER connector is used, the pins of the connector are as follows:



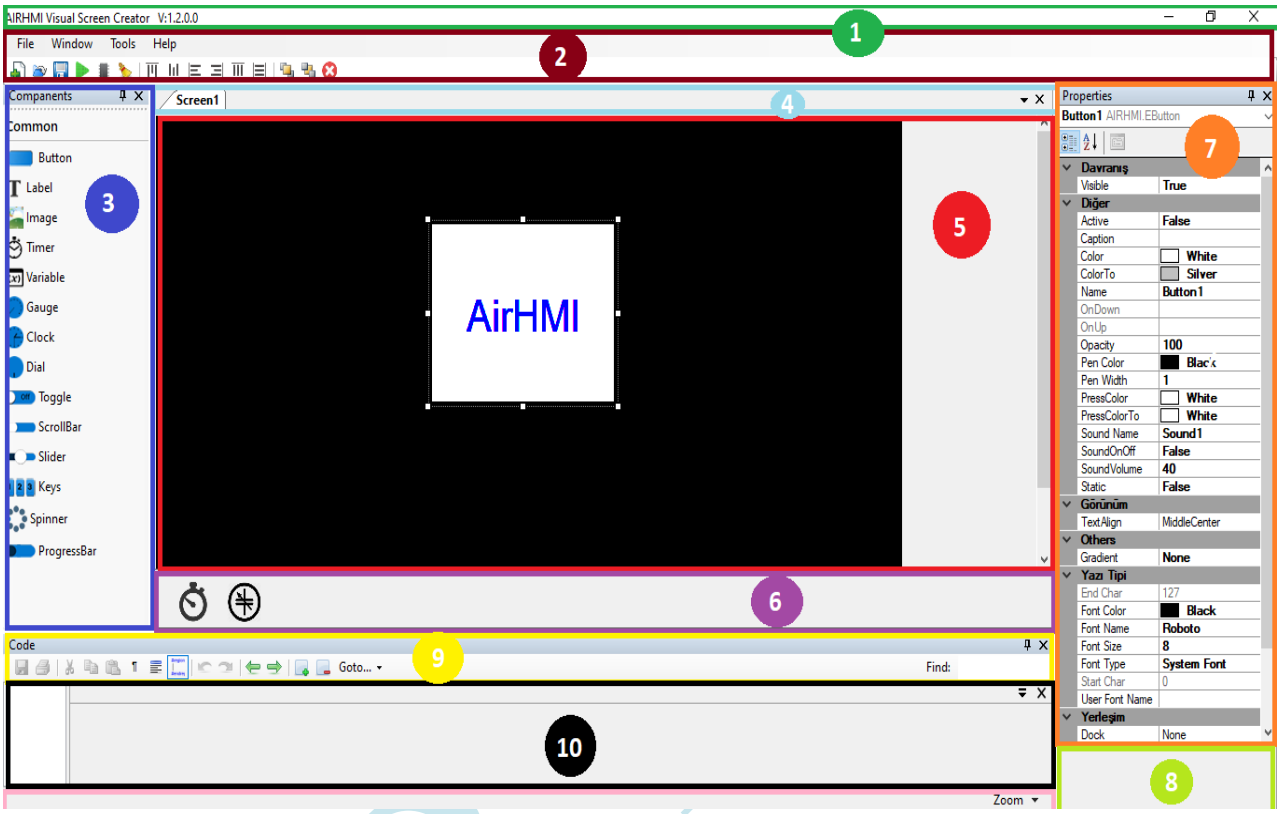
GREEN = GND

YELLOW= RX

WHITE =TX

BROWN = SUPPLY

4. AirHMI EDITOR INTERFACE



4.1 TITLE BAR

The title bar contains the application name and version number when an AirHMI project is opened.

4.2 MAIN MENU AND TOOLBARS

File Menu

There are commands for users such as Open New Project, Save Project, Save Project As, Open an Existing Project and Exit. The important point here is that if you want to open a new project while an existing project is open, the save message on the screen should be given confirmation if the old project is stored on the computer or the changes are not lost.

Windows

Within the window area;

- Creating a new working screen in addition to the main screen used in the project (Add Screen)
- Loading of designed interface screen to AirHMI LCD Card via selected USB port (Download to Flash)
- Export the designed interface screen to a desired file in the computer as external files (Download to SD Card). It is used to load Bootloader via SD Card when USB installation is not desired. When the files are copied to the SD card and the project is run via the SD Card, the files are loaded from the SD Card as if they were loaded via USB.

Tools

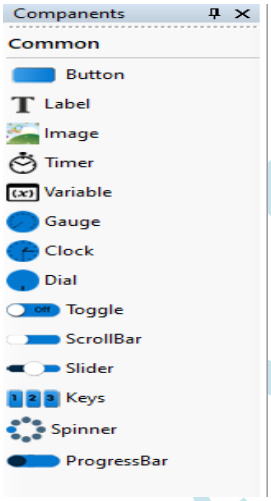
In the tools, there is a section in the Options section for selecting USB ports and setting the baud rate. Since USB loading works at many baud rates, the user can select the desired baud rate setting and load it.

4.3 **Alignment**

Align Left, Align Right, Align Upper and Align Down; By means of the vertical and horizontal averaging properties, the determined objects are aligned or centered as desired.

The Bring to Front and Send to Back features can be used to determine which nested objects are in the foreground and are used for objects that are required to stand in the background.

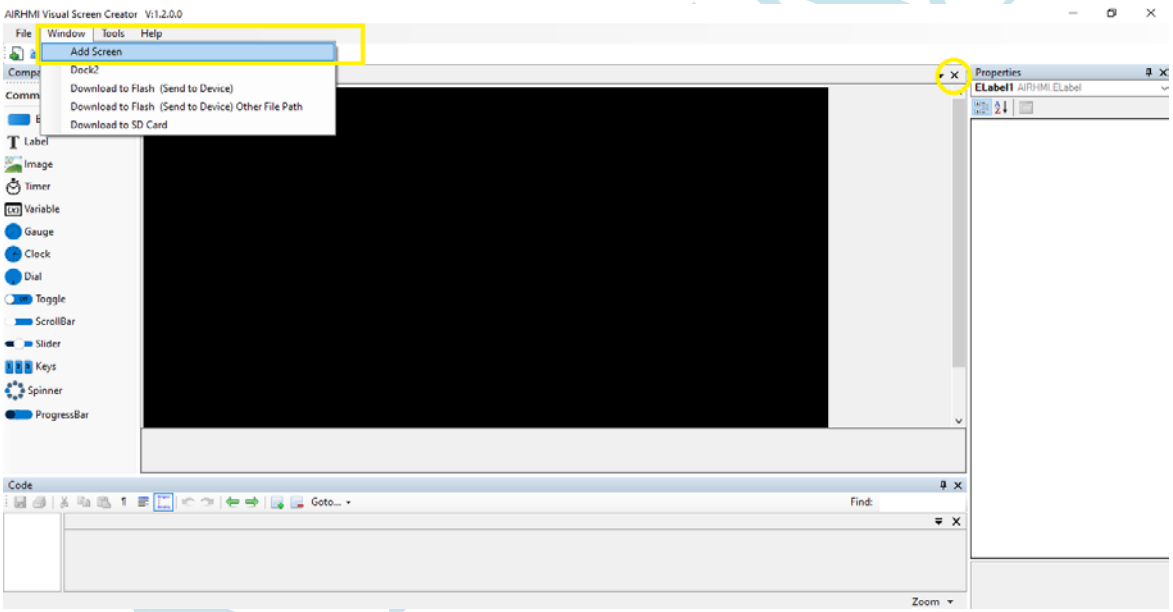
Component Pane



The AirHMI LCD Design Screen contains the ready-made objects to be displayed. Click on the object you want to use and drag to the screen area is added to the project. External objects not shown on the screen are also included in this section: Timer and Variable. These objects are located in the Area of Non-Visual Components section at the bottom of the screen area. Setting the properties (location, size, name, etc.) of the objects in the project-specific project is found in the Attribute Field of Objects section.

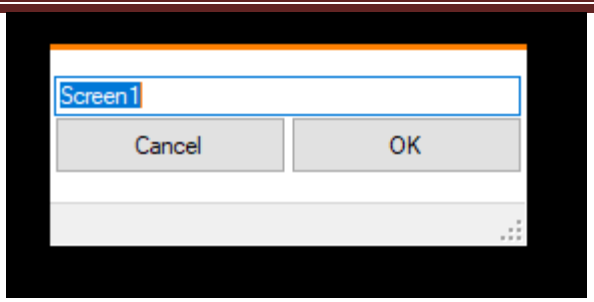
4.4 SCREEN / COMMAND TAB

Design projects are not generally used as single screens but require different screens at the same time. Opening general display screen, menu setting screen, detailed display screen etc… For this reason, AirHMI can create multiple original and screen designs from the editor in line with user requests. The screen / command tab is used to select which screen to work on.



The Window / Add Screen tab can be used to add a new worksheet, or you can select Add Screen by right-clicking in an empty place on the worksheet. To delete the opened worksheet, simply press the cross (x) at the end of the Display / Command Tab line.

To change the name of the screen, right-click in an empty area of the screen and click the Rename tab. The name of the screen can be changed from the opened tab.



4.5Main Screen of Design Area

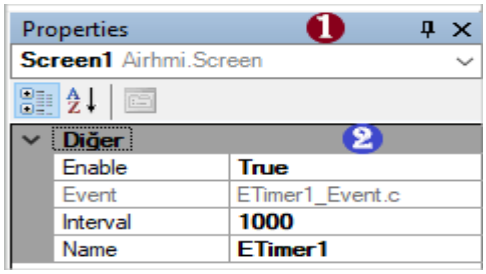
The AIRHMI Designer worksheet is the design visual area. In the LCD screen design, features such as which objects can be found on the screen, dimensions, writing characteristics are shown in this area.

AHMI SCREEN EDITOR

4.6 Area of Non-Visual Components

In a project, not all components are displayed on the LCD screen. there are components that do not need to be displayed on the LCD screen, such as timer and variable. It is important to display the components that are running in the background in the editor so that they are not shown on the LCD screen but this is easy to use and understandable during design.

4.7 Attribute Field of Objects



3.7.1 Display Area of Objects Used in the Project

Many objects can be used in LCD screen design. Each object has its own settings. In projects where more detail is required, it can be complicated to find the object to be adjusted from the design screen. This is the list of all objects used in the design to avoid confusion. In this way, the desired object can be selected and adjusted in the attribute field.

3.7.2 Attributes of Objects Display / Setting Area

In AirHMI Editor, objects are automatically added with their initial settings when they are included in the project. Users can edit many properties of the objects they add, such as names, sizes, appearance, colors, according to their intended use and wishes.

4.8 DESCRIPTION AREA OF FEATURES

The settings of the objects are performed in the attribute field. But there is only the attribute name. In the Description Field of Attributes, there is a description part of the attributes. The functions that attribute headers perform are generally explained.

4.9 User Project Code Menu and Toolbars

The most important part of the project is the code phase. According to the project basis, the coding structure is used to determine what situations are shown on the design screen. The Code Menu contains some basic components that can help the user save code, copy and paste, search for keywords in code, and so on.

4.10 USER PROJECT CODE AREA

Contains a valid AirHMI PICOC Code Instruction for the User Project Code. This section will not teach programming, but will generally help the user to add code. Within this area, users will be able to write C-based codes to the events of the Timer component or to the events of the objects they use on the screen. Screen Editor supports ready-made library codes to minimize the software difficulty of this section, ready-made functions under the third heading (3. Functions) you can examine in detail. In addition to the functions mentioned there, all C-based codes can be written to this field and executed simultaneously in the program.

4.11 Code Area, Zoom Area

In the project design, the code area is the area where the font size can zoom in and out to the desired extent for ease of use.

5. USER CODE STRUCTURE

One of the most important advantages of AirHMI Editor's is its solution-oriented, time and effort-oriented design that creates the most efficient point, and its easy and understandable code structure. The code structure is prepared in C programming language. However, in order to be user-oriented and provide ease of use for the user, the necessary functions are prepared under the “stk.h” library. In this layout, where basic C libraries are attached, you can create your code using the C programming language and add the necessary functions to the beginning of your code. In addition to the ready-made C functions, you can find the ready-made functions in this manual along with descriptions of many important subjects such as control / adjustment functions of objects, LCD display sleep mode, timer code layout. The important point here is that the "stk.h" library must be included at the beginning of each code structure for these functions to work effectively.

ETimer1_Event.c

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

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

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

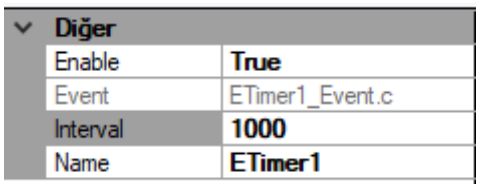
    DrawScreenGet();

}
```

Sample code structure is prepared with timer. A detailed description of the timer code structure is described under **2.1 TIMER.**

Code structure can be in the Timer according to the desired situation, or the code structure that we want to operate when objects are touched can be created for resistive screens. While the code that you will create in the Event will be active in the timer program in the whole program, the code structure that you want to be active when the objects are tapped must be added to the OnUp section of the attribute.

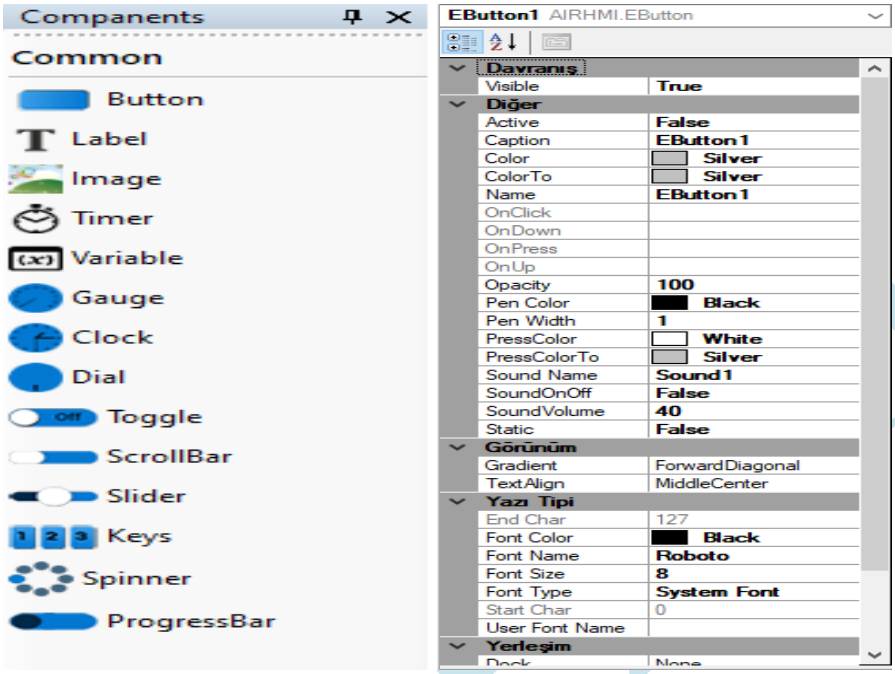
5.1 TIMER



Perhaps the most important point in the code structure is the use of Timer. The changes that will occur in real-time operation of the designed editor screen in the project and the intervals at which these changes will occur are set within the Timer Attributes. The enable, selects whether the Timer is active or not. Interval is the interval in milliseconds at which the code is selected. Name, as the name implies, is the name of the Timer. The Event section is the opening section of the code to be generated for the project design. ETimer1_Event.c is the name of the C file where the generated code is saved.

When using the timer, the code structure activates the code sequence at that intervals according to the time set in the Interval, regardless of the status of the objects. If the user uses a Resistive screen in its project and wants to perform an operation when an object is touched; When touched, the object that you want to process should come to "OnUp" section from the Setting attributes section and add the code under this attribute. Thus, regardless of the timer, only the code that is written when the object is touched will be active.

5.2 AirHMI SCREEN OBJECTS



5.2.1 Button

The button object is the object that makes any action when pressed. For instance, it can be used to send data received from the user to a location, to process the received data, or to send messages. You can drag the position of the button to the desired location and adjust its size by pulling the edges.

5.2.2 Label

It is a display object used to inform the user that any identifiable label is written for any object on the screen.

5.2.3 Image

You should ensure that the images you add are not larger than the screen size. If the image you added is larger than the screen size, your entire image will not be displayed because there is no in-editor resizing.

Another important point to note is that the format of the images to be added must be in PNG image format.

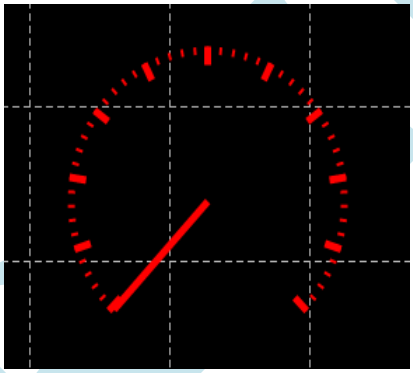
5.2.4 ProgressBar

It is used in cases where the execution steps of a long process need to be graphically displayed. An example of using the Progress Bar is: showing the remaining time of a video or audio file that is currently playing on the Progress Bar, graphically showing the fill rate of a fuel tank using the Progress Bar.

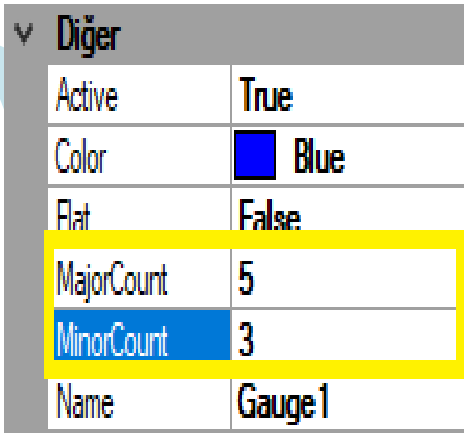
5.2.5 ScrollBar

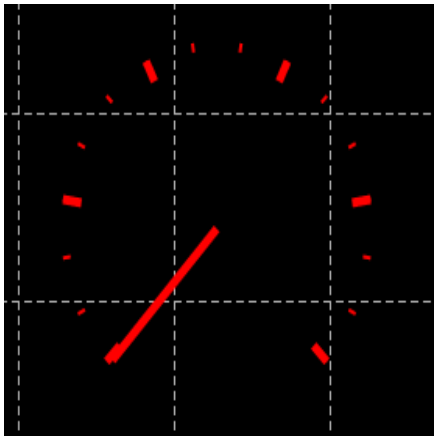
An object that allows the screen to be scrolled when the design does not fit the screen or when the design screen is to be shifted.

5.2.6 Gauge



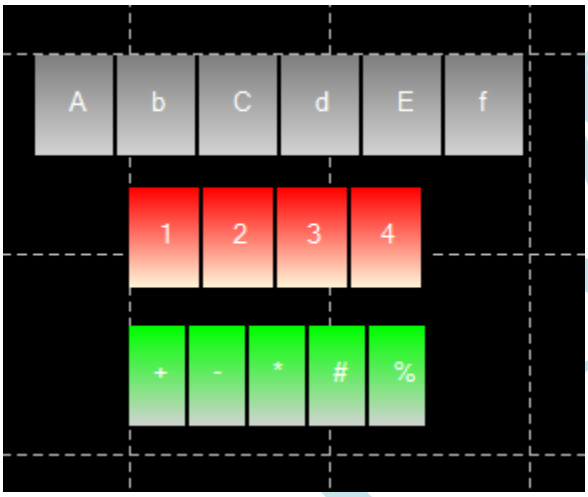
The period range on the scale can be edited from the attributes section for the gauge object.



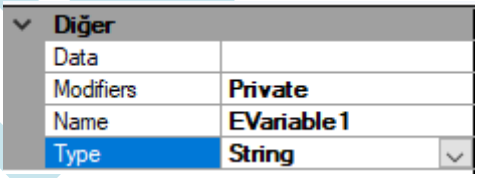


5.2.7 Keys

An object that you can use as an onscreen keyboard. You can design the keyboard by entering letters, numbers, characters as you want with Caption feature from the attribute section.

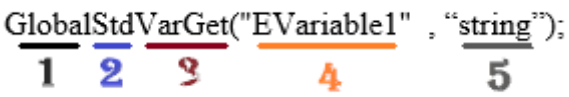


5.3 VARIABLE



Variables take part in the struggle a very important role in situations where the last values of the variables within the code structure or the value in each embodiment of the code are desired not to be lost. As the code structure is generally compiled and restarted every time the Timer is active or when touch-enabled is active in Resistive screen projects, the normal variables generated are reset. This is a major problem for the user when it is desired to use data from the previous location or situation. In order to prevent such a problem, variables come into play. The name of the variables is given in the Attributes section under the heading Name.

If the type of the variable to be used is Type, then char should be String and numeric value should be Integer. Another feature, Modifiers, Attributes section that we want to use Private (local) or Public (global) should be selected. Local-global distinction is made in projects where multiple screen designs will be used. If the operation is to be performed on a single screen, the Private variable can perform the desired state. However, if you want to use more than one screen, for example, if a value found in the second screen is changed to the first screen, Public (Global) variable should be used here. Below is a description of the use of variables in the code structure.



Variable:

1. Global or Local
2. String or Integer
3. Set or Get Value Definition
4. Name
5. Variable to retrieve new value or old value

The desired function should be used according to the situation.

```
int value;
LocalStdVarSet("EVariable1" , "string");  // Set a String variable that is local
GlobalIntVarGet("EVariable2", &value);  //  Get the global integer variable
```

5.4 GPIO

There are 4 different GPIO pins on the AirHMI graphics card for users to use according to their wishes. These pins can be set to INPUT or OUTPUT optionally in projects. Code structure;

void halGpioSet(unsigned char *pin, unsigned char *value)

halGpioSet(* pinname , *read/write)

```
halGpioSet("GPIO_OUT_1", "Write");  // Real time example code structure
halGpioSet("GPIO_OUT_2", "Read");  // Real time example code structure
```

it should be this way.

5.5 UART

In the real life of the project, a processor card or ready-made development kits are used. By sending data from the uart output of these cards, operations can be performed on the AirHMI Editor screen according to the data from the UART. The process of receiving data from the UART within the code scheme is performed as shown below.

- The uartDataGet () command is used to retrieve data from UART.

void uartDataGet(char *value , int *uartsize)

```
char uartData[30];           //  The string where data from UART will be stored.
int uartsize;               //  Size of data from UART
uartDataGet(uartData , &uartsize); //Processing the data from UART
```

- Printf command is used to **send data** from UART.

```
printf("Hello world");
```

5.6 READ&WRITE COMMAND WITH FLASH

Air HMI LCD Card hardware has 32 Mbyte Flash in order to avoid internal memory problems in the projects and enough space for the files to be uploaded by the user. In addition, basic operations such as loading and reading from flash in PICOC code structure are designed for the data that the user sees important and wants to be saved in any case in cases of sudden power outages or power loss in the code directory. In this way, the user can add the desired data to the internal flash for safe use and store it safely. The process of writing, reading and taking the size of a file in the flash in the code layout is done with the functions below.

- **Read from Flash**

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

File_read(the name of the .txt file , the name of the string array , reading size , 1);

- **Write to Flash**

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

File_write(the name of the .txt file , the name of the string array , the size of the directory to be written , 1);

Example;

```
Char example[200];
File_write("Message.txt" , example, sizeof(example), 1);
```

- **Getting File Size Information**

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

File_size(the name of the.txt file, an integer variable);

5.7 DATE AND TIME DISPLAY WITH RTC

The AirHMI LCD Card hardware has an external RTC integration for use in designs. In this way, you can adjust the time and date settings without any external integration or data and display a date and time information with a Text on the LCD screen. In this system, which is realized for the convenience of the user, the things to be done to get and set the clock data are quite simple. With RTC, date and time settings and usage are done with the codes below.

```
unsigned char day, month, year, hour, min; // Example Date-Time variables in the
code index

dateGet(&day , &month , &year);        // Get date data from the RTC

timeGet(&hour , &min);               // Get time data from the RTC

dateSet(&day , &month , &year);    // Renew / set up Date data in RTC

timeSet(&hour , &min);           // Renew / set up Time data in RTC
```

5.8 AUDIO FILE PLAYBACK

It is important for the project to report a warning when certain limitations and values are exceeded in industrial projects. The structure of the AirHMI LCD Card, with its structure aiming to meet many needs, provides the opportunity to play an external sound file and to play the warning sounds in itself. The audio output is in the form of PWM signal. After adding the sound request file to the project via AirHMI Editor, the user can play with the code number 1 below. Code structure number 2 is used to play the audio files in itself. Code structures:

- ```
  int volume;              // Sound level

  AudioPlay("NameoftheSoundFile" , &volume );
  ```
  ```

In the AudioPlay function, the name of the audio file that is desired to be played on the first variable, and the second variable should be given the desired sound level between 0-255.

- `PlaySound(integerValue);`

Prepared warning sounds are installed in the project. It can be selected by typing 1.2, .., 5 into the integerVeger section desired from 5 different warning sounds prepared with PlaySound function.

5.9 LCD DISPLAY SCREEN

The indispensable point for the changes made in the code structure in the projects designed to be seen on the screen is the renewal process of the LCD screen. In order to set the new properties of the objects renewed over the code structure within the prepared project, the screen renewal process should be performed. The user can perform this process by adding the code located below to the point where the page refresh is desired in the code.

- `DrawScreenGet();  // Page refresh command`

Multiple screen designs can be realized in the project where the user wants to prepare. Switching between pages constitutes an important situation in such designs. It can switch the screen by adding the name of the screen that it wants to switch from the code directory to the code below, in line with the user request.

void ChangeScreenSet(unsigned char *value)

- `ChangeScreenSet ("NameofScreentoSwitch");`

5.10 PASSIVE SLEEP MODE

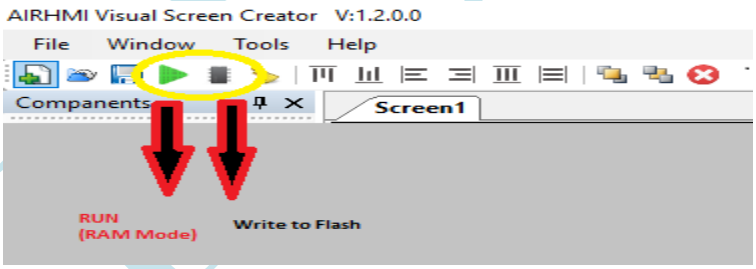
The main purpose of industrial projects is to realize the project design with the most efficient values in terms of time and labor. For this purpose, it is necessary to be careful in the

integrations and coding system used in the projects. AirHMI has been designed by considering these issues in LCD Screen design and offers sleep mode support to the user in order to save energy. When the project is actively working, when the image is not required on the LCD screen, the LCD screen will put itself to sleep using the LCDsleep function. In this way, the LCD screen is prevented from drawing extra power while the project is actively working without losing data. In the same way, when the user wants the image to come back on the LCD screen, they can have the screen again using the LCDwakeup function.

- `LCDsleep();` `// LCD sleep mode activated`
- `LCDwakeup();` `// LCD sleep mode deactivated`

6. DESIGN LOADING

There are two options to upload the design created in AirHMI Editor to the device: Run (RAM Mode) and Write to Flash.

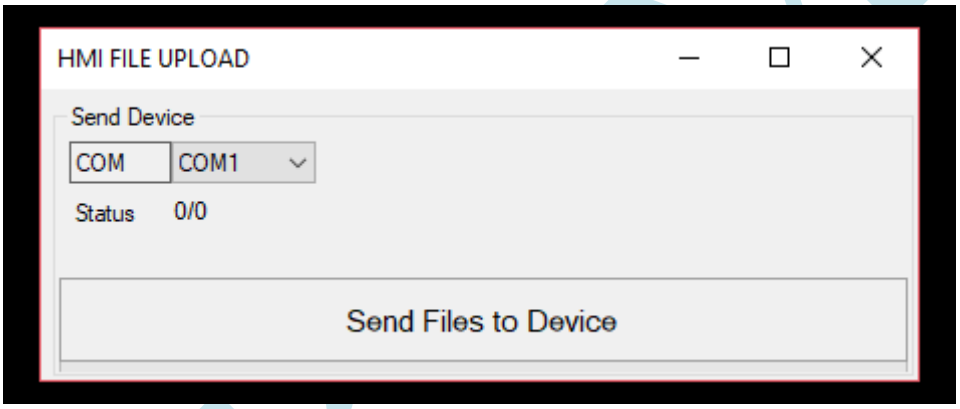


6.1 Run(RAM Mode)

During design development, it is used to quickly load the design prepared in its editor and view it on the screen. However, when you cut the power of the device, the design you have installed is also deleted. The advantage over installing in Flash mode is that it loads your design on the device quickly.

6.2 Write to Flash

The AirHMI LCD Card hardware has 32Mbyte Flash in order to ensure that there is no internal memory problem in the projects and there is enough space for the files to be uploaded by the user. Projects created via AirHMI Editor are uploaded to this template. In addition, the flash memory feature has been added for the data that the user deems important and wants to be recorded in any case in case of sudden power outages or power loss in the code directory. In this way, the user can add the desired data to the internal flash for safe use and store it safely.



When you proceed to the loading of the design on the device, you will see a screen like the figure for both loading methods. You can start the installation by selecting the COM port your device is connected to from the screen that opens. You can also follow the progress of the installation from this screen.

7. FUNCTIONS

Descriptions of the basic functions used in the code structure in the AirHMI Screen Editor and sample code structure are found under this directory. The parameters that can be changed in the functions related to the objects can be easily understood by looking at the object properties via the AirHMI Screen Editor.

7.1delay( )

Explanation

It is a command that allows to wait for the specified time in the line it is used.

Function

void delay (int ms)

Parameter	Explanation
ms	Specifies the time period

Example Code

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

delay(1000);
```

7.2KeyGet ( )

Explanation

It is the command used to inform the user which letter / number is pressed when the Key object is touched. It stores the value printed in keyboard and similar operations in a fixed variable for use elsewhere in the code directory.

Function

void KeyGet(unsigned char *value)

Parameter	Explanation
value	The variable where the value is stored

Example Code

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

unsigned char value;

KeyGet(&value);
```

7.3 ClockSet ( )

Explanation

It is the command that regulates the parameter settings of the clock object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed

value	The new value of the parameter to be changed

Example Code

```
Visible adjustment command
        ClockSet ("EImage1" , "Visible" , "1");

Left adjustment command
        ClockSet ("Clock1" , "Left" , "10");


Top adjustment command
        ClockSet ("Clock1" , "Top" , "255");

Radius adjustment command
        ClockSet ("Clock1" , "Radius" , "35");

Color adjustment command
        ClockSet ("Clock1" , "Color" , "255");

Press_Color adjustment command
        ClockSet ("Clock1" , "Press_Color" , "1745238");

Hour adjustment command
        ClockSet ("Clock1" , "Hour" , "23");

Minute adjustment command
        ClockSet ("Clock1" , "Minute" , "30");

Flat adjustment command
        ClockSet ("Clock1" , "Flat" , "1");

Center adjustment command
        ClockSet ("Clock1" , "Center" , "50");

NoSecond adjustment command
        ClockSet ("Clock1" , "NoSecond" , "0");

NoBackGround adjustment command
        ClockSet ("Clock1" , "NoBackGround" , "1");

NoHands adjustment command
        ClockSet ("Clock1" , "NoHands" , "0");

NoTicks adjustment command
        ClockSet ("Clock1" , "NoTicks" , "1");

OnDown adjustment command
        ClockSet ("Clock1" , "OnDown" , "Clock2_OnDown.c");
```

```
OnUp adjustment command
        ClockSet ("Clock1" , "OnUp" , "Clock 1_OnUp.c");
```

7.4GaugeSet ( )

Explanation

It is the command that regulates the parameter settings of the Gauge object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

Example Code

```
Visible adjustment command
        GaugeSet ("Gauge1" , "Visible" , "1");

Left adjustment command
        GaugeSet ("Gauge1" , "Left" , "10");

Top adjustment command
        GaugeSet ("Gauge1" , "Top" , "255");

Radius adjustment command
        GaugeSet ("Gauge1" , "Radius" , "35");

Color adjustment command
        GaugeSet ("Gauge1" , "Color" , "255");
```

```
Press_Color adjustment command
        GaugeSet ("Gauge1" , "Press_Color" , "1745238");

Value adjustment command
        GaugeSet ("Gauge1" , "Value" , "100");

Range adjustment command
        GaugeSet ("Gauge1" , "Range" , "30");

Major_Count adjustment command
        GaugeSet ("Gauge1" , "Major_Count" , "10");

Minor_Count adjustment command
        GaugeSet ("Gauge1" , "Minor_Count" , "5");

Flat adjustment command
        GaugeSet ("Gauge1" , "Flat" , "1");

Center adjustment command
        GaugeSet ("Gauge1" , "Center" , "50");

NoBackGround adjustment command
        GaugeSet ("Gauge1" , "NoBackGround" , "1");

NoHands adjustment command
        GaugeSet ("Gauge1" , "NoHands" , "0");

NoTicks adjustment command
        GaugeSet ("Gauge1" , "NoTicks" , "1");

OnDown adjustment command        GaugeSet ("Gauge1" , "OnDown" ,
"Gauge2_OnDown.c");

OnUp adjustment command
        GaugeSet ("Gauge1" , "OnUp" , "Gauge1_OnUp.c");
```

7.5DialSet ( )

Explanation

It is the command that regulates the parameter settings of the Dial object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

Visible adjustment command

```
GaugeSet ("Gauge1" , "Visible" , "1");
```

Left adjustment command

```
GaugeSet ("Gauge1" , "Left" , "10");
```

Top adjustment command

```
GaugeSet ("Gauge1" , "Top" , "255");
```

Radius adjustment command

```
GaugeSet ("Gauge1" , "Radius" , "35");
```

Color adjustment command

```
GaugeSet ("Gauge1" , "Color" , "255");
```

Press_Color adjustment command

```
GaugeSet ("Gauge1" , "Press_Color" , "1745238");
```

Value adjustment command

```
GaugeSet ("Gauge1" , "Value" , "100");
```

BackGround_Color adjustment command

```
GaugeSet ("Gauge1" , "BackGround_Color" , "65280");
```

Flat adjustment command

```
GaugeSet ("Gauge1" , "Flat" , "1");
```

Center adjustment command

```
GaugeSet ("Gauge1" , "Center" , "50");
```

OnDown adjustment command

```
GaugeSet ("Gauge1" , "OnDown" , "Gauge2_OnDown.c");
```

OnUp adjustment command

```
GaugeSet ("Gauge1" , "OnUp" , "Gauge1_OnUp.c");
```

7.6SliderSet ( )

Explanation

It is the command that regulates the parameter settings of the Slider object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

```
Visible adjustment command
        SliderSet  ("Slider1" , "Visible" , "1");

Left adjustment command
        SliderSet  ("Slider1" , "Left" , "10");

Top adjustment command
        SliderSet  ("Slider1" , "Top" , "255");

Width adjustment command
        SliderSet  ("Slider1" , "Width" , "90");


Height adjustment command
        SliderSet  ("Slider1" , "Height" , "70");

Color adjustment command
        SliderSet  ("Slider1" , "Color" , "255");

Press_Color adjustment command
        SliderSet  ("Slider1" , "Press_Color" , "1745238");
```

```
Thumb_Color adjustment command
        SliderSet  ("Slider1" , "Thumb_Color" , "65280");

BackGround_Color adjustment command
        SliderSet  ("Slider1" , "BackGround_Color" , "1458269");

Range adjustment command
        SliderSet  ("Slider1" , "Range" , "100");

Value adjustment command
        SliderSet  ("Slider1" , "Value" , "10");

Flat adjustment command
        SliderSet  ("Slider1" , "Flat" , "1");

OnDown adjustment command
        SliderSet  ("Slider1" , "OnDown" , "Slider2_OnDown.c");

OnUp adjustment command
        SliderSet  ("Slider1" , "OnUp" , "Slider1_OnUp.c");
```

7.7LabelSet ( )

Explanation

Label nesnesinin parametre ayarlarını düzenleyen komuttur.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

```
Visible adjustment command
        LabelSet ("ELabel1" , "Visible" , "1");

Left adjustment command
        LabelSet ("ELabel1" , "Left" , "10");

Top adjustment command
        LabelSet ("ELabel1" , "Top" , "255");

FontHandle adjustment command
        LabelSet ("ELabel1" , "FontHandle" , "23");

FontName adjustment command
        LabelSet ("ELabel1" , "FontName" , "Roboto");

Font_Color adjustment command
        LabelSet ("ELabel1" , "Font_Color" , "16777215");

Caption adjustment command
        LabelSet ("ELabel1" , "Caption" , "Button");

Center adjustment command
        LabelSet ("ELabel1" , "Caption" , "Center");
```

7.8 ToggleSet ( )

Explanation

Toggle is the command that adjusts the parameter settings of its object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

```
Visible adjustment command
        ToggleSet ("Toggle1" , "Visible" , "1");

Left adjustment command
        ToggleSet ("Toggle1" , "Left" , "10");

Top adjustment command
        ToggleSet ("Toggle1" , "Top" , "255");

Width adjustment command
        ToggleSet ("Toggle1" , "Width" , "90");

State adjustment command
        ToggleSet ("Toggle1" , "State " , "1");

StateOffCaption adjustment command
        ToggleSet ("Toggle1" , "StateOffCaption " , "No");


StateOnCaption adjustment command
        ToggleSet ("Toggle1" , "StateOnCaption " , "Yes");

Color adjustment command
        ToggleSet ("Toggle1" , "Color" , "255");

Press_Color adjustment command
        ToggleSet ("Toggle1" , "Press_Color" , "1745238");

BackGround_Color adjustment command
        ToggleSet ("Toggle1" , "BackGround_Color " , "65280");

Flat adjustment command
        ToggleSet ("Toggle1" , "Flat" , "1");
```

```
OnDown adjustment command
        ToggleSet ("Toggle1" , "OnDown" , "Toggle2_OnDown.c");

OnUp adjustment command
        ToggleSet ("Toggle1" , "OnUp" , "Toggle1_OnUp.c");
```

7.9ScrollBarSet ( )

Explanation

It is the command that regulates the parameter settings of the Scroll Bar object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

```
Visible adjustment command
        ScrollBarSet ("ScrollBar1" , "Visible" , "1");

Left adjustment command
        ScrollBarSet ("ScrollBar1" , "Left" , "10");

Top adjustment command
        ScrollBarSet ("ScrollBar1" , "Top" , "255");

Width adjustment command
        ScrollBarSet ("ScrollBar1" , "Width" , "90");

Height adjustment command
        ScrollBarSet ("ScrollBar1" , "Height" , "70");
```

```
Color adjustment command
        ScrollBarSet ("ScrollBar1" , "Color" , "255");

Press_Color adjustment command
        ScrollBarSet ("ScrollBar1" , "Press_Color" , "1745238");

Range adjustment command
        ScrollBarSet ("ScrollBar1" , "Range" , "100");

Size adjustment command
        ScrollBarSet ("ScrollBar1" , "Size" , "20");

Value adjustment command
        ScrollBarSet ("ScrollBar1" , "Value" , "50");

BackGround_Color adjustment command
        ScrollBarSet ("ScrollBar1" , "BackGround_Color" , "1458269");

Flat adjustment command
        ScrollBarSet ("ScrollBar1" , "Flat" , "1");

OnDown adjustment command
        ScrollBarSet ("ScrollBar1" , "OnDown" ,
      "ScrollBar_OnDown.c");

OnUp adjustment command
        ScrollBarSet ("ScrollBar1" , "OnUp" ,
       "ScrollBar_OnUp.c");
```

7.10 ProgressBarSet ( )

Explanation

It is the command that regulates the parameter settings of the Progress Bar object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

Example Code

```
Visible adjustment command
        ProgressBarSet  ("ProgressBar1" , "Visible" , "1");

Left adjustment command
        ProgressBarSet  ("ProgressBar1" , "Left" , "10");

Top adjustment command
        ProgressBarSet  ("ProgressBar1" , "Top" , "255");

Width adjustment command
        ProgressBarSet  ("ProgressBar1" , "Width" , "90");

Height adjustment command
        ProgressBarSet  ("ProgressBar1" , "Height" , "70");

Color adjustment command
        ProgressBarSet  ("ProgressBar1" , "Color" , "255");

Press_Color adjustment command
        ProgressBarSet  ("ProgressBar1" , "Press_Color" , "1745238");

Range adjustment command
        ProgressBarSet  ("ProgressBar1" , "Range" , "100");

Value adjustment command
        ProgressBarSet  ("ProgressBar1" , "Value" , "50");

BackGround_Color adjustment command
        ProgressBarSet  ("ProgressBar1" , "BackGround_Color" , "1458269");

Flat adjustment command
        ProgressBarSet  ("ProgressBar1" , "Flat" , "1");
```

```
OnDown adjustment command
        ProgressBarSet  ("ProgressBar1" , "OnDown" ,
      "ProgressBar_OnDown.c");

OnUp adjustment command
        ProgressBarSet  ("ProgressBar1" , "OnUp" ,
       "ProgressBar_OnUp.c");
```

7.11 KeySet ( )

Explanation

It is the command that regulates the parameter settings of the Key object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

Example Code

```
Visible adjustment command
        KeySet  ("Key1" , "Visible" , "1");

Left adjustment command
        KeySet  ("Key4" , "Left" , "10");

Top adjustment command
        KeySet  ("Key7" , "Top" , "255");

Width adjustment command
        KeySet  ("Key1" , "Width" , "90");
```

```
Height adjustment command
        KeySet  ("Key4" , "Height" , "70");

Color adjustment command
        KeySet  ("Key7" , "Color" , "255");

ColorTo adjustment command
        KeySet  ("Key1" , "ColorTo" , "65280");

Press_Color adjustment command
        KeySet  ("Key4" , "Press_Color" , "1745238");

Max_Lenght adjustment command
        KeySet  ("Key7" , "Max_Lenght " , "1458269");

FontHandle adjustment command
        KeySet  ("Key1" , "FontHandle" , "23");

FontName adjustment command
        KeySet  ("Key4" , "FontName" , "Roboto");

Font_Color adjustment command
        KeySet  ("Key7" , "Font_Color" , "16777215");

Caption adjustment command
        KeySet  ("Key4" , "Caption" , "Button");

OnDown adjustment command
        KeySet  ("Key7" , "OnDown" , "Key7_OnDown.c");

OnUp adjustment command
        KeySet  ("Key7" , "OnUp" , "Key7_OnUp.c");
```

7.12 ButtonSet ( )

Explanation

It is the command that regulates the parameter settings of the button object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

Example Code

```
Visible adjustment command
        ButtonSet ("EButton1" , "Visible" , "1");

Left adjustment command
        ButtonSet ("EButton4" , "Left" , "10");

Top adjustment command
        ButtonSet ("EButton7" , "Top" , "255");

Width adjustment command
        ButtonSet ("EButton1" , "Width" , "90");

Height adjustment command
        ButtonSet ("EButton4" , "Height" , "70");

Color adjustment command
        ButtonSet ("EButton7" , "Color" , "255");

ColorTo adjustment command
        ButtonSet ("EButton1" , "ColorTo" , "65280");

Press_Color adjustment command
        ButtonSet ("EButton4" , "Press_Color" , "1745238");

Press_ColorTo adjustment command
        ButtonSet ("EButton7" , "Press_ColorTo" , "1458269");

FontHandle adjustment command
        ButtonSet ("EButton1" , "FontHandle" , "23");

FontName adjustment command
        ButtonSet ("EButton4" , "FontName" , "Roboto");

Font_Color adjustment command
        ButtonSet ("EButton7" , "Font_Color" , "16777215");
```

```
Gradient adjustment command
        ButtonSet ("EButton1" , "Gradient" , "3");

Caption adjustment command
        ButtonSet ("EButton4" , "Caption" , "Button");

OnDown adjustment command
        ButtonSet ("EButton7" , "OnDown" , "EButton2_OnDown.c");

OnUp adjustment command
        ButtonSet ("EButton7" , "OnUp" , "EButton1_OnUp.c");
```

7.13 ImageSet ( )

Explanation

It is the command that regulates the parameter settings of the Image object.

Function

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

Parameter	Explanation
name	Name of the object
type	The parameter of the object to be changed
value	The new value of the parameter to be changed

Example Code

```
Visible adjustment command
        ImageSet ("EImage1" , "Visible" , "1");

Left adjustment command
        ImageSet ("EImage1" , "Left" , "10");
```

```
Top adjustment command
        ImageSet ("EImage1" , "Top" , "255");

RotationAngle adjustment command
        ImageSet ("EImage1" , "RotationAngle" , "90");

RotationCenterLeft adjustment command
        ImageSet ("EImage1" , "RotationCenterLeft" , "400");

RotationCenterTop adjustment command
        ImageSet ("EImage1" , "RotationCenterTop" , "240");

ColorTo adjustment command
        ImageSet ("EImage1" , "ScaleX" , "65280");

ScaleY adjustment command
        ImageSet ("EImage1" , "ScaleY" , "1745238");

OnDown adjustment command
        ImageSet ("EImage1" , "OnDown" , "EImage2_OnDown.c");

OnUp adjustment command
        ImageSet ("EImage1" , "OnUp" , "EImage1_OnUp.c");
```

7.14 LocalStdVarGet ( )

Explanation

It is a local string data reading command.

Function

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

Parameter	Explanation
name	Name of the object
value	The new value of the parameter to be changed

Example Code

```
#include "stdio.h"

#include "stk.h"

LocalStdVarGet("EVariable1" , “string”);  // Get the String variable that is local
```

7.15 LocalStdVarSet ( )

Explanation

Local string is the value assignment command.

Function

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

Parameter	Explanation
name	Name of the object
value	The new value of the parameter to be changed

Example Code

```
#include "stdio.h"

#include "stk.h"

LocalStdVarSet("EVariable1" , “string”);  // Set the String variable that is local
```

7.16 LocalIntVarGet ( )

Explanation

Local integer is a data reading command.

Function

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

Parameter	Explanation
name	Name of the object
value	The new value of the parameter to be changed

Explanation

```
#include "stdio.h"

#include "stk.h"

int value;

LocalIntVarGet("EVariable2", &value);  //  Get the Integer variable that is local
```

7.17 LocalIntVarSet ( )

Explanation

Local integer value assignment command.

Function

void LocalIntVarSet(unsigned char *name , int value)

Parameter	Explanation
name	Name of the object
value	The new value of the parameter to be changed

Explanation

```
#include "stdio.h"

#include "stk.h"

int value;

LocalIntVarSet("EVariable2", value);  //  Set the Integer variable that is local
```

7.18 GlobalStdVarGet ( )

Explanation

It is a global string data reading command.

Function

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

Parameter	Explanation
name	Name of the object
value	The new value of the parameter to be changed

Example Code

```
#include "stdio.h"

#include "stk.h"

GlobalStdVarGet("EVariable1" , “string”); // Get the String variable that is
global
```

7.19 GlobalStdVarSet ( )

Example

Global string is the value assignment command.

Function

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

Parameter	Explanation
Name	Name of the object
value	The new value of the parameter to be changed

Example Code

```
#include "stdio.h"

#include "stk.h"

GlobalStdVarSet("EVariable1" , “string”); // Setting the global variable String
```

7.20 GlobalIntVarGet ( )

Explanation

It is a global integer data reading command.

Function

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

Parameter	Explanation
name	Name of the object
value	The new value of the parameter to be changed

Example Code

```
#include "stdio.h"

#include "stk.h"

int value;

GlobalIntVarGet("EVariable2", &value);  // Get the Integer variable that is global
```

7.21 GlobalIntVarSet ( )

Explanation

It is a global integer value assignment command.

Function

void GlobalIntVarSet(unsigned char *name , int value)

Parameter	Explanation
name	Name of the object
value	The new value of the parameter to be changed

Example Code

```
#include "stdio.h"

#include "stk.h"

int value;

GlobalIntVarSet("EVariable2", value);  // Set the Integer variable that is global
```

7.22 halGpioSet ( )

Explanation

There are 4 different GPIO pins for users to use according to their wishes. This is the command that can optionally set pins to INPUT or OUTPUT.

Function

void halGpioSet(unsigned char *pin, unsigned char *value)

Parameter	Explanation
pin	Name of the pin
value	Write / Read

Example Code

```
#include "stdio.h"

#include "stk.h"

halGpioSet("GPIO_OUT_1" , "Write");      // Sets pin 1 as input

halGpioSet("GPIO_OUT_2" , "Read");      // Sets pin 2 as output
```

7.23 uartDataGet ( )

Explanation

Operations can be made on the AIRHMI Editor screen according to the data from UART. It is a command to receive data from UART in the code order.

Function

void uartDataGet(char *value , int *uartsize)

Parameter	Explanation
value	The string where the data from UART will be stored
uartsize	Size of data from UART

Example Code

```
#include "stdio.h"

#include "stk.h"

char uartData[30];                    // The string where the data from the UART
will be stored
int uartsize;                            //  The size of the data from Uart

uartDataGet(uartData , &uartsize);   //  Reading data from Uart
```

7.24 ChangeScreenSet ( )

Explanation

It is a command that allows to switch between the screens in the code.

Function

void ChangeScreenSet(unsigned char *value)

Parameter	Explanation
value	Name of the screen to be switched

Example Code

```
#include "stdio.h"

#include "stk.h"

ChangeScreenSet ("Nameofthescreentobeswitched");
```

7.25 DrawScreen( )

Explanation

The page refresh command.

Function

void DrawScreen()

Example Code

```
#include "stdio.h"

#include "stk.h"

DrawScreen();        // Page refresh command
```

7.26 dateSet ( )

Explanation

It is the command to refresh / set the date data in RTC.

Function

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

Parameter	Explanation
days	Indicates days
months	Indicates months
years	Indicates years

Example Code

```
#include "stdio.h"

#include "stk.h"

unsigned char day, month, year;     // Example Date-Time variables in the code
index

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

dateSet(&day, &month , &year);     // Setting Date data from RTC
```

7.27 timeSet ( )

Explanation

It is the command to refresh / adjust clock data in RTC.

Function

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

Parameter	Explanation
hours	Indicates hours
mins	Indicates minutes

Example Code

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

unsigned char hour, min;          // Example Date-Time variables in the code index

hour = 16;
min  = 30;

timeSet(&hour , &min);            // Renew / set Clock data in RTC
```

7.28 dateGet ( )

Explanation

It is the command to get the date data from RTC.

Function

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

Parameter	Explanation
days	Indicates days
months	Indicates months
years	Indicates years

Example Code

```
#include "stdio.h"

#include "stk.h"

unsigned char day, month, year;      // Example Date-Time variables in the code
index

dateGet(&day, &month , &year);     // Retrieving Date data from RTC
```

7.29 timeGet ( )

Explanation

It is the command to get clock data from RTC.

Function

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

Parameter	Explanation
hours	Indicates hours
mins	Indicates minutes

Example Code

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

unsigned char hour, min;          // Example Date-Time variables in the code index
timeSet(&hour , &min);           // Reading Clock data in RTC
```

7.30 AudioPlay( )

Explanation

The user can perform the playback process with this function after adding the sound file of the request to play to the project via AirHMI Editor.

Function

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

Parameter	Explanation
audioname	Name of the audio file
volume	Sound volume

Example Code

```
#include "stdio.h"

#include "stk.h"

int volume;   // Sound volume

AudioPlay("NameoftheSoundFile" , &volume );
```

7.31 PlaySound ( )

Explanation

There are ready warning sounds installed in the device. With the PlaySound function, it can be selected from 5 different warning sounds prepared before by typing in the int option section such as 1,2, .., 5.

Function

void PlaySound(int option)

Parameter	Explanation
option	You can choose from 1-5.

Example Code

```
#include "stdio.h"

#include "stk.h"

PlaySound(1)
```

7.32 ScreenColorSet ( )

Explanation

It is the command that allows to set the background color of the page.

Function

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

Parameter	Explanation
name	Name of the page whose background color will be changed
value	Background color code of the page (int)

Example Code

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

ScreenColorSet("Screen1", "0");   // The background color of Screen1 screen has been set to black
```

7.33 LCDsleep ( )

Explanation

It is the command that puts the LCD screen to sleep mode when the image is not required on the LCD screen while the project is actively working.

Function

void LCDsleep()

Example Code

```
#include "stdio.h"

#include "stk.h"

LCDsleep();             //  LCD sleep mode activated
```

7.34 LCDwakeup ( )

Explanation

It is the command that restores the image on the LCD screen in sleep state.

Function

void LCDwakeup()

Example Code

```
#include "stdio.h"

#include "stk.h"

LCDwakeup();            //  LCD sleep mode deactivated
```

7.35 SoundFileState( )

Explanation

It is a command that gives information about the volume of the sound file.

Function

void SoundFileState(int *sound_state)

Parameter	Explanation
sound_state	Sound file preset volume

Example Code

```
#include "stdio.h"
#include "stk.h"
int volume;

SoundFileState(&volume);        // Get the data of Adjusted Sound Level
```

7.36 File_write ( )

Explanation

It is a command to write to Flash.

Function

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

Parameter	Explanation
name	Name of the TobeUsed.txt
buffer	The name of the string array
size	The size of the array to be written
nmemb	1

Example Code

```
#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);

// A file named Message.txt was created in Flash and x_file data was written in
this file as much as the size of sizeof (x_file).
```

7.37 File_read( )

Explanation

Reading command from Flash.

Function

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

Parameter	Explanation
name	Name of the Tobeused .txt
buffer	The name of the string array
size	Reading Size
nmemb	1

Example Code

```
#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);

// From the data in a file named Message.txt in Flash, sizeof (x_file) was read to
the x_file variable.
```

7.38 File_size( )

Explanation

It is the command to learn the file size.

Function

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

Parameter	Explanation
name	The name of the file to be used
size	An integer variable in which the file size will be kept

Example Code

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

int f_size;

File_size("Message.txt" , &f_size);     // Get the size data of the Message.txt
file in Flash.
```

7.39 RotateScreen ( )

Explanation

It is the command that allows the screen to be horizontal or vertical.

Function

void RotateScreen(int option)

Parameter	Explanation
option	Whether the screen is horizontal or vertical

Example Code

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

RotateScreen(1);    // It is the command to adjust the working area of the LCD screen to the vertical position

RotateScreen(0);    // It is the command to adjust the working area of the LCD screen to the horizontal position
```

7.40 RoleSet ( )

Explanation

It is the command that controls the on-off status of relays.

Function

void RoleSet(char *name , char *value)

Parameter	Explanation
name	Relays name
value	On-off status of the relay

Example Code

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

RoleSet("Relay1" , "1");     // The Relay1 Activated

RoleSet("Relay2" , "0");     // The Relay2 Deactivated
```

7.41 RS485Set ( )

Explanation

It is a command to send and receive data with RS485 communication protocol.

Function

void RS485Set(char *name, char *value)

Parameter	Explanation
name	Determines whether data will be sent or received
value	Data to be sent or received

Example Code - 1

```
// Assigning data from RS485 to the receive variable

#include "string.h"
#include "stdio.h"
#include "stdlib.h"
#include "stk.h"

char receive[100];

memset(receive , 0x00 , sizeof(receive));

RS485Set("Receive" , receive);
```

Example Code – 2

```
// Send data in “send variable” via RS485

#include "string.h"
#include "stdio.h"
#include "stdlib.h"
#include "stk.h"

char send[100];

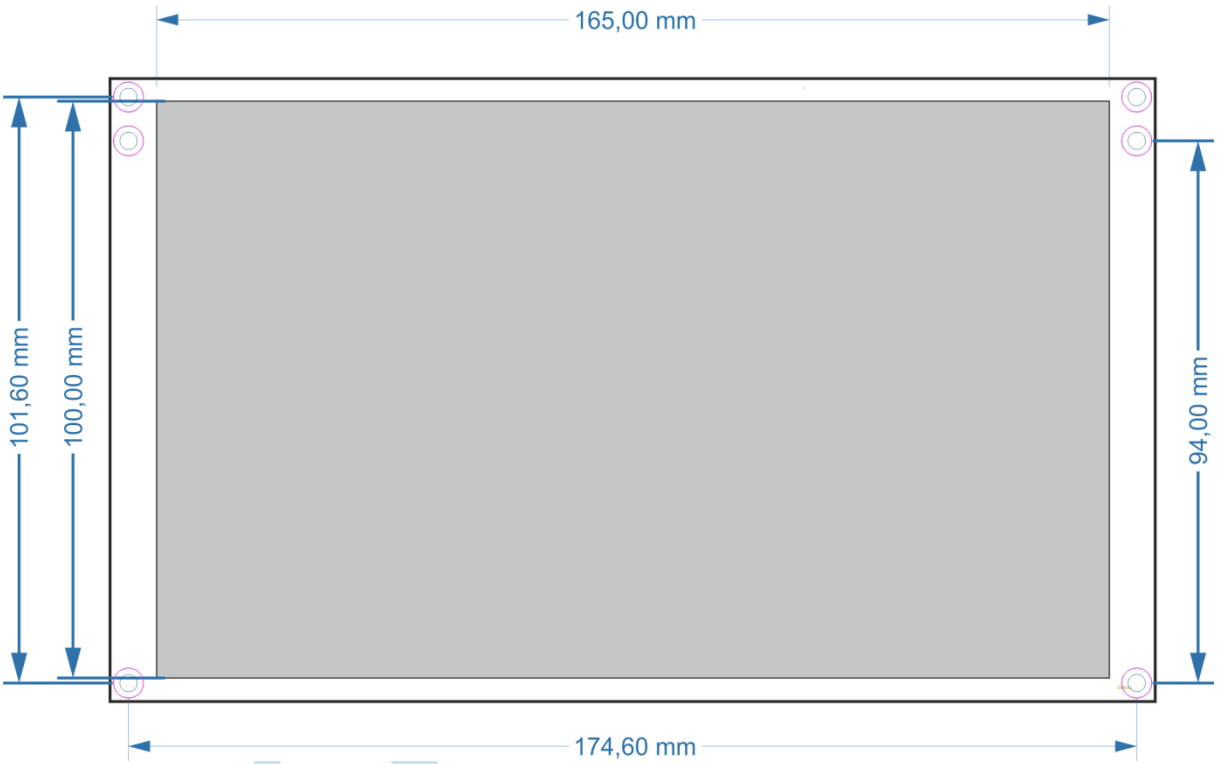
memset(send, 0x00 , sizeof(send));

sprintf(send , "%s" , "Hello World !!!");

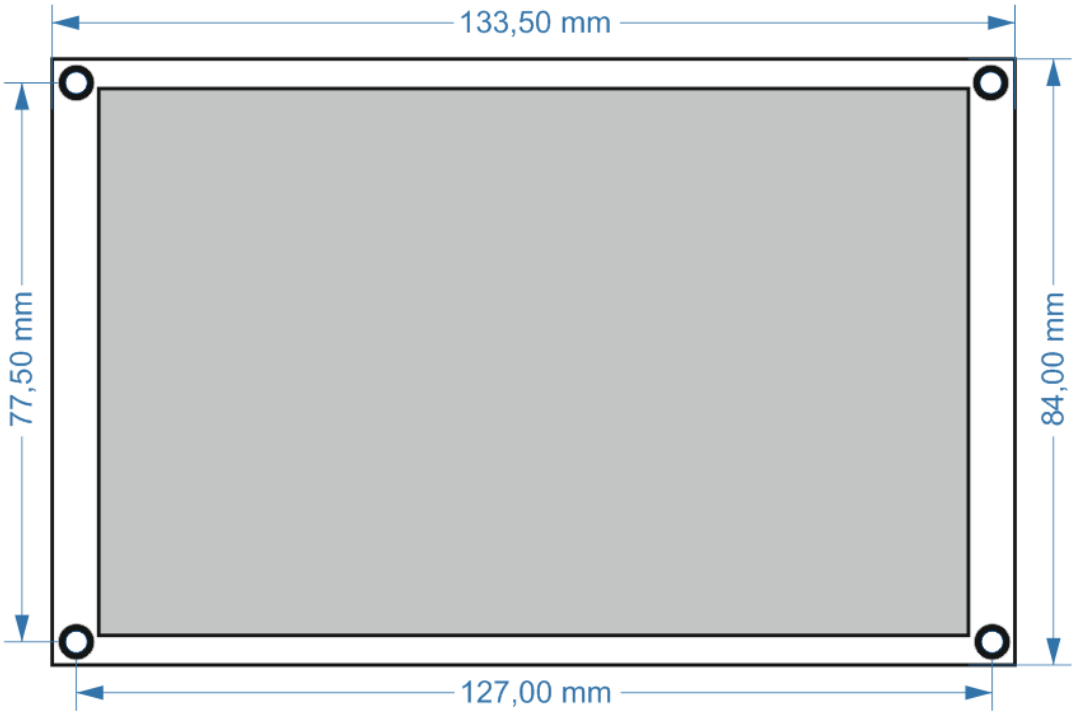
RS485Set("Send" , send);
```

8. Products Dimensions

8.1 BT800X480S70_RT



8.2BT800X480S50_RT



8.3BT480X272S43_RT