Rapor

Raporun konusu: “ ESP32-Cam ile Nesne Tanıma”

Doç. Dr. Erdal Özbay

Fırat Üniversitesi Bilgisayar Mühendisliği

3. Sınıf Bahar Dönemi Yazılım Mühendisliği Dersi

Son teslim tarihi: 12 Haziran 2025

Proje Üyeleri, Okul Numarası:

Hazan YÜCEL, 220260042

Hacer ÇADIRCI, 220260004

Esra ASLANBOĞA, 220260048

1. GİRİŞ

Projenin Amacı

Bu projenin amacı, ESP32-CAM modülü ve Edge Impulse platformunu kullanarak, düşük maliyetli ve enerji verimli bir nesne tanıma sistemi gerçekleştirmektir. Amaç, makine öğrenmesi tabanlı modelleri kaynak kısıtlı gömülü sistemlerde çalıştırabilir hale getirerek, edge AI yaklaşımıyla bulut bağımlılığını azaltmak ve verileri yerel olarak işlemek yoluyla gecikmeyi minimize etmektir.

Hedeflenen Uygulama ve Çıktı

Gerçek zamanlı nesne tanıma sistemi geliştirilerek, elde edilen sonuçlar LCD ekran ya da UART üzerinden kullanıcıya sunulacaktır. Eğitilen model ESP32-CAM modülüne deploy edilecek ve belirli nesneleri algıladığında ilgili çıktılar üretecektir.

Kapsam ve Kısıtlar

Proje, belirli sayıda sınıfa (meyve-sebze gibi) ait görüntülerle eğitilecek bir modeli kapsamaktadır. Kullanılacak cihaz ESP32-CAM olup, donanım kısıtları (RAM, flash hafıza, işlemci kapasitesi vb.) göz önüne alınarak model boyutu ve karmaşıklığı optimize edilecektir.

Kısaca Kullanılan Teknolojiler

- **ESP32-CAM**: Görüntü toplamak ve model çıktısını çalıştırmak için kullanılacak IoT modülü
- **Edge Impulse**: Veri toplama, öznitelik çıkarma, model eğitimi ve deployment süreçlerinin yürütüldüğü platform
- **FOMO(MobileNetV2 0.35)**: Nesne tanıma algoritması, alternatif yöntem olarak değerlendirilmektedir
- **TinyML**: Mikrodenetleyiciler üzerinde çalışabilecek şekilde optimize edilmiş ML yaklaşımı

2. PROJENİN GENEL TANIMI

Sistem Bileşenleri

- **ESP32-CAM**: Veri toplama, modele input sağlama ve sonuçları yönetme
- **LCD veya UART**: Model sonuçlarını ekrana yazma veya seri porttan iletme
- **Edge Impulse Cloud**: Model eğitimi ve öznitelik çıkarma
- **Veri seti elde edilmesi**: Eğitim verisi kaynağı oluşturma

Kullanılan Donanım ve Yazılım

- Donanım: ESP32-CAM, USB-UART modülü, breadboard, jumper kablolar
- Yazılım: Edge Impulse Studio, Arduino IDE, Github

Geliştirme Ortamı

Arduino IDE ESP32 için firmware yükleme ve kod düzenleme için kullanılmaktadır. Ek olarak proje geliştirme süreci Jira üzerinde Kanban yaklaşımıyla yapılmıştır. Aynı zamanda projeye ait kaynaklar ve Jira'daki işlemler Githubdaki depoya eklenmiştir.

Eğitim Süreci

1. **Veri Toplama**: ESP32-CAM modülü ile 4 sınıf sebzeye ait (salatalık, biber, patates, soğan) görüntüler toplandı. (50 foto/sınıf)
2. **Model Oluşturma**: Edge Impulse platformunda impulse oluşturuldu.
3. **Öznitelik Çıkarma**: Görüntülerden gerekli özellikler elde edildi, etiketlemeler yapıldı.
4. **Model Eğitimi**: TinyML uyumlu bir sinir ağı eğitildi model doğruluk oranı kontrol edildi.
5. **Deployment**: Eğitilen model ESP32-CAM'e yüklenerek sahada test edildi.

Projenin Ele Almayı Amaçladığı Sorun ve İhtiyaçlar

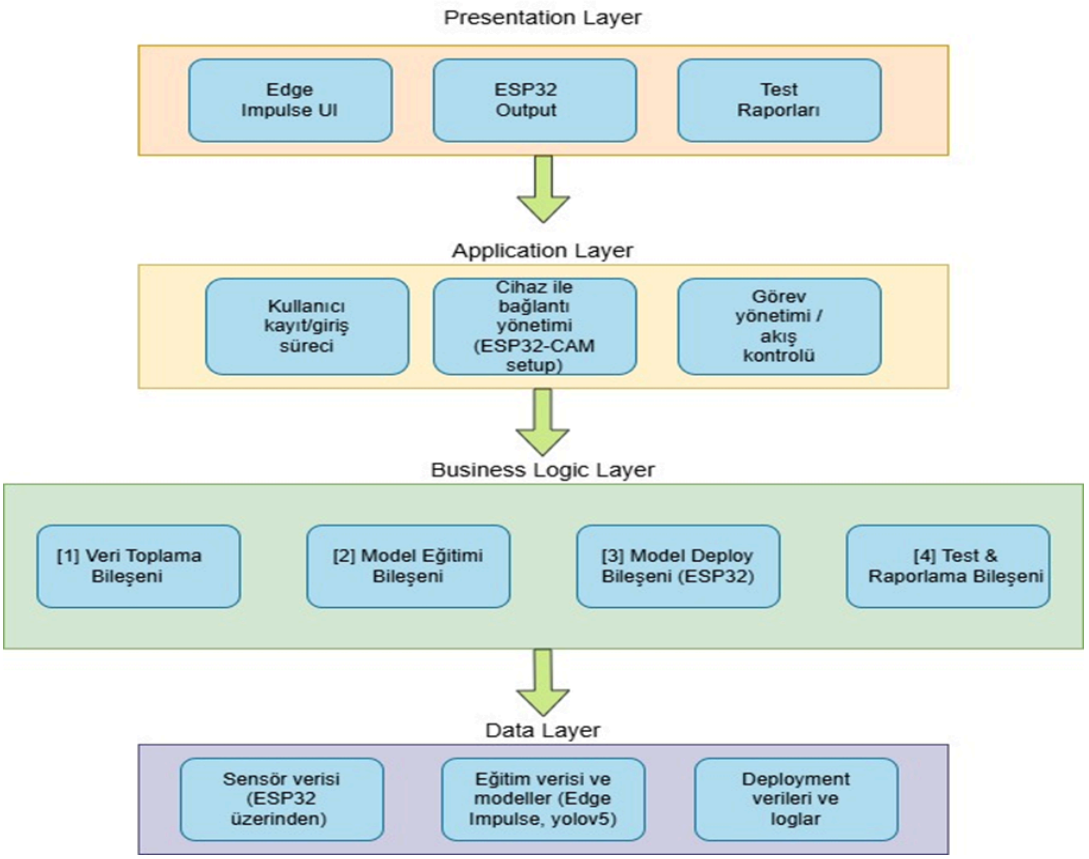
- Düşük donanımlı cihazlarda ML modeli çalıştırmak
- Gecikmeyi azaltmak ve bulut bağımlılığını kaldırmak
- Tarım, güvenlik gibi alanlarda yerel karar verebilen, enerji verimli sistemler tasarlamak

TinyML ve Edge AI gibi yaklaşımlar, bu sorunların çözümüne yönelik yüksek potansiyele sahiptir. Bu proje, bu potansiyelin gerçek dünya uygulamasını ortaya koymayı hedeflemektedir.

3. YAZILIM MİMARİSİ

Tercih Edilen Mimari Yapılar

Projemiz olan ESP32-CAM tabanlı, Edge Impulse ile geliştirilen nesne tanıma sistemi için Katmanlı Mimari ile Bileşen Tabanlı Mimarinin bir arada kullanıldığı hibrit bir mimari yapı tercih edilmiştir (Şekil 1). Bu tercih, projenin modüler, sürdürülebilir ve geliştirilebilir bir yapıya sahip olmasını sağlamıştır.



Şekil 1. Sistemin Katmanlı ve Bileşen Tabanlı Hibrit Mimarisi

Layered Architecture (Katmanlı Mimari)

Katmanlı mimari sayesinde sistem; sunum, uygulama, iş mantığı ve veri katmanları şeklinde mantıksal olarak ayrılmış, her katmanın belirli sorumluluklara sahip olması sağlanmıştır. Bu yapı, görevlerin düzenli ilerlemesine ve katmanlar arası bağımsızlığa imkân tanımıştır.

Katmanlar:

- **Sunum Katmanı**: Edge Impulse kullanıcı arayüzü, ESP32 çıktı ekranı, test ekranı
- **Uygulama Katmanı**: ESP32 bağlantı ve görev yönetimi
- **İş Mantığı Katmanı**:
    - Veri Toplama Bileşeni
    - Model Eğitimi Bileşeni
    - Model Dağıtımı Bileşeni
    - Test & Raporlama Bileşeni
- **Veri Katmanı**: ESP32 verileri, eğitim modelleri, çıktı logları

Component-Based Architecture (Bileşen Tabanlı Mimari)

İş Mantığı Katmanı, her biri belirli bir görevi gerçekleştiren bileşenlere ayrılmıştır. Her bileşen, sistemin tamamından bağımsız geliştirilebilir ve gerektiğinde yeniden kullanılabilir.

Bileşen	Açıklama
[1] Veri Toplama	ESP32-Cam modülü ile görüntülerin toplanması, Edge Impulse'a aktarılması
[2] Model Eğitimi	Özellik çıkarımı, eğitim, doğrulama
[3] Model Deploy	Eğitilen modelin ESP32'ye uygun hale getirilmesi
[4] Test & Raporlama	Gerçek zamanlı test ve çıktı yönetimi

Mimari Yapının Uygunluğu ve Sonuç

- Katmanlı yapı; veri işleme sürecinin ardışıklığına uyum sağlar.
- Bileşenler, modülerlik ve esneklik sağlar.
- Katmanlar arasında sorumluluk ayrımı, geliştirme ve bakım kolaylığı sunar.
- Edge Impulse ve ESP32 süreçlerinin ayrıştırılması, hibrit yapıya geçişi doğal kılar.

4. KULLANILAN TASARIM DESENLERİ (Design Patterns)

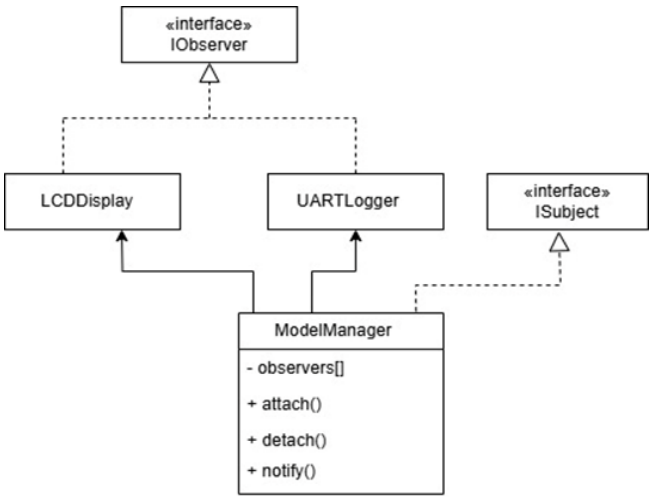
1. Observer Pattern (Gözlemci Deseni)

Intent: Bir nesnede değişiklik olduğunda, bu değişiklikten haberdar olması gereken diğer nesnelerin otomatik olarak bilgilendirilmesini sağlar.

Motivation: ESP32-CAM üzerinde çalışan TinyML modeli bir nesne algıladığında farklı bileşenlerin (örneğin LCD ekran, UART çıkışı) buna tepki vermesi gerekir. Observer pattern bu durumda bileşenlerin loosely-coupled şekilde yönetilmesini sağlar.

Applicability: Gözlemlenen bir nesnede (subject) durum değişiklikleri varsa ve bu değişikliklere tepki vermesi gereken bağımlı nesneler varsa kullanılır.

Structure: Model çıktı yöneticisi merkezi bir Subject görevi görür. LCD, UART gibi bileşenler Observer göreviyle bağlıdır. Model sonucu değişince hepsi bilgilendirilir.



Şekil 2. Gözlemci Deseni Diyagramı

Implementation: Kodda attach() veya notify() gibi fonksiyonlar tanımlanmasa da model çıktısı alındığında LCD ve UART eş zamanlı tepki verir. Her bileşen doğrudan değil, model sonucu yöneticisinden veri alır. Bu gevşek bağlı (loosely coupled) yapı Observer deseninin uygulandığını gösterir.

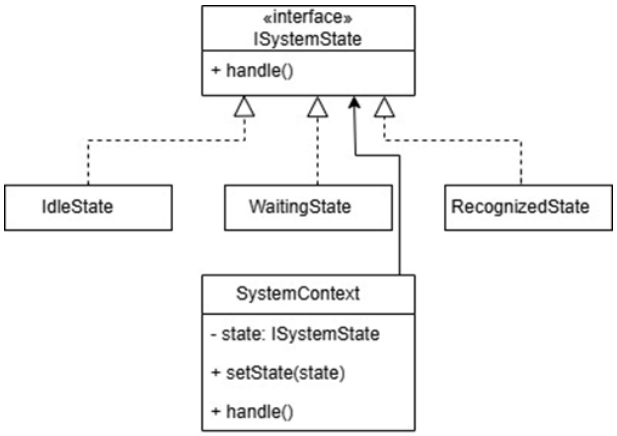
2. State Pattern (Durum Deseni)

Intent: Bir nesnenin iç durumu değiştiğinde, davranışını da değiştirmesini sağlar.

Motivation: Sistem üç ana durumda farklı davranır: Model yüklenmedi, model yüklendi ama nesne yok, nesne algılandı. Bu davranışlar farklı kontrol yapılarıyla yönetilir.

Applicability: Açıkça tanımlı birden fazla sistem durumu varsa ve her durumda farklı işlemler yapılması gerekiyorsa uygundur.

Structure: Sistem durumu bir context nesnesi ile temsil edilir. Her durum, farklı bir davranış sınıfı ile tanımlanır. Durumlar arasında geçiş context üzerinden yapılır.



Şekil 3. Durum Deseni Diyagramı

Implementation: Kod içinde Waiting, Processing, Detected gibi sınıflar kullanılmasa da her durum farklı davranış bloğu ile tanımlanmıştır. State Pattern yapısına mantıksal olarak uygunluk gösterir.

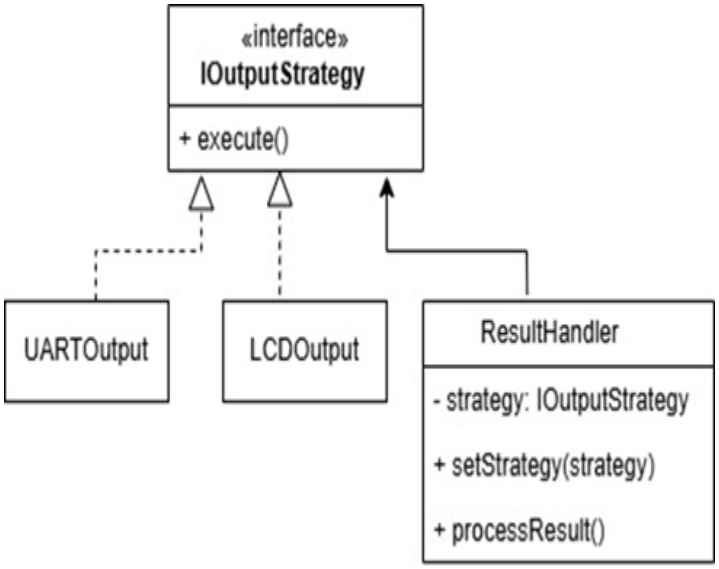
3. Strategy Pattern (Strateji Deseni)

Intent: Bir işlemi gerçekleştirme yöntemini dışarıdan belirlenebilir hale getirerek, sistemin esnekliğini artırır.

Motivation: Farklı nesneler farklı şekilde işlenmek istenebilir (LCD'ye yazdırmak, UART'tan göndermek gibi). Bu farklı stratejiler çalışma anında uygulanmalıdır.

Applicability: Farklı davranışlar aynı çatı altında yönetilmek isteniyorsa ve çalışma zamanında değiştirme gerekiyorsa uygundur.

Structure: ResultHandler gibi bir yapı, farklı strateji sınıfları ile çalışır. Her biri farklı bir davranış implementasyonu sunar.



Şekil 4. Strateji Deseni Diyagramı

Implementation: Kodda IStrategy sınıfı tanımlı olmasa da, if-else blokları veya switch-case kullanılarak farklı işlem yolları tanımlanmıştır. Kullanılan yöntem Strategy mantığını işlevsel olarak yansıtmaktadır.

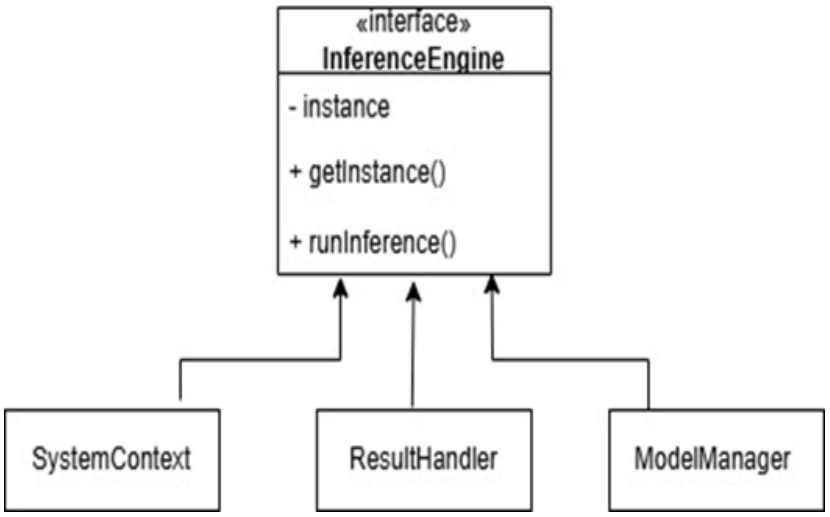
4. Singleton Pattern

Intent: Bir sınıfın yalnızca tek bir örneğinin olmasını sağlar ve bu örneğe global erişim sunar.

Motivation: UART, LCD gibi donanım kaynakları yalnızca bir kez oluşturulmalı ve sistemin her yerinde bu örnek kullanılmalıdır.

Applicability: Tek örneğin yeterli olduğu, global erişimin gerektiği ve tekrar oluşturmanın maliyetli olduğu durumlarda uygundur.

Structure: Tekil sınıf, private constructor ve static getInstance() fonksiyonu ile erişim sağlar.



Şekil 5. Singleton Pattern Diyagramı

Implementation: Kod içinde klasik Singleton yapısı olmasa da, LCD ve UART nesneleri yalnızca bir defa tanımlanmakta ve her yerden bu örnekler kullanılmaktadır. Bu yapı Singleton desenini işlevsel olarak sağlamaktadır.

5. YAZILIM GELİŞTİRME SÜRECİ (Jira & GitHub Entegrasyonu)

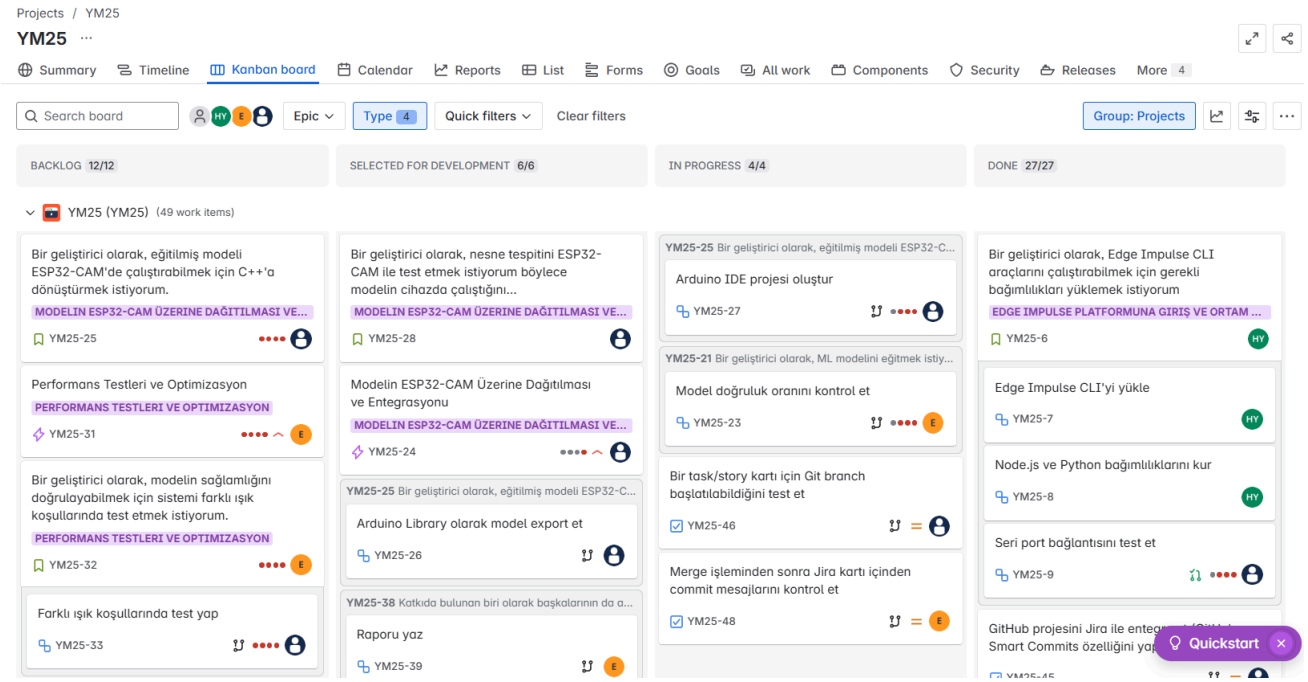
Bu projede yazılım geliştirme süreci, çevik (Agile) yazılım geliştirme yaklaşımlarından Kanban yöntemi temel alınarak yönetilmiştir. Takım üyeleri arasındaki görev paylaşımı, iş akışının takibi ve kod geliştirme süreçleri Jira ve GitHub platformlarının entegrasyonu ile yürütülmüştür. Süreç boyunca hem yazılımın teknik gelişimi hem de ekip içi koordinasyon bu iki platform üzerinden sağlanmıştır.

Jira Kanban Yönetimi Nasıl Yapıldı?

Proje için Jira üzerinde **Kanban şablonu** kullanılarak bir proje yönetim ortamı oluşturulmuştur.
Görevler, **Epic → Story → Task/Sub-task** yapısı altında tanımlanmış; iş akışı, dört temel sütun üzerinden yürütülmüştür:

- **Backlog:** Tanımlanmış ancak henüz planlamaya alınmamış işler
- **Selected for Development:** Geliştirmeye hazır işler
- **In Progress:** Üzerinde aktif olarak çalışılan işler
- **Done:** Tamamlanan görevler

Her görev kartına açıklayıcı başlıklar, açıklamalar ve varsa Jira-Git bağlantıları eklenmiştir. İşlerin takibi ve ilerleyişi bu yapı sayesinde görsel olarak izlenebilir hale getirilmiştir.



Görsel 1. Jira Yapısı

Epic, Story ve Task Örnekleri

Projede oluşturulan bazı Epic'ler ve ilgili alt görevler şu şekildedir:

- **Epic:** Edge Impulse Platformunda Model Eğitimi
    - **Story:** Verilerin toplanması ve yüklenmesi
        - **Task:** İnternetten esp32 cam modülüyle sebze görsellerinin toplanması
        - **Task:** Görsellerin Edge Impulse'a yüklenmesi
    - **Story:** Modelin eğitilmesi ve optimize edilmesi
        - **Task:** Impulse oluşturulması
        - **Task:** Özelliklerin çıkarılması ve model eğitimi
- **Epic:** Donanımsal Entegrasyon ve Testler
    - **Story:** ESP32-CAM modülünün yapılandırılması
        - **Task:** Seri port bağlantısının test edilmesi
        - **Task:** Kamera başlatma kodlarının çalıştırılması

- **Task:** LCD modülünün test edilmesi

Her bir görev Jira'da tanımlandıktan sonra, sorumlu takım üyelerine atandı.

Jira ile GitHub Entegrasyonu (Bağlantı Süreci)

Projede **GitHub – Jira entegrasyonu**, Atlassian'ın sunduğu "GitHub for Jira" eklentisi kullanılarak kurulmuştur.
Bu entegrasyon sayesinde:

- Jira görev kartlarının içinde ilgili GitHub **branch**, **commit** ve **pull request** (PR) bağlantıları görüntülenebilmiştir.

- Git commit mesajlarına görev ID'si (YM25-5 gibi) yazıldığında, bu commit otomatik olarak ilgili Jira kartında görünmüştür.

Entegrasyon sayesinde yazılım geliştirme süreçleri ve görev yönetimi senkronize biçimde ilerletilmiştir.

Feature Branch → Pull Request → Merge Süreci

Her task veya sub-task için, GitHub üzerinde **main** branch'ten türeyen bir **feature branch** açılmıştır.
Feature branch'ler şu formatta isimlendirilmiştir:

feature/YM25-5-esp32-cam-test

feature/YM25-8-lcd-display

Bu branch üzerinde yapılan değişiklikler commit'lenmiş, ardından GitHub üzerinden **Pull Request (PR)** açılmıştır. PR açıklamalarına ilgili Jira görev kodu da eklenmiştir.

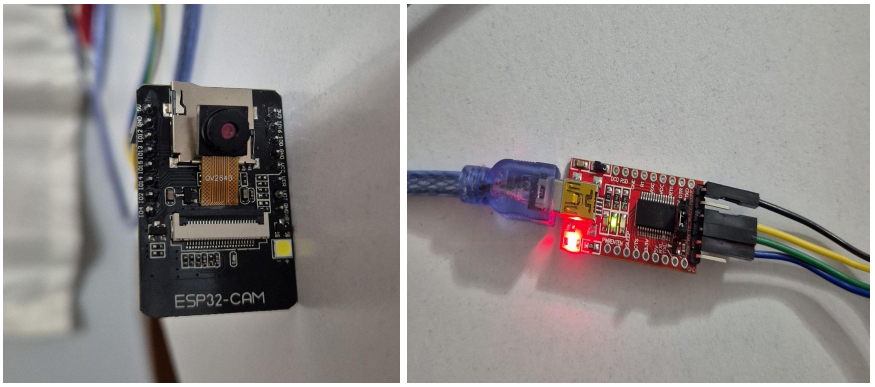
Takım üyeleri karşılıklı olarak PR'ları **code review** (kod inceleme) sürecine tabi tutmuş, öneri ve düzeltmeler PR üzerinde gerçekleştirilmiştir. Onaylanan PR'lar **main branch** ile birleştirilmiş (merge) ve feature branch'ler silinmiştir. Bu sayede GitHub'daki kod tabanı daima stabil ve yayınlanabilir durumda kalmıştır.

Takım İçinde Görev Dağılımı Nasıl Yapıldı?

Takım içindeki her üye belirli görevlerden sorumlu tutulmuştur. Jira üzerinde açılan görev kartları, üyeler arasında eşit ve ilgi alanlarına uygun şekilde atanmıştır.
Kodlama, test, dokümantasyon ve entegrasyon işlemleri belirli task'lara bölünerek farklı üyelerce gerçekleştirilmiştir. Ayrıca PR'ların onay süreci de takım üyeleri arasında karşılıklı olarak yürütülmüş, iş birliği içinde kod kalitesi artırılmıştır.

Bu yapı sayesinde proje süreci hem teknik olarak hem de proje yönetimi açısından çevik prensiplere uygun olarak tamamlanmıştır.

Devre Bağlantıları:



Görsel 2-3. Devre Elemanları

6. TEST SÜRECİ

Projemizde kullanılan ESP32-CAM donanımının ve çevresel bileşenlerin doğru bir şekilde çalıştığını teyit etmek amacıyla çeşitli test senaryoları hazırlanmış ve sistematik olarak uygulanmıştır. Bu testler, hem donanımın doğruluğunu kontrol etmeye hem de yazılım katmanlarının temel işlevlerini yerine getirip getirmediğini anlamaya yöneliktir. Test sürecinin sonunda tüm testler başarılı bir şekilde geçildiğinden dolayı model dağıtım aşamasına (deployment) geçilmesine karar verilmiştir.

ESP32-CAM donanımının test sürecinin yanı sıra, oluşturulan model de bilgisayar kamerasıyla test edilmiş ve sonuçları gözlemlenmiştir.

Yazılan Unit Test Örnekleri (Varsayılan Model Çıktısına Göre)

ESP32-CAM üzerinde çalışan sistemin model yüklenmeden önce temel bileşenlerinin sağlıklı çalıştığını test etmek amacıyla aşağıdaki **unit testleri** gerçekleştirilmiştir. Modülün her bir bileşeninin başarıyla çalıştığı gözlemlenince sorunun yetersiz güç veren v3 kablosunda olduğu anlaşılmıştır ve daha iyi veri aktarımı yapabilen yeni bir v3 kablosu takılarak sorun çözülmüştür :

1. **Seri port (UART) bağlantı testi**
2. **Kamera modül başlatma testi**
3. **Kamera pin konfigürasyonu doğrulama testi**
4. **PSRAM (harici bellek) testi**
5. **Model Testi**

Hangi Fonksiyonlar Test Edildi?

- Serial.begin(115200) → Seri iletişim başlatıldı mı?
- esp_camera_init(&config) → Kamera başlatılabiliyor mu?
- psramFound() → Harici PSRAM algılanabiliyor mu?
- heap_caps_get_total_size() / get_free_size() → Bellek ölçümleri doğru mu?
- Model yeni verilerle doğru tahmin yapabiliyor mu?

Test Senaryoları ve Çıktıları

Test 1: Basit Seri Port Bağlantı Testi

```
//basit seri port baglanti testi

void setup() {

 Serial.begin(115200);

 delay(1000); // Seri portun hazırlanmasını bekle

 Serial.println("Merhaba dünya! ESP32-CAM ayakta 🚀");}

void loop() {

 Serial.println("Loop çalışıyor...");

 delay(2000);}
```

Sonuç: Seri port üzerinden sürekli veri aktarımı başarıyla gözlemlenmiştir. ESP32-CAM cihazı çevre birimiyle iletişim kurabilmektedir.

Test 2: Kamera Başlatma Testi

```
#include "esp_camera.h"

void setup() {

 Serial.begin(115200);

 delay(1000);
```

```
Serial.println("📷 Kamera testi başlıyor...");

if (!psramFound()) {

  Serial.println("❌ PSRAM bulunamadı!");

} else {

  Serial.println("✅ PSRAM bulundu!");}

camera_config_t config;

config.ledc_channel = LEDC_CHANNEL_0;

config.ledc_timer = LEDC_TIMER_0;

config.pin_d0 = 5;

config.pin_d1 = 18;

config.pin_d2 = 19;

config.pin_d3 = 21;

config.pin_d4 = 36;

config.pin_d5 = 39;

config.pin_d6 = 34;

config.pin_d7 = 35;

config.pin_xclk = 0;

config.pin_pclk = 22;

config.pin_vsync = 25;

config.pin_href = 23;

config.pin_sscb_sda = 26;

config.pin_sscb_scl = 27;

config.pin_pwdn = 32;

config.pin_reset = -1;
```

```
 config.xclk_freq_hz = 20000000;
 config.pixel_format = PIXFORMAT_JPEG;
 if(psramFound()){
   config.frame_size = FRAMESIZE_QVGA;
   config.jpeg_quality = 12;
   config.fb_count = 1;
 } else {
   config.frame_size = FRAMESIZE_QQVGA;
   config.jpeg_quality = 12;
   config.fb_count = 1;}
 Serial.println("📷 Kamera başlatılıyor...");
 esp_err_t err = esp_camera_init(&config);
 if (err != ESP_OK) {
   Serial.printf("❌ Kamera başlatılamadı. Hata kodu: 0x%x\n", err);
   return;}
 Serial.println("🎉 Kamera başarıyla başlatıldı!");}
void loop() {
 delay(1000);}
```

Sonuç: Kamera konfigürasyonu doğru yapılmış, başlatma işlemi başarılı bir şekilde tamamlanmıştır.

Test 3: Kamera Giriş-Çıkış Pin Konfigürasyonu (AI Thinker Model)

```
#include "esp_camera.h"
// AI-THINKER modelini kullanıyoruz
#define PWDN_GPIO_NUM     32
```

```
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.println("📷 Kamera başlatılıyor...");
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer   = LEDC_TIMER_0;
  config.pin_d0       = Y2_GPIO_NUM;
  config.pin_d1       = Y3_GPIO_NUM;
```

```
config.pin_d2       = Y4_GPIO_NUM;
config.pin_d3       = Y5_GPIO_NUM;
config.pin_d4       = Y6_GPIO_NUM;
config.pin_d5       = Y7_GPIO_NUM;
config.pin_d6       = Y8_GPIO_NUM;
config.pin_d7       = Y9_GPIO_NUM;
config.pin_xclk     = XCLK_GPIO_NUM;
config.pin_pclk     = PCLK_GPIO_NUM;
config.pin_vsync    = VSYNC_GPIO_NUM;
config.pin_href     = HREF_GPIO_NUM;
config.pin_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn     = PWDN_GPIO_NUM;
config.pin_reset    = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
config.frame_size   = FRAMESIZE_QVGA;
config.jpeg_quality = 12;
config.fb_count     = 1;
if (psramFound()) {
  Serial.println("✅ PSRAM tekrar tespit edildi.");
  config.fb_count = 2;
  config.jpeg_quality = 10;}
esp_err_t err = esp_camera_init(&config);
```

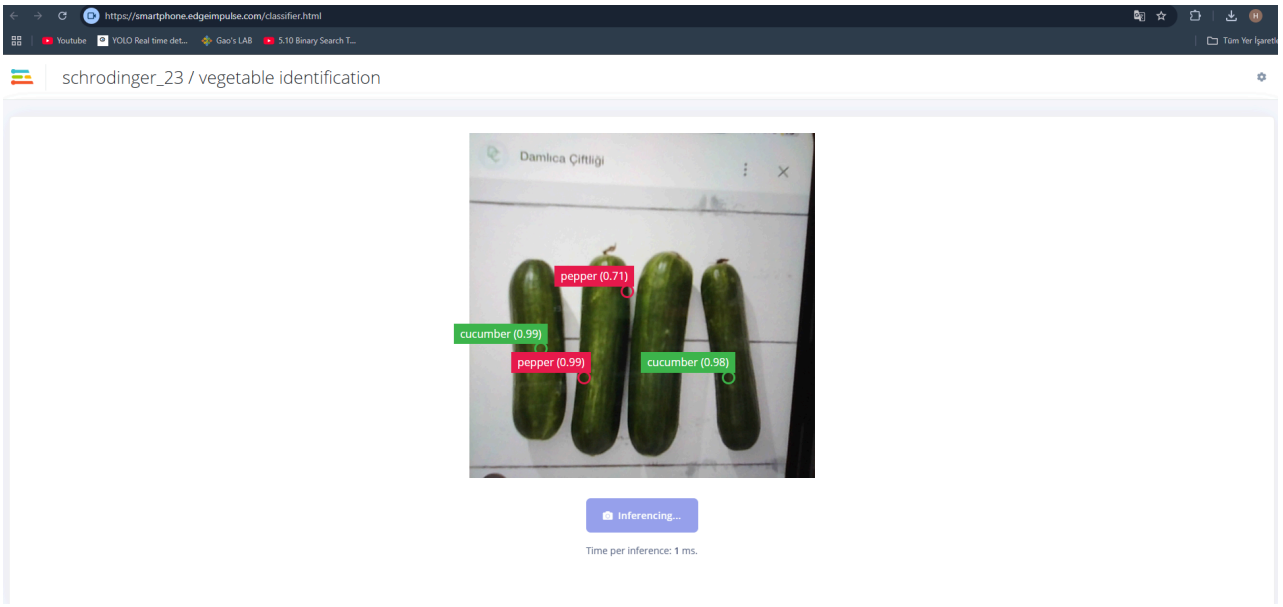
```
 if (err != ESP_OK) {
   Serial.printf("❌ Kamera başlatılamadı. Hata kodu: 0x%x\n", err);
   return;}
 Serial.println("✅ Kamera başarıyla başlatıldı!");}
void loop() {
 delay(1000);}
```

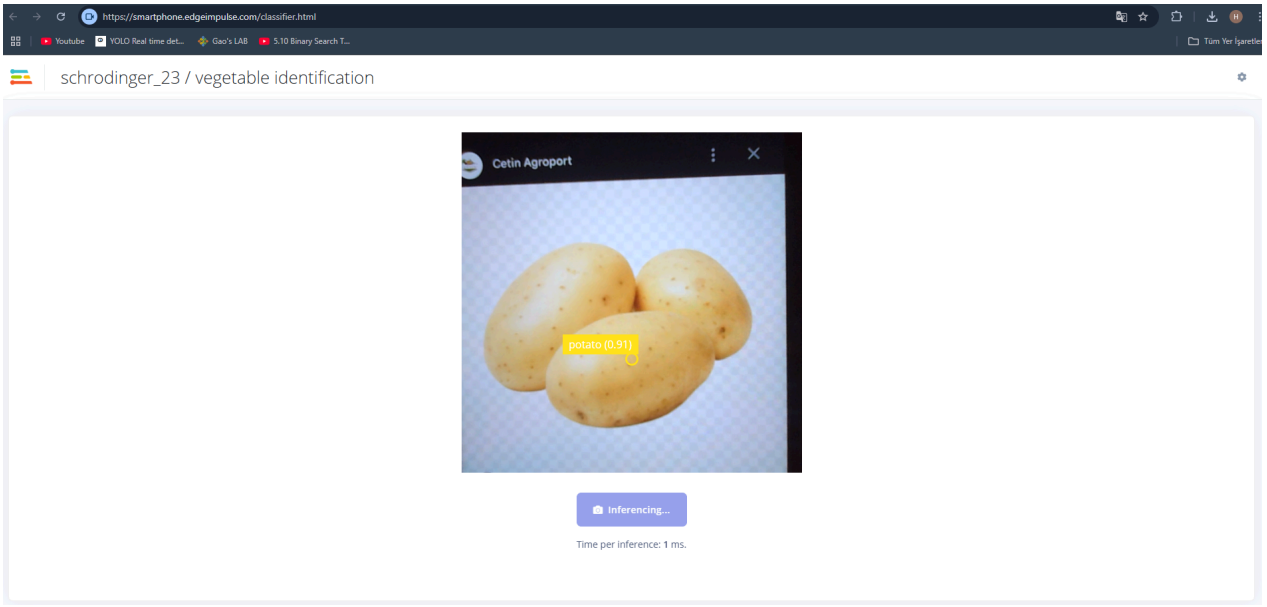
Sonuç: Kullanılan pin yapılandırması AI-THINKER ESP32-CAM modülü ile birebir uyumlu olarak yapılandırılmış, kamera başlatma işlemi sorunsuz gerçekleşmiştir.

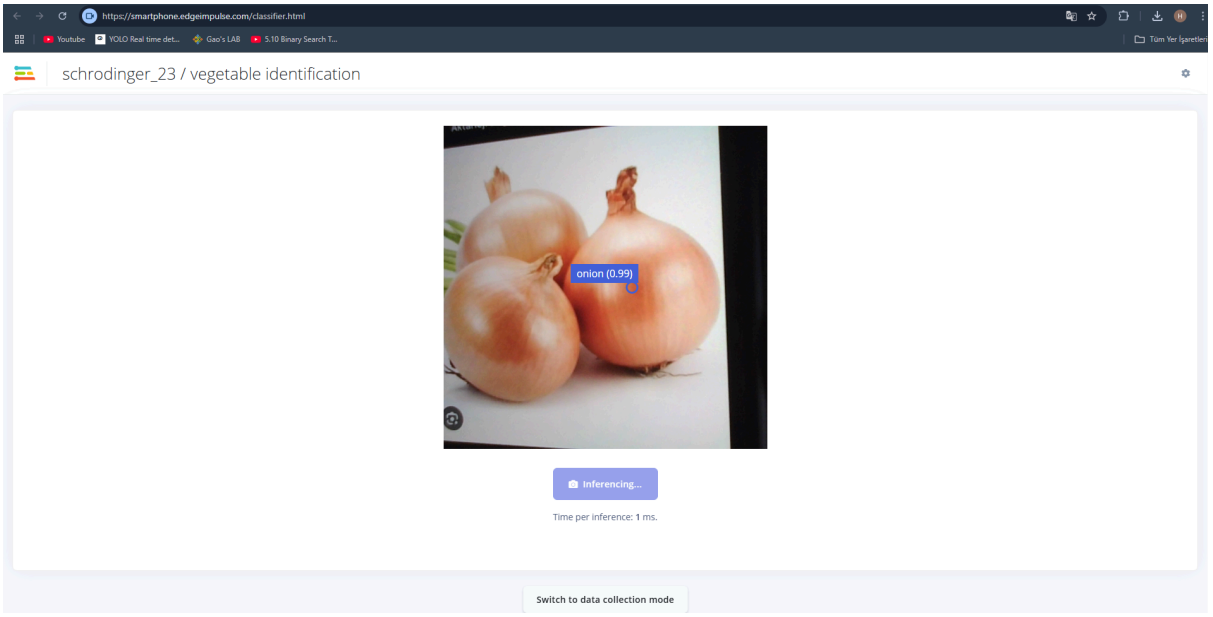
Test 4: PSRAM Bellek Testi

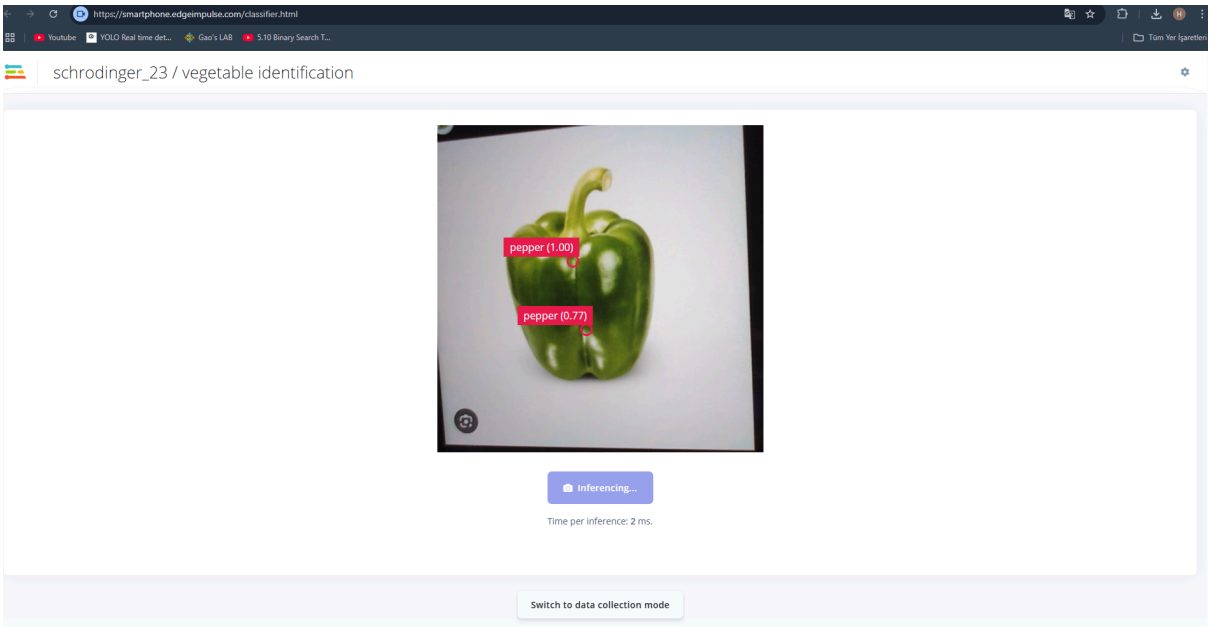
```
#include "esp_system.h"
#include "esp_heap_caps.h"
void setup() {
 Serial.begin(115200);
 delay(1000);
 if (psramFound()) {
   Serial.println("✅ PSRAM bulundu!");
   size_t total_psram = heap_caps_get_total_size(MALLOC_CAP_SPIRAM);
   size_t free_psram = heap_caps_get_free_size(MALLOC_CAP_SPIRAM);
   Serial.printf("📦 Toplam PSRAM: %u byte\n", total_psram);
   Serial.printf("📭 Boş PSRAM: %u byte\n", free_psram);
 } else {
   Serial.println("❌ PSRAM bulunamadı veya çalışmıyor.");}}
void loop() {
 delay(5000);}
```

Test 5: Model Testi









Sonuç: Bazı sınıfları tanımada zorluk çekiyor, bu da veri setinin ve modelin küçüklüğünden kaynaklanıyor olabilir.

Değerlendirme

Yukarıda yer alan testler sonucunda:

- ESP32-CAM donanımının temel iletişim altyapısı,
- Kamera biriminin donanımsal uyumluluğu ve yazılım üzerinden kontrolü,
- Harici belleğin tanınması ve belleğe erişim,

başarıyla doğrulanmıştır. Bu testlerin ardından, Edge Impulse üzerinden eğitilen modelin cihaz üzerine dağıtım aşamasına (deployment) geçilmesi uygun bulunmuştur. Böylece sistemin güvenilirliği temel düzeyde doğrulanarak sonraki adımlar için sağlam bir altyapı oluşturulmuştur.

8. KODLAR (KAYNAK KODLAR)

```
#YOLOv5 kurulumu
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
%pip install -r requirements.txt roboflow

#Roboflow veri setini indirme
from roboflow import Roboflow
rf = Roboflow(api_key="fiiEBNce3fTbjoRqdpHY")
project = rf.workspace("hazan").project("meyve-sebze-nan8q-uondr")
dataset = project.version(2).download("yolov5")

#YOLOv5n modeli ile eğitim
!python train.py --img 96 --batch 16 --epochs 50 --data
{dataset.location}/data.yaml --weights yolov5n.pt --cache

#Eğitilen modeli quantized TFLite formatında dışa aktarma
!python export.py --weights runs/train/exp/weights/best.pt --include
tflite
#TFLite modelini indirme
from google.colab import files
files.download('runs/train/exp/weights/best.tflite')
```

Sonuç: Bu model ile eğitim sonucu Edge Impulse sistemine aktarılmış fakat boyutu 6MB'ı aştığı için ESP32-CAM'e aktarılamamıştır. Bu nedenle daha düşük bir model ile Edge Impulse üzerinde eğitim yapılıp ESP32-CAM'e aktarılmıştır.

Edge İmpulse'tan deploy edilen Arduino kütüphanesi:

```
/* Includes ---------------------------------------------------------------- */

#include <vegetable_identification_inferencing.h>

#include "edge-impulse-sdk/dsp/image/image.hpp"

#include "esp_camera.h"

// Select camera model - find more camera models in camera_pins.h file here

//
https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Ca
mera/CameraWebServer/camera_pins.h

//#define CAMERA_MODEL_ESP_EYE // Has PSRAM

#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#if defined(CAMERA_MODEL_ESP_EYE)

#define PWDN_GPIO_NUM    -1

#define RESET_GPIO_NUM   -1

#define XCLK_GPIO_NUM    4

#define SIOD_GPIO_NUM    18

#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM      36

#define Y8_GPIO_NUM      37

#define Y7_GPIO_NUM      38

#define Y6_GPIO_NUM      39

#define Y5_GPIO_NUM      35

#define Y4_GPIO_NUM      14

#define Y3_GPIO_NUM      13

#define Y2_GPIO_NUM      34
```

```
#define VSYNC_GPIO_NUM   5
#define HREF_GPIO_NUM    27
#define PCLK_GPIO_NUM    25

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#else
#error "Camera model not selected"
```

```
#endif

/* Constant defines ------------------------------------------------------- */

#define EI_CAMERA_RAW_FRAME_BUFFER_COLS           320

#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS           240

#define EI_CAMERA_FRAME_BYTE_SIZE                 3

/* Private variables ------------------------------------------------------- */

static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal

static bool is_initialised = false;

uint8_t *snapshot_buf; //points to the output of the capture

static camera_config_t camera_config = {

    .pin_pwdn = PWDN_GPIO_NUM,

    .pin_reset = RESET_GPIO_NUM,

    .pin_xclk = XCLK_GPIO_NUM,

    .pin_sscb_sda = SIOD_GPIO_NUM,

    .pin_sscb_scl = SIOC_GPIO_NUM,

    .pin_d7 = Y9_GPIO_NUM,

    .pin_d6 = Y8_GPIO_NUM,

    .pin_d5 = Y7_GPIO_NUM,

    .pin_d4 = Y6_GPIO_NUM,

    .pin_d3 = Y5_GPIO_NUM,

    .pin_d2 = Y4_GPIO_NUM,

    .pin_d1 = Y3_GPIO_NUM,

    .pin_d0 = Y2_GPIO_NUM,
```

```
    .pin_vsync = VSYNC_GPIO_NUM,
    .pin_href = HREF_GPIO_NUM,
    .pin_pclk = PCLK_GPIO_NUM,

    //XCLK 20MHz or 10MHz for OV2640 double FPS (Experimental)
    .xclk_freq_hz = 20000000,
    .ledc_timer = LEDC_TIMER_0,
    .ledc_channel = LEDC_CHANNEL_0,

    .pixel_format = PIXFORMAT_JPEG, //YUV422,GRAYSCALE,RGB565,JPEG
     .frame_size = FRAMESIZE_QVGA,     //QQVGA-UXGA Do not use sizes above QVGA when not JPEG

    .jpeg_quality = 12, //0-63 lower number means higher quality
     .fb_count = 1,       //if more than one, i2s runs in continuous mode. Use only with JPEG
    .fb_location = CAMERA_FB_IN_PSRAM,
    .grab_mode = CAMERA_GRAB_WHEN_EMPTY,
};

/* Function definitions ------------------------------------------------- */
bool ei_camera_init(void);
void ei_camera_deinit(void);
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) ;
```

```
/**
* @brief      Arduino setup function
*/
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);
    //comment out the below line to start inference immediately after upload
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");
    if (ei_camera_init() == false) {
        ei_printf("Failed to initialize Camera!\r\n");
    }
    else {
        ei_printf("Camera initialized\r\n");
    }

    ei_printf("\nStarting continious inference in 2 seconds...\n");
    ei_sleep(2000);
}
/**
* @brief      Get data and run inferencing
*
* @param[in]  debug  Get debug info if true
```

```
*/

void loop()

{



    // instead of wait_ms, we'll wait on the signal, this allows threads to cancel us...

    if (ei_sleep(5) != EI_IMPULSE_OK) {

        return;

    }



        snapshot_buf = (uint8_t*)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS *
EI_CAMERA_RAW_FRAME_BUFFER_ROWS * EI_CAMERA_FRAME_BYTE_SIZE);



    // check if allocation was successful

    if(snapshot_buf == nullptr) {

        ei_printf("ERR: Failed to allocate snapshot buffer!\n");

        return;}

    ei::signal_t signal;

                        signal.total_length      =      EI_CLASSIFIER_INPUT_WIDTH      *
EI_CLASSIFIER_INPUT_HEIGHT;

    signal.get_data = &ei_camera_get_data;

                        if      (ei_camera_capture((size_t)EI_CLASSIFIER_INPUT_WIDTH,
(size_t)EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) == false) {

        ei_printf("Failed to capture image\r\n");

        free(snapshot_buf);

        return;}
```

```
    // Run the classifier
    ei_impulse_result_t result = { 0 };
    EI_IMPULSE_ERROR err = run_classifier(&signal, &result, debug_nn);
    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", err);
        return;}
    // print the predictions
    ei_printf("Predictions (DSP: %d ms., Classification: %d ms., Anomaly: %d ms.): \n",
                result.timing.dsp, result.timing.classification, result.timing.anomaly);
#if EI_CLASSIFIER_OBJECT_DETECTION == 1
    ei_printf("Object detection bounding boxes:\r\n");
    for (uint32_t i = 0; i < result.bounding_boxes_count; i++) {
        ei_impulse_result_bounding_box_t bb = result.bounding_boxes[i];
        if (bb.value == 0) {
            continue;}
        ei_printf("  %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\r\n",
                bb.label,
                bb.value,
                bb.x,
                bb.y,
                bb.width,
                bb.height);}


    // Print the prediction results (classification)
```

```
#else
    ei_printf("Predictions:\r\n");
    for (uint16_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT; i++) {
        ei_printf("  %s: ", ei_classifier_inferencing_categories[i]);
        ei_printf("%.5f\r\n", result.classification[i].value);
    }
#endif


    // Print anomaly result (if it exists)
#if EI_CLASSIFIER_HAS_ANOMALY
    ei_printf("Anomaly prediction: %.3f\r\n", result.anomaly);
#endif


#if EI_CLASSIFIER_HAS_VISUAL_ANOMALY
    ei_printf("Visual anomalies:\r\n");
    for (uint32_t i = 0; i < result.visual_ad_count; i++) {
        ei_impulse_result_bounding_box_t bb = result.visual_ad_grid_cells[i];
        if (bb.value == 0) {
            continue;
        }
        ei_printf("  %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\r\n",
                bb.label,
                bb.value,
                bb.x,
```

```
                bb.y,
                bb.width,
                bb.height);
    }
#endif
    free(snapshot_buf);}
/**
 * @brief   Setup image sensor & start streaming
 *
 * @retval  false if initialisation failed
 */
bool ei_camera_init(void) {
    if (is_initialised) return true;
#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif
    //initialize the camera
    esp_err_t err = esp_camera_init(&camera_config);
    if (err != ESP_OK) {
      Serial.printf("Camera init failed with error 0x%x\n", err);
      return false;}
    sensor_t * s = esp_camera_sensor_get();
    // initial sensors are flipped vertically and colors are a bit saturated
```

```
    if (s->id.PID == OV3660_PID) {
      s->set_vflip(s, 1); // flip it back
      s->set_brightness(s, 1); // up the brightness just a bit
      s->set_saturation(s, 0); // lower the saturation
    }
#if defined(CAMERA_MODEL_M5STACK_WIDE)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
#elif defined(CAMERA_MODEL_ESP_EYE)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
    s->set_awb_gain(s, 1);
#endif
    is_initialised = true;
    return true;}
/**
 * @brief      Stop streaming of sensor data
 */
void ei_camera_deinit(void) {
    //deinitialize the camera
    esp_err_t err = esp_camera_deinit();
    if (err != ESP_OK){
        ei_printf("Camera deinit failed\n");
        return;}
```

```
    is_initialised = false;

    return;}

/**

 * @brief      Capture, rescale and crop image

 *

 * @param[in]  img_width     width of output image

 * @param[in]  img_height    height of output image

 * @param[in]  out_buf       pointer to store output image, NULL may be used

 *                           if ei_camera_frame_buffer is to be used for capture and resize/cropping.

 *

 * @retval     false if not initialised, image captured, rescaled or cropped failed

 *

 */

bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) {

    bool do_resize = false;


    if (!is_initialised) {

        ei_printf("ERR: Camera is not initialized\r\n");

        return false;

    }

    camera_fb_t *fb = esp_camera_fb_get();

    if (!fb) {

        ei_printf("Camera capture failed\n");
```

```
      return false;
    }
  bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT_JPEG, snapshot_buf);
  esp_camera_fb_return(fb);
  if(!converted){
      ei_printf("Conversion failed\n");
      return false;
  }
   if ((img_width != EI_CAMERA_RAW_FRAME_BUFFER_COLS)
       || (img_height != EI_CAMERA_RAW_FRAME_BUFFER_ROWS)) {
       do_resize = true;
    }
   if (do_resize) {
       ei::image::processing::crop_and_interpolate_rgb888(
       out_buf,
       EI_CAMERA_RAW_FRAME_BUFFER_COLS,
       EI_CAMERA_RAW_FRAME_BUFFER_ROWS,
       out_buf,
       img_width,
       img_height);
    }
   return true;}
static int ei_camera_get_data(size_t offset, size_t length, float *out_ptr)
{
```

```
    // we already have a RGB888 buffer, so recalculate offset into pixel index
    size_t pixel_ix = offset * 3;
    size_t pixels_left = length;
    size_t out_ptr_ix = 0;
    while (pixels_left != 0) {
        // Swap BGR to RGB here
        // due to https://github.com/espressif/esp32-camera/issues/379
        out_ptr[out_ptr_ix] = (snapshot_buf[pixel_ix + 2] << 16) + (snapshot_buf[pixel_ix + 1] << 8) + snapshot_buf[pixel_ix];
        // go to the next pixel
        out_ptr_ix++;
        pixel_ix+=3;
        pixels_left--;}
    // and done!
    return 0;}
#if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_CAMERA
#error "Invalid model for current sensor"
#endif
```

9. SONUÇ ve DEĞERLENDİRME

Bu proje kapsamında, düşük kaynak tüketimli gömülü bir sistem olan **ESP32-CAM** ile gerçek zamanlı nesne tanıma uygulaması gerçekleştirilmiştir. **Edge Impulse** platformu sayesinde hızlı ve verimli bir eğitim süreci yürütülmüş, **FOMO (MobileNetV2 0.35)** mimarisi ile donanımsal kısıtlamalara uygun, hafif ancak işlevsel bir model deploy edilmiştir.

ESP32-CAM modülü üzerinde çalışan sistem, dört farklı sebze sınıfını (salatalık, biber, patates, soğan) ayırt edebilmiş; UART üzerinden tahmin çıktıları başarıyla kullanıcıya iletilmiştir. Kodun modüler yapısı sayesinde, farklı donanımlarla entegrasyonu ve sınıf sayısının artırılması mümkündür.

Süreç içerisinde veri toplama, model eğitme, test etme ve dağıtma aşamaları sistematik olarak uygulanmıştır. Nihai olarak:

- ESP32-CAM cihazı kullanılarak yerel veri toplama işlemi başarıyla gerçekleştirilmiş,
- Etiketlenen veriler Edge Impulse platformuna yüklenmiş ve buradan model eğitimi gerçekleştirilmiş,
- Eğitilen model cihaza deploy edilmiş ve test edilmiştir,
- Gerçek zamanlı nesne tanıma başarıyla sağlanmıştır.

Nihai modelin test sürecinde elde edilen F1 skorları ortalama olarak %78–%94 aralığında değişmiştir. Bu değerler, gömülü sistem üzerinde çalışan bir TinyML modeline göre oldukça tatmin edicidir.

Proje sadece teknik bir başarı değil; aynı zamanda **TinyML** ve **Edge AI** kavramlarının gerçek dünyada uygulanabilirliğini de ortaya koymuştur. Sistem, buluta bağlı olmadan yerel olarak karar verebilmekte, bu da veri güvenliği ve işlem hızı açısından büyük avantaj sağlamaktadır.

Gelecekteki Uygulama Alanları

- **Tarım**: Meyve/sebze tanıma, hastalıklı ürün ayıklama
- **Güvenlik**: Kamera tabanlı alan izleme
- **Perakende**: Otomatik ürün tanıma sistemleri
- **Endüstriyel otomasyon**: Hat üzerindeki nesne denetimi

Karşılaşılan Sorunlar ve Alınan Önlemler

Proje süreci boyunca birçok farklı yöntem ve senaryo denenmiş, bunların bir kısmı yeterli performans sağlamamış veya sistemin donanımsal sınırları nedeniyle uygun olmamıştır. Aşağıda bu durumlar ve çözüm yolları özetlenmiştir:

1. İlk Plan: Sebze Tanıma

- Başlangıçta hedeflenen sınıflar: *domates, biber, patlıcan, salatalık, havuç*
- Hazır veri setleri araştırıldı fakat görüntülerin çözünürlükleri yüksekti → **ESP32 belleği yetersiz kaldı**
  Çözüm: Farklı nesne sınıflarına yönelmek düşünüldü.

2. Alternatif Plan: Çiçek Tanıma

- Takım üyesi tarafından “çiçek türleri” içeren yüksek kaliteli bir veri seti bulundu ve etiketleme işlemleri gerçekleştirildi.
- Edge Impulse'a veri yüklendi, model eğitildi (F1 skoru: **%39,6**) ancak modelin F1 skoru çok düşüktü, model öğrenme yapamadı ve çok büyük bir model olduğu için ESP32-CAM’e sığmadı.

Çözüm:
Edge Impulse üzerinde model dışarıda eğitilip platforma .onnx olarak yüklendi. Yine de deploy sırasında cihaz sınırına takıldı.
Sonunda çiçekler yerine tekrar sebze tanıma problemine dönüldü.

3. Sonuç: Görüntüleri ESP32-CAM ile Kendimiz Topladık

- Donanım üzerinden (ESP32-CAM) 4 sınıf için 50-60 görüntü çekildi.
- Edge Impulse ile bu veriler etiketlenerek model eğitildi.
- İlk modelin F1 skoru: **%94** fakat yüksek skorun sebebi veri miktarının azlığı ve modelin veriye aşırı uyum göstermesi (overfitting) olabilir.

Çözüm:
Learning rate değeri düşürüldü, model yeniden eğitildi → F1 skoru: **%78**
Genelleme kabiliyeti artarken doğruluk dengelenmiş oldu.

Model Seçimi: Neden FOMO (MobileNetV2) Kullandık?

Model eğitimi sürecinde, nesne tanıma görevine uygun birçok farklı mimari araştırılmıştır. İlk olarak, nesne tespiti alanında yaygın kullanılan **YOLO (You Only Look Once)** mimarisi değerlendirilmiş; ancak YOLO tabanlı modellerin yüksek parametre sayısı ve büyük model boyutu nedeniyle, **ESP32-CAM gibi bellek ve işlem gücü açısından kısıtlı donanımlarda çalıştırılamayacağı** görülmüştür.

Bu nedenle, **Edge Impulse tarafından TinyML uygulamaları için önerilen FOMO (Fast Object Detection for Mobile)** model mimarisi tercih edilmiştir. FOMO, **MobileNetV2** tabanlı bir yapıya sahiptir ve hem sınıflandırma hem de konumlama (bounding box yerine grid tabanlı nokta tahmini) işlemlerini düşük kaynak tüketimiyle gerçekleştirebilmektedir.

FOMO modelinin tercih edilme sebepleri şunlardır:

- ESP32-CAM gibi mikrodenetleyici sınıfı cihazlarda **çalıştırılabilecek kadar hafif** olması,
- Gerçek zamanlı nesne tespiti için **yüksek hızda çalışabilmesi**,
- Eğitim ve deploy sürecinde **Edge Impulse platformu ile tam uyumlu olması**,
- Küçük boyutlu veri setlerinde **aşırı öğrenmeye (overfitting) daha dayanıklı** bir yapı sunması.

Bu sayede model, cihaz üzerinde hem enerji verimliliğini hem de çalışma kararlılığını koruyarak başarılı sonuçlar vermiştir.

Işık Koşullarının Model Performansına Etkisi

Sistem, farklı aydınlatma düzeylerinde test edilerek modelin karanlık, düşük ışık, doğal ışık ve yapay ışık gibi koşullarda nesne tanıma başarımı gözlemlenmiştir.

Test sırasında aşağıdaki yöntem uygulanmıştır:

- ESP32-CAM ile tanıma yapan cihaz sabit tutulmuş,
- Farklı nesne görüntüleri mobil telefon ekranından gösterilmiş,
- Telefonun ekran parlaklığı değiştirilerek aydınlatma koşulları simüle edilmiştir.

Gözlem:

Telefon ekranının parlaklığı düşürüldüğünde, modelin nesneleri daha isabetli şekilde tanıdığı fark edilmiştir. Yüksek ekran parlaklığında ise görüntüde parlamalar (glare) oluşmuş ve modelin tespit kararlılığı azalmıştır.

Sonuç:

ESP32-CAM'in kamera sensörü, aşırı parlak görüntülerde doğru pozlamayı sağlayamayarak modelin performansını olumsuz etkileyebilir. Ekran parlaklığının azaltılması, daha doğal bir görüntü sunarak nesne tanıma performansını iyileştirmiştir. Bu sonuç, sistemin ışık duyarlılığı yüksek olduğunu ve gerçek dünya koşullarında dikkatli kalibrasyon yapılması gerektiğini göstermektedir.

Karşılaşılan Sorunlar ve Çözümler:

Karşılaşılan Sorun	Açıklama	Çözüm
PSRAM Algılanamaması	Kamera çalıştırılmadan önce PSRAM (harici bellek) bazen tanınmadı	Daha kaliteli bir USB-V3 kablosu kullanıldı, kablo değişimi ile sorun çözüldü
Görüntü Dönüşüm Hataları	JPEG formatından RGB888'e dönüştürme sırasında görüntü bozulmaları oluştu	fmt2rgb888() fonksiyonu optimize edilerek dönüştürme işlemi iyileştirildi
Kamera başlatılamadı hatası	esp_camera_init() çağrısında bazı kartlarda hata alındı	Pin tanımlamaları AI Thinker modeline uygun şekilde güncellendi ve PSRAM desteği doğrulandı
Model hafıza aşımı (Out of memory)	Eğitim sonrası modelin boyutu ESP32 RAM kapasitesini aştı	Model mimarisi olarak FOMO + MobileNetV2 (0.35) seçildi; bu daha

		küçük parametreli bir modeldir
İletişim gecikmesi	Seri portta veri gecikmeleri yaşandı	Baud rate ve delay süreleri optimize edilerek iletişim stabil hale getirildi

Geliştirilebilir Yönler

Projenin temel amacı olan düşük güçlü cihazlarda nesne tanıma görevi başarıyla sağlanmış olsa da, sistemin daha ileri seviyelere taşınabilmesi için aşağıdaki geliştirme önerileri sunulmuştur:

Geliştirilebilir Alan	Açıklama
Nesne Sınıfı Sayısının Artırılması	Mevcut sistem 4 sebze sınıfını tanıyacak şekilde eğitilmiştir. Farklı gıda ürünleri, meyveler veya endüstriyel nesnelerle daha büyük bir sınıf yelpazesi oluşturularak modelin kapsamı genişletilebilir.
LCD/GUI Entegrasyonu	Seri port üzerinden verilen çıktılar, küçük bir LCD ekran ya da dokunmatik arayüz ile kullanıcı dostu hale getirilebilir.
Model Performans Analizi (F1, Confusion Matrix)	Şu an sadece doğruluk (accuracy) değeri baz alındı. Precision, recall, F1-score gibi daha detaylı ölçütlerle model kalitesi daha bilimsel olarak değerlendirilebilir.
Mobil veya Web Arayüz Desteği	ESP32-CAM ile algılanan nesneler, bir mobil uygulama veya web arayüzü ile uzaktan görüntülenebilir hale getirilebilir.