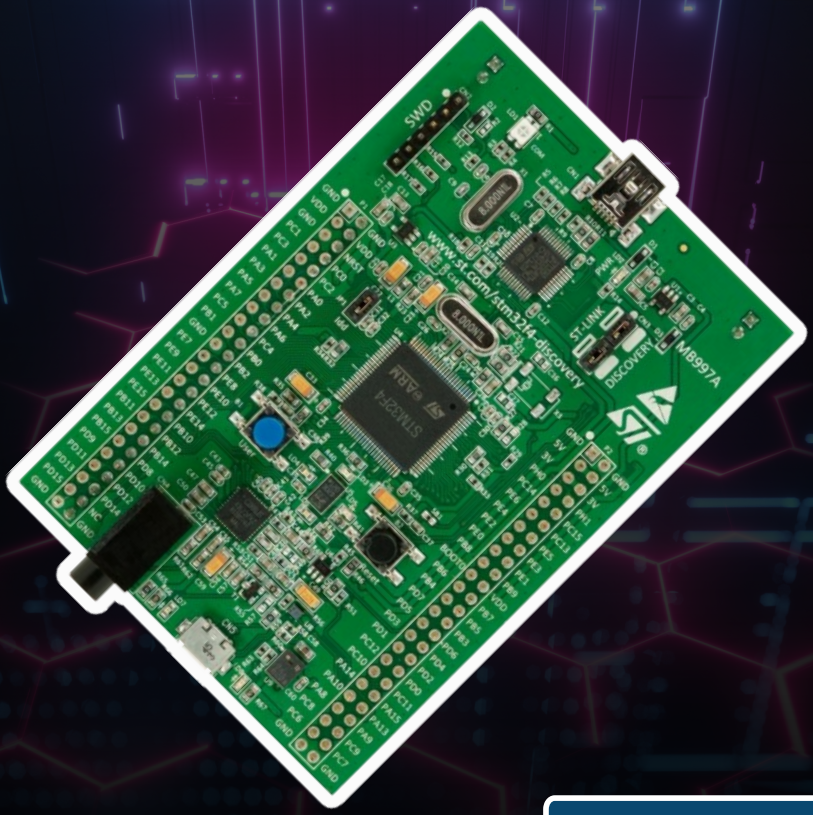
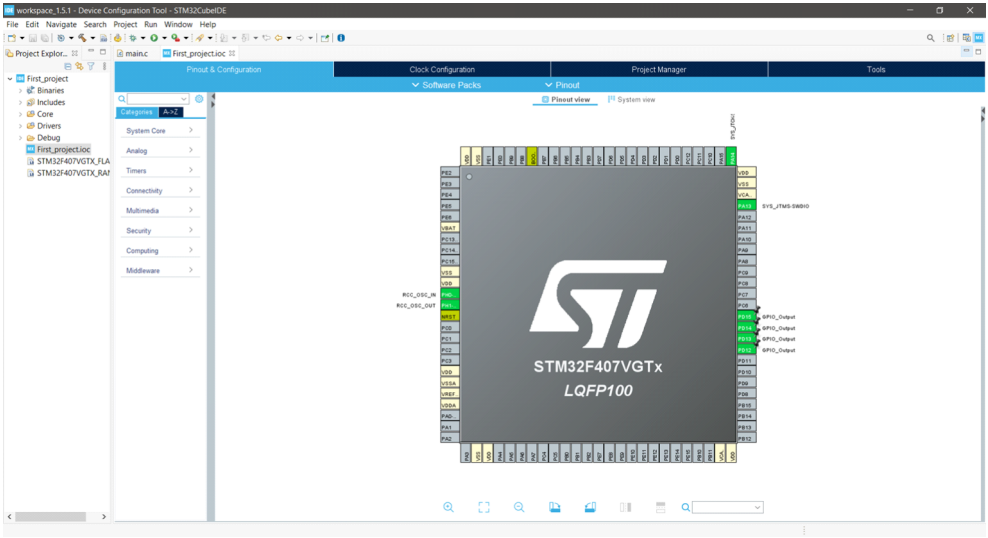
STM32

STM32F407VG Discovery Süper Başlangıç Kitabı



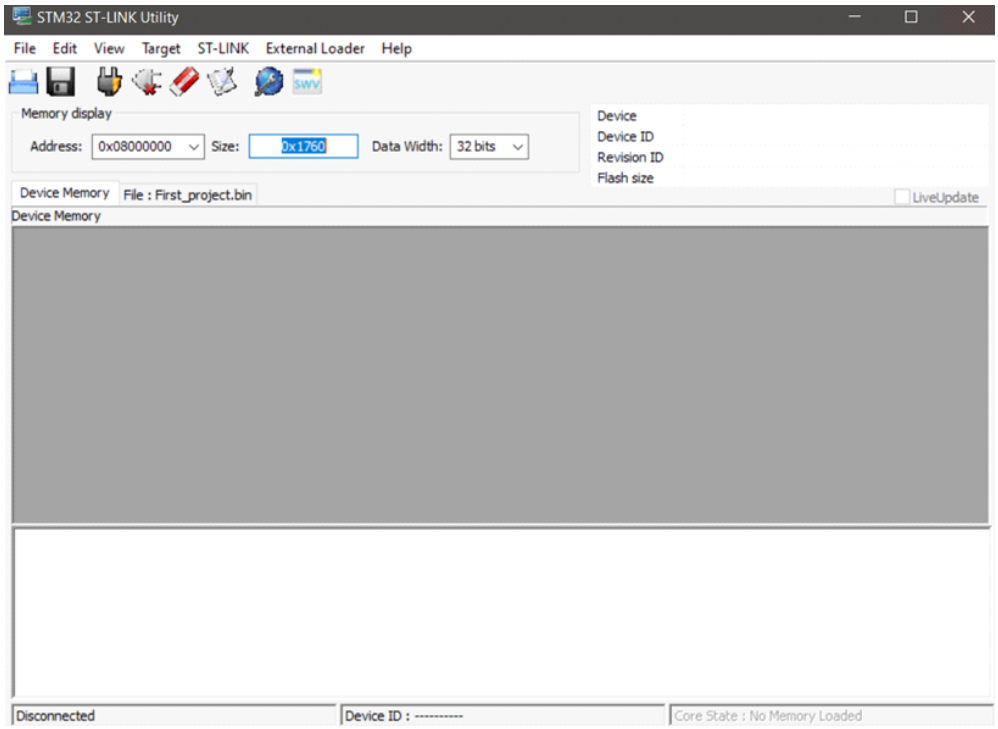
robotistan



STM32CubeIDE tüm özelliklerin tek geliştirme ortamında bulunduğu bir IDE dir.

STM32CubeMX'in tüm özellikleri, TrueSTUDIO derleyiciyi ve diğer IDE araçları ile STM32CubeIDE altında birleşti. İhtiyacınız olan her şey artık tek bir pakette. Üstelik tüm işletim sistemlerinde (Windows, Linux, macOS) ve ücretsiz.

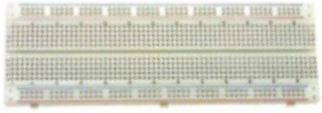
Farklı bir derleyici ile çalışmayı tercih edenler için ise STM32CubeMX halen aktif. Hem MX hem de IDE; aynı kod tabanı, özellikler ve güncellemeler ile desteklenmeye devam edecek. Bu noktada önemli bir hatırlatma yapmak gerekirse: Keil'ın geliştirme ortamı MDK'yı; sadece IoT değil, tüm sektör ve uygulamalarda çok yaygın hale gelen ST Arm Cortex M0/M0+ tabanlı mikrodenetleyici aileleri (STM32G0/L0/F0) için kod boyutu sınırlaması olmadan ücretsiz olarak kulanabilirsiniz. Biz bu serüvende pin durumlarımızı ayarlayabilmek ve kodlarımızı yazabilmek için STM32CubeIDE'yi kullanacağız.



Kodlarımızı yazıp derledikten sonra programımızı kartımıza yüklememiz için STM32 ST-LINK Utility programını kullanıyoruz.

Bu program silme, doğrulama, yazma, okuma gibi işlemleri yapar ve hex dosyalarını mikrodenetleyiciye yükler.

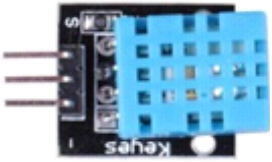
STM32F407 Discovery Geliştirme Kartı
STM32F407 kısaca 168 Mhz saat frekansına sahip 32-bit'lik bir mikrodenetleyicidir. Özellikle DSP uygulamalarında , içerisinde bir çok alt sistemler olan yapıların ana beyni olarak kullanılmaktadır. Üzerinde STM32F4VG çekirdeğini barındırmaktadır. 3-eksen ivme ölçer , dahili ses girişi ve buna bağlı dijital analog dönüştürücü , 4 adet programlanabilir LED , 2 adet buton barındırır.



Breadboard Nedir?
Breadboard üzerinde devrelerimizi test ettiğimiz araçtır. Kurduğumuz devreleri birbirlerine lehimlemeden kolaylıkla test etmemizi sağlar. Tasarladığımız devreleri baskı devre veya delikli plaketler üzerine aktarmadan önce denememize olanak sağlar.

RTC Saat Modülü
DS1302 gerçek zamanlı saat devresi entegresi olup saat bilgisini anlık ve sürekli olarak okuyabileceğiniz çok kullanışlı bir modüldür.

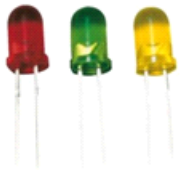
Kart üzerinde CR2032 pil konnektörü bulunur.



Dht 11 Sıcaklı Ve Nem Sensörü
DHT11 sıcaklık ve nem algılayıcı kalibre edilmiş dijital sinyal çıkışı veren gelişmiş bir algılayıcı birimidir. Yüksek güvenilirliktedir ve uzun dönem çalışmalarda dengelidir. 0 ile 50°C arasında 2°C hata payı ile sıcaklık ölçer.

Buton
Buton, operatörün yaptığı işlemi (basma, çevirme gibi) makineye aktaran, operatörün daha doğru kararlar vermesine yardımcı olup ve sistemin kontrolünü elinde tutmasını sağlayan elektriksel cihazdır.

ne555
Birçok elektronik projesinde sıkça kare dalga üreteci olarak kullanılan popüler zamanlayıcı (timer) entegresidir. DIP8 kılıftadır. 555 entegresi Osilasyon, zaman gecikmesi ve darbe sinyali üretmek için kullanılabilecek kararlı bir tümleşik devredir.
555 Timer (zamanlayıcı) entegresinin zamanlama aralığı mikrosaniyeler ile saatler arasındadır. Ayarlanabilir çıkış frekansı gibi özellikleri vardır.



Diyot LED
Işık yayan diyot (LED), adından da anlaşılacağı gibi enerji verildiği zaman görülebilir bir ışık yayan diyottur. Genel olarak kırmızı, sarı ve yeşil olmak üzere üç değişik renkte yapılırlar. Çalışma akımları 5 mA ile 50 mA arasındadır.

Potansiyometre
Potansiyometrenin özelliği kontrol edilebilir direnç olmasıdır. Elektroniğin temel elemanlarından biridir ve kontrol gerektiren devrelerin birçoğunda bulunmaktadır. Sembolü de normal bir direncin üzerine ok eklenmesiyle meydana gelir. Bunun sebebi de direnç değerinin anlık kontrol edilebildiğini göstermesidir.

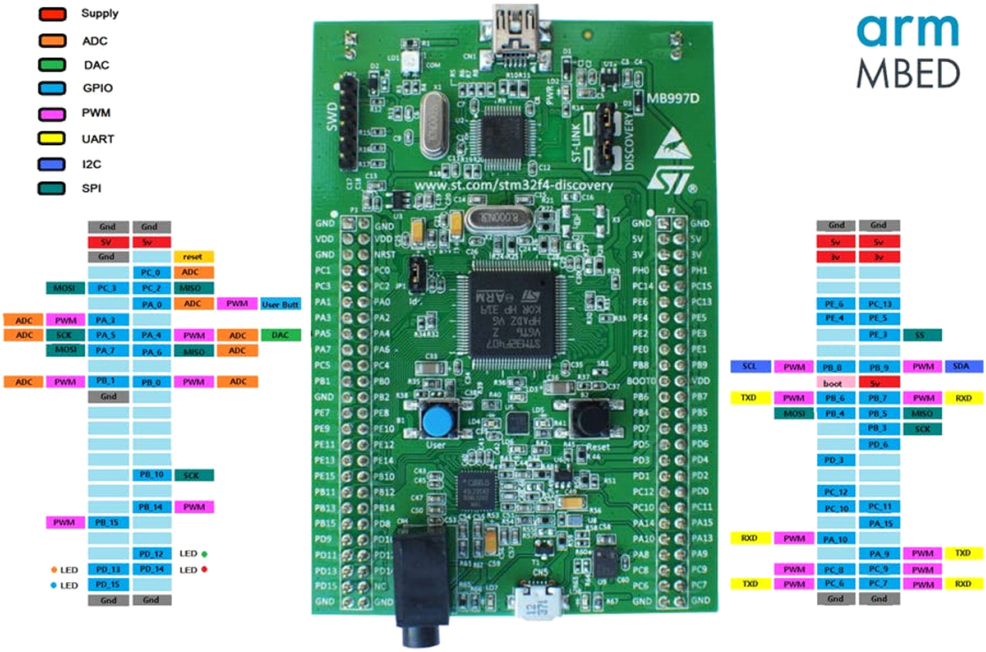
LDR
LDR (Light Dependent Resistor), Türkçede "Işığa Bağımlı Direnç" anlamına gelmektedir. Bir diğer adı da foto dirençtir. LDR her ne kadar bir direnç çeşidi olsa da aynı zamanda pasif bir sensördür. LDR'ler bulundukları devrelerde değişen direnç değerleri ile bir çıkış sağlarlar fakat bu işlemi dış ortamdan aldıkları fiziksel bir değişim ile gerçekleştirdiklerinden dolayı bir sensör görevi görmüş olurlar.

Sg90 Servo Motor
Tower Pro SG90 küçük mekanizmalarınız için ideal bir servo motordur. Her marka uzaktan kumanda alıcılarına tam uyumlu olup RC araclarınızda kullanabilirsiniz. Bunun yanı sıra birçok mikrodenetleyiciden alabileceğiniz PWM sinyali ile kendi yapt1ğınız robot projelerinizde de kolaylıkla kullanabilirsiniz.

Buzzer
Buzzer; mekanik, elektromekanik ya da piezoelektrik prensiplerine bağlı olarak çalışan işitsel ikaz cihazı çeşididir. Kullanım alanları oldukça fazla olan buzzerlar, genel itibarıyla piezoelektrik prensibiyle çalışmaktadırlar. Buzzerlar, kullanım alanlarına da bağlı olarak alarm, zamanlayıcı, onaylama cevap ikazı gibi işlevlerde kullanılabilmektedirler.

Unutmayın ki her proje led yakmakla başlar. Bizde bu durumu baz alarak yeni kartımızla ilk ledimizi yakalım.İlk yapacağımız proje kartımızın üzerinde bulunan dahili ledleri yakıp söndürmek olucaktır. Kartımızın üzerinde dahili olarak 4 adet led bulunmaktadır. Zaten kartımıza ilk enerjiyi verdiğimizde bu 4 ledin minik bir gösterisi ile karşılaşacağız.
Hadi başlayalım.

Gerekli Malzemeler
Stm32F407 Geliştirme Kartı
4 adet LED
Breadboard
Yeteri kadar jumper kablo (dişi-erkek)



Yukarıdaki pinout(pin dağılımı) tablosu, yazılımı CubeIDE ile yazacağımız için STM32F407 geliştirme kartının pinlerini bu arayüzde nasıl kullanacağımızı bilmemiz gerekiyor. Kartımızda dahili olan pinler şu şekildedir :
Yeşil Led PD12 , Turuncu Led PD13, Kırmızı Led PD14, Mavi Led PD12. Setimizle beraber gelen kablo aracılığıyla kartımızı bilgisayara bağlayı programlama işlemlerine geçebiliriz.

CubeIDE derleyicimizi açtığımızfa ilk olarak File=>New=>STM32 Project butonuna tıklayalım.

Açılan sayfada arama kısmına stm32f407vg yazıp çıkan denetleyicimize tıklayarak Next butonuna basalım.

Yeni açılan pencerede projemizin ismini belirleyeceğiz. Led yazmayı tercih ettim. Projemizin adını belirledikten sonra Next => Finish butonlarına basıyoruz. Gelen uyarıda CubeIDE arayüzünü açalım mı diyecek r . Yes butonuna basıyoruz.

Konfigürasyon sayfamız açıldığında yapmamız gereken ilk ayar programlama ayarıdır.
System Core => SYS seç kten sonar Debug kısmını Serial Wire Yapıyoruz. Daha Sonra
System Core =>RCC seçip osilatör ayarlarımızı yapacağız. HSE kısmını Crystal olarak seçiyoruz.

Dahili ledlerimizin bağlı olduğu pinler üzerinde işlem yapabilmemiz için pinleri output olarak belirlememiz gerekiyor.Sağ tarfta bulunan entegrede PD12,PD13,PD14,PD15 pinlerimizin üzerine tıklayıp GPIO_Output seçenekleriniz seçiyoruz. Pin konfigürasyon ayarlarımız bitti şimdi Clock konfigürasyon ayarlarımızı yapacağız.

Pinout & Configura on menusunun hemen sağ tara nda bulunan Clock Configura on menüsüne tıklıyoruz.

Input frequency kısmında 25 olan frekansımızı 8 yapıyoruz. Sistemin clock ayarınıda (System Clock Mux) kısmınıda HSE yapıyoruz. Bu sayede tüm birimlere 8MHz clock vermiş olduk.

Clock ayarlarımızda bitti şimdi projemizi kaydedip (File=> Save) kodlarımızı olurşturalım...

Kodlarımız oluştu peki bunları nasıl görüp değişiklikler yapacağız ?
Core=>Src=>main.c Bizim ana kaynak kodlarımız burda. Kart üzerindeki ledleri sürekli kontrol edebilmemiz için while döngüsünün olduğu yere yazacağız.

```
HAL_GPIO_WritePin(GPIOx, GPIO_Pin, Pinstate);
```

Bu komut ile çıkış olarak ayarlanan pinin durumunu lojik 0 veya 1 yaparız. 3 parametre alır bunlar sırayla port ,gpio pini ve pinin durumudur. 12 nolu pine bağlı ledimizin yanması için

```
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
```

Tek satırla aslında ledimizi yakmış olduk ama bir blink işlemi için bekleme süresi koyup ledimizi söndürüp tekrar bir bekleme süresi koymalıyız. Hadi yapalım !

```
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);  //Ledi Yak
HAL_Delay(1000);                           // 1 saniye bekle
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 0); // Ledi Söndür
HAL_Delay(1000);                          // 1 Saniye bekle
```

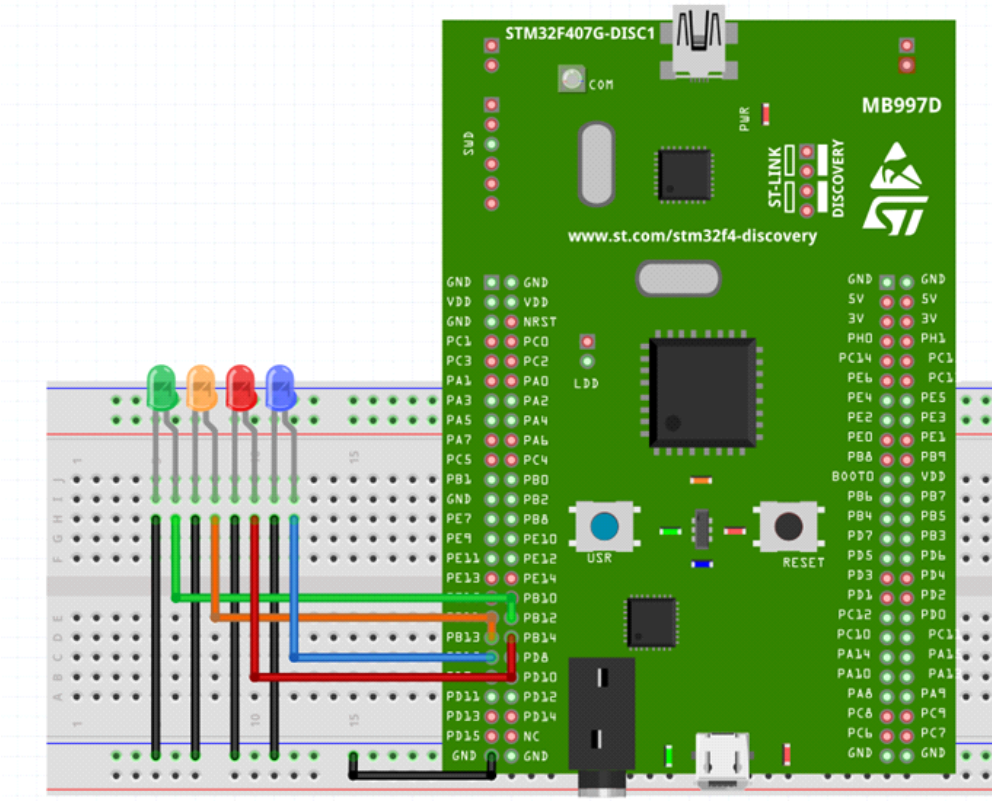
Kodumuzu derlemek için build işare ne tıklayalım. Kaydedilsin mi sorusuna Yes diyelim. Kodu kartımıza yüklememiz için proje yolunu biliyor olmamız gerek bunun için Led dosyamıza sağ tıklayıp proper es e tıklayalım. Loca on kısmında projemizin yolu yazılıdır bunu kopyalayalım. Blink için kodlarımızı yazdık ve derledik şimdi nasıl kartımıza yükleyeceğimizi öğrenelim.

STM32 ST-LINK programını açalım File => Open file seçeneklerine tıklayıp kopyaladığımız yolu yapıştıralım. Debug=> Led dosyamızı açalım. Kartımıza yüklemek için Program verify butonuna tıklayıp start butonuna tıklıyoruz.

Yükleme işlemi tamamlandı ledimizin yandığını gözlemleyelim. Tabiki bize bir led yetmeyecek r hemen dahili olarak kalan 3 ledide sırayla yakalım.

```
int t =250;  // sürekli olarak süreleri değiştirmek için bir
değişken atıyorum.

        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
        HAL_Delay(t);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 0);
        HAL_Delay(t);

        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 1);
        HAL_Delay(t);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 0);
        HAL_Delay(t);

        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 1);
        HAL_Delay(t);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 0);
        HAL_Delay(t);

        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, 1);
        HAL_Delay(t);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, 0);
        HAL_Delay(t);
```

Kodumuzu kaydedip STM32 ST_LINK programımızda tekrar aynı işlemleri tekrarlayarak kodu kartıma yüklüyorum. Artık tüm ledlerimin sırayla yandığını görebiliyorum. Setimizin içerisinden çıkan ledlerimizi aşağıdaki şekildeki gibi bağlayarak harici ve dahili ledlerin aynı anda nasıl yandığını gözlemleyebiliriz.



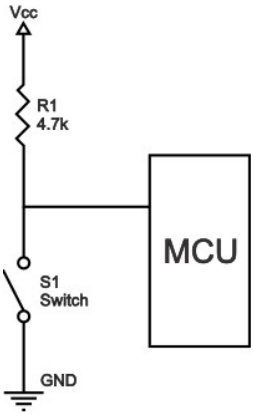
Led yakmayı öğrendik artık şimdi bir sonraki projemize geçelim.

Gerekli Malzemeler
Stm32F407 Geliştirme Kartı
1 adet Buton
1 adet 10K direnç
1 adet LED
Breadboard
Yeteri kadar jumper kablo (dişi-erkek)

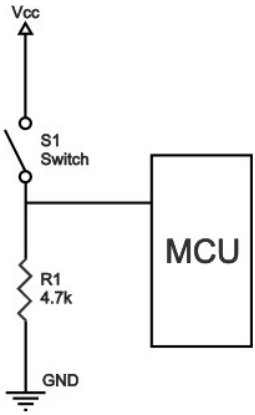
Bu projemizde herhangi bir gpio digital olarak nasıl okunur , gpio okuma fonksiyonu bize nasıl bir değer döndürür bunu göreceğiz. İlk olarak derleyicimizi açalım File=>New=>STM32 Project butonuna tıklayalım. Açılan sayfada arama kısmına stm32f407vg yazıp çıkan denetleyicimize tıklayarak Next butonuna basalım.

Proje ismimi Buton olarak belirliyorum siz istediğiniz bir ismi belirleyebilirsiniz. Pin Configuration sayfamız açıldı. System Core => SYS seçtikten sonar Debug kısmını Serial Wire yapıyoruz. Daha Sonra System Core =>RCC seçip osilatör ayarlarımızı yapacağız. HSE kısmını Crystal olarak seçiyoruz.

Buton için bir adet giriş pini ayarlamamız gerekli bunun için PA1 pinini GPIO_Input olarak ayarlıyoruz.



Pull-Up Direnç



Pull-Down Direnç

Bir mikrodenetleyicinin başlangıçta lojik-0 olarak atanan bir pini, lojik-0'dan lojik-1'e getirilmek istendiğinde pull-down direnç kullanılır. Pull-down dirençte yukarıdaki şekilde görüldüğü gibi mikrodenetleyicinin giriş (input) olarak atanan pini 4.7K'lık direnç üzerinden toprağa bağlanarak lojik-0 (low) değerini alır. Buton ile toprak arasındaki direncin kullanılmasının nedeni mikrodenetleyici pininin kararsız durumda kalmamasını sağlamaktır. Pull-Up ise bu durumun tam tersidir. Biz bu örneğimizde Pull-Down direnç kullanacağız.

Pull-Down kullanacağımızı denetleyicimize bildirmemiz lazım. Bunuda GPIO kısmında tıklayım GPIO Pull up/Pull down kısmından Pull down seçeneğini seçerek yapıyoruz.

Şimdi ledimizi takacağımız pini belirleyelim ben PE10 numaralı pini seçiyorum.Tıklayıp GPIO_Output moduna ayarlıyorum.

Bu aşamada istersek belirlediğimiz pinlere isim verebiliriz.Pinimize sağ tıklayıp Enter User Label seçeneğini seçiyoruz. PE10 pinine mavi led takacağım için adını mavi olarak belirliyorum. Butonumun bağlı olduğu pinede aynı işlemi yaparak adını buton olarak değiştiriyorum.

Pin ayarlarımız bitti şimci Clock Configuration ayarlarımıza geçiyoruz. Input frequency kısmında 25 olan frekansımızı 8 yapıyoruz. Sistemin clock ayarınıda (System Clock Mux) kısmınıda HSE yapıyoruz. Bu sayede tüm birimlere 8MHz clock vermiş olduk.

Clock ayarlarımızda bitti şimdi projemizi kaydedip (File=> Save) kodlarımızı olurşturalım...

```
HAL_GPIO_ReadPin(GPIOx, GPIO_Pin)
```

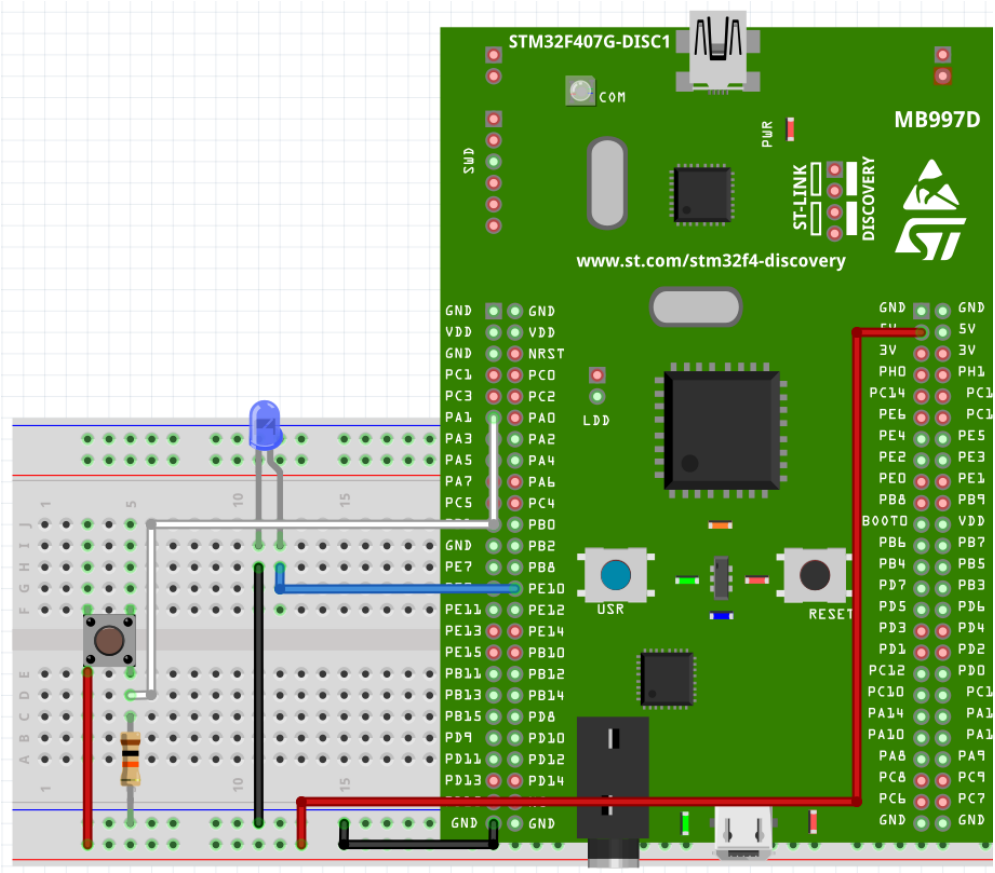
Bu komut ile bir gpıo pinini digital olarak okuyabiliriz. 2 parametre alır bunlar sırayla port ,gpio pinidir.
Bir önceki işlemde isimlendirdiğimiz için pinleri bunları kullanabiliriz.Bu komut bizlere 1 veya 0 olarak değer döndürür butona basıldıysa 1 basılmadıysa 0 değerlerini döndürür.

Şimdi butona basıldığında mavi ledimizin yanıp butondan elimizi çektiğimizde ledimizin söndüğünü gözlemlemek için gerekli kodları yazalım.

```
if(HAL_GPIO_ReadPin(buton_GPIO_Port,buton_Pin)==1) // Butona basıldı
{
HAL_GPIO_WritePin(mavi_GPIO_Port, mavi_Pin, 1);    // Ledi Yak
}

else                                               // Butona basılmadı
{
HAL_GPIO_WritePin(mavi_GPIO_Port, mavi_Pin, 0);    // Ledi Söndür
}
```

Kodumuzu derlemek için build işaretine tıklayalım. Kaydedilsin mi sorusuna Yes diyelim. Kodu kartımıza yüklememiz için proje yolunu biliyor olmamız gerek bunun için Buton dosyamıza sağ tıklayıp properties e tıklayalım. Location kısmında projemizin yolu yazılıdır bunu kopyalayalım.

Devremizi kurup daha sonra kodumuzu kartımıza yükleyelim.



Devremizi yukarıdaki gibi kurup kodumuzu yükleyelim. STM32 ST-LINK programını açalım File => Open file seçeneklerine tıklayıp kopyaladığımız yolu yapıştıralım. Debug=> Buton dosyamızı açalım. Kartımıza yüklemek için Program verify butonuna tıklayıp start butonuna tıklıyoruz.

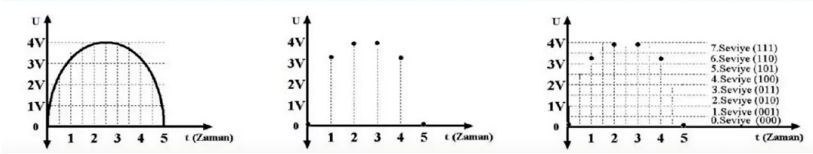
Yükleme işlemi tamamlandı butona bastığımızda ledimizin yandığını elimizi butondan çekince ledimizin söndüğünü gözlemleyelim.

Bu projemizde herhangi bir gpio digital olarak nasıl okunur , gpio okuma fonksiyonu bize nasıl bir değer döndürür bunları bir buton ve led aracılığıyla öğrenmiş olduk.

Gerekli Malzemeler
Stm32F407 Geliş rme Kartı
1 Adet 10K Potansiyometre
1 Adet LED
Breadboard
Yeteri kadar jumper kablo (dişi-erkek)

Bu projede “ADC” uygulaması yapacağız. ADC (Analog Digital Conventer) donanımı Analog bir veriyi Digital bir veriye dönüştürme işlemleri yapmaktadır. Mikrodenetleyici sistemlerde işlemler digital olarak işlenmektedir. Yani 1 ve 0 sinyalleri gelmektedir. Analog bir sinyali okumak is yorsak bunu ADC donanımı ile digital veriye döndürmemiz gerekmektedir.

ADC Dönüşümü Nasıl Yapılıyor ?



STM32 Üzerinde 6 bit ,8 bit ,10 bit ve 12 bit olarak ADC çözünürlüğü seçilebilmektedir. Örneğin 10 bit seçtik giriş sinyalimizin o anki değeri 3.3V geldi bu durumda ADC işlemi sonrasında digital işlemi sonrasında 2^10 (2 üzeri 10) değerinden max olarak 1023 değerini görürüz. (İşlemleri yaparken 0 ile bir değer aralığını aldığımız için genel formül (2^n)-1 şeklinde düşünülebilir. Örn: 2^10 = 1024 yapar. (2^n)-1 şeklinde düşündüğümüzde 1024 – 1 = 1023 şeklinde olacaktır.)

Bu formülle baktığımızda max 6 bit = 63 , 8 bit = 255 , 10 bit = 1023, 12 bit = 4095 değerleri verdiğini gözlemleriz. Şimdi derleyicimizi açıp işlemlerimize başlayabiliriz.

CubeIDE derleyicimizi açtığımızfa ilk olarak File=>New=>STM32 Project butonuna tıklayalım. Açılan sayfada arama kısmına stm32f407vg yazıp çıkan denetleyicimize tıklayarak Next butonuna basalım.

Yeni açılan pencerede projemizin ismini belirleyeceğiz. Potansiyometre yazmayı tercih ettim. Projemizin adını belirledikten sonra Next => Finish butonlarına basıyoruz. Gelen uyarıda CubeIDE arayüzünü açalım mı diyecektir. Yes butonuna basıyoruz.

Konfigürasyon sayfamız açıldığında yapmamız gereken ilk ayar programlama ayarıdır. System Core => SYS seçtikten sonar Debug kısmını Serial Wire Yapıyoruz. Daha Sonra System Core =>RCC seçip osilatör ayarlarımızı yapacağız. HSE kısmını Crystal olarak seçiyoruz.

Devremizde 1 adet potansiyometre kullanacağız bunuda PA1 kanalına bağlayalım.

Bu işlem için PA1'e tıklayıp ADC1_IN1 seçeneğini seçiyoruz. Sonrasında ADC ayarlarını yapmak için hemen sol kısımda bulunan Analog penceresine tıklıyoruz.ADC1 bölümünü seçiyoruz ve IN1 kanalının enable(aktif) olduğunu göreceğiz. ADC_Settings=>Resolution kısmından çözünürlüğümüzü seçmemiz gerekiyor. Biz 12bit' I seçelim. Continuous Conversion Mode kısmı sürekli çevrim yapmak için mevcuttur bunu Enable yapalım. Pin konfigürasyon ayarlarımız bitti şimdi Clock konfigürasyon ayarlarımızı yapacağız.

Pinout & Configuration menusunun hemen sağ tarafında bulunan Clock Configuration menüsüne tıklıyoruz.

Input frequency kısmında 25 olan frekansımızı 8 yapıyoruz. Sistemin clock ayarınıda (System Clock Mux) kısmınıda HSE yapıyoruz. Bu sayede tüm birimlere 8MHz clock vermiş olduk.

Clock ayarlarımızda bitti şimdi projemizi kaydedip (File=> Save) kodlarımızı olurşturalım... Core=>Src=>main.c seçip while döngüsünün olduğu yere kodlarımızı yazacağız.

İlk olarak ADC çevirimini başlatmamız lazım.

```
HAL_ADC_Start (&handc1);
```

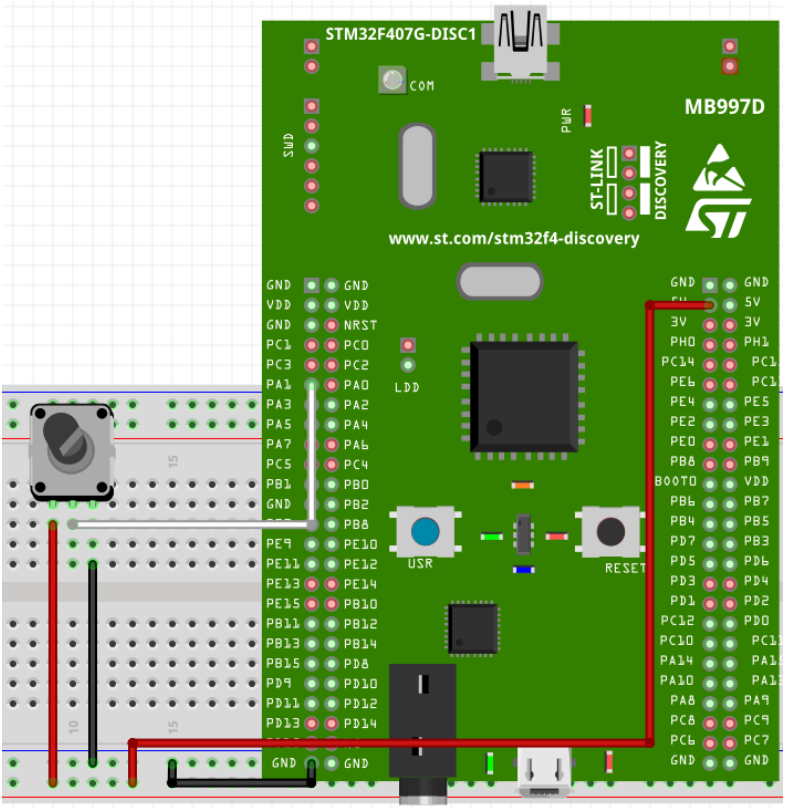
Daha sonra çevirimin bitmesini beklememiz lazım. Bunun için ;

```
if(HAL_ADC_PollForConversion(&handc1,100000) == HAL_OK)
```

Çevirimin bittiği durumda bize 0 değerini döndürüyor bunuda “HAL_OK” ile sağlıyor bu durumdan dolayı “HAL_OK” yerine 0 da yazabiliriz. Değerimizi okudu artık bize return olarak bir tam sayı gönderiyor. Bunuda int değer=0;şeklinde tanımlayabiliriz. Şimdi kodların hepsini yazıp açıklayalım.

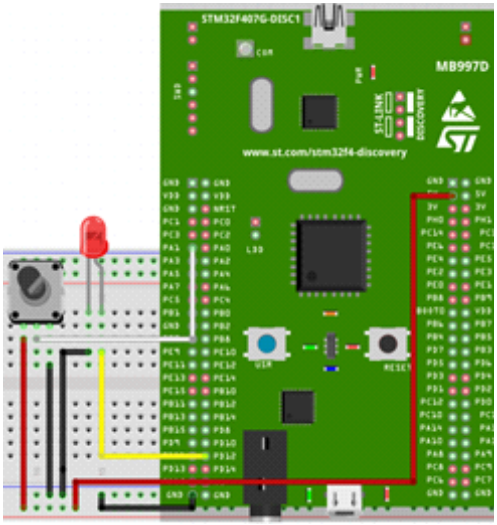
```
int deger=0;

HAL_ADC_Start(&handc1);
if(HAL_ADC_PollForConversion(&handc1,100000)==HAL_OK)
{

deger = HAL_ADC_GetValue(&handc1);  // biten çevirim değerini okuyoruz.
}
HAL_ADC_Stop(&handc1);        // çevirimi bitiriyoruz.
```

Kodumuzu derlemek için build işaretine tıklayalım. Şimdi devremizi kuralım.



Potansiyometreden değerimizi okuyabilmemiz için CubeIDE üzerinden Debug butonuna gelip OK tuşuna basıyoruz. Kodumuzu derleyip kartımıza yükleyip debug moduna geçtiğini gözlemleyelim. butonuna basıyoruz. Add new expression kısmına deger yazıyoruz. Şimdi yazılımı kart üzerinde çalıştırıp değerleri izleyelim. F8(Resume) tuşuna tıklıyoruz. Potansiyometremizi çevirerek value değerinin değiştiğini gözlemleyebilriz. Max konumuna geldiğinde 4095 değerini vericektir çünkü 12 bit olarak seçtik.

Artık potansiyometreyi nasıl kullandığımızı öğrendik hadi potansiyometre kullanarak bir led yakalım !
Aydı dosya üzerinden PD12 numaralı pini output moduna alıp kodlarımızı yazalım ve yeni devremizi oluşturalım.



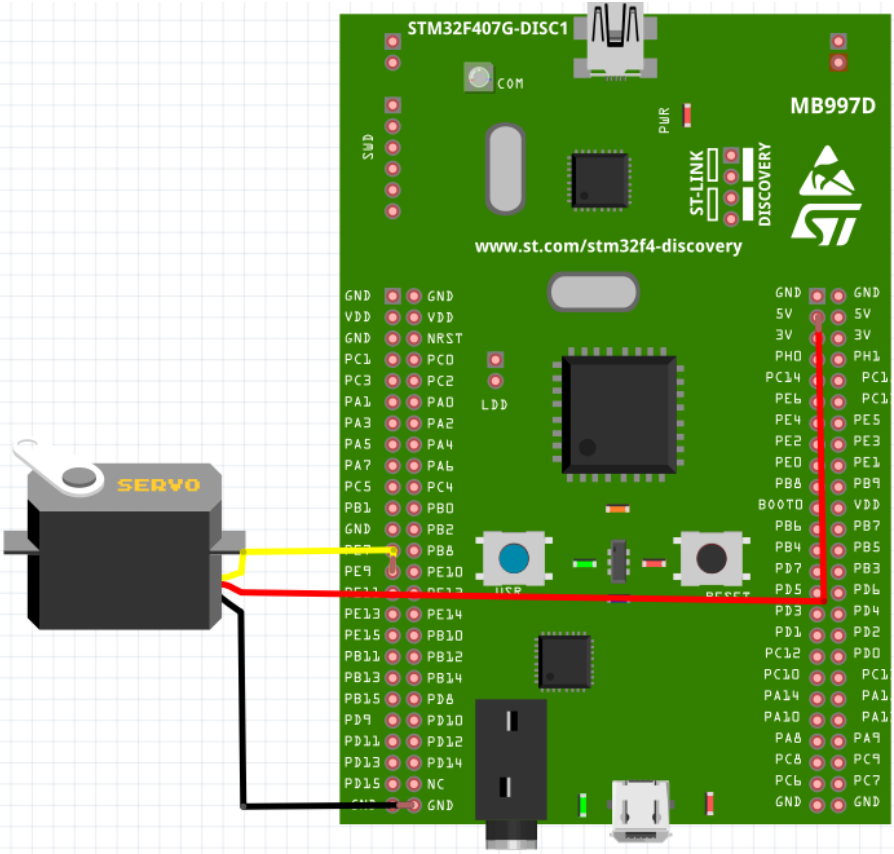

```
int deger =0;
        HAL_ADC_Start(&hadc1);
        if(HAL_ADC_PollForConversion(&hadc1,100000) == HAL_OK){
                deger = HAL_ADC_GetValue(&hadc1);
        }
                HAL_ADC_Stop(&hadc1);
                if(deger>=2000){    //deger 2000 ve üstünde ise
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1); // ledi yak

                }
               else{ //deger 2000  altında ise
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 0); // ledi söndür
}
```

Potansiyometreden okuduğumuz değer 2000 ve üstünde ise ledimizin yandığını 2000 in altında ise ledimizin söndüğünü gözlemleyebiliriz.

Gerekli Malzemeler
Stm32F407 Geliştirme Kartı
1 adet Servo Motor
Yeteri kadar jumper kablo (dişi-erkek)
Bu projedemizde STM32F407 de PWM sinyallerinin nasıl kullanıldığını öğreneceğiz. Projeyi desteklemesi ve daha iyi kavramamız içinde bir servo motorla çalışma yapacağız. Hadi başlayalım !

Servo motorumuzun datasheetini incelediğimizde 20ms yani 50 Hz lik bir pwm sinyaliyle sürülebildiğini gözlemleyebiliriz. Pozisyon 0 da iken 1.5ms 90 da iken 2ms -90 1ms lik bir pulse verilmesi gerektiği yazıyor.

Servo motorumuzun sarı renkli kablosu => Sinyal kırmızı => VCC siyah =>GND dir.



CubeIDE derleyicimizi açtığımızfa ilk olarak File=>New=>STM32 Project butonuna tıklayalım. Açılan sayfada arama kısmına stm32f407vg yazıp çıkan denetleyicimize tıklayarak Next butonuna basalım.

Yeni açılan pencerede projemizin ismini belirleyeceğiz. Servo yazmayı tercih ettim. Projemizin adını belirledikten sonra Next => Finish butonlarına basıyoruz. Gelen uyarıda CubeIDE arayüzünü açalım mı diyecektir. Yes butonuna basıyoruz.

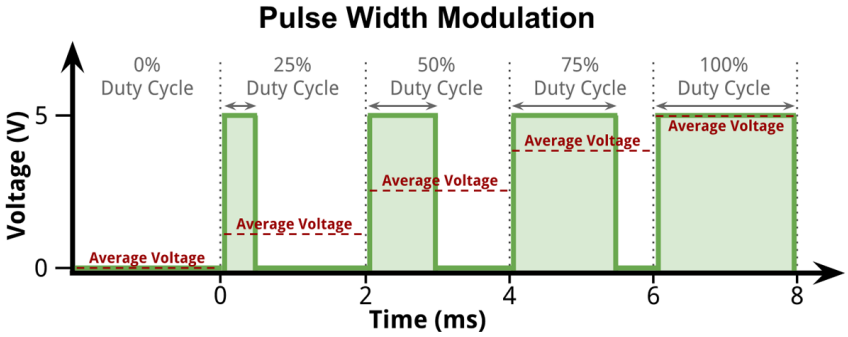
System Core => SYS seçtikten sonar Debug kısmını Serial Wire Yapıyoruz. Daha Sonra System Core =>RCC seçip osilatör ayarlarımızı yapacağız. HSE kısmını Crystal olarak seçiyoruz.

Pwm sinyalini belirlerken 2 tane değer girmemiz gerekir.

Formül : `Prescaler=(Timer Source Freg)/PWM Freq*PWM Resulution)-1`

Bizim timer clock'umuz 8MHz, Pwm Freq 1KHz olsun, PWM Resolution 500 Count olsun.
 Formüle gore hesapladığımızda 8000000/1000 = 8000 8000/500 = 16-1 =15 =Prescaler değeri. 500 Count -1 499 Count.

Pwm sinyalimiz için Timers menüsü altından TIM1 e tıklayıp Channel1'i PWM Generation CH1 olarak seçiyoruz.Prescaler'i 15 Counter Period'u 499 yapıyoruz. Bu şekilde seçince Program otomatik olarak PE9 u TIM1_CH1 olarak atadı. Programımızı derleyelim projemizi kaydedip (File=> Save) kodlarımızı olurşturalım.
Core=>Src=>main.c seçip while döngüsünün olduğu yere kodlarımızı yazalım.



Bizim şuan ki ayarlarımızda denetleyici 0 dan 500'e kadar sayıcak. %50 duty cycle oluşturmak için 500/2 = 250 olarak yazmamız gerekir.

```
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); // Timer'ı başlatıyoruz.

TIM1->CCR1 = 250;// %50 genişlik veriyoruz.
```

Kodumuzu derlemek için build işaretine tıklayalım.

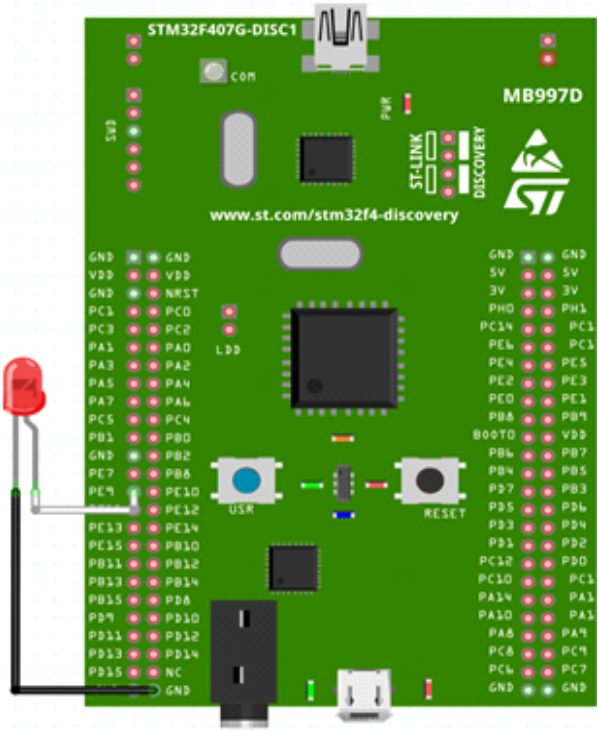
Servo motorumuzun açılarını nasıl değiştirebileceğimizi gözlemledik. Şimdi tüm duty cycle ları deneyeceğimiz kodlarımızı yazalım.

```
TIM1->CCR1 = 0;
HAL_Delay(1000);
TIM1->CCR1 = 125;
HAL_Delay(1000);
TIM1->CCR1 = 250;
HAL_Delay(1000);
TIM1->CCR1 = 375;
HAL_Delay(1000);
TIM1->CCR1 = 500;
HAL_Delay(1000);
```

F8(Run) butonuna basarak kodumuzu yükleyelim.
Servo motorumuzun açılarını 1 saniye aralıklarla değiştirdik. Şimdi bu işlemi for döngüsü ilede yapalım.

```
for(int i=0;i<500;i+=125){
        TIM1->CCR1 = i;
HAL_Delay(1000);}
for(int i=500;i>0;i-=125){
        TIM1->CCR1 = i;
HAL_Delay(1000);}
```

Pwm sinyallerini servo motorlar üzerinde deneyimledik hemen basit bir şekilde led parlaklığını değiştirebileceğimiz bir proje yapalım.



Aynı pin üzerine servo motorumuzu çıkarıp ledimizi taktık. Led daha efektif yanıp sönsün diye kodda geçişleri daha kısa tutalım.

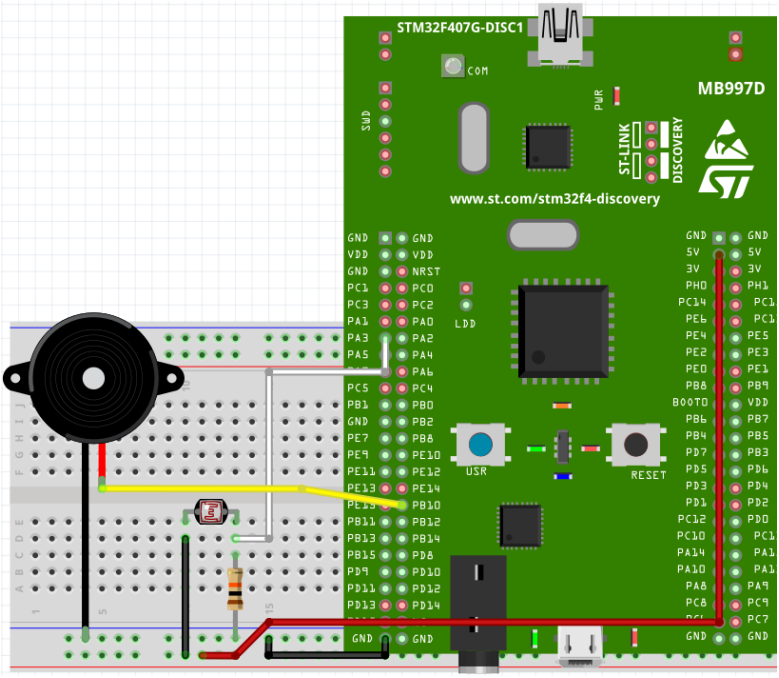
```
for(int i=0;i<500;i++){
                TIM1->CCR1 = i;
        HAL_Delay(1);}
        for(int i=500;i>0;i--){
                TIM1->CCR1 = i;
        HAL_Delay(1);}
```

Kodu çalıştıralım 0 ile 500 arasındaki tüm duty cycleları ledimizde parlaklık değişimi olarak gözlemleyelim.

Bu projede ldr ile ortamın ışık seviyesini ölçüp belli değerlerde buzzerı çalıştıracağız. Bu projeyi en sona bırakmak istedim çünkü analog bir değeri okumayı ve digital bir pini nasıl kullanacağımızı ayrı ayrı öğrenmiş k bu projede 2 farklı modülü direkt olarak kullanıp öğrendiklerimizi pekiştirelim.

Gerekli Malzemeler:
Stm32F407 Geliştirme Kartı
1 Adet Ldr
1 Adet 10K direnç
1 adet Buzzer
Breadboard
Yeteri kadar jumper kablo (dişi-erkek)

Artık çok fazla ön bilgiye ihtiyacımız olmadığından direct olarak devremizi kurup kodlarımızı yazalım.



Devrede buzzerı PB10 nolu pine Ldr yi PA3 nolu pine bağladım. Şimdi derleyicimizi açıp işlemleri yapmaya başlayalım.

CubeIDE derleyicimizi açtığımızfa ilk olarak File=>New=>STM32 Project butonuna tıklayalım. Açılan sayfada arama kısmına stm32f407vg yazıp çıkan denetleyicimize tıklayarak Next butonuna basalım.

Yeni açılan pencerede projemizin ismini belirleyeceğiz. Potansiyometre yazmayı tercih ettim. Projemizin adını belirledikten sonra Next => Finish butonlarına basıyoruz. Gelen uyarıda CubeIDE arayüzünü açalım mı diyecektir. Yes butonuna basıyoruz.

System Core => SYS seçtikten sonar Debug kısmını Serial Wire Yapıyoruz. Daha Sonra System Core =>RCC seçip osilatör ayarlarımızı yapacağız. HSE kısmını Crystal olarak seçiyoruz.

PA1'e tıklayıp ADC1_IN1 yapıyoruz daha sonra PB10 pinini Output olarak ayarlıyoruz. Sonrasında ADC ayarlarını yapmak için hemen sol kısımda bulunan Analog penceresine tıklıyoruz.ADC1 bölümünü seçiyoruz ve IN1 kanalının enable(aktif) olduğunu göreceğiz. ADC_Settings=>Resolution kısmından çözünürlüğümüzü seçmemiz gerekiyor. Bu defada 10bit' i seçelim. Continuous Conversion Mode kısmı sürekli çevrim yapmak için mevcuttur bunu Enable yapalım. Pin konfigürasyon ayarlarımız bitti şimdi Clock konfigürasyon ayarlarımızı yapacağız.

Clock ayarlarımızda bitti şimdi projemizi kaydedip (File=> Save) kodlarımızı olurşturalım... Core=>Src=>main.c seçip while döngüsünün olduğu yere kodlarımızı yazacağız.

Artık öğrendiğimiz kodları hemen yazabiliriz!...

10 bit kullandığımız için 0 ila 1023 arasında değer okuyacağımızı biliyoruz daha once yaptığımız
potansiyometre örneğimizden bunu baz alarak kodlarımızı şekillendirelim.

```
int deger =0;
        HAL_ADC_Start(&hadc1);
        if(HAL_ADC_PollForConversion(&hadc1,100000) == HAL_OK){
                deger = HAL_ADC_GetValue(&hadc1);
        }
                HAL_ADC_Stop(&hadc1);
                if(deger>=500){     //deger 500 ve üstünde ise
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, 1); //buzzer aktif

                }
              else{ //deger 500 altında ise
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, 0); //buzzer pasif
}
```

Kodumuzu derlemek için build işaretine tıklayalım. Potansiyometreden değerimizi okuyabilmemiz için CubeIDE üzerinden Debug butonuna gelip OK tuşuna basıyoruz. Kodumuzu derleyip kartımıza yükleyip debug moduna geçtiğini gözlemleyelim. butonuna basıyoruz. Add new expression kısmına deger yazıyoruz. Şimdi yazılımı kart üzerinde çalıştırıp değerleri izleyelim. F8(Resume) tuşuna tıklıyoruz. Ldr üzerine ışık tutup yada parmağımızla kapatıp value değerinin değiştiğini gözlemleyebilriz.

Ldr den okuduğumuz değer 500 ve üzerindeyse buzzer aktif olacaktır 500 altında ise değer buzzer pasif konuma geçecektir.

YouTube

youtube.com/robotistan

FORUM

robotistan

forum.robotistan.com

BLOG

robotistan

maker.robotistan.com

Robotistan Elektronik Ticaret AŞ

Muhammed Ali ÖZEN (İçerik) - Mehmet AKÇALI (Editör) - (Mehmet Nasır KARAER (Grafik)
info@robotistan.com - www.robotistan.com
Phone: 0850 766 0 425