T.C BİTLİS EREN ÜNİVERSİTESİ
GÖMÜLÜ SİSTEM PROGRAMLAMA DERSİ

MÜHENDİSLİK – MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ARDUİNO TABANLI DONANIM CÜZDANI SİMÜLASYONU

PROJE ADI: Arduino Tabanlı Donanım Cüzdanı Simülasyonu

DERSİ VEREN AKADEMİSYEN: Dr. Öğrt. Üyesi İrfan ÖKTEN

ÖĞRENCİNİN ADI SOYADI: Zekeriya KALKAN

ÖĞRENCİNİN NUMARASI: 23080410034

1. Projenin Amacı

Bu projenin amacı, kripto para ekosisteminde yaygın olarak kullanılan donanım cüzdanlarının temel çalışma prensiplerini, gömülü sistemler perspektifinden ele alarak Arduino tabanlı bir simülasyon geliştirmektir. Proje kapsamında; kullanıcı doğrulama mekanizmaları, işlem onay süreçleri, basitleştirilmiş imzalama mantığı ve işlem kayıtlarının bütünlüğünü koruyan blok benzeri bir veri yapısı eğitim odaklı olarak modellenmiştir.

Geliştirilen sistem, gerçek kriptografik algoritmaların birebir uygulanmasını hedeflememekte; bunun yerine donanım cüzdanlarının güvenlik yaklaşımını ve mimari tasarım felsefesini anlaşılır ve öğretici bir şekilde ortaya koymayı amaçlamaktadır. Bu doğrultuda proje, gömülü sistemlerde sınırlı donanım kaynaklarıyla güvenlik odaklı bir yapı tasarlanabileceğini göstermeyi hedeflemektedir.

2. Projenin Kapsamı

Bu proje, Arduino tabanlı bir donanım cüzdanı simülasyonu geliştirilmesini kapsamaktadır. Proje kapsamında gerçekleştirilen başlıca işlevler aşağıda özetlenmiştir:

• Kullanıcı kimlik doğrulaması için PIN tabanlı giriş mekanizmasının oluşturulması
• PIN doğrulamasına ek olarak iki faktörlü kimlik doğrulama (2FA) yaklaşımının simülasyonu
• Kullanıcı tarafından onaylanan işlemler için basitleştirilmiş bir imza üretim mantığının uygulanması
• Gerçek blockchain bağlantısı olmaksızın, işlemlerin blok benzeri bir yapı ile SD kart üzerinde kayıt altına alınması
• LCD ekran, LED’ler ve buzzer kullanılarak kullanıcıya görsel ve işitsel geri bildirim sağlanması

Proje, gerçek bir kripto para ağıyla haberleşme veya gerçek özel anahtarların güvenli donanım bileşenleri üzerinde saklanması gibi ileri seviye güvenlik uygulamalarını kapsamamaktadır. Bu tercihin temel nedeni, projenin eğitim amaçlı bir simülasyon olarak tasarlanmış olması ve gömülü sistemlerin temel prensiplerine odaklanılmasıdır.

3. Projenin Kısıtları

Projenin geliştirilme sürecinde karşılaşılan donanımsal ve yazılımsal kısıtlar, sistemin nihai mimarisinin ve kapsamının belirlenmesinde doğrudan etkili olmuştur. Bu kısıtlar doğrultusunda proje, işlevselliğini koruyan ancak kaynak kullanımını gözeten bir yaklaşımla sınırlandırılmıştır.

• Fiziksel 4’lü keypad donanımının temin edilememesi nedeniyle kullanıcı girişleri IR tabanlı uzaktan kumanda aracılığıyla gerçekleştirilmiştir. Bu durum, giriş mekanizmasının yazılımsal olarak yeniden modellenmesini gerektirmiştir.

• Arduino Uno platformunun sınırlı RAM kapasitesi, proje geliştirme sürecinde belirleyici bir faktör olmuştur. Özellikle LCD, SD kart, IR alıcı ve seri haberleşme bileşenlerinin eş zamanlı kullanımı bellek tüketimini artırmış; bu durum bazı ileri seviye geliştirmelerin uygulanmasını kısıtlamıştır.

• RAM taşması ve kararsız çalışma riskleri göz önünde bulundurularak, sistem belirli bir noktadan sonra yeni özellikler eklenmeden, stabil ve işlevsel bir hâlde bırakılmıştır.

• Projenin eğitim amaçlı niteliği dikkate alınarak, mevcut donanım imkânlarıyla temel seviyede ancak çalışma mantığını net biçimde ortaya koyan bir simülasyon hedeflenmiştir.

Bu kısıtlar çerçevesinde proje, maksimum özellik ekleme yaklaşımı yerine, mevcut kaynakların dengeli kullanıldığı, kararlı ve öğretici bir sistem olarak tamamlanmıştır.

---

2. Sistem Tasarımı ve Mimari

2.1. Genel Sistem Mimarisi ve Donanım Blok Yapısı

Geliştirilen sistem, merkezinde Arduino UNO bulunan ve çevresel donanım bileşenleri ile desteklenen modüler bir mimari üzerine kurulmuştur. Sistem tasarımında, her donanım bileşeni belirli bir işlevi yerine getirecek şekilde konumlandırılmış ve bileşenler arası görev dağılımı net bir biçimde tanımlanmıştır.

Arduino UNO, sistemin ana kontrol birimi olarak görev yapmakta; kullanıcıdan alınan girişlerin işlenmesi, doğrulama süreçlerinin yönetilmesi, işlem akışlarının kontrol edilmesi ve çevresel birimlerle haberleşmenin sağlanmasından sorumludur. Kullanıcı ile sistem arasındaki etkileşim, LCD ekran, IR tabanlı giriş birimi, LED’ler ve buzzer aracılığıyla gerçekleştirilmiştir.

Sistemde kullanılan başlıca donanım bileşenleri ve görevleri aşağıda özetlenmiştir:

• **Arduino UNO:** Sistem kontrolü, durum yönetimi ve işlem akışlarının yürütülmesi
• **LCD Ekran (I2C):** Kullanıcıya metin tabanlı görsel geri bildirim sağlanması
• **IR Alıcı + Uzaktan Kumanda:** Kullanıcı girişlerinin alınması ve sanal keypad işlevinin sağlanması
• **LED’ler (Yeşil, Sarı, Kırmızı):** Sistem durumlarının görsel olarak ifade edilmesi
• **Buzzer:** Kullanıcıya işitsel geri bildirim verilmesi
• **SD Kart Modülü:** İşlem kayıtlarının blok benzeri bir yapı ile kalıcı olarak saklanması

• **ESP8266:** İki faktörlü kimlik doğrulama kapsamında TOTP doğrulamasının gerçekleştirilmesi

Bu yapı sayesinde, güvenlik doğrulama işlemleri, kullanıcı arayüzü ve veri kayıt mekanizmaları birbirinden ayrılmış; sistem, hem donanımsal hem de yazılımsal açıdan okunabilir ve genişletilebilir bir mimari hâline getirilmiştir. Donanım bileşenleri arasındaki iletişim, görev odaklı bir yaklaşımla tasarlanmış olup, her modül yalnızca kendi sorumluluk alanı dahilinde çalışmaktadır.

2.2. Yazılım Mimarisi (Durum Makinesi Yaklaşımı)

Geliştirilen yazılım, kullanıcı etkileşimlerinin ve güvenlik süreçlerinin karmaşıklaşmasını
önlemek amacıyla durum makinesi (state machine) yaklaşımı kullanılarak tasarlanmıştır. Bu yaklaşım sayesinde sistem, belirli durumlar arasında kontrollü geçişler yaparak her bir işlem adımını deterministik ve yönetilebilir bir biçimde yürütmektedir.

Durum makinesi mimarisi, özellikle kullanıcı doğrulama, işlem onayı, zaman kilidi ve hata senaryolarının bulunduğu gömülü sistemlerde; kontrolsüz akışları ve beklenmeyen
durumları önlemek açısından önemli avantajlar sunmaktadır. Bu projede her durum, yalnızca kendi sorumluluk alanına ait girişleri işlemekle yükümlüdür ve geçerli koşullar sağlandığında bir sonraki duruma geçiş yapılmaktadır.

Sistemde tanımlanan temel durumlar aşağıda özetlenmiştir:

• **LOCKED:** Kullanıcının PIN kodunu girdiği başlangıç durumu
• **TWO_FA:** PIN doğrulaması sonrası iki faktörlü kimlik doğrulama (TOTP) aşaması
• **MENU:** Kullanıcının sistem fonksiyonlarına eriştiği ana menü durumu
• **TX_INPUT:** İşlem miktarının girildiği durum
• **TX_CONFIRM:** Girilen işlemin kullanıcı tarafından onaylandığı durum
• **LOG_CLEAR_CONFIRM:** İşlem kayıtlarının silinmesi için onay alınan durum
• **TIME_LOCK:** Hatalı girişler sonrası sistemin geçici olarak kilitlendiği durum

Her durum, kullanıcıdan gelen girişleri (IR tabanlı sanal keypad aracılığıyla) kendi bağlamında değerlendirir. Örneğin PIN doğrulama yalnızca LOCKED durumunda aktifken, işlem miktarı girişi yalnızca TX_INPUT durumunda mümkündür. Bu sayede farklı işlem adımlarının birbiriyle karışması engellenmiş ve yazılımın okunabilirliği artırılmıştır.

Durum makinesi yapısı, aynı zamanda güvenlik açısından da önemli katkılar sağlamaktadır. Hatalı PIN veya TOTP girişleri belirlenen eşik değerlerini aştığında sistem doğrudan TIME_LOCK durumuna geçirilmekte ve belirli bir süre boyunca yeni girişler engellenmektedir. Bu yaklaşım, brute-force denemelerine karşı temel seviyede bir koruma sağlamaktadır.

2.3. Modüller Arası Veri Akışı

Geliştirilen sistemde, kullanıcıdan alınan girişlerin güvenli bir şekilde işlenebilmesi için modüller arası veri akışı belirli bir sıra ve kontrol mekanizması çerçevesinde tasarlanmıştır. Bu akış, doğrulama adımlarının birbirini tamamlayacak şekilde çalışmasını ve yetkisiz işlemlerin engellenmesini amaçlamaktadır.

Sistem, kullanıcı etkileşimini PIN doğrulama aşaması ile başlatmakta ve başarılı doğrulama sonrasında sırasıyla iki faktörlü kimlik doğrulama (2FA), menü erişimi, işlem oluşturma, işlem onayı ve kayıt altına alma adımlarını izlemektedir. Her adım, bir önceki adımın başarıyla tamamlanmasına bağlıdır.

Veri akışının temel adımları aşağıda özetlenmiştir:

1. **PIN Doğrulama:**
   Kullanıcıdan alınan PIN bilgisi doğrulanır. Hatalı girişler belirlenen eşik değerini aştığında sistem zaman kilidi durumuna geçirilir.
2. **İki Faktörlü Kimlik Doğrulama (2FA):**
   PIN doğrulamasının başarılı olması durumunda kullanıcıdan TOTP tabanlı ikinci doğrulama kodu istenir. Bu aşama, ek bir güvenlik katmanı oluşturarak yetkisiz erişim riskini azaltır.
3. **Ana Menü Erişimi:**
   Doğrulama adımlarının tamamlanmasının ardından kullanıcı ana menüye yönlendirilir. Menü üzerinden işlem görüntüleme, yeni işlem oluşturma ve kayıt yönetimi gibi işlevlere erişim sağlanır.
4. **İşlem Oluşturma ve Onay:**
   Kullanıcı tarafından girilen işlem bilgileri sistem tarafından geçici olarak saklanır ve kullanıcıdan açık bir onay alınmadan işleme alınmaz.
5. **Kayıt (Log) Aşaması:**
   Onaylanan işlemler, blok benzeri bir yapı kullanılarak SD kart üzerinde kalıcı olarak
   kaydedilir. Bu aşamada işlem verileri ve bütünlük kontrol bilgileri birlikte saklanır.

Bu sıralı veri akışı sayesinde, sistem içerisinde yetkisiz veya eksik doğrulanmış bir işlemin kayıt altına alınması engellenmiştir. Modüller arası geçişlerin kontrollü olması, yazılımın hem güvenliğini hem de bakım ve geliştirme kolaylığını artırmıştır.

2.4. Arduino – ESP8266 Haberleşme Mimarisi

Sistemde iki faktörlü kimlik doğrulama (2FA) mekanizmasının uygulanabilmesi amacıyla, Arduino UNO ile ESP8266 modülü arasında seri haberleşmeye dayalı bir iletişim mimarisi kurulmuştur. Bu yaklaşım ile doğrulama işlemleri, ana kontrol biriminden ayrıştırılarak farklı bir modül üzerinde gerçekleştirilmiştir.

Arduino UNO, sistemin merkez kontrol birimi olarak görev yapmakta; kullanıcıdan alınan TOTP kodunu ESP8266 modülüne iletmekte ve doğrulama sonucuna göre yazılım akışını yönlendirmektedir. ESP8266 ise kablosuz ağ bağlantısı ve zaman

senkronizasyonu gibi ek yetenekleri sayesinde TOTP doğrulama işlemini bağımsız bir şekilde gerçekleştirmektedir.

İki modül arasındaki haberleşme, yazılımsal seri port (SoftwareSerial) kullanılarak sağlanmıştır. Arduino tarafından gönderilen veri, belirli bir formatta ESP8266'ya iletilmekte; ESP8266 doğrulama sonucunu basit ve deterministik bir yanıt mesajı ile geri döndürmektedir. Bu sayede haberleşme protokolü hem okunabilir hem de hata ayıklamaya uygun bir yapı sunmaktadır.

Haberleşme sürecinin genel işleyişi aşağıdaki gibidir:

• Arduino, kullanıcıdan aldığı 6 haneli doğrulama kodunu seri hat üzerinden ESP8266'ya gönderir.
• ESP8266, aldığı doğrulama kodunu mevcut zaman bilgisi ile karşılaştırarak doğrular.
• Doğrulama sonucuna bağlı olarak ESP8266, Arduino'ya olumlu veya olumsuz bir yanıt iletir.
• Arduino, gelen yanıt doğrultusunda kullanıcıyı ana menüye yönlendirir veya hata/ kilit mekanizmasını devreye alır.

Bu mimari yaklaşım sayesinde, Arduino UNO üzerindeki işlem yükü ve bellek kullanımı azaltılmış; doğrulama işlemleri daha uygun donanımsal yeteneklere sahip bir modüle devredilmiştir. Ayrıca bu yapı, sistemin ileride farklı doğrulama yöntemleri veya haberleşme protokolleri ile genişletilebilmesine olanak tanımaktadır.

---

3. Uygulama ve Gerçekleme

Bu bölümde, geliştirilen donanım cüzdanı simülasyonunun yazılım ve donanım düzeyinde nasıl gerçekleştirildiği, modüler bir yapı içerisinde ele alınmıştır. Her bir modül; amacı, uygulama detayları, karşılaşılan problemler ve geliştirilen çözümler ile birlikte sunulmuştur. Bu yaklaşım, sistemin tasarım sürecinin daha anlaşılır ve takip edilebilir olmasını sağlamaktadır.

Uygulama sürecinde, gömülü sistemlerin doğası gereği karşılaşılan donanımsal kısıtlar göz önünde bulundurulmuş; sistem, kararlılık ve işlevsellik önceliklendirilerek adım adım geliştirilmiştir.

3.1. IR Tabanlı Sanal Keypad Emülasyonu

Amaç

Bu modülün amacı, kullanıcıdan PIN, doğrulama kodu ve menü komutlarını alabilecek

bir giriş mekanizması oluşturmaktır. Fiziksel keypad donanımı bulunmadığından, kullanıcı girişlerinin alternatif bir yöntemle sisteme aktarılması hedeflenmiştir.

Uygulama Detayı

Kullanıcı girişleri, IR alıcı ve uzaktan kumanda kullanılarak alınmıştır. Uzaktan kumandadan gönderilen ham (raw) IR kodları analiz edilerek her bir tuş, yazılımsal olarak belirli bir karakter veya komut ile eşleştirilmiştir. Bu sayede, fiziksel bir keypad davranışı yazılım seviyesinde emüle edilmiştir.

Elde edilen karakterler; rakamsal girişler, onay, iptal ve yönlendirme komutları olacak şekilde sınıflandırılmıştır. Bu sanal keypad yapısı, PIN girişi, iki faktörlü doğrulama, menü navigasyonu ve işlem onayı gibi tüm kullanıcı etkileşimlerinde ortak bir giriş katmanı olarak kullanılmıştır.

Karşılaşılan Sorun

Fiziksel keypad donanımının bulunmaması, kullanıcı girişlerinin doğrudan alınmasını engellemiştir. Ayrıca IR sinyallerinin ham kodlarının farklı kumandalarda değişkenlik gösterebilmesi, girişlerin güvenilir şekilde işlenmesini zorlaştırmıştır.

Çözüm

Kumandadan gelen IR kodları tek tek okunarak sabit değerler hâlinde yazılıma tanımlanmış ve her bir kod, sistemde karşılığı olan bir tuş davranışı ile eşleştirilmiştir. Böylece girişler standartlaştırılmış ve yazılımın geri kalan bölümleri, giriş kaynağından bağımsız hâle gelmiştir.

Bu yaklaşım sayesinde, ileride fiziksel bir keypad eklenmesi durumunda yazılım mimarisinde köklü değişiklik yapılmasına gerek kalmadan sistem genişletilebilir hâle gelmiştir.

Test / Çıktı

• Rakam tuşlarının PIN ve doğrulama kodu girişlerinde doğru şekilde algılandığı doğrulanmıştır.
• Onay, iptal ve yönlendirme tuşlarının menü ve işlem akışlarında beklenen davranışı sergilediği gözlemlenmiştir.
• Farklı sistem durumlarında (PIN girişi, menü, işlem onayı) girişlerin çakışmadan işlendiği test edilmiştir.

Görsel / Kod Referansı

• IR alıcı devre bağlantısına ait fotoğraflar
• Uzaktan kumanda tuşları ile sistem komutları arasındaki eşleştirmeyi gösteren kod bloğu

(Bu bölümde, IR kodlarının karakterlere eşlendiği fonksiyon raporun ekler kısmında sunulmuştur.)

3.2. PIN Doğrulama, Deneme Limiti ve Zaman Kilidi Mekanizması

Amaç

Bu modülün amacı, sisteme yetkisiz erişimi engellemek için kullanıcı doğrulamasını temel seviyede güvence altına alan bir PIN doğrulama mekanizması oluşturmaktır. Ayrıca, tekrarlanan hatalı girişleri sınırlayarak brute-force denemelerine karşı koruyucu bir yapı kurulması hedeflenmiştir.

Uygulama Detayı

Sisteme erişim, kullanıcıdan alınan dört haneli bir PIN kodu ile başlatılmaktadır. Kullanıcı tarafından girilen PIN bilgisi, sistemde tanımlı doğru PIN ile karşılaştırılarak doğrulanmaktadır. PIN doğrulaması başarılı olduğunda, yazılım akışı iki faktörlü kimlik doğrulama aşamasına yönlendirilmektedir.

Hatalı PIN girişlerinin güvenlik zafiyetine yol açmaması amacıyla, sistemde bir deneme
sayacı tutulmaktadır. Belirlenen maksimum deneme sayısına ulaşıldığında sistem otomatik olarak zaman kilidi (time lock) durumuna geçirilmekte ve belirli bir süre boyunca yeni girişler engellenmektedir.

Zaman kilidi süresince kullanıcıya, LCD ekran üzerinden bilgilendirici mesajlar gösterilmekte; LED ve buzzer bileşenleri kullanılarak kilit durumuna ilişkin görsel ve işitsel geri bildirim sağlanmaktadır. Kilit süresi sona erdiğinde sistem başlangıç durumuna geri dönmekte ve kullanıcıdan tekrar PIN girişi istenmektedir.

Karşılaşılan Sorun

Arduino Uno’nun sınırlı bellek kapasitesi nedeniyle, PIN doğrulama, giriş yönetimi, geri
bildirim mekanizmaları ve diğer modüllerin aynı anda çalıştırılması bellek kullanımını artırmıştır. Bu durum, daha karmaşık doğrulama yapılarına geçilmesini zorlaştırmıştır.

Ayrıca, kullanıcı deneyimini olumsuz etkilemeden güvenlik önlemlerinin uygulanması, deneme sayısı ve kilit süresi gibi parametrelerin dikkatli belirlenmesini gerektirmiştir.

Çözüm

Bellek kullanımını kontrol altında tutmak amacıyla PIN doğrulama mekanizması sade bir yapı ile tasarlanmış, yalnızca gerekli değişkenler ve kontrol akışları kullanılmıştır. Deneme limiti ve zaman kilidi süreleri, sistemin hem güvenli hem de kullanılabilir kalmasını sağlayacak şekilde dengelenmiştir.

Zaman kilidi sürecinde yazılım, yalnızca kilit durumuna ait işlemleri yürütecek biçimde sınırlandırılmış; böylece sistemin kararlılığı korunmuştur. Bu yaklaşım, sınırlı donanım kaynakları altında güvenlik önlemlerinin uygulanabilirliğini artırmıştır.

Test / Çıktı

• Doğru PIN girildiğinde sistemin bir sonraki doğrulama aşamasına geçtiği doğrulanmıştır.
• Hatalı PIN girişlerinde deneme sayacının doğru şekilde arttığı gözlemlenmiştir.
• Maksimum deneme sayısına ulaşıldığında sistemin zaman kilidi durumuna geçtiği ve bu süre boyunca yeni girişleri kabul etmediği test edilmiştir.
• Kilit süresi sonunda sistemin başlangıç durumuna sorunsuz şekilde döndüğü gözlemlenmiştir.

Görsel / Kod Referansı

• PIN giriş ekranını ve kilit durumunu gösteren LCD görüntüleri
• LED ve buzzer geri bildirimlerinin aktif olduğu sistem durumlarına ait devre fotoğrafları
• PIN doğrulama ve zaman kilidi mekanizmasını gerçekleştiren kodlar Ek A bölümünde sunulmuştur.

3.4. Menü Sistemi ve Kullanıcı Etkileşimi

Amaç

Bu modülün amacı, doğrulama süreçlerini başarıyla tamamlayan kullanıcının sistem fonksiyonlarına kontrollü ve anlaşılır bir şekilde erişebilmesini sağlamaktır. Menü sistemi, kullanıcı ile sistem arasındaki etkileşimi sadeleştirerek işlem yönetimini kolaylaştırmayı hedeflemektedir.

Uygulama Detayı

Kullanıcı, PIN ve iki faktörlü kimlik doğrulama aşamalarını başarıyla tamamladıktan sonra ana menü ekranına yönlendirilmektedir. Menü sistemi, LCD ekran üzerinden kullanıcıya sunulmakta ve IR tabanlı sanal keypad aracılığıyla kontrol edilmektedir.

Menü yapısı, aşağıdaki temel işlevleri içerecek şekilde tasarlanmıştır:

• Son işlemin görüntülenmesi
• Yeni işlem oluşturulması
• İşlem kayıtlarının silinmesi
• Sistemden çıkış yapılması

Menüde gezinme işlemleri yukarı ve aşağı yön komutları ile gerçekleştirilmekte, seçilen işlem ise onay komutu ile aktive edilmektedir. Kullanıcıdan alınan her giriş, mevcut sistem durumuna göre değerlendirilmekte ve geçersiz girişlerin sistem akışını bozması engellenmektedir.

Karşılaşılan Sorun

Menü sistemi tasarlanırken, Arduino Uno'nun sınırlı bellek kapasitesi nedeniyle dinamik veri yapılarının ve karmaşık kullanıcı arayüzü bileşenlerinin kullanımı mümkün olmamıştır. Ayrıca, farklı sistem durumlarında aynı giriş tuşlarının farklı anlamlara gelmesi, kullanıcı girişlerinin yanlış yorumlanma riskini artırmıştır.

Çözüm

Menü sistemi, sabit ve önceden tanımlı seçenekler kullanılarak sade bir yapıda tasarlanmıştır. Menü metinleri, bellek kullanımını azaltmak amacıyla program belleğinde saklanmış ve yalnızca gerekli durumlarda LCD ekran üzerinde gösterilmiştir.

Her sistem durumu için geçerli kullanıcı girişleri net bir şekilde sınırlandırılmış; böylece menü navigasyonu sırasında yanlış veya beklenmeyen durum geçişlerinin önüne geçilmiştir. Bu yaklaşım, hem yazılımın kararlılığını artırmış hem de kullanıcı deneyimini iyileştirmiştir.

Test / Çıktı

• Doğrulama sonrası kullanıcının ana menüye sorunsuz şekilde eriştiği doğrulanmıştır.
• Menü seçenekleri arasında gezinmenin doğru çalıştığı gözlemlenmiştir.
• Yanlış veya geçersiz girişlerin sistem akışını bozmadığı test edilmiştir.
• Menüden çıkış yapıldığında sistemin başlangıç durumuna geri döndüğü doğrulanmıştır.

Görsel / Kod Referansı

• Menü ekranına ait LCD görüntüleri
• Menü navigasyonunun çalıştığını gösteren sistem fotoğrafları
• Menü yönetimini gerçekleştiren yazılım kodları Ek A bölümünde sunulmuştur.

3.5. İşlem Oluşturma, Onay ve İmza Simülasyonu

Amaç

Bu modülün amacı, donanım cüzdanlarının temel işlevlerinden biri olan işlem oluşturma ve kullanıcı onayı sürecini simüle etmektir. Kullanıcının açık onayı alınmadan herhangi bir işlemin gerçekleştirilmemesi ve işlemin sistem içerisinde doğrulanabilir bir şekilde temsil edilmesi hedeflenmiştir.

Uygulama Detayı

Ana menü üzerinden "yeni işlem" seçeneğini seçen kullanıcıdan, işlem miktarı bilgisi alınmaktadır. Girilen işlem verisi, sistem tarafından geçici olarak bellekte

tutulmakta ve kullanıcıdan açık bir onay alınana kadar kalıcı hâle getirilmemektedir.

Kullanıcı onayı alındıktan sonra, işlem için basitleştirilmiş bir imza üretim süreci başlatılmaktadır. Bu imza, gerçek kriptografik algoritmaların birebir uygulanması yerine, işlem verisi, kullanıcıya ait doğrulama bilgileri ve işlem sırası gibi parametrelerin bir araya getirilmesiyle oluşturulan eğitim amaçlı bir simülasyon olarak tasarlanmıştır.

Üretilen imza değeri, işlemle ilişkilendirilerek sistem içerisinde gösterilmekte ve bir sonraki aşamada kayıt altına alınmak üzere hazırlanmaktadır. Bu süreç, gerçek donanım cüzdanlarında yer alan “işlem oluştur → kullanıcı onayı → imzalama” akışının temel mantığını yansıtmaktadır.

Karşılaşılan Sorun

Arduino Uno’nun sınırlı bellek ve işlem kapasitesi, gerçek kriptografik imzalama algoritmalarının uygulanmasını mümkün kılmamıştır. Ayrıca, işlem oluşturma sürecinde kullanıcıdan alınan verilerin güvenli ve tutarlı şekilde yönetilmesi gerekliliği, yazılım akışının dikkatli bir şekilde kurgulanmasını zorunlu kılmıştır.

Çözüm

Bu kısıtlar doğrultusunda, imzalama işlemi basitleştirilmiş bir model ile ele alınmış ve yalnızca işlem mantığını temsil edecek şekilde tasarlanmıştır. Kullanıcı onayı, sistemde ayrı bir durum olarak ele alınmış ve onay alınmadan hiçbir işlem kayıt altına alınmamıştır.

Bu yaklaşım sayesinde, sistem hem donanım kısıtlarına uygun kalmış hem de donanım cüzdanlarının temel güvenlik felsefesini öğretici bir şekilde yansıtan bir yapı sunmuştur.

Test / Çıktı

• Kullanıcıdan işlem miktarı alınmadan sürecin ilerlemediği doğrulanmıştır.
• İşlem onayı verilmeden imza üretimi ve kayıt işleminin gerçekleştirilmediği test edilmiştir.
• Onaylanan işlemler için imza değerinin üretildiği ve ekranda doğru şekilde gösterildiği gözlemlenmiştir.
• Kullanıcı tarafından iptal edilen işlemlerin sistem tarafından tamamen yok sayıldığı doğrulanmıştır.

Görsel / Kod Referansı

• İşlem miktarı girişi ve onay ekranına ait LCD görüntüleri
• İmza değerinin ekranda gösterildiği sistem çıktıları
• İşlem oluşturma ve imza simülasyonunu gerçekleştiren kodlar Ek A bölümünde sunulmuştur.

3.5. İşlem Oluşturma, Onay ve İmza Simülasyonu

Amaç

Bu modülün amacı, donanım cüzdanlarının temel işlevlerinden biri olan işlem oluşturma ve kullanıcı onayı sürecini simüle etmektir. Kullanıcının açık onayı alınmadan herhangi bir işlemin gerçekleştirilmemesi ve işlemin sistem içerisinde doğrulanabilir bir şekilde temsil edilmesi hedeflenmiştir.

Uygulama Detayı

Ana menü üzerinden “yeni işlem” seçeneğini seçen kullanıcıdan, işlem miktarı bilgisi alınmaktadır. Girilen işlem verisi, sistem tarafından geçici olarak bellekte tutulmakta ve kullanıcıdan açık bir onay alınana kadar kalıcı hâle getirilmemektedir.

Kullanıcı onayı alındıktan sonra, işlem için basitleştirilmiş bir imza üretim süreci başlatılmaktadır. Bu imza, gerçek kriptografik algoritmaların birebir uygulanması yerine, işlem verisi, kullanıcıya ait doğrulama bilgileri ve işlem sırası gibi parametrelerin bir araya getirilmesiyle oluşturulan eğitim amaçlı bir simülasyon olarak tasarlanmıştır.

Üretilen imza değeri, işlemle ilişkilendirilerek sistem içerisinde gösterilmekte ve bir sonraki aşamada kayıt altına alınmak üzere hazırlanmaktadır. Bu süreç, gerçek donanım cüzdanlarında yer alan “işlem oluştur → kullanıcı onayı → imzalama” akışının temel mantığını yansıtmaktadır.

Karşılaşılan Sorun

Arduino Uno’nun sınırlı bellek ve işlem kapasitesi, gerçek kriptografik imzalama algoritmalarının uygulanmasını mümkün kılmamıştır. Ayrıca, işlem oluşturma sürecinde kullanıcıdan alınan verilerin güvenli ve tutarlı şekilde yönetilmesi gerekliliği, yazılım akışının dikkatli bir şekilde kurgulanmasını zorunlu kılmıştır.

Çözüm

Bu kısıtlar doğrultusunda, imzalama işlemi basitleştirilmiş bir model ile ele alınmış ve yalnızca işlem mantığını temsil edecek şekilde tasarlanmıştır. Kullanıcı onayı, sistemde ayrı bir durum olarak ele alınmış ve onay alınmadan hiçbir işlem kayıt altına alınmamıştır.

Bu yaklaşım sayesinde, sistem hem donanım kısıtlarına uygun kalmış hem de donanım cüzdanlarının temel güvenlik felsefesini öğretici bir şekilde yansıtan bir yapı sunmuştur.

Test / Çıktı

• Kullanıcıdan işlem miktarı alınmadan sürecin ilerlemediği doğrulanmıştır.
• İşlem onayı verilmeden imza üretimi ve kayıt işleminin gerçekleştirilmediği

test edilmiştir.
• Onaylanan işlemler için imza değerinin üretildiği ve ekranda doğru şekilde gösterildiği gözlemlenmiştir.
• Kullanıcı tarafından iptal edilen işlemlerin sistem tarafından tamamen yok sayıldığı doğrulanmıştır.

Görsel / Kod Referansı

• İşlem miktarı girişi ve onay ekranına ait LCD görüntüleri
• İmza değerinin ekranda gösterildiği sistem çıktıları
• İşlem oluşturma ve imza simülasyonunu gerçekleştiren kodlar Ek A bölümünde sunulmuştur.

3.7. LED ve Buzzer ile Kullanıcı Geri Bildirim Mekanizmaları

Amaç

Bu modülün amacı, sistemin mevcut durumunu ve kullanıcı etkileşimlerine verdiği tepkileri görsel ve işitsel geri bildirimler aracılığıyla kullanıcıya açık bir şekilde iletmektir. Böylece kullanıcı, ekrana sürekli odaklanmadan sistemin durumu hakkında hızlı ve sezgisel bilgi edinebilmektedir.

Uygulama Detayı

Sistemde, farklı durumları temsil etmek amacıyla üç farklı LED (yeşil, sarı ve kırmızı) kullanılmıştır. Her LED, sistemin belirli bir çalışma durumunu ifade edecek şekilde konfigüre edilmiştir. Buna ek olarak, buzzer bileşeni kullanılarak kullanıcıya işitsel uyarılar sağlanmıştır.

LED’ler; doğrulama başarı durumu, işlem süreci, hata ve kilitlenme gibi kritik durumları ayırt edilebilir biçimde göstermek üzere yazılım tarafından kontrol edilmektedir. Buzzer ise kısa, uzun veya ardışık ses desenleri ile kullanıcıya onay, hata veya kilit durumu hakkında geri bildirim vermektedir.

Bu geri bildirim mekanizmaları, yazılımın durum makinesi yapısı ile entegre çalışmakta ve sistem durumu değiştikçe otomatik olarak güncellenmektedir.

Karşılaşılan Sorun

Birden fazla modülün aynı anda geri bildirim üretme ihtiyacı, LED ve buzzer kontrolünde çakışma riskini beraberinde getirmiştir. Özellikle hata, kilitlenme ve kullanıcı onayı gibi durumların eş zamanlı oluşması, geri bildirimlerin belirsizleşmesine neden olabilecek bir durumdur.

Çözüm

Bu sorunun önüne geçmek için geri bildirim mekanizmaları, sistem durumlarına öncelik tanımlanarak yönetilmiştir. Kritik durumlar (örneğin zaman kilidi veya

hata durumu), diğer geri bildirimlerin önüne geçirilmiş; böylece kullanıcıya iletilen mesajların netliği korunmuştur.

LED ve buzzer kontrolü, merkezi bir yapı üzerinden gerçekleştirilerek geri bildirimlerin sistem genelinde tutarlı olması sağlanmıştır. Bu yaklaşım, kullanıcı deneyimini iyileştirdiği gibi yazılımın bakımını ve genişletilmesini de kolaylaştırmıştır.

Test / Çıktı

• Doğru doğrulama ve işlem onayı durumlarında görsel ve işitsel geri bildirimlerin beklendiği şekilde çalıştığı doğrulanmıştır.
• Hatalı giriş ve zaman kilidi durumlarında LED ve buzzer uyarılarının kullanıcıyı açık şekilde bilgilendirdiği gözlemlenmiştir.
• Farklı sistem durumlarında geri bildirimlerin çakışmadan ve tutarlı biçimde sunulduğu test edilmiştir.

Görsel / Kod Referansı

• LED ve buzzer bileşenlerinin Arduino'ya entegre edildiğini gösteren devre fotoğrafları
• Farklı sistem durumlarında LED ve buzzer geri bildirimlerini gösteren çalışma anı görüntüleri
• Geri bildirim mekanizmalarını yöneten yazılım kodları Ek A bölümünde sunulmuştur.

---

4. Test ve Doğrulama

Bu bölümde, geliştirilen donanım cüzdanı simülasyonunun tanımlanan işlevleri doğru ve kararlı bir şekilde yerine getirip getirmediği test edilmiştir. Test süreci, sistemin güvenlik, kullanıcı etkileşimi ve veri bütünlüğü açısından beklenen davranışları sergileyip sergilemediğini doğrulamaya yönelik olarak senaryo tabanlı bir yaklaşımla yürütülmüştür.

Testler, gerçek kullanım senaryolarını temsil edecek şekilde tasarlanmış; özellikle doğrulama adımları, hata durumları ve kenar durumlar (edge case) önceliklendirilmiştir. Her test senaryosu için beklenen sonuçlar belirlenmiş ve sistemin gerçek çıktıları ile karşılaştırılmıştır.

4.1. Test Senaryoları

Aşağıdaki tabloda, sistem üzerinde gerçekleştirilen temel test senaryoları ve elde edilen sonuçlar sunulmuştur:

Test No	Test Senaryosu	Girdi / Koşul	Beklenen Sonuç	Gerçek Sonuç
T1	Doğru PIN girişi	Geçerli PIN girilmesi	2FA ekranına geçiş	Başarılı
T2	Yanlış PIN denemeleri	PIN'in art arda 3 kez yanlış girilmesi	Zaman kilidi (time lock)	Başarılı
T3	Zaman kilidi süreci	Kilit süresi boyunca giriş yapılması	Girişlerin engellenmesi	Başarılı
T4	Kilit süresi sonu	Kilit süresinin dolması	PIN ekranına dönüş	Başarılı
T5	Doğru TOTP girişi	Geçerli 2FA kodu	Ana menüye geçiş	Başarılı
T6	Yanlış TOTP denemeleri	TOTP'nin art arda yanlış girilmesi	Zaman kilidi	Başarılı
T7	Menü navigasyonu	Menüde yukarı/aşağı gezinme	Doğru menü seçimi	Başarılı
T8	İşlem oluşturma	Geçerli işlem miktarı girilmesi	Onay ekranına geçiş	Başarılı
T9	İşlem iptali	Onay ekranında iptal seçilmesi	İşlemin yok sayılması	Başarılı
T10	İşlem onayı	İşlem onaylanması	İmza üretimi ve kayıt	Başarılı
T11	SD kart olmadan çalışma	SD kart çıkarılmış durum	Sistem çalışmaya devam eder	Başarılı
T12	Sistem yeniden başlatma	Güç kesilip tekrar verilmesi	Son kaydın geri yüklenmesi	Başarılı
T13	Log silme işlemi	Log silme onaylanması	Kayıtların sıfırlanması	Başarılı

4.2. Test Sonuçlarının Değerlendirilmesi

Gerçekleştirilen testler sonucunda, sistemin tanımlanan tüm senaryolarda beklenen davranışları sergilediği gözlemlenmiştir. Özellikle güvenlik odaklı senaryolarda (PIN doğrulama, iki faktörlü kimlik doğrulama ve zaman kilidi mekanizması), sistemin tutarlı ve kararlı bir şekilde çalıştığı doğrulanmıştır.

Ayrıca, SD kart erişiminin olmadığı veya sistemin yeniden başlatıldığı durumlarda dahi yazılımın beklenmeyen bir davranış sergilemediği görülmüştür. Bu durum, geliştirilen sistemin sınırlı donanım kaynaklarına rağmen kararlı ve güvenilir bir yapı sunduğunu göstermektedir.

EKLER

Bu bölümde, proje kapsamında geliştirilen yazılım kodları ve donanım uygulamalarına ait görseller sunulmuştur. Ana rapor metninde açıklanan sistem bileşenleri ve modüller, bu bölümde detaylandırılarak belgelendirilmiştir. Ekler bölümü, raporun ana akışını bozmadan uygulamaya ait teknik kanıtların incelenebilmesini amaçlamaktadır.

Ek A – Arduino Uno Uygulama Kodları

Bu ekte, Arduino Uno üzerinde çalışan donanım cüzdanı simülasyonuna ait yazılım kodları sunulmaktadır. Kodlar, sistemin modüler yapısını yansıtacak şekilde alt başlıklar hâlinde düzenlenmiştir.

Ek A.1 – Ana Kontrol Yapısı ve Durum Makinesi

Bu bölümde, sistemin genel kontrol akışını ve durum makinesi yapısını yöneten ana yazılım kodları yer almaktadır. PIN doğrulama, iki faktörlü kimlik doğrulama, menü yönetimi ve zaman kilidi gibi durum geçişleri bu yapı üzerinden kontrol edilmektedir.

```
#include <IRremote.hpp>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <SD.h>
#include <avr/pgmspace.h>
#include <SoftwareSerial.h>

// ------------------ LCD ------------------
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
// ------------------ PINLER ------------------
#define IR_PIN        2
#define BUZZER_PIN    8
#define LED_GREEN     9
#define LED_YELLOW    10
#define LED_RED       7

// ------------------ ESP (TOTP Verify) ------------------
SoftwareSerial espSerial(5, 6);
#define OTP_LEN           6
#define OTP_MAX_ATTEMPTS   3

char enteredOtp[OTP_LEN + 1];
byte otpIndex = 0;
byte otpAttempts = 0;

// ------------------ SD ------------------
#define SD_CS_PIN      4
#define SD_FILE_NAME   "blocks.txt"

bool sdReady = false;
uint16_t blockIndex = 0;
uint32_t prevBlockHash = 0;

// ------------------ SABİTLER ------------------
#define PIN_LENGTH     4
#define MAX_ATTEMPTS   3
#define LOCK_TIME      30000UL

// ------------------ STATE MACHINE ------------------
enum State {
  LOCKED,
  TIME_LOCK,
  TWO_FA,
  MENU,
  TX_INPUT,
  TX_CONFIRM,
  LOG_CLEAR_CONFIRM
};

State currentState = LOCKED;
```

Ek A.2 – IR Tabanlı Sanal Keypad Kodları

Bu bölümde, IR alıcıdan alınan ham sinyallerin sanal keypad tuşlarına dönüştürülmesini sağlayan yazılım kodları sunulmaktadır. Kullanıcı girişleri bu yapı üzerinden standartlaştırılmıştır.

```
char mapIRtoKey(unsigned long code) {
  switch (code) {
    case 0xBA45FF00: return '1';
    case 0xB946FF00: return '2';
    case 0xB847FF00: return '3';
    case 0xBB44FF00: return '4';
    case 0xBF40FF00: return '5';
    case 0xBC43FF00: return '6';
    case 0xF807FF00: return '7';
    case 0xEA15FF00: return '8';
    case 0xF609FF00: return '9';
    case 0xE619FF00: return '0';
    case 0xE31CFF00: return 'O';   // Onay
    case 0xF20DFF00: return 'X';   // İptal
    case 0xE718FF00: return 'U';   // Yukarı
    case 0xAD52FF00: return 'D';   // Aşağı
    case 0xE916FF00: return 'C';   // Temizle
    default:
      return 0;
  }
}
```

Ek A.3 – PIN Doğrulama ve Zaman Kilidi Mekanizması

Bu bölümde, PIN doğrulama işlemini, deneme limitini ve zaman kilidi mekanizmasını gerçekleştiren yazılım kodları yer almaktadır. Güvenlik ihlallerine karşı alınan önlemler bu kodlar üzerinden uygulanmaktadır.

```
static inline void clearPinEntry() {
  memset(enteredPin, 0, sizeof(enteredPin));
  pinIndex = 0;
}

void clearOtpEntry() {
  memset(enteredOtp, 0, sizeof(enteredOtp));
  otpIndex = 0;
}

void handleKey(char key) {

  if (key >= '0' && key <= '9') {
    if (pinIndex < PIN_LENGTH) {
      enteredPin[pinIndex++] = key;
      lcd.setCursor(pinIndex - 1, 1);
      lcd.print('*');
      buzzerStart(BEEP_SHORT);
    }
    return;
  }

  if (key == 'C') {
    clearPinEntry();
    lcdFooterClear();
    buzzerStart(BEEP_SHORT);
    return;
  }

  if (key == 'O' && pinIndex == PIN_LENGTH) {
    enteredPin[PIN_LENGTH] = '\0';
    verifyPin();
    return;
  }

  if (key == 'X') {
    clearPinEntry();
    lcdFooterClear();
    buzzerStart(BEEP_SHORT);
  }
}

void verifyPin() {

  if (strcmp(enteredPin, correctPin) == 0) {

    lcdMessageF(F("PIN Dogru"), F("2FA Gerekli"), 450);
    buzzerStart(BEEP_DOUBLE);
```

```
    attempts = 0;
    viewingTx = false;

    setState(TWO_FA);
    clearOtpEntry();

    lcdHeaderF(F("2FA KOD GIR"));
    lcdFooterClear();

    buzzerStart(BEEP_SHORT);

  } else {

    attempts++;
    lcdMessageF(F("Yanlis PIN"), F("Tekrar Dene"), 450);
    buzzerStart(BEEP_ERROR);

    if (attempts >= MAX_ATTEMPTS) {

      lockStartTime = millis();
      lastLockBlinkMs = 0;
      redBlink = true;
      lastShownSec = 255;

      viewingTx = false;
      setState(TIME_LOCK);

      showLockCountdown(30);
      buzzerStart(BEEP_LOCK);

    } else {

      setState(LOCKED);
      lcdHeaderF(F("PIN Giriniz"));
      lcdFooterClear();
    }
  }

  clearPinEntry();
}
```

Ek A.4 – Menü Sistemi ve Kullanıcı Etkileşimi Kodları

Bu bölümde, LCD tabanlı menü yapısını ve kullanıcı etkileşimlerini yöneten yazılım kodları sunulmaktadır. Menü navigasyonu ve seçim işlemleri bu yapı üzerinden gerçekleştirilmektedir.

```
// ------------------ MENU (PROGMEM) ------------------

const char menu0[] PROGMEM = "Islem Goster";
const char menu1[] PROGMEM = "Yeni Islem";
const char menu2[] PROGMEM = "Log Sil";
const char menu3[] PROGMEM = "Cikis";

const char* const menuItems[] PROGMEM = {
  menu0,
  menu1,
  menu2,
  menu3
};

#define MENU_COUNT 4

void lcdPrintMenuItem(byte idx) {
  char buf[17];
  strcpy_P(buf, (PGM_P)pgm_read_ptr(&menuItems[idx]));
  lcd.print(buf);
}

void showMenu() {
  lcdHeaderF(F("MENU"));
  lcd.setCursor(0, 1);
  lcd.print(F(" "));
  lcd.setCursor(0, 1);
  lcdPrintMenuItem(menuIndex);
}

void handleMenu(char key) {

  if (viewingTx) {
    if (key == 'X') {
      viewingTx = false;
      buzzerStart(BEEP_SHORT);
      showMenu();
    }
    return;
  }

  if (key == 'U' && menuIndex > 0) {
    menuIndex--;
    buzzerStart(BEEP_SHORT);
  }
```

```
  if (key == 'D' && menuIndex < (MENU_COUNT - 1)) {
    menuIndex++;
    buzzerStart(BEEP_SHORT);
  }

  if (key == 'O') {

    if (menuIndex == 0) {
      showTransactionLCD();
      return;
    }

    if (menuIndex == 1) {
      setState(TX_INPUT);
      clearAmountEntry();
      lcdHeaderF(F("Miktar Gir"));
      lcdFooterClear();
      buzzerStart(BEEP_SHORT);
      return;
    }

    if (menuIndex == 2) {
      setState(LOG_CLEAR_CONFIRM);
      lcdHeaderF(F("Log Sil?"));
      lcd.setCursor(0, 1);
      lcd.print(F("OK / X"));
      buzzerStart(BEEP_ERROR);
      return;
    }

    if (menuIndex == 3) {
      setState(LOCKED);
      viewingTx = false;
      otpAttempts = 0;
      clearOtpEntry();
      lcdMessageF(F("Cikis Yapildi"), F("PIN Giriniz"), 450);
      lcdHeaderF(F("PIN Giriniz"));
      lcdFooterClear();
      buzzerStart(BEEP_SHORT);
      return;
    }
  }

  showMenu();
}
```

Ek A.5 – İşlem Oluşturma ve İmza Simülasyonu Kodları

Bu bölümde, işlem miktarı girişi, kullanıcı onayı ve basitleştirilmiş imza üretim sürecini yöneten yazılım kodları sunulmaktadır.

```
static inline void clearAmountEntry() {
  memset(amountBuffer, 0, sizeof(amountBuffer));
  amountIndex = 0;
}
void handleTransaction(char key) {
  if (currentState == TX_INPUT) {
    if (key >= '0' && key <= '9' && amountIndex < 5) {
      amountBuffer[amountIndex++] = key;
      lcd.setCursor(amountIndex - 1, 1);
      lcd.print(key);
      buzzerStart(BEEP_SHORT);
      return;
    }
    if (key == 'C') {
      clearAmountEntry();
      lcdFooterClear();
      buzzerStart(BEEP_SHORT);
      return;
    }

    if (key == 'O') {

      if (amountIndex == 0) {
```

```
      buzzerStart(BEEP_ERROR);
      lcdMessageF(F("Miktar Bos"), F("Giriniz"), 450);
      lcdHeaderF(F("Miktar Gir"));
      lcdFooterClear();
      return;
    }
    amountBuffer[amountIndex] = '\0';
    setState(TX_CONFIRM);
    lcdHeaderF(F("Islem Onay"));
    lcd.setCursor(0, 1);
    lcd.print(F("OK / X"));
    buzzerStart(BEEP_SHORT);
    return;
  }
  if (key == 'X') {
    setState(MENU);
    buzzerStart(BEEP_SHORT);
    showMenu();
    return;
  }
}
else if (currentState == TX_CONFIRM) {
  if (key == 'O') {
    lcdMessageF(F("Imzalaniyor"), F("..."), 220);
    txNonce++;
```

```
      fakeSignature = generateSignature();
      uint16_t sig16 = (uint16_t)((fakeSignature >> 16) & 0xFFFF);
      bool ok = appendBlockToSD(amountBuffer, sig16, txNonce);
      if (ok) {
        lcdMessageF(F("Islem"), F("SD Kaydedildi"), 450);
        buzzerStart(BEEP_DOUBLE);
      } else
     {
        lcdMessageF(F("SD Hata"), F("Kayit Yok"), 520);
        buzzerStart(BEEP_ERROR);
      }

      setState(MENU);
      showMenu();
      return;
    }

    if (key == 'X') {
      setState(MENU);
      buzzerStart(BEEP_SHORT);
      showMenu();
      return;
    }
  }
}
```

Ek A.6 – SD Kart Blok Loglama ve Geri Yükleme Kodları

Bu bölümde, işlemlerin SD kart üzerinde blok benzeri bir yapı ile kaydedilmesini ve sistem yeniden başlatıldığında son kaydın geri yüklenmesini sağlayan yazılım kodları sunulmaktadır.

```
static inline void clearLastTxCache() {
  memset(amountBuffer, 0, sizeof(amountBuffer));
  amountIndex = 0;
  fakeSignature = 0;
}

static inline bool isDigitC(char c) {
  return (c >= '0' && c <= '9');
}

static inline int hexVal(char c) {
  if (c >= '0' && c <= '9') return c - '0';
  if (c >= 'A' && c <= 'F') return 10 + (c - 'A');
  if (c >= 'a' && c <= 'f') return 10 + (c - 'a');
  return -1;
}

static uint32_t parseHex(const char* s) {
  uint32_t v = 0;
  while (*s) {
    int hv = hexVal(*s++);
    if (hv < 0) break;
    v = (v << 4) | (uint32_t)hv;
  }
  return v;
}

static uint32_t parseDec(const char* s) {
  uint32_t v = 0;
  while (*s && isDigitC(*s)) {
    v = v * 10 + (uint32_t)(*s - '0');
    s++;
  }
  return v;
}

void sdInit() {

  lcdMessageF(F("SD Kontrol"), F("Basliyor..."), 350);
  sdReady = SD.begin(SD_CS_PIN);

  if (sdReady) {
    sdEnsureHeader();
    lcdMessageF(F("SD Hazir"), F("Log aktif"), 450);
  } else {
    lcdMessageF(F("SD Yok/Ariza"), F("Log pasif"), 600);
  }
}
```

```
bool sdEnsureHeader() {

  if (!sdReady) return false;

  if (!SD.exists(SD_FILE_NAME)) {
    File f = SD.open(SD_FILE_NAME, FILE_WRITE);
    if (!f) return false;

    f.println(F("idx,ms,nonce,amount,sig16,prevHash,thisHash"));
    f.flush();
    f.close();
  }
  return true;
}

uint32_t fnv1a_update_u8(uint32_t h, uint8_t b) {
  h ^= b;
  h *= 16777619UL;
  return h;
}

uint32_t fnv1a_update_cstr(uint32_t h, const char* s) {
  while (*s) {
    h = fnv1a_update_u8(h, (uint8_t)(*s++));
  }
  return h;
}

uint32_t fnv1a_update_dec(uint32_t h, uint32_t v) {
  char buf[11];
  ultoa(v, buf, 10);
  return fnv1a_update_cstr(h, buf);
}

uint32_t fnv1a_update_hex(uint32_t h, uint32_t v) {
  char buf[9];
  ultoa(v, buf, 16);
  return fnv1a_update_cstr(h, buf);
}

bool appendBlockToSD(const char* amount, uint16_t sig16, uint16_t nonce) {

  if (!sdReady) return false;
  if (!sdEnsureHeader()) return false;

  File f;
  for (uint8_t i = 0; i < 3; i++) {
    f = SD.open(SD_FILE_NAME, FILE_WRITE);
    if (f) break;
    delay(10);
  }
  if (!f) return false;

  uint32_t h = 2166136261UL;
```

```
  f.print(blockIndex);
  h = fnv1a_update_dec(h, blockIndex);
  f.print(',');

  uint32_t ms = millis();
  f.print(ms);
  h = fnv1a_update_dec(h, ms);
  f.print(',');

  f.print(nonce);
  h = fnv1a_update_dec(h, nonce);
  f.print(',');

  f.print(amount);
  h = fnv1a_update_cstr(h, amount);
  f.print(',');

  f.print(sig16, HEX);
  h = fnv1a_update_hex(h, sig16);
  f.print(',');

  f.print(prevBlockHash, HEX);
  h = fnv1a_update_hex(h, prevBlockHash);
  f.print(',');

  f.println(h, HEX);

  f.flush();
  f.close();

  prevBlockHash = h;
  blockIndex++;

  return true;
}

bool sdClearLog() {

  if (!sdReady) return false;

  if (SD.exists(SD_FILE_NAME)) {
    if (!SD.remove(SD_FILE_NAME)) return false;
  }

  blockIndex = 0;
  prevBlockHash = 0;
  txNonce = 0;
  clearLastTxCache();

  return sdEnsureHeader();
}

bool sdRestoreLastRecord_tail() {
```

```
  if (!sdReady) return false;
  if (!SD.exists(SD_FILE_NAME)) return false;

  File f = SD.open(SD_FILE_NAME, FILE_READ);
  if (!f) return false;

  uint32_t sz = f.size();
  if (sz < 20) {
    f.close();
    return false;
  }

  const uint8_t N = 80;
  char buf[N + 1];

  uint32_t start = (sz > N) ? (sz - N) : 0;
  f.seek(start);

  uint8_t i = 0;
  while (f.available() && i < N) {
    buf[i++] = (char)f.read();
  }
  buf[i] = '\0';
  f.close();

  int end = (int)i - 1;
  while (end >= 0 && (buf[end] == '\n' || buf[end] == '\r')) end--;
  if (end < 0) return false;

  int j = end;
  while (j >= 0 && buf[j] != '\n') j--;
  int lineStart = j + 1;

  buf[end + 1] = '\0';
  char* line = &buf[lineStart];
  char* p = line;

  char* c1 = strchr(p, ','); if (!c1) return false;
  *c1 = 0;
  uint16_t idx = (uint16_t)parseDec(p);

  p = c1 + 1;
  char* c2 = strchr(p, ','); if (!c2) return false;
  *c2 = 0;

  p = c2 + 1;
  char* c3 = strchr(p, ','); if (!c3) return false;
  *c3 = 0;
  uint16_t nonce = (uint16_t)parseDec(p);

  p = c3 + 1;
  char* c4 = strchr(p, ','); if (!c4) return false;
  *c4 = 0;
  strncpy(amountBuffer, p, sizeof(amountBuffer) - 1);
  amountBuffer[sizeof(amountBuffer) - 1] = '\0';
```

```
  amountIndex = strlen(amountBuffer);

  p = c4 + 1;
  char* c5 = strchr(p, ','); if (!c5) return false;
  *c5 = 0;
  uint16_t sig16 = (uint16_t)parseHex(p);

  p = c5 + 1;
  char* c6 = strchr(p, ','); if (!c6) return false;
  *c6 = 0;

  p = c6 + 1;
  uint32_t thisH = parseHex(p);

  txNonce = nonce;
  fakeSignature = ((unsigned long)sig16) << 16;
  prevBlockHash = thisH;
  blockIndex = idx + 1;

  return true;
}
```

Ek A.7 – LED ve Buzzer Geri Bildirim Mekanizmaları

Bu bölümde, sistem durumlarına göre LED ve buzzer geri bildirimlerini yöneten yazılım kodları yer almaktadır.

```
void setState(State s) {
  currentState = s;
  applyIndicatorsForState();
}

void applyIndicatorsForState() {

  allLedsOff();

  switch (currentState) {

    case LOCKED:
      digitalWrite(LED_YELLOW, HIGH);
      break;

    case TWO_FA:
      digitalWrite(LED_YELLOW, HIGH);
      break;

    case MENU:
      digitalWrite(LED_GREEN, HIGH);
      break;

    case TX_INPUT:
    case TX_CONFIRM:
      digitalWrite(LED_YELLOW, HIGH);
```

```
      break;

    case LOG_CLEAR_CONFIRM:
      digitalWrite(LED_RED, HIGH);
      break;

    case TIME_LOCK:
      digitalWrite(LED_RED, HIGH);
      break;
  }
}

void allLedsOff() {
  digitalWrite(LED_GREEN, LOW);
  digitalWrite(LED_YELLOW, LOW);
  digitalWrite(LED_RED, LOW);
}

void buzzerOffNow() {
  digitalWrite(BUZZER_PIN, LOW);
  beepPattern = BEEP_NONE;
  beepStep = 0;
  beepNextMs = 0;
}

byte beepPriority(BeepPattern p) {

  switch (p) {
    case BEEP_LOCK:   return 5;
    case BEEP_ERROR:  return 4;
    case BEEP_DOUBLE: return 3;
    case BEEP_SHORT:  return 2;
    case BEEP_TICK:   return 1;
    default:          return 0;
  }
}

void buzzerStart(BeepPattern p) {

  if (p == BEEP_NONE) return;

  if (beepPattern != BEEP_NONE && beepPriority(p) < beepPriority(beepPattern)) return;

  beepPattern = p;
  beepStep = 0;
  beepNextMs = 0;
}

void buzzerUpdate() {

  if (beepPattern == BEEP_NONE) return;

  unsigned long now = millis();

  if (beepNextMs != 0 && now < beepNextMs) return;
```

```
switch (beepPattern) {

  case BEEP_SHORT:
    if (beepStep == 0) {
      digitalWrite(BUZZER_PIN, HIGH);
      beepNextMs = now + 70;
      beepStep++;
    } else {
      digitalWrite(BUZZER_PIN, LOW);
      beepPattern = BEEP_NONE;
    }
    break;

  case BEEP_DOUBLE:
    if (beepStep == 0) {
      digitalWrite(BUZZER_PIN, HIGH);
      beepNextMs = now + 60;
      beepStep++;
    } else if (beepStep == 1) {
      digitalWrite(BUZZER_PIN, LOW);
      beepNextMs = now + 70;
      beepStep++;
    } else if (beepStep == 2) {
      digitalWrite(BUZZER_PIN, HIGH);
      beepNextMs = now + 90;
      beepStep++;
    } else {
      digitalWrite(BUZZER_PIN, LOW);
      beepPattern = BEEP_NONE;
    }
    break;

  case BEEP_ERROR:
    if (beepStep == 0) {
      digitalWrite(BUZZER_PIN, HIGH);
      beepNextMs = now + 200;
      beepStep++;
    } else {
      digitalWrite(BUZZER_PIN, LOW);
      beepPattern = BEEP_NONE;
    }
    break;

  case BEEP_LOCK:
    if (beepStep == 0) {
      digitalWrite(BUZZER_PIN, HIGH);
      beepNextMs = now + 700;
      beepStep++;
    } else {
      digitalWrite(BUZZER_PIN, LOW);
      beepPattern = BEEP_NONE;
    }
    break;
```

```
    case BEEP_TICK:
      if (beepStep == 0) {
        digitalWrite(BUZZER_PIN, HIGH);
        beepNextMs = now + 40;
        beepStep++;
      } else {
        digitalWrite(BUZZER_PIN, LOW);
        beepPattern = BEEP_NONE;
      }
      break;

    default:
      buzzerOffNow();
      break;
  }
}
```

Ek B – ESP8266 TOTP Doğrulama Kodları

Bu ekte, iki faktörlü kimlik doğrulama (2FA) kapsamında ESP8266 üzerinde çalışan yazılım kodları sunulmaktadır.

Ek B.1 – WiFi Bağlantısı ve Zaman Senkronizasyonu

Bu bölümde, ESP8266 modülünün kablosuz ağa bağlanmasını ve zaman senkronizasyonunu sağlayan yazılım kodları yer almaktadır.

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <TOTP.h>

const char* WIFI_SSID = "Zekeriya Wifi";
const char* WIFI_PASS = "12345678";

// Secret (20 byte)
uint8_t secret[] = {
  0x11, 0x22, 0x33, 0x44, 0x55,
  0x66, 0x77, 0x88, 0x99, 0xAA,
  0xBB, 0xCC, 0xDD, 0xEE, 0xF0,
  0x0F, 0x12, 0x34, 0x56, 0x78
};

TOTP totp(secret, sizeof(secret));

// ===== NTP =====
```

```
WiFiUDP udp;
const char* NTP_SERVER = "pool.ntp.org";
const int NTP_PORT = 123;
const int NTP_PACKET_SIZE = 48;
byte packetBuffer[NTP_PACKET_SIZE];

unsigned long baseUnix = 0;
unsigned long baseMillis = 0;

void sendNtpPacket(const char* address) {
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  packetBuffer[0] = 0b11100011;
  packetBuffer[1] = 0;
  packetBuffer[2] = 6;
  packetBuffer[3] = 0xEC;
  udp.beginPacket(address, NTP_PORT);
  udp.write(packetBuffer, NTP_PACKET_SIZE);
  udp.endPacket();
}

unsigned long getNtpTime() {
  sendNtpPacket(NTP_SERVER);
  delay(1000);
  int cb = udp.parsePacket();
  if (!cb) return 0;
  udp.read(packetBuffer, NTP_PACKET_SIZE);

  unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
  unsigned long lowWord  = word(packetBuffer[42], packetBuffer[43]);
  unsigned long secsSince1900 = (highWord << 16) | lowWord;
  const unsigned long seventyYears = 2208988800UL;
  unsigned long epoch = secsSince1900 - seventyYears;
  return epoch;
}

void syncTime() {
  udp.begin(2390);
  unsigned long t = getNtpTime();
  if (t > 0) {
    baseUnix = t;
    baseMillis = millis();
  }
}

unsigned long unixNow() {
  unsigned long elapsedSec = (millis() - baseMillis) / 1000;
  return baseUnix + elapsedSec;
}

void setup() {
  Serial.begin(9600);
  Serial.println();
  Serial.println("ESP TOTP verifier basliyor...");

  WiFi.mode(WIFI_STA);
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
  }

  syncTime();
  Serial.print("Epoch: ");
  Serial.println(unixNow());
}
```

Ek B.2 – TOTP Üretim ve Doğrulama Mekanizması

Bu bölümde, zaman tabanlı tek kullanımlık parola üretimi ve doğrulamasını gerçekleştiren yazılım kodları sunulmaktadır.

```
bool verifyTotp6(const char* code6) {

  unsigned long now = unixNow();

  char* nowCode = totp.getCode(now);
  if (strcmp(nowCode, code6) == 0) {
    return true;
  }

  char* prevCode = totp.getCode(now - 30);
  if (strcmp(prevCode, code6) == 0) {
    return true;
  }

  char* nextCode = totp.getCode(now + 30);
  if (strcmp(nextCode, code6) == 0) {
    return true;
  }

  return false;
}
```

Ek B.3 – Arduino – ESP8266 Seri Haberleşme Protokolü

Bu bölümde, Arduino Uno ile ESP8266 modülü arasındaki seri haberleşmeyi yöneten yazılım kodları yer almaktadır.

```
// Arduino'dan komut formatı:
//   "PING\n" -> "PONG"
//   "Vxxxxxx\n" -> OK / NO

void loop() {
```

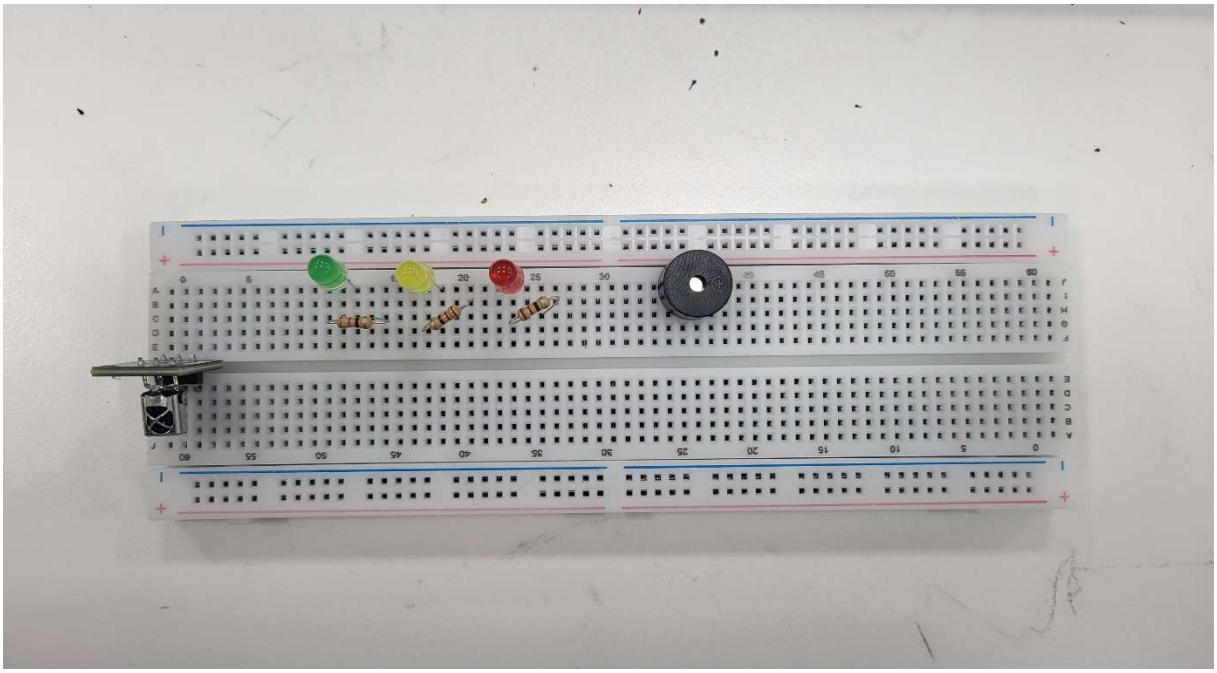
```
  static char buf[32];
  static uint8_t idx = 0;

  while (Serial.available()) {

    char c = (char)Serial.read();
    if (c == '\r') continue;

    if (c == '\n') {

      buf[idx] = 0;

      if (strcmp(buf, "PING") == 0) {
        Serial.println("PONG");
        idx = 0;
        return;
      }

      // Verify komutu: Vxxxxxx
      if (buf[0] == 'V' && strlen(buf) == 7) {

        const char* code6 = buf + 1;

        if (verifyTotp6(code6)) Serial.println("OK");
        else Serial.println("NO");

      } else {
        Serial.println("ERR");
      }

      idx = 0;
      return;
    }

    if (idx < sizeof(buf) - 1) buf[idx++] = c;
  }
}
```

Ek C – Donanım Uygulama Görselleri

Bu ekte, proje kapsamında kurulan donanım devresine ait görseller sunulmaktadır.

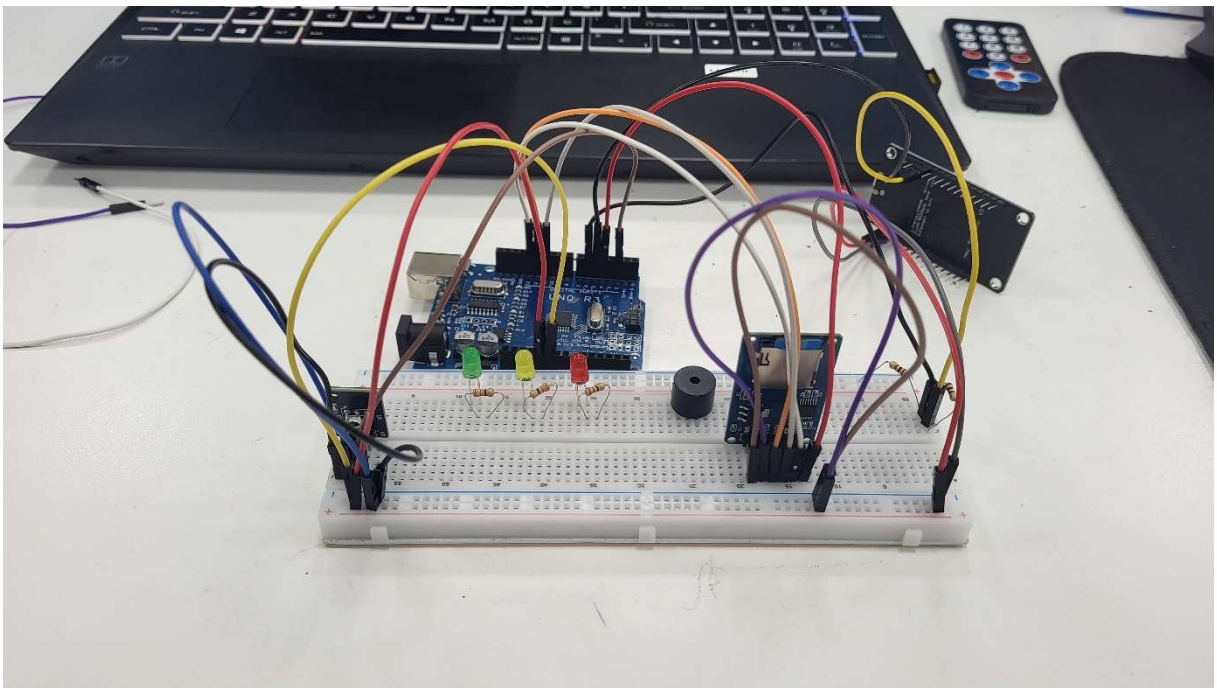
Ek C.1 – Giriş ve Geri Bildirim Bileşenleri

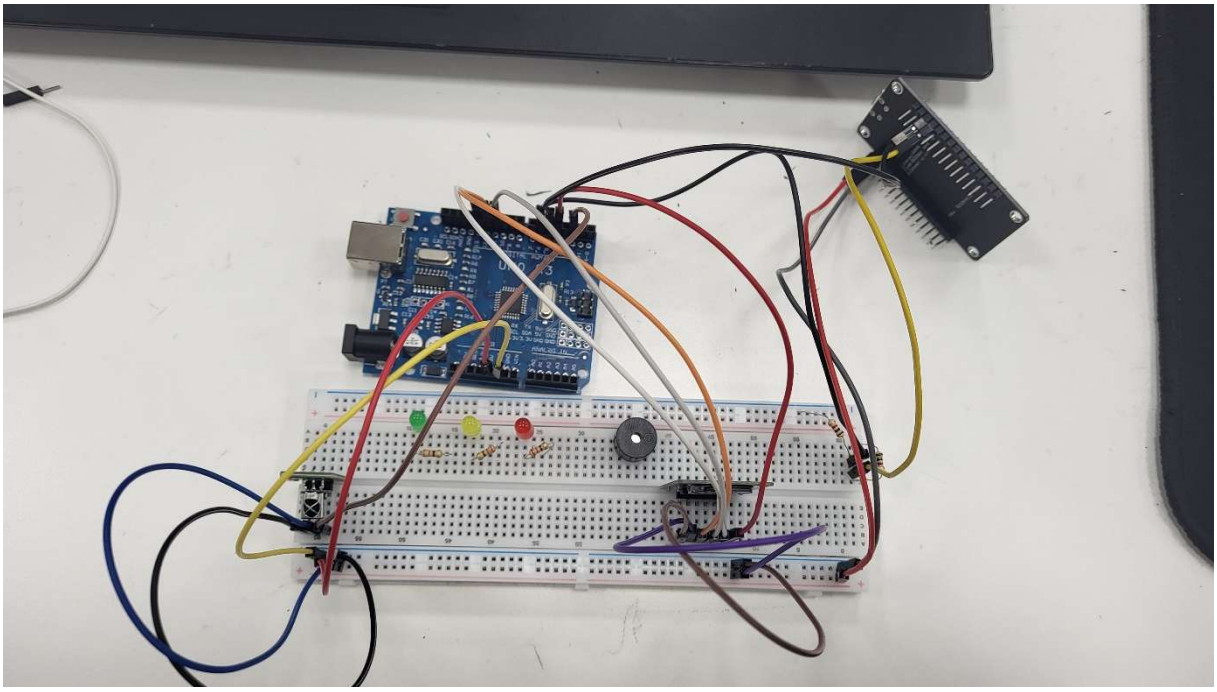
IR alıcı, LED ve buzzer bileşenlerinin devre üzerindeki konumlarını gösteren detaylı fotoğraflar.

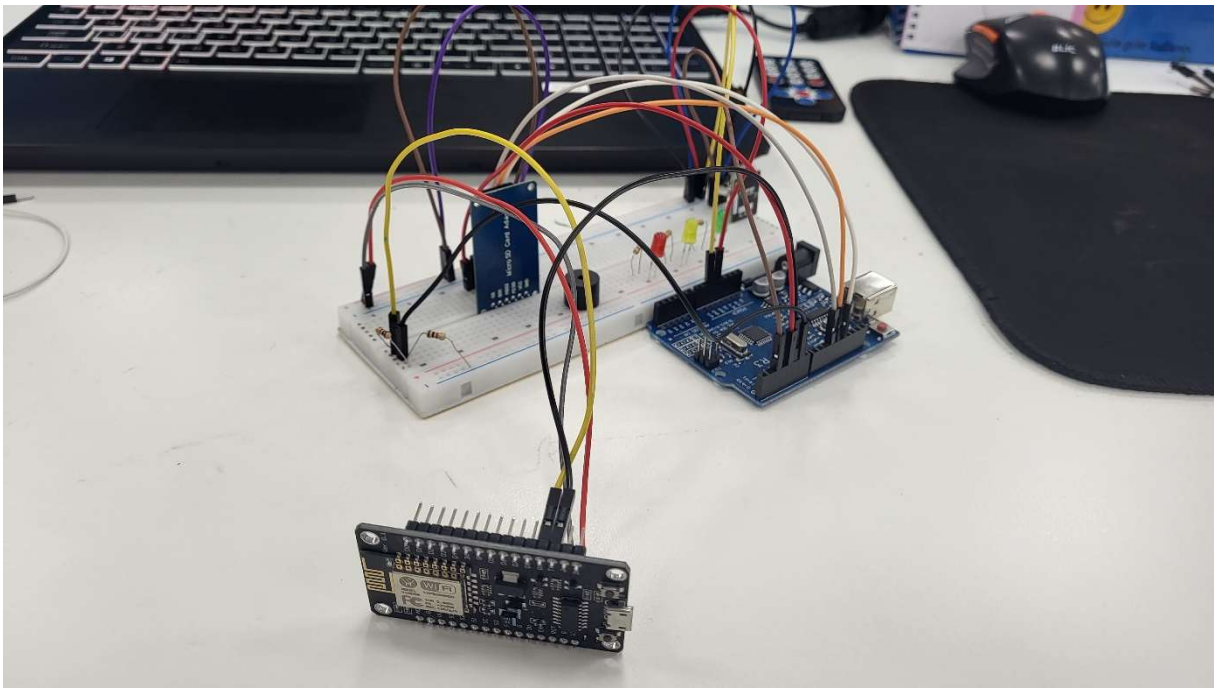


Ek C.2 – SD Kart ve ESP8266 Bağlantıları

SD kart modülü ve ESP8266 modülünün Arduino ile olan bağlantılarını gösteren yakın plan görseller.

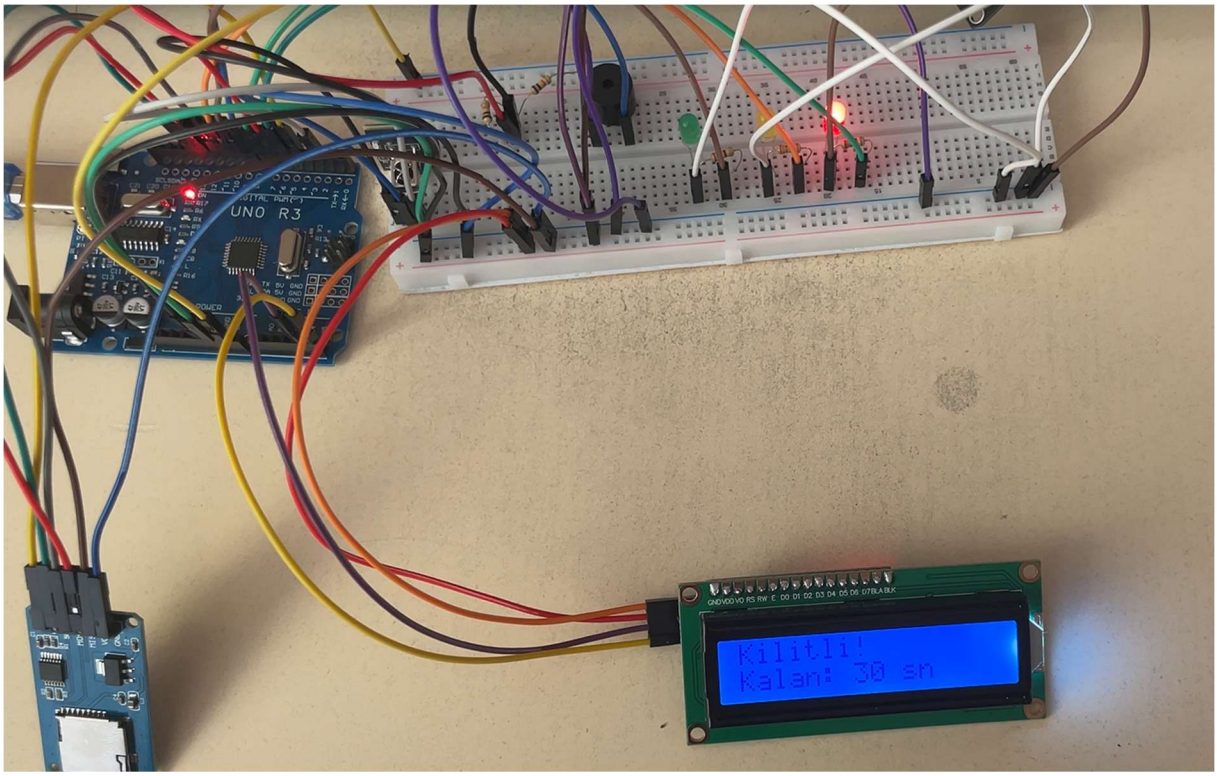






Ek C.3 – Genel Devre Görünümü

Arduino Uno, LCD ekran, IR alıcı, SD kart modülü, LED’ler, buzzer ve ESP8266 modülünün entegre edilmiş hâlini gösteren genel devre anlatımı, fotoğraf ve video linki:



VİDEO LİNKİ:

https://www.youtube.com/watch?v=KcW8VPC15Lg&t=1s