

Vision-Enhanced Adaptive Traffic Controller

Microprocessor Dersi Proje Raporu

1. Özet

Bu projede, Explorer 8 geliştirme kartı üzerindeki PIC16F1719 mikrodenetleyici ile **trafik ışığı kontrolü** gerçekleştirilmiştir. Sistem, bir **YOLO tabanlı görüntü işleme** uygulamasından gelen anlık araç yoğunluğu bilgisini UART üzerinden alır ve **kırmızı ışık süresini adaptif olarak uzatır**. Ayrıca, yaya geçişi isteği için bir buton kullanılmış, bu istekte sistem anlık olarak kırmızı faza geçmiştir. Tüm faz durumu ve ölçümler LCD üzerinde gösterilmiştir.

2. Hedefler

- Kırmızı/Yeşil LED'lerle temel trafik fazlarını oluşturmak.
- Python (YOLO) tarafından gönderilen araç yoğunluğunu **her 1 saniyede** almak.
- Sadece kırmızı fazda** yoğunluğa bağlı süre uzatma yapmak.
- Yaya butonu (S1) ile **anında kırmızıya geçiş** yapmak.
- LCD'de **faz, kalan süre, araç sayısı, yaya durumu ve uzatma bilgisi** göstermek.

3. Donanım Yapısı

Mikrodenetleyici: PIC16F1719

Geliştirme kartı: Microchip Explorer 8

Programlayıcı: PICkit 3

LCD: 16x2 karakter LCD (MCP23S17 SPI expander üzerinden)

LED: D1 (RD0) kırmızı, D2 (RD1) yeşil

Buton: S1 (RB0, active-low)

UART köprü: MCP2221 (Explorer 8 üzerindeki USB-UART)

LCD bağlantı notları (Explorer 8):

- MCP23S17 kullanılır (LCD doğrudan PIC'e bağlı değildir).
- Jumpers: **J59 (CS)**, **J60 (RESET)**, **J61 (LCD güç)**.

4. Yazılım Mimarisi

Sistem iki parçadan oluşur:

1. Python (YOLO) Uygulaması

- Video üzerinden araç tespiti ve takibi yapar.
- Anlık yoğunluğu (ekrandaki takip edilen araç sayısı) hesaplar.
- Her 1 saniyede bir UART üzerinden **C:<nn>\n** formatında gönderir.

2. PIC16F1719 Firmware (XC8 C)

- UART'dan **C:<nn>** mesajını parse eder.
- 1 saniyelik zamanlayıcıyla **bloklamasız** state machine çalıştırır.
- Kırmızı/yeşil LED'leri sürer.
- Pedestrian butonunu izler.
- LCD'ye durumu yazar.

5. UART Protokolü

Python → PIC

- Format: **C:<nn>\n**
- Örnek: **C:07\n**
- **nn** iki haneli 0–99 arası yoğunluk değeridir.

PIC tarafı

- Satır bazlı parsing yapılır.
- **vehicle_count** değişkeni güncellenir.

6. PIC Firmware Detayları (XC8)

PIC tarafında kullanılan temel modüller ve fonksiyonlar aşağıdaki gibidir:

6.1 Saat ve Konfigürasyon

- Dahili osilatör 8 MHz (**FOSC = INTOSC**), **#define _XTAL_FREQ 8000000**.
- WDT/LVP kapalı, MCLR açık.

6.2 SPI + MCP23S17 (LCD Sürücü Hattı)

- **SPI_Init()**
 - PPS yönlendirme (RC3=SCK, RC4=SDI, RC5=SDO).
 - **SSP1CON1**, **SSP1STAT** ile SPI Master mode (Fosc/16).
- **SPI_Transfer()**
 - Tek byte SPI gönderir ve yanıtı döner.
- **MCP_Write(reg, data)**
 - MCP23S17'e register yazımı için CS toggling yapar.
- **MCP_Init()**
 - Reset (J60), IOCON ayarı, portların output yapılması.

6.3 LCD Fonksiyonları (8-bit, MCP23S17 üzerinden)

- **LCD_Init()**
 - 8-bit init sırası, display on, entry mode, clear.
- **LCD_Command()**, **LCD_Char()**, **LCD_String()**
 - Komut/data gönderimi.
- **LCD_SetCursor()**
 - Satır ve kolon seçimi.
- **lcd_draw()**
 - Her 1 saniyede faz/süre/araç/ped/ext bilgisi basar.

6.4 UART / EUSART (Araç Yoğunluğu Alma)

- **UART_Init()**
 - 9600 baud, **BRG16=1**, **BRGH=1**, **SP1BRG=207**.
 - RC7=RX, RC6=TX yönlendirmesi.
- **UART_Read()**
 - OERR durumunda CREN resetlenir, RX devam eder.
- **UART_Write()** / **UART_WriteStr()**
 - Debug amaçlı kullanılabilir.
- **parse_line_and_update()**
 - **C:<nn>** formatını parse eder, **vehicle_count** güncellenir.

6.5 Timer1 ve Kesme Yapısı

- **Timer1_Init_10ms()**

- Timer1 prescaler 1:8, preload **0xF63C**.
- 10 ms kesme üretir.
- **__interrupt() isr()**
 - 10 ms sayacı ile 1 saniyelik tick üretir (**g_tick_1s**).

6.6 Trafik Fazı Kontrolü

- **set_phase()**
 - RED/GRN LED çıkışlarını ayarlar.
- Ana döngü:
 - UART satır okuma (bloklamasız).
 - 1 saniyede bir faz geri sayım.
 - Pedestrian request (S1) algılanırsa RED'e geçiş.
 - Kırmızı fazda yoğunluk eşiği aşılsa süre uzatma.

7. Kontrol Algoritması

Sistem iki faza sahiptir:

- **GREEN (Yeşil):** Sabit süre (10s).
- **RED (Kırmızı):** Taban süre (10s) + adaptif uzatma.

Adaptif Uzatma:

- Sadece RED fazında çalışır.
- Eğer **vehicle_count >= THRESHOLD (20)** ise **EXT_STEP (3s)** kadar süre uzatılır.
- Toplam kırmızı süre **RED_MAX (30s)** ile sınırlandırılır.

Yaya İsteği:

- S1 (RB0) basıldığında sistem anlık olarak kırmızıya geçer.
- **PED_HOLD** süresi kadar kırmızı kalır.
- LCD'de **PED:1** görünür.

8. LCD Arayüzü

LCD, her 1 saniyede bir güncellenir.

Satır 1:

PH:RED T:12 veya PH:GRN T:08

Satır 2:

CAR:07 PED:1 EXT

Alanlar:

- PH: aktif faz (RED/GRN)
- T: fazın kalan süresi (saniye)
- CAR: anlık araç yoğunluğu (00–99)
- PED: yaya isteği (1=aktif, 0=pasif)
- EXT: kırmızı uzatma aktifse görünür

9. Zamanlama ve Timer1 Hesabı

PIC iç osilatörü **8 MHz** kullanır.

- Instrüksiyon frekansı: $F_{osc}/4 = 2 \text{ MHz}$
- Timer1 prescaler: 1:8
- Timer1 tick: $4 \mu\text{s}$
- 10 ms için gereken tick sayısı: $10,000 \mu\text{s} / 4 \mu\text{s} = 2500$
- Preload: $65536 - 2500 = \text{0xF63C}$

Bu yapı ile **10 ms** kesme üretilir; yazılımsal sayaç ile **1 saniye** tick elde edilir.

10. Test Planı ve Sonuçlar

Test 1: LCD Başlatma

- LCD'de ilk satırda faz, ikinci satırda CAR/PED/EXT görünür.

Test 2: UART İletişimi

- Python'dan C:10\n gönderilince LCD'de CAR:10 güncellenir.

Test 3: Adaptif Kırmızı

- RED fazında araç sayısı THRESHOLD üstüne çıkınca EXT görünür ve kırmızı süre uzar.

Test 4: Yaya Butonu

- S1 basıldığında anlık kırmızıya geçer, **PED:1** olur.

11. Kullanılan Dosyalar

- **main.c** (PIC firmware, XC8)
- **main.py** (YOLO + UART gönderimi)

12. Sınırlamalar

- YOLO yoğunluğu anlık takip edilen araç sayısıdır; gerçek yoğunluk için zaman penceresi ortalaması daha doğru olabilir.
- Pedestrian davranışı “anlık kırmızı” şeklindedir; gerçek trafik güvenliği için “yeşili tamamla, sonra kırmızı” mantığı eklenebilir.
- UART tek yönlüdür; PIC geri bildirim (ACK) göndermemektedir.

13. Gelecek Çalışmalar

- Yoğunluğu son 5–10 saniye ortalaması ile hesaplamak.
- UART üzerinden çift yönlü protokol (ACK, hata kodu).
- Sarı faz eklemek ve güvenli geçişleri uzatmak.
- LCD’de geçmiş yoğunluk grafiği veya ortalama değer göstermek.

14. Çalıştırma Adımları

1. Jumpers: **J59, J60, J61** (LCD) ve **J54** (UART) takılı olsun.
2. MPLAB X ile **main.c** derle ve PICkit 3 ile programla.
3. Python tarafında **main.py** çalıştır.
4. LCD’de faz ve değerlerin güncellendiğini doğrula.