AKILLI EV AYDINLATMA SİSTEMİ

Deneyin Amacı

- Ortamın ışık seviyesini ve hareketi algılayarak enerji tasarrufu sağlayan akıllı bir aydınlatma sistemi oluşturmak.
- PIR sensörü ile hareket algılandığında ve/veya ortam karanlık olduğunda LED'leri yakmak.
- Mikrodenetleyiciyi kullanarak sensörlerden gelen verileri işlemek ve LED'leri uygun şekilde kontrol etmek.

Kullanılan Malzemeler ve Görevleri

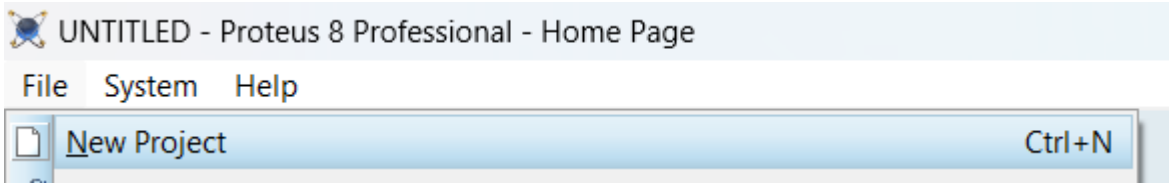
1. **Kristal Osilatör (4 MHz):**
    a. **Görev:** Mikrodenetleyicinin çalışması için sabit bir saat sinyali üretir.
2. **Kondansatörler (2 adet, 22 pF):**
    a. **Görev:** Kristal osilatörün kararlı çalışmasını sağlar.
3. **LDR (Işığa Duyarlı Direnç):**
    a. **Görev:** Ortam ışık seviyesini algılar ve mikrodenetleyiciye analog bir sinyal gönderir.
4. **PIR Sensör:**
    a. **Görev:** Ortamdaki hareketi algılar ve mikrodenetleyiciye dijital bir sinyal gönderir.
5. **LED'ler (3 adet):**
    a. **Görev:** Sistem tarafından kontrol edilen aydınlatma elemanlarını temsil eder.
6. **Dirençler (1 adet 10 kΩ, 1 adet 330 Ω, 1 adet 220 Ω, 1 adet 110 Ω):**
    a. **Görev:**
        i. LDR'nin çalışması için gerilim bölücü devre oluşturur (10 kΩ).
        ii. LED'ler için akımı sınırlayarak zarar görmelerini engeller (330 Ω , 220Ω, 110Ω).
7. **Logic Toggle (Opsiyonel):**
    a. **Görev:** Simülasyonda PIR sensörün çalışmasını elle kontrol etmek için kullanılır.
8. **PIC16F877A Mikrodenetleyici:**
    a. **Görev:** Sensörlerden gelen verileri işleyerek LED'leri kontrol eder.

Devre Elemanları ve Görevleri

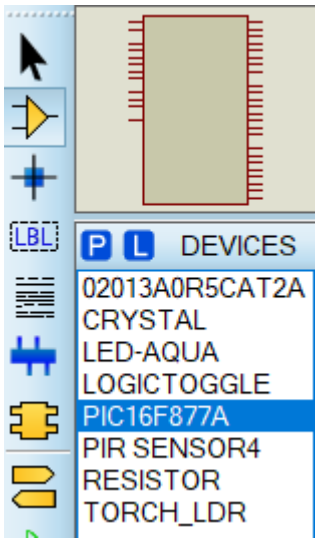
1. **Kristal Osilatör (4 MHz):**
    a. **Görev:** Mikrodenetleyicinin çalışması için sabit bir saat sinyali üretir.
2. **Kondansatörler (2 adet, 22 pF):**
    a. **Görev:** Kristal osilatörün kararlı çalışmasını sağlar.
3. **LDR (Işığa Duyarlı Direnç):**
    a. **Görev:** Ortam ışık seviyesini algılar ve mikrodenetleyiciye analog bir sinyal gönderir.
4. **PIR Sensör:**
    a. **Görev:** Ortamdaki hareketi algılar ve mikrodenetleyiciye dijital bir sinyal gönderir.
5. **LED'ler (3 adet):**
    a. **Görev:** Sistem tarafından kontrol edilen aydınlatma elemanlarını temsil eder.
6. **Dirençler (1 adet 10 kΩ, 3 adet 330 Ω):**
    a. **Görev:**
        i. LDR'nin çalışması için gerilim bölücü devre oluşturur (10 kΩ).
        ii. LED'ler için akımı sınırlayarak zarar görmelerini engeller (330 Ω).
7. **Logic Toggle (Opsiyonel):**
    a. **Görev:** Simülasyonda PIR sensörün çalışmasını elle kontrol etmek için kullanılır.
8. **PIC16F877A Mikrodenetleyici:**
    a. **Görev:** Sensörlerden gelen verileri işleyerek LED'leri kontrol eder.

Devre Elemanlarının Bağlantıları Ve Deneyin Yapılışı

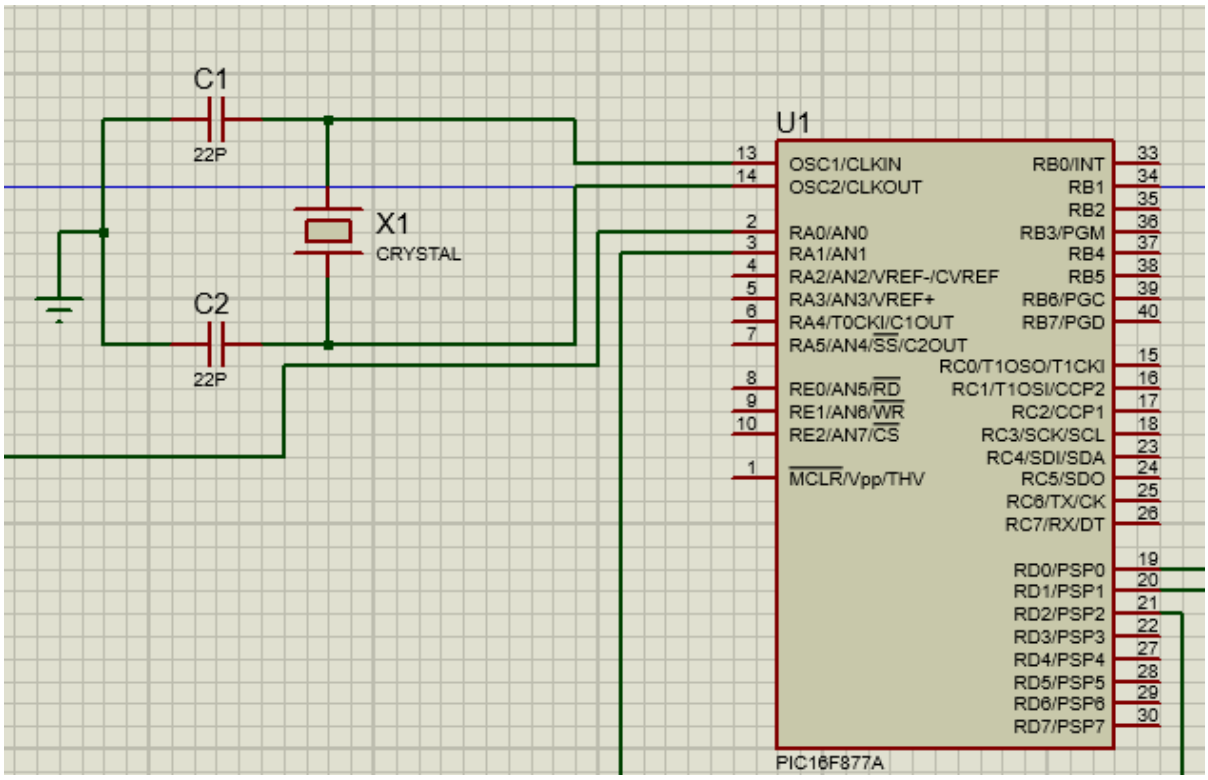
Bu deney için gerekli olan 2 program deneyleri simüle etmek için Proteus, mikrodenetliyiciyi kodlamak için MPLAB X IDE uygulamalarıdır.



Proteus uygulamasına girdikten sonra File sekmesinden New Project sekmesine tıklayıp yeni proje oluşturuyoruz.



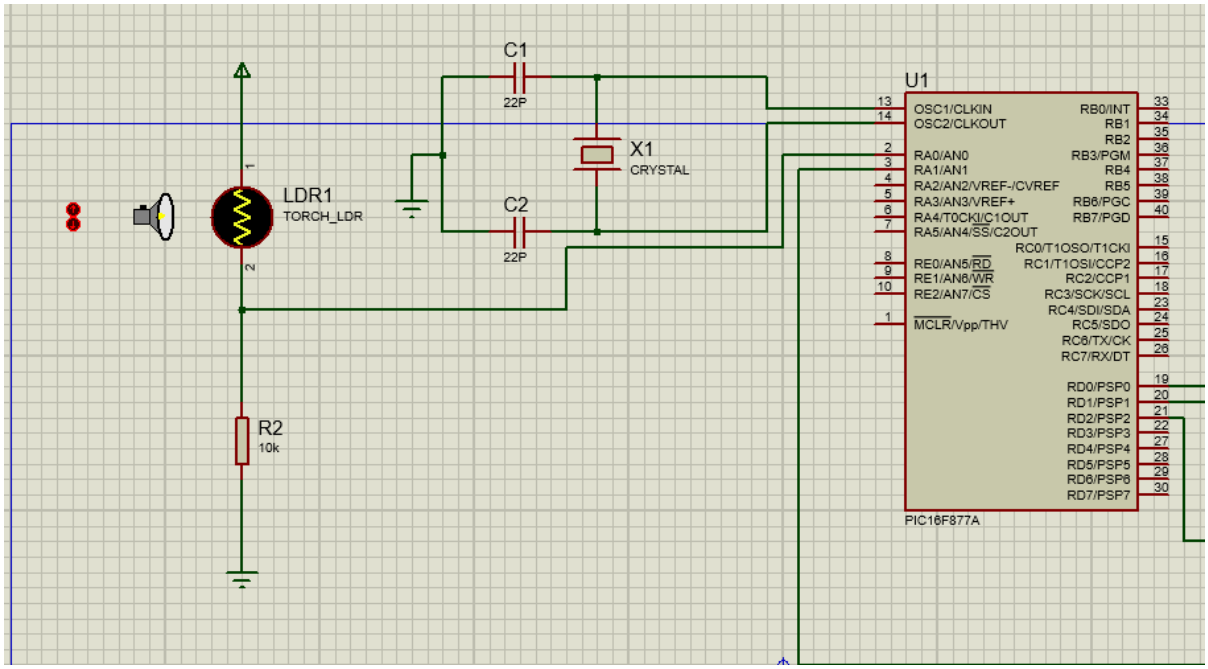
Projeyi oluşturduktan sonra karşımıza gelen ekranın sol kısmındaki Component Modu'u seçip Devices kısmındaki P butonuna tıkladıktan sonra devre elemanlarını teker teker seçiyoruz.



Daha sonra devre elemanlarının kuruluşuna başlanır.

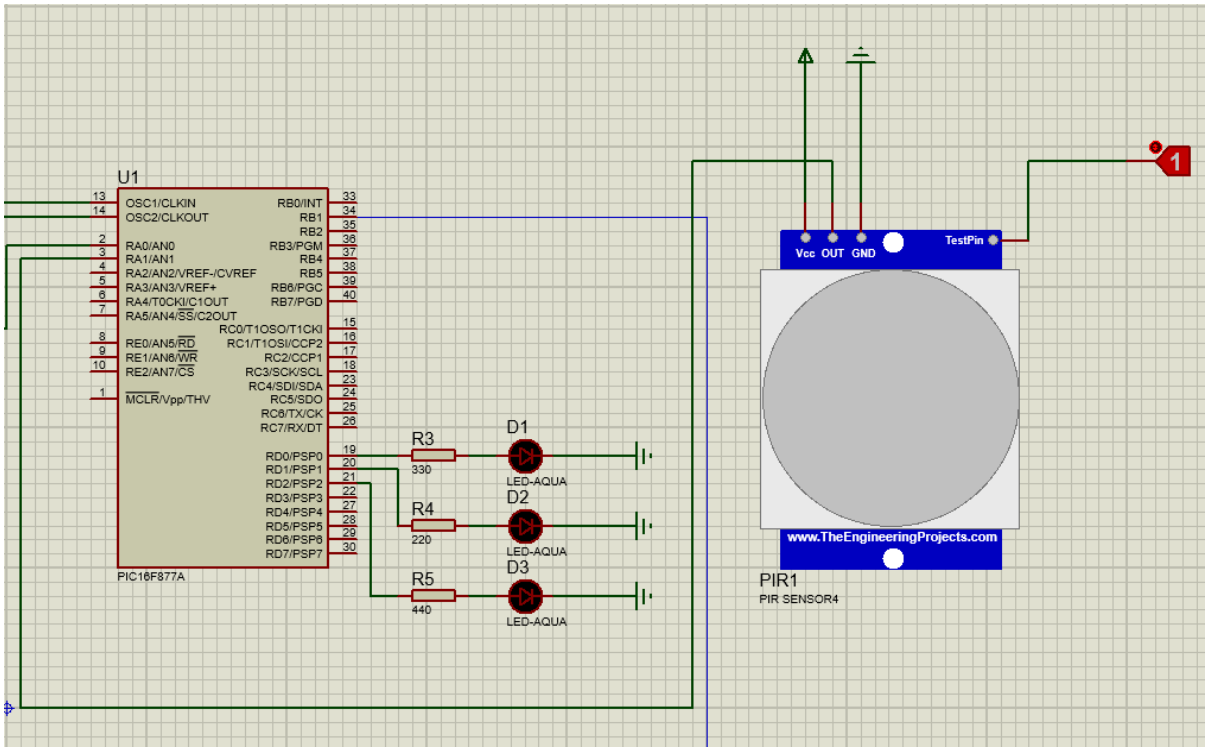
Kristal Osilatör ve Kondansatörler:

- **Kristal:**
  - Bir ucu OSC1 (13. pin), diğer ucu OSC2 (14. pin) pinlerine bağlanır.
- **Kondansatörler:**
  - Kristalin her iki ucu ile toprak arasına bağlanır (22 pF).



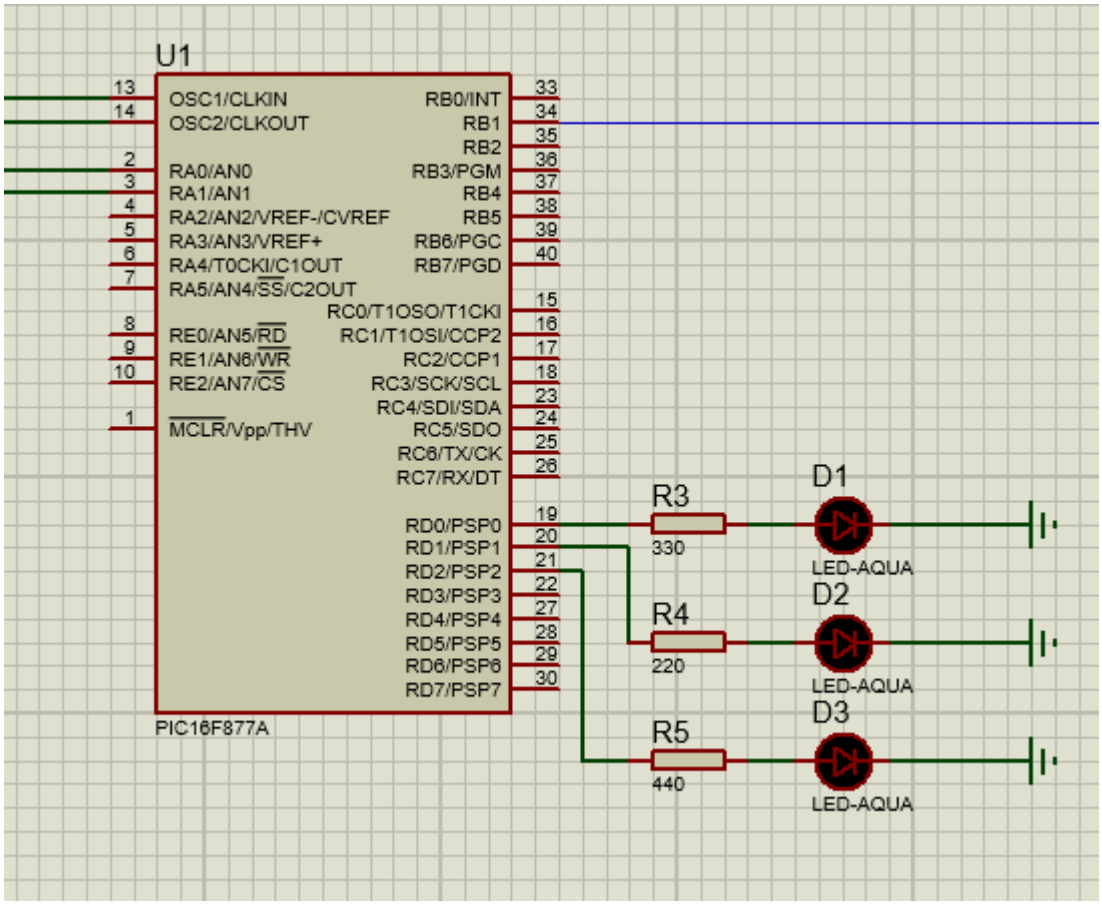
LDR (Gerilim Bölücü Devre):

- LDR'nin bir ucu 5V'a, diğer ucu bir 10 kΩ dirençle birlikte topraklanır.
- LDR ve direnç arasındaki bağlantı **RA0 (AN0, 2. pin)** pinine bağlanır.



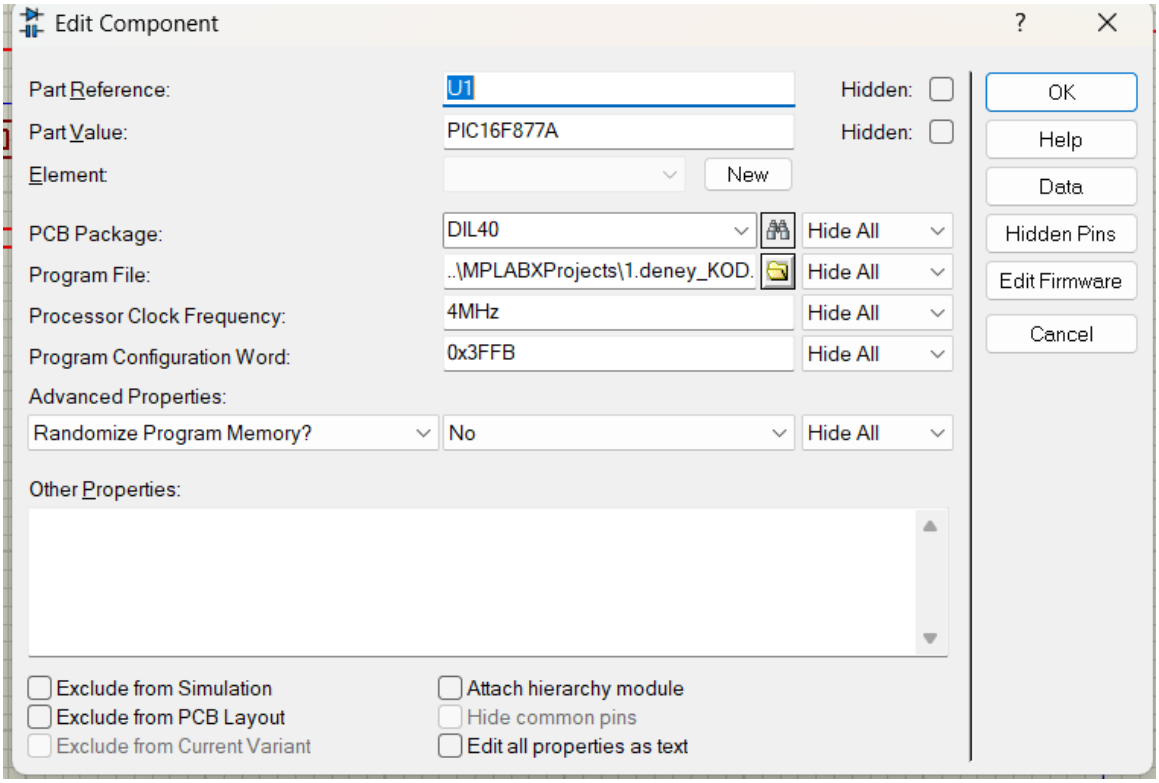
PIR Sensör:

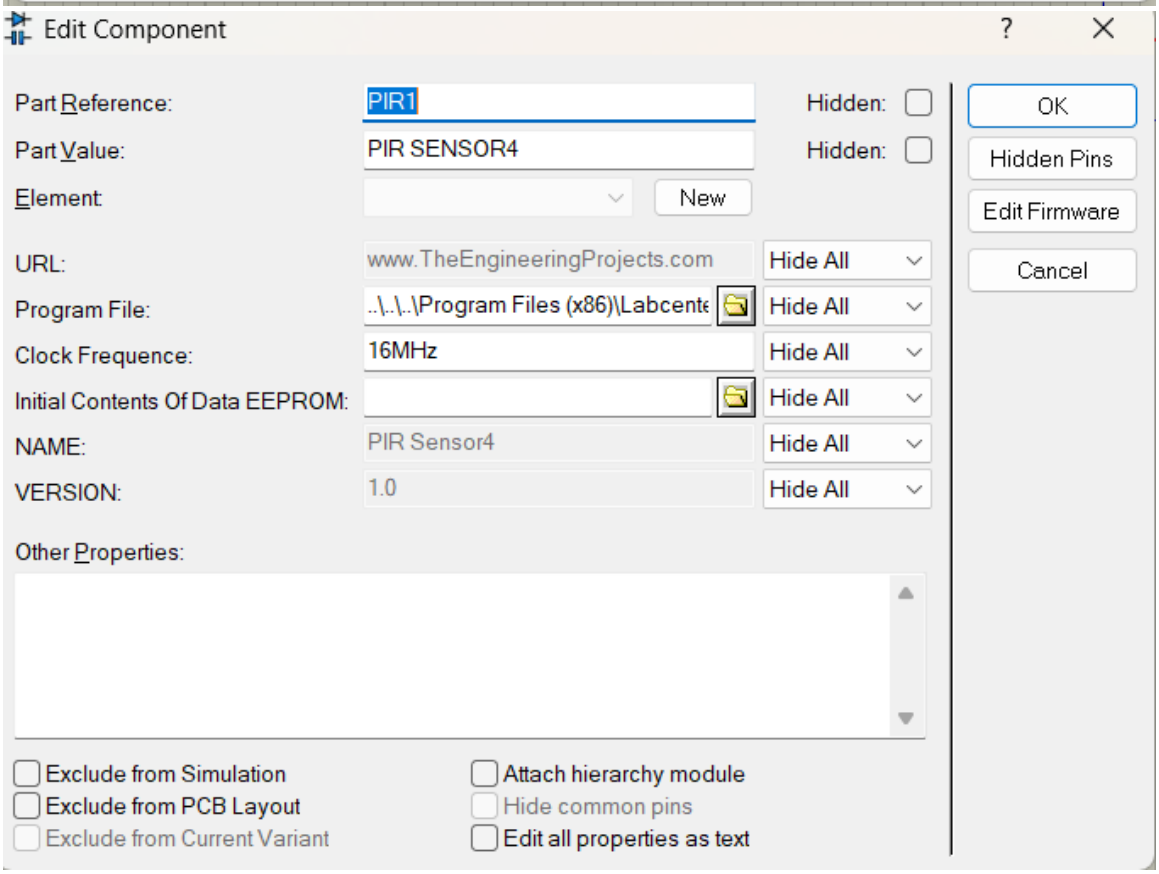
- PIR sensörün çıkış pini **RA1 (3. pin)** pinine bağlanır.
- Güç (5V) ve toprak bağlantıları yapılır.



LED'ler ve Dirençler:

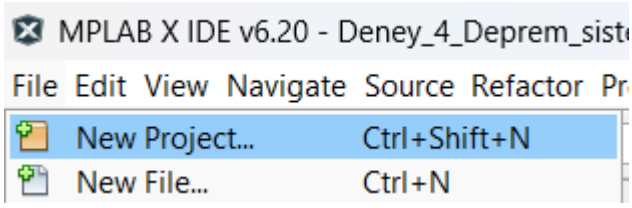
- LED'lerin anot uçları sırasıyla **PORTD** üzerindeki **RD0 (19. pin)**, **RD1 (20. pin)** ve **RD2 (21. pin)** pinlerine bağlanır.
- LED'lerin katot uçlarına 330 Ω 220 Ω 440 Ω dirençler bağlanarak topraklanır.
- Dirençlerin farklı olma sebebi ledlerin yanıp yanmamasının dirençle alakalı olmadığını göstermektir.



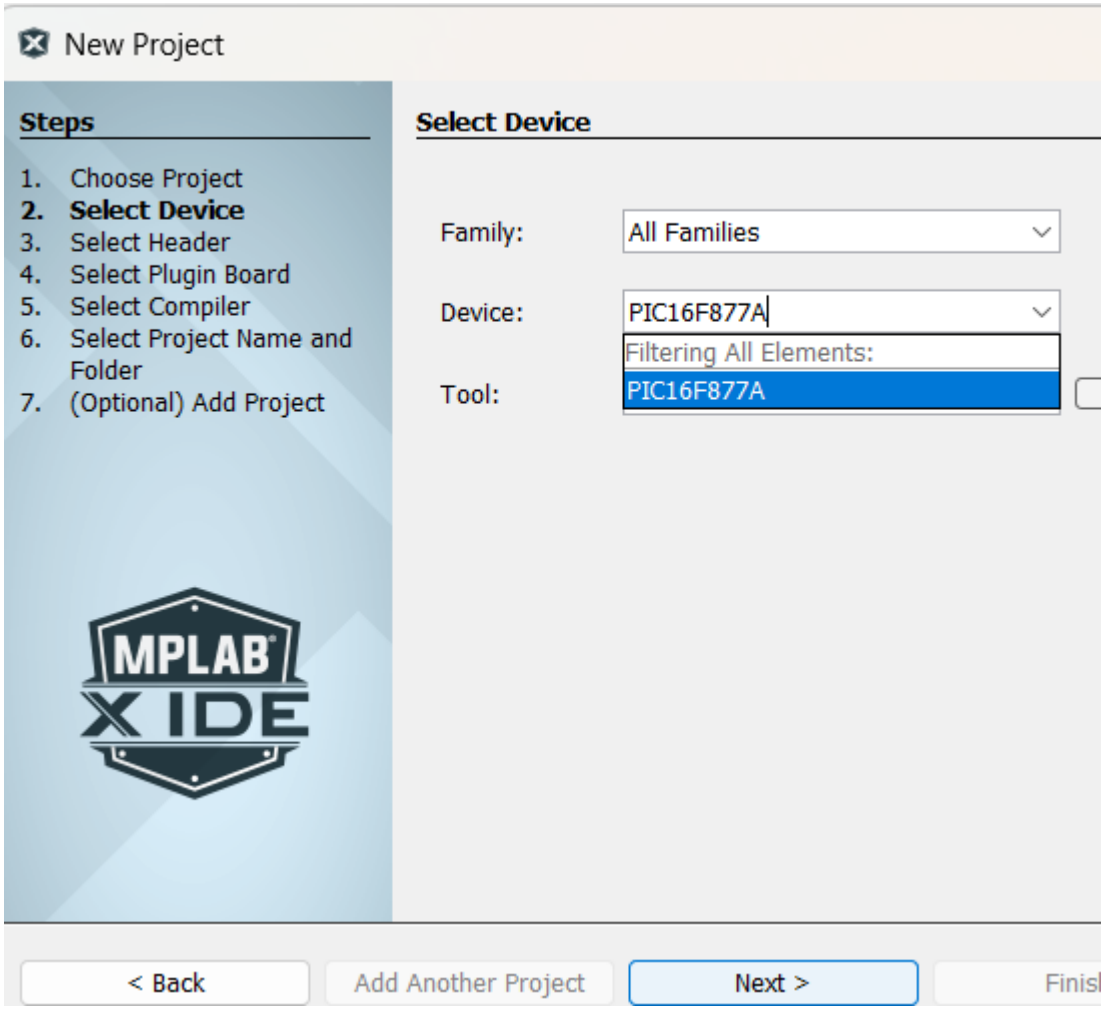


En son Pır sensör ve Mikrodenetliyicinin program dosyaları seçilir ve proteusta işimiz biter.

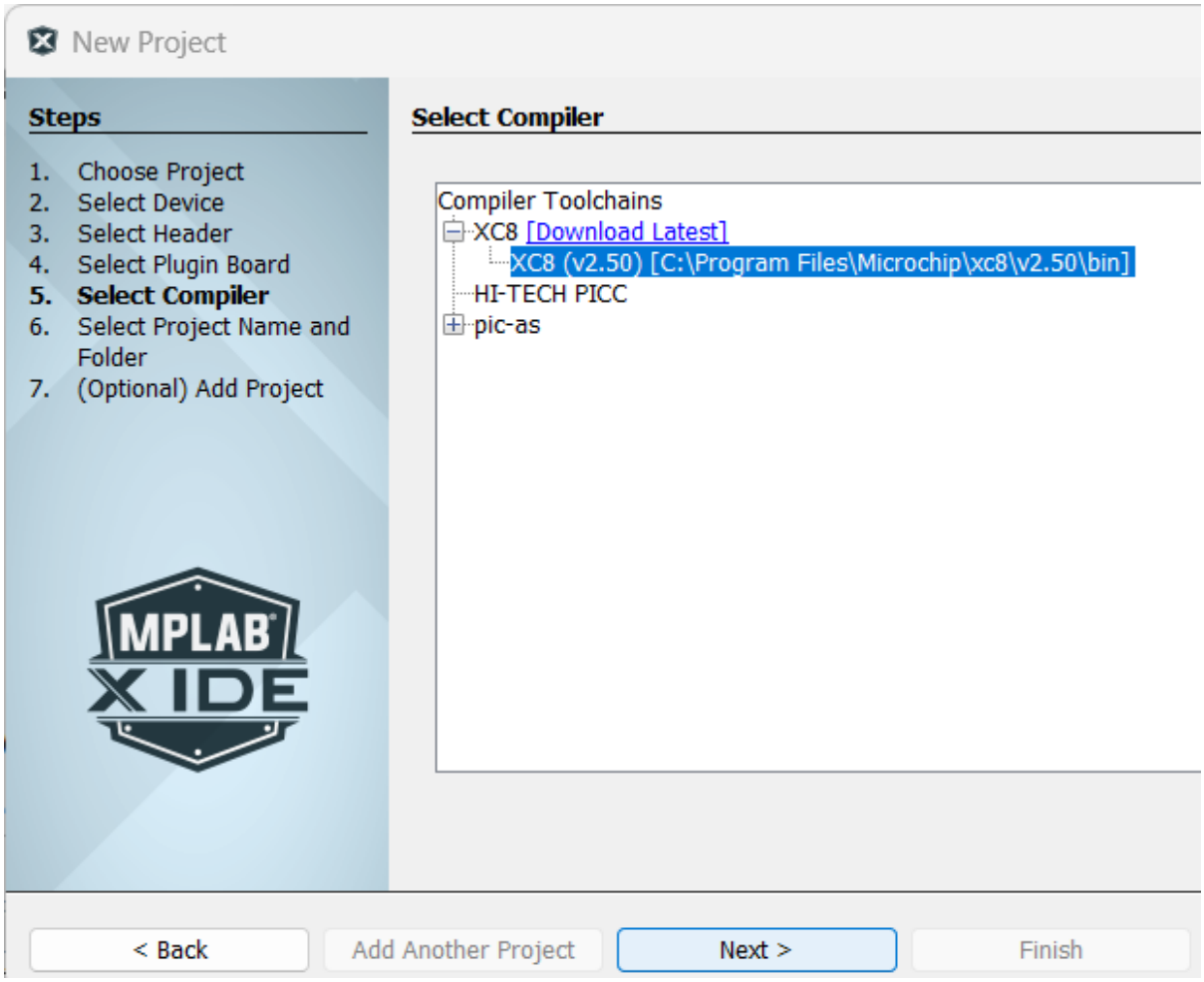
Mikrodenetliyicinin Kodlanması



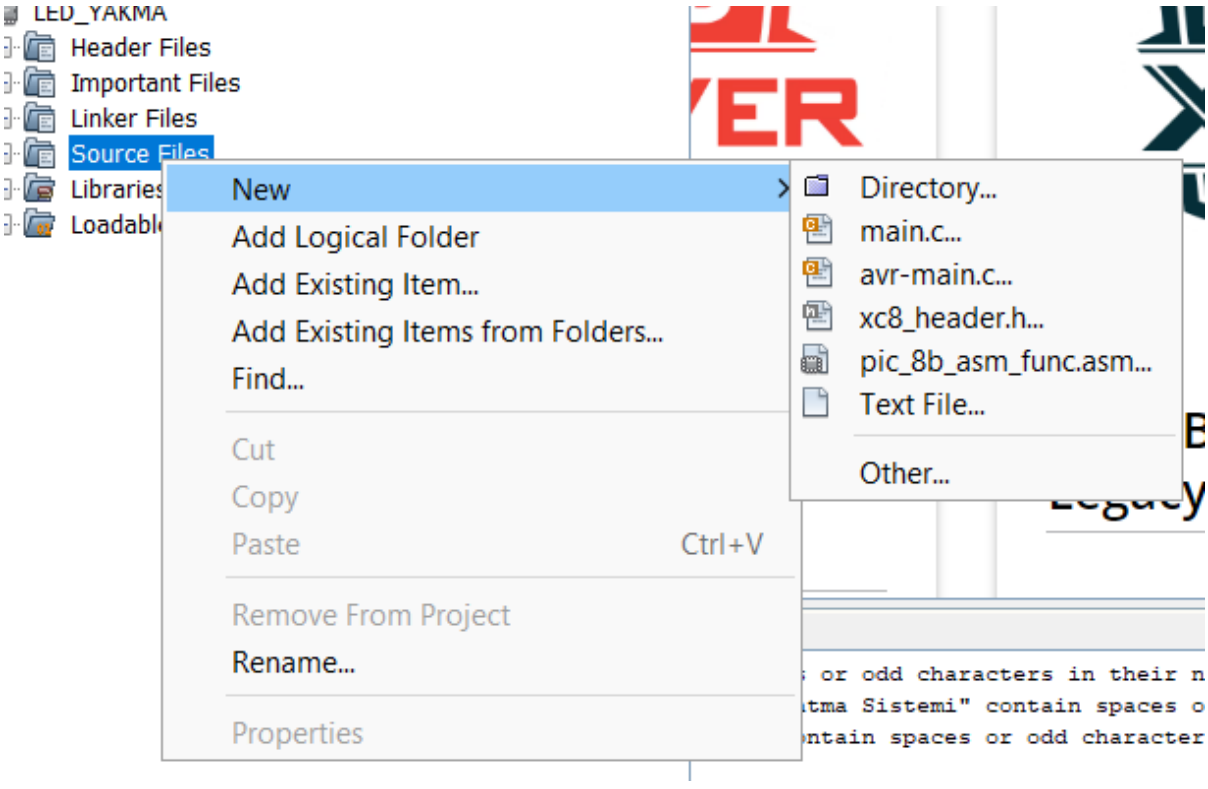
MPLAB uygulamasına girişin ardından New Project kısmı seçilir.



Üzerinde işlem yapılan mikrodenetliyici device kısmına yazılır.



Next dedikten sonra XC8 compileri seçilerek proje oluşturulur.



Projenin kodlanabilmesi için main.c klasörü oluşturulur ve ardından kodlamaya geçilir.

Projenin kodları aşağıda yanında açıklamalarıyla beraber verilmiştir.

Deney Kodları

1. Kodun Genel Yapısı

Kod, şu işlevleri içerir:

- LDR'den gelen **analog ışık seviyesi** sinyalini okumak için **ADC (Analog-Dijital Dönüştürücü)** kullanır.
- PIR sensöründen gelen **dijital hareket sinyalini** okur.
- Gelen verilere göre, bir LED grubunu kontrol eder.

2. Kütüphane ve Frekans Tanımlamaları

```
#include <xc.h>

// Mikrodenetleyicinin osilatör frekansı (4 MHz olarak
tanımlanmıştır)
#define _XTAL_FREQ 4000000
```

- **<xc.h> kütüphanesi**: Mikrodenetleyici için gerekli olan sabitleri ve donanım yapılandırmalarını içerir.
- **_XTAL_FREQ tanımı**: Mikrodenetleyicinin çalışma frekansı olarak 4 MHz belirtilmiştir. Bu, zamanlama işlemleri için önemlidir (örneğin `__delay_ms()` fonksiyonu).

3. Konfigürasyon Ayarları

```
#pragma config FOSC = XT      // XT Osilatör seçildi
#pragma config WDTE = OFF     // Watchdog Timer devre dışı bırakıldı
#pragma config PWRTE = ON     // Güç açılış zamanlayıcısı aktif
#pragma config BOREN = ON     // Brown-out Reset aktif
#pragma config LVP = OFF      // Düşük voltaj programlama devre dışı
bırakıldı
#pragma config CPD = OFF      // Veri belleği koruması devre dışı
```

```
#pragma config WRT = OFF      // Flash bellek yazma koruması devre
dışı
#pragma config CP = OFF       // Program belleği koruması devre dışı
```

Bu ayarlar, mikrodenetleyicinin çalışma modlarını belirler:

- **`FOSC = XT`**: XT tipi osilatör (kristal) seçilmiştir.
- **`WDTE = OFF`**: Watchdog Timer (sistem kontrol mekanizması) kapalıdır.
- **`PWRTE = ON`**: Güç açıldığında sistemin stabil hale gelmesini sağlayan zamanlayıcı aktif.
- **`BOREN = ON`**: Düşük voltaj durumunda mikrodenetleyici otomatik olarak resetlenir.
- Diğer ayarlar, düşük voltaj programlama ve bellek koruma özelliklerini devre dışı bırakır.

4. `main()` Fonksiyonu

Kodun ana fonksiyonunda, sensörlerin verileri okunur ve LED'ler kontrol edilir.

4.1 PORT Yapılandırması

```
TRISD = 0x00;  // PORTD'nin tamamını çıkış olarak ayarla (LED'ler
burada bağlanacak)
TRISA = 0x02;  // RA1 pini giriş, diğer pinler çıkış olarak
ayarlandı
ADCON1 = 0x06; // AN0 analog giriş, diğer pinler dijital giriş/çıkış
olarak ayarlandı
```

- **`TRISD`**: PORTD'nin tamamını çıkış olarak ayarlar. Bu port, LED'lerin bağlı olduğu pinlerdir.
- **`TRISA`**: PORTA'nın RA1 pini giriş, diğerleri çıkış olarak ayarlanır. **RA1**, PIR sensörden gelen dijital hareket sinyali için kullanılır.
- **`ADCON1`**: Analog-Dijital dönüştürme için **AN0 pini (RA0)** analog giriş olarak yapılandırılır. Diğer PORTA pinleri dijital olarak kullanılır.

4.2 Değişkenler

```
int isikSeviyesi;      // LDR'den gelen analog sinyali temsil eden
değişken
```

```
int isikEsigi = 512;   // Ortam ışık seviyesi eşik değeri
```

- **`isikSeviyesi`**: LDR sensöründen gelen analog değeri saklar.
- **`isikEsigi`**: Karar verme mekanizması için belirlenen eşik değeri. Eğer ışık seviyesi bu değerin altında ise ortam karanlık kabul edilir.

4.3 Sonsuz Döngü (while(1))

Kodun ana işlemleri, sonsuz bir döngü içinde sürekli olarak tekrar eder:

a) ADC Ayarları ve Dönüştürme

```
ADCON0 = 0x01;          // ADC girişini AN0 olarak seç
__delay_ms(2);          // ADC'nin stabilize olması için gecikme

GO_nDONE = 1;           // ADC dönüştürme işlemini başlat
while (GO_nDONE);       // Dönüştürme tamamlanana kadar bekle

isikSeviyesi = ((ADRESH << 8) + ADRESL); // 10-bit ADC sonucunu
hesapla
```

- **`ADCON0 = 0x01`**: ADC girişini **AN0 (RA0)** olarak seçer ve ADC'yi etkinleştirir.
- **`__delay_ms(2)`**: ADC'nin stabilize olması için kısa bir bekleme yapılır.
- **`GO_nDONE = 1`**: ADC dönüştürme işlemi başlatılır.
- **`while (GO_nDONE)`**: Dönüştürme işlemi tamamlanana kadar bekler.
- **`isikSeviyesi`**: ADC'nin 10-bit çıkışı hesaplanır ve değişkene atanır. ADC sonucu iki parçalıdır: üst 8 bit **ADRESH**, alt 2 bit **ADRESL**.

b) Koşullar ve LED Kontrolü

```
if (isikSeviyesi < isikEsigi || PORTAbits.RA1 == 1) {
    PORTD = 0x07; // Tüm LED'leri yak (PORTD'nin ilk üç pini)
} else {
    PORTD = 0x00; // LED'leri kapat
}
```

- **Karanlık Ortam (LDR)**: `isikSeviyesi < isikEsigi` olduğunda ortam karanlık kabul edilir ve LED’ler yanar.

- **Hareket Algılandığında (PIR)**: Eğer PIR sensöründen gelen sinyal aktif (RA1 = 1) ise yine LED'ler yanar.
- **Diğer Durumlar**: LED'ler kapalıdır.

5. Çalışma Prensibi

1. **Başlangıçta Yapılanlar**:
    a. PORTD çıkış, RA1 giriş ve RA0 analog giriş olarak ayarlanır.
    b. ADC modülü devreye alınır.
2. **Döngü İçindeki İşlemler**:
    a. LDR sensöründen gelen ışık seviyesi analog olarak okunur ve ADC ile dijital değere çevrilir.
    b. PIR sensöründen dijital hareket sinyali okunur.
    c. LDR veya PIR sensörlerinden birinde belirli bir eşik aşılırsa, LED'ler yakılır.
    d. Aksi takdirde LED'ler kapalı kalır