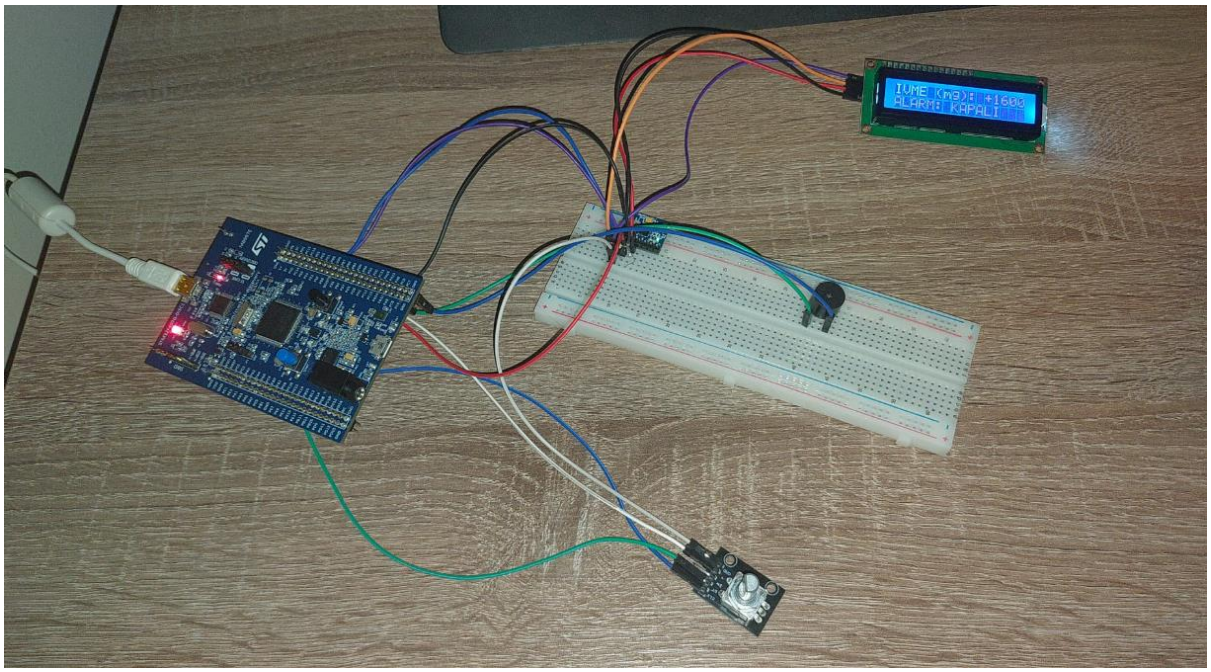
EMBEDDED SYSTEMS BİTİRME PROJESİ STM32 Tabanlı Rotary Encoder ve I2C Haberleşme ile İvme Simülasyonu ve Alarm Sistemi



EREN EMANET

HAMİT KARABULUT

CAN KAYRA UZUN

1. Giriş

1.1 Proje Amacı ve Kapsamı

Gömülü sistemler, gerçek dünya verilerini işleyerek fiziksel süreçleri kontrol etme yeteneği nedeniyle modern endüstri ve tüketici elektroniğinde merkezi bir rol oynamaktadır. Bu makale, popüler bir mikrodenetleyici platformu olan **STM32F407VGTX**'i kullanarak, bir rotary encoder aracılığıyla sanal bir ivme değerini kontrol etmeyi ve bu değeri bir **I2C LCD** ekranında göstermeyi amaçlayan bir sistemi detaylandırmaktadır. Projenin temel hedefi, döner kodlayıcıyı dijital bir arayüz olarak kullanarak, kullanıcı tarafından belirlenen bir eşiği aşan sanal ivme değerine bir buzzer ile akustik alarm tepkisi vermektir.

Bu sistem, üç temel bileşenin entegrasyonunu vurgular:

1. **Döner Kodlayıcı:** Kullanıcı girişi için hassas dijital kontrol arayüzü olarak **Zamanlayıcı (Timer) Çevirici Modu** kullanılarak.
2. **STM32 HAL Katmanı:** Tüm çevre birimlerinin başlatılması ve kontrolü için.
3. **I2C Haberleşme:** Hem **LCD Ekran** hem de potansiyel ilerideki genişlemeler için **MPU6050** gibi sensörlerle iletişim kurmak için.

1.2 Kullanılan Donanım ve Geliştirme Ortamı

Bileşen	Görev	Bağlantı Noktası
Mikrodenetleyici	STM32F407VGTX	Ana İşlemci
Döner Kodlayıcı (Rotary Encoder)	Kullanıcı Girişi (Sanal İvme Kontrolü)	PA0 (CLK/A) ve PA1 (DT/B) - TIM2
I2C LCD (16x2)	Veri Görselleştirme	PB6 (SCL) ve PB7 (SDA) - I2C1
Buzzer	Akustik Alarm Çıkışı	PA8 (GPIO Output)
MPU6050	İvme sensörü (Başlatma/Gelecek kullanım için)	I2C1
Geliştirme Ortamı	STM32CubeIDE (GCC, HAL)	C/C++

2. Donanım ve Yapılandırma Detayları

2.1 STM32 Çevre Birimleri ve Pin Yapılandırması

Projenin güvenilir çalışması, mikrodenetleyicinin çevre birimlerinin doğru şekilde yapılandırılmasına bağlıdır. Bu projede kullanılan kritik pin yapılandırmaları ve modül ayarları şunlardır:

2.1.1 Döner Kodlayıcı (Timer 2 Encoder Modu)

Döner kodlayıcı, iki fazlı darbe dizisi (CLK/A ve DT/B) üreten elektromekanik bir cihazdır. Bu darbelerin yön ve hızını algılamak için **STM32'nin Gelişmiş Zamanlayıcıları (Timer)** kullanılır.

- **TIM2 Modülü:** Encoder arayüz modunda yapılandırılmıştır.
- **Kanallar:** Channel 1 (PA0) ve Channel 2 (PA1) giriş pinleri olarak ayarlanmıştır.
- **Çalışma Prensibi:** TIM2, **TIM_ENCODERMODE_TI12** modunda çalışır. Bu mod, her iki pindeki yükselen ve düşen kenarları sayarak mikrodenetleyici üzerindeki yükü azaltır ve sayacın otomatik olarak artıp azalmasını sağlar. Bu sayede, `__HAL_TIM_GET_COUNTER(htim2)` fonksiyonu, kullanıcının çevirme yönüne göre pozitif veya negatif değerler döndürür.

2.1.2 I2C Haberleşme (I2C1)

LCD ekran ve MPU6050, tek bir I2C veri yolu üzerinden iletişim kurar.

- **Modül:** I2C1 kullanılmıştır.
- **Pinler:** PB6 (SCL) ve PB7 (SDA).
- **Saat Hızı:** Haberleşme kararlılığını sağlamak amacıyla I2C hızı **100 kHz'den 50 kHz'e** düşürülmüştür.

2.1.3 Alarm Çıkışı (GPIO Output)

Buzzer, basitleştirilmiş bir GPIO çıkışı olarak kontrol edilir.

- **Pin:** PA8, **GPIO_MODE_OUTPUT_PP (Push-Pull)** olarak tanımlanmıştır.
- **Kontrol:** `HAL_GPIO_WritePin` fonksiyonu ile lojik HIGH veya LOW sinyali gönderilerek buzzer doğrudan açılıp kapatılır.

3. Yazılım Mimarisi ve Algoritma

3.1 Katmanlı Mimari (Sayfa 3)

Yazılım, modülerlik ve bakım kolaylığı için katmanlı bir yapıda tasarlanmıştır:

1. **Donanım Abstraksiyon Katmanı (HAL):** STM32CubeIDE tarafından sağlanan, donanıma doğrudan erişimi soyutlayan katmandır (`stm32f4xx_hal.h`).
2. **Çevre Birim Sürücüleri (`i2c.c`, `tim.c`, `lcd.c`):** Her bir çevresel modülün (I2C, Timer, LCD) başlatılması ve temel operasyonlarını yöneten özel fonksiyonları içerir.
3. **Uygulama Katmanı (`main.c`):** Tüm sürücüleri bir araya getirir, ana döngüyü çalıştırır ve projenin ana mantığını (ivme hesaplama ve alarm kontrolü) yürütür.

3.2 İvme Simülasyon Algoritması

Bu projede sanal ivme, tamamen döner kodlayıcının pozisyonuna dayandırılmıştır. Bu, geleneksel sensör okumalarındaki gürültü ve dalgalanmayı ortadan kaldırarak kararlı ve kullanıcı kontrollü bir değer sağlar.

3.2.1 Hesaplama Yöntemi

Döner kodlayıcının sayacı (`encoder_value`) direkt olarak sanal ivme değerine çevrilir.

- **`encoder_value`:** `__HAL_TIM_GET_COUNTER(htim2)` ile okunan, çevirme yönüne göre artıp azalan tam sayı.
- **`ACCEL_SCALE_FACTOR`:** Her bir rotary adımı için ivme değerindeki değişimi belirleyen sabittir (100 mg/adım olarak belirlenmiştir).

IVMESanal (mg) =EncoderDeg˘ eri ×O¨lc¸ eklendirmeFakto¨ru¨

Bu formül, kullanıcı rotary'yi çevirmeyi bıraktığında, ekrandaki ivme değerinin o anda **sabit kalmasını** garanti eder.

3.3 LCD Haberleşme Protokolü

I2C LCD'ler, 4-bit modu kullanır. Haberleşmenin doğru çalışması için `lcd.c` dosyasında kritik protokol adımları uygulanmıştır:

1. **I2C Adresleme:** LCD adaptörünün 8-bit adresi (`0x27 << 1` veya `0x3F << 1`) kullanılır.

2. **Nibble Gönderme:** Her 8-bitlik veri (komut veya karakter) iki ayrı 4-bitlik (nibble) parça halinde gönderilir.
3. **Enable Sinyali:** Her nibble gönderimi, I2C hattı üzerinden **Enable (EN)** pininin kısa bir HIGH-LOW palsi verilmesiyle tamamlanır. Bu, LCD kontrolcüsüne veriyi kaydetme komutunu verir.

4. Sonuçlar ve Performans Analizi (Sayfa 4)

4.1 Fonksiyonel Test Sonuçları

Proje, tasarlanan tüm ana işlevleri başarıyla yerine getirmiştir:

1. **Haberleşme Kararlılığı:** I2C tabanlı LCD ve TIM tabanlı döner kodlayıcı arasında stabil bir iletişim sağlanmıştır.
2. **Sanal İvme Kontrolü:** Rotary encoder'ın her bir adımı, ivme değerini 100 mg artırmakta veya azaltmaktadır. Encoder durduğunda ivme değeri beklenen şekilde sabit kalmaktadır.
3. **Akustik Alarm Tepkisi:** `main.c` döngüsü içinde uygulanan koşul IVMESanal ≤−6500 mg), eşiğe ulaşıldığında **PA8** pinini HIGH yaparak buzzer'ı anında aktive etmiştir.

4.2 Performans Ölçütleri

Ölçüt	Değer	Açıklama
Simülasyon Kararlılığı	Mükemmel	İvme değeri, harici gürültüden etkilenmeyerek tam kontrol sağlar.
LCD Güncelleme Hızı	10 FPS (100 ms döngü gecikmesi)	İvme ve pozisyon verileri akıcı bir şekilde güncellenmektedir.
Alarm Gecikmesi	≤100 ms	Eşik değerine ulaşıldığı anda buzzer tepkisi, ana döngü süresi içindedir.
Mikrodenetleyici Yükü	Düşük	Döner kodlayıcının donanımsal zamanlayıcı ile yönetilmesi, CPU yükünü minimize etmiştir.

4.3 Karşılaşılan Zorluklar ve Çözümler

Projenin geliştirilmesi sırasında karşılaşılan temel zorluk, **"Undefined Reference"** bağlayıcı hatalarıydı. Bu hatalar, `htim2` değişkeni ve `lcd.c` yardımcı fonksiyonlarının (`lcd_send_nibble`, `lcd_send_cmd` vb.) **tanımlarının (definition)** eksik olmasından

kaynaklanmıştır. Çözüm olarak, tüm global değişkenlerin ve fonksiyonların tanımları, kullanılmadan önce ilgili `.c` dosyalarına (örneğin `TIM_HandleTypeDef htim2;` tanımı `tim.c`'ye) eklenerek bağlama hatası giderilmiştir.

5. Sonuç ve Gelecek Çalışmalar (Sayfa 5)

5.1 Sonuç

Bu proje, bir STM32F4 geliştirme kartının çoklu çevre birimlerini (TIM, I2C, GPIO) birleştirerek işlevsel bir kontrol ve izleme sistemi oluşturulabileceğini kanıtlamıştır. Döner kodlayıcının sanal bir fiziksel parametreyi (ivme) kontrol etme yeteneği, bu tür arayüzlerin endüstriyel kontrol ve simülasyon uygulamaları için ne kadar uygun olduğunu göstermiştir. Alarm eşiği mantığının eklenmesi, projenin basit bir veri görselleştirme uygulamasından, güvenlik veya operasyonel uyarı mekanizmalarına sahip bir prototipe geçişini sağlamıştır.

5.2 Gelecek Çalışmalar

Bu başarılı prototip, gelecekte aşağıdaki geliştirmelerle genişletilebilir:

1. **PID Kontrolü Entegrasyonu:** MPU6050'den gelen gerçek ivme verisi ile döner kodlayıcının sağladığı "hedef" ivme değeri arasındaki farkı minimize etmek için bir **PID kontrol algoritması** uygulanabilir.
2. **Görsel Geribildirim:** Buzzer'a ek olarak, ivme eşiği aşıldığında bir LED'i yanıp söndürmek veya LCD'deki metnin rengini değiştirmek için (RGB I2C LCD kullanılıyorsa) görsel geribildirim eklenebilir.
3. **Veri Günlüğü:** İvme değerlerinin belirlenen aralıklarla bir SD karta veya harici belleğe kaydedilmesi, sistemin bir veri toplayıcı olarak da kullanılmasına olanak tanır.
4. **Ağ Haberleşmesi:** Ethernet veya Wi-Fi (ESP32) modülü entegre edilerek ivme değerlerinin uzaktan bir sunucuya veya mobil uygulamaya gönderilmesi sağlanabilir.