

Bluetooth İnsan Arayüz Cihazı Profili ile Kablosuz Hata Ayıklama

Denis Davidoglu

İstanbul Teknik Üniversitesi, Bilgisayar Mühendisliği Öğrencisi
ENTES Ar-Ge Gömülü Yazılım Stajyeri

Temmuz, 2023

İçindekiler

1	Giriş/Amaç	2
2	Teorik Bilgi	2
2.1	Bluetooth Yığını	2
2.2	İnsan Arayüz Cihazı Profili	3
2.3	HID Rapor Haritası	4
2.4	PS/2 Klavye Protokolü	6
3	Uygulama	7
3.1	Donanım prototipi	7
3.2	ESP-IDF kütüphaneleri	10
3.3	Gömülü veriler	10
3.4	Fonksiyonlar	12
4	Sonuç	13

1 Giriş/Amaç

Cihazlar, seri üretimine başlamadan önce Ar-Ge ve diğer böümlerde sıkı testlerden geçmek zorundadır. Eksiklik tespit edildiğinde Ar-Ge'deki gömülü yazılım mühendisleri geleneksel hata ayıklama yöntemlerine başvurabilmektedir. Fakat eğer cihaz müşteriye ulaşmış durumundaysa ve birçok teste rağmen arıza ortaya çıkmışsa, mühendislerin doğrudan müdaħalesi zorlaşmaktadır. Ayrıca, seri üretimine hazırlanmış devre kartı tasarımlarında JTAG, SWD, UART gibi hata ayıklama portları mevcut değildir. Sahada bulunan arızalı cihazların durumunu en detaylı şekilde üreticiye iletilmesini sağlayabilen Bluetooth İnsan Arayüz Cihazı¹ profili ile kablosuz hata ayıklama projesi bu raporda sunulmaktadır. Proje kapsamında, Bluetooth özelliği olan ESP32 mikrodenetleyicisi ile telefondan bilgisayara kadar her cihaza bağlanabilen Bluetooth klavye profili geliştirilmiştir. Söz konusu yazılım modulünün ekstra bir mesaj portu olarak kullanılması amaçlanmaktadır. Örneğin, arızalı cihaz cep telefonuna bağlandığı zaman, klavye yetkisini alarak müşteri hizmetlerinin e-posta adresi veya WhatsApp hattına teknik bilgilerini otomatik olarak yazabilmektedir. Bu yöntemin en büyük avantajı, ekramı ve internet bağlantısı olmayan en minimalist cihazlarda bile uygulanabilmesidir.

2 Teorik Bilgi

Bluetooth zor bir protokol olduğundan dolayı yazılım yazmaktan daha çok araştırma yapılmıştır. Bu bölümde Bluetooth ve USB hakkında temel kavramlar ve uygulamaya özel spesifikasyonlar hakkında bilgi verilmektedir. Ayrıca, yazılım modulünün gerçek bir klavyeye uygun olduğunu göstermek için prototipte PS/2 çıkışlı klavye kullanılmıştır ve protokolünü açıklayan kısım da aşağıda yer almaktadır.

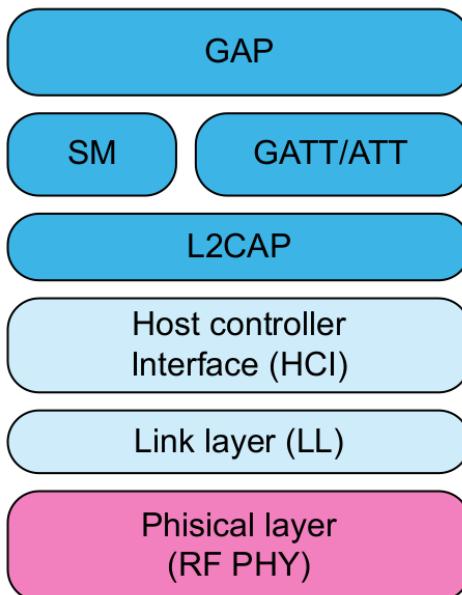
2.1 Bluetooth Yiğini

Fiziksel bağlantısı olmayan Bluetooth standartı 2.4 GHz civarındaki radyo dalgalarını kullanmaktadır, zamanlama açısından kritiktir ve büyük işlem gücü gerektirmektedir. Bu sebeple Bluetooth Yiğini² denen donanım parçalarına ve yazılıma ESP32, STM32WB gibi bazı mikrodenetleyicilerde özel ayrılmış bir işlemci çekirdeği bulunmaktadır. Projenin amacı kısa mesafede küçük veri paketlerini aktarmak olduğu için Bluetooth Düşük Enerji (BLE)³ teknolojisi incelenecaktır.

¹Human Interface Device (HID)

²Bluetooth Stack

³Bluetooth Low Energy



Görsel 1: Bluetooth Low Energy Yiğimi Mimarisi [1]

BLE Yiğini, kontrolcü ve sunucu iki komponentten oluşmaktadır. Kontrolcünün dahiline fiziksel radyo frekansı katmanı⁴ ve bağlantı katmanı⁵ varken mantık bağlantı kontrol ve adaptasyon protokolü (L2CAP)⁶, güvenlik yöneticisi (SM)⁷, nitelik protokolü (ATT)⁸, jenerik nitelik protokolü (GATT)⁹ ve jenerik erişim profili (GAP)¹⁰ sunucuya aittir. Sunucu kontrol arayüzü (HCI)¹¹, kontrolcü ile sunucuyu bağlayan, en düşük seviyeli yazılımla yönetilen yiğin parçasıdır [1]. API kullanan yüksek seviyeli kod, yiğin elemanları aracılığıyla paketler halinde veri alışverişi yapmaktadır ve frekans atlama algoritmasıyla haberleşme sağlamaktadır.

2.2 İnsan Arayüz Cihazı Profili

Bluetooth'tan iletilen verilerin diğer ucta doğru değerlendirilmesi için her veri servisi ve içindeki parametreler evrensiz benzersiz tanımlayıcıya (UUID)¹² sahiptir. USB standardından kaynaklanmasına rağmen Bluetooth GATT sunucusu ile HID profilinin tanımlanması mümkün değildir. HID Bluetooth cihazının en az üç tane servis sunması gerekmektedir: HID servisi (UUID: 0x1812), pil servisi (UUID: 0x180F) ve cihaz bilgi servisi (UUID: 0x180A). Veri özellikle bu numaralı servislere ayrıldığı zaman cihaz kendini bir klavye, fare veya başka bir girdi cihazı olarak tanıtmaktadır [2] [3].

⁴radio frequency physical layer

⁵link layer

⁶logical link control and adaptation protocol

⁷security manager

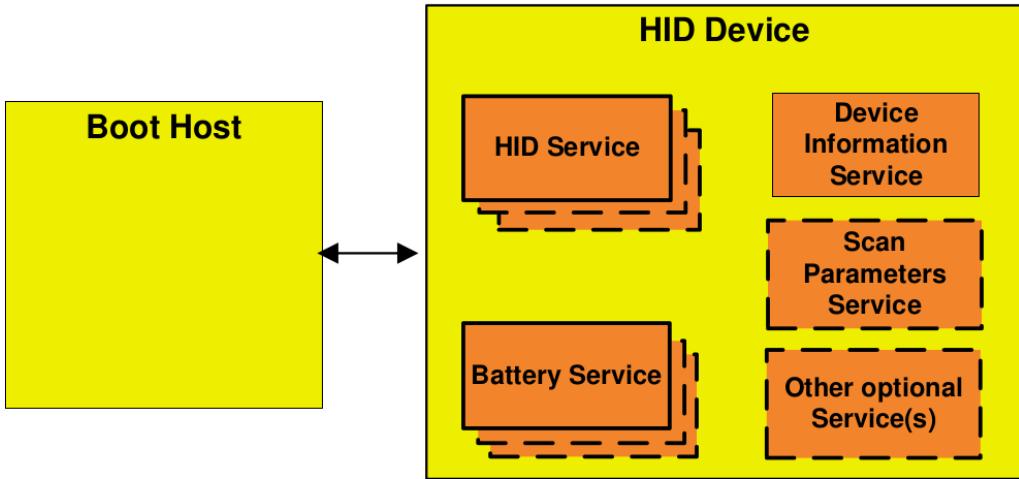
⁸attribute protocol

⁹generic attribute profile

¹⁰generic access profile

¹¹host controller interface

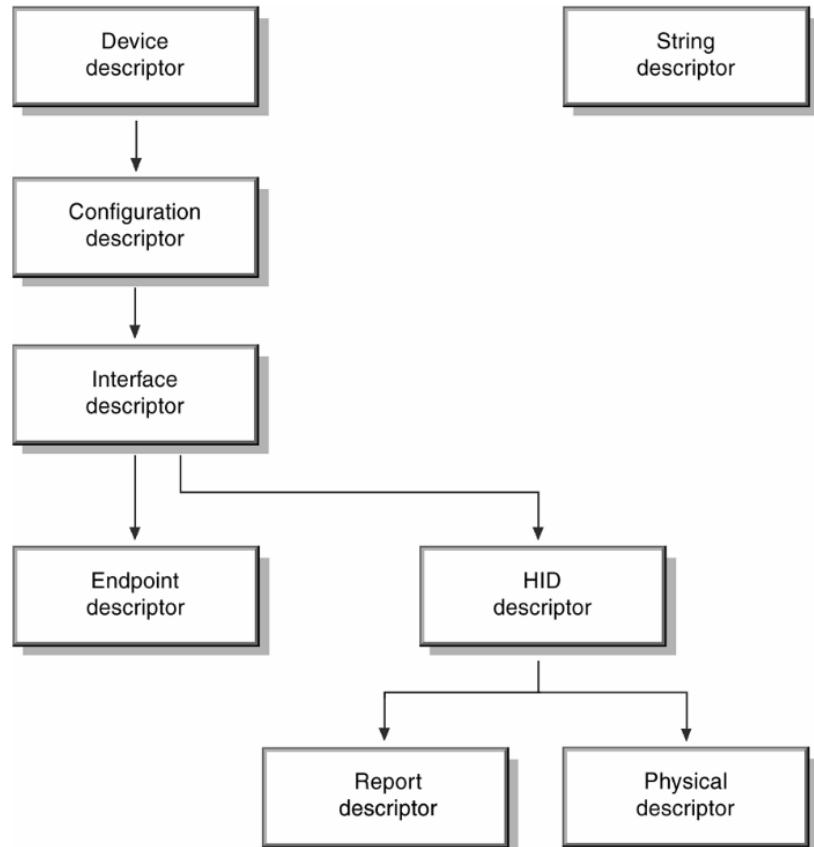
¹²universally unique identifier



Görsel 2: Sunucu ile cihaz rolleri/hizmetleri arasındaki ilişki [2]

2.3 HID Rapor Haritası

HID servisi içindeki 0x2A4B UUID'li tanımlayıcı, değerini değişmez ve USB HID спецификациина dayanılarak doldurulmaktadır. Aşağıdaki şema bir USB HID cihazının tanımlayıcılarının tümünü özetlemektedir, fakat Bluetooth HID cihazını yalnızca "HID descriptor" dalı ilgilendirmektedir.



Görsel 3: USB İnsan Arayüz Cihazı sınıf tanımlanması [4]

HID cihazları "fare", "joystick" gibi basit sınıflarla sınırlanmamaktadır. Aksine bir HID cihazı, insanın fiziksel olarak arayüzleyebildiği herhangi yöntemini uygulayabilmektedir ve birkaç profili kapsayabilmektedir. HID cihazının rapor haritası, sadece okunabilir bellekte (ROM) yer almaktır ve HID sürücüsüne gönderdiği ve aldığı veri formatını belirtmektedir. Dokümanda klavye için verilen hazır rapor haritası ileride kullanılmak üzere C programlama dili dizini şeklinde yazılmıştır [4]:

```

1 const unsigned char keyboardReportMap[] = {
2     0x05, 0x01, /* Usage Page (Generic Desktop), */
3     0x09, 0x06, /* Usage (Keyboard), */
4     0xA1, 0x01, /* Collection (Application), */
5         0x05, 0x07, /* Usage Page (Key Codes); */
6         0x19, 0xE0, /* Usage Minimum (224), */
7         0x29, 0xE7, /* Usage Maximum (231), */
8         0x15, 0x00, /* Logical Minimum (0), */
9         0x25, 0x01, /* Logical Maximum (1), */
10        0x75, 0x01, /* Report Size (1), */
11        0x95, 0x08, /* Report Count (8), */
12        0x81, 0x02, /* Input (Data, Variable, Absolute), */
13        0x95, 0x01, /* Report Count (1), */
14        0x75, 0x08, /* Report Size (8), */
15        0x81, 0x01, /* Input (Constant), */
16        0x95, 0x05, /* Report Count (5), */
17        0x75, 0x01, /* Report Size (1), */
18        0x05, 0x08, /* Usage Page (Page# for LEDs), */
19        0x19, 0x01, /* Usage Minimum (1), */
20        0x29, 0x05, /* Usage Maximum (5), */
21        0x91, 0x02, /* Output (Data, Variable, Absolute), */
22        0x95, 0x01, /* Report Count (1), */
23        0x75, 0x03, /* Report Size (3), */
24        0x91, 0x01, /* Output (Constant), */
25        0x95, 0x06, /* Report Count (6), */
26        0x75, 0x08, /* Report Size (8), */
27        0x15, 0x00, /* Logical Minimum (0), */
28        0x25, 0x65, /* Logical Maximum(101), */
29        0x05, 0x07, /* Usage Page (Key Codes), */
30        0x19, 0x00, /* Usage Minimum (0), */
31        0x29, 0x65, /* Usage Maximum (101), */
32        0x81, 0x00, /* Input (Data, Array), */
33    0xC0 /* End Collection */
34 };

```

Klavye en yaygın girme metodu olduğu için formatı standartlaşmış ve verilen örnekte fazla değişiklik yapılamamaktadır. Haritayı anlamak için dikkat edilmesi gereklidir. Klavyenin gönderdiği basılan buton kodlarına *Input* denir. Sadece *Input* ve onlardan önceki *Report Size* ve *Report Count* maddeleri vurgulanırsa çıktı paketinin formatı öğrenilebilir:

```

10    0x75, 0x01, /* Report Size (1), */
11    0x95, 0x08, /* Report Count (8), */
12    0x81, 0x02, /* Input (Data, Variable, Absolute), */

```

(Sekiz tane tek bitli öğe. Her bit Shift, Alt, Ctrl gibi değiştirici butonu işaretlemektedir.)

```

13    0x95, 0x01, /* Report Count (1), */
14    0x75, 0x08, /* Report Size (8), */

```

```
15     0x81, 0x01, /* Input (Constant), */
```

(Bir tane sekiz bitli öğe. Sabit sıfır değeri vardır ve gelecek kullanım için ayrılmıştır.)

```
25     0x95, 0x06, /* Report Count (6), */
26     0x75, 0x08, /* Report Size (8), */
27     ...
32     0x81, 0x00, /* Input (Data, Array), */
```

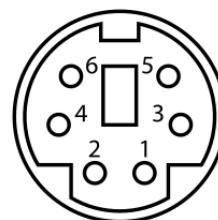
(Altı tane sekiz bitli öğe. Aynı anda basılan en fazla altı tuş hakkında bilgi taşımaktadır.)

Toplamba aktarılan verinin büyüklüğü 8 bayttır. Ancak son kısımdaki altı tuş özelliği metin girmek için gereksizdir, bu yüzden sadece tek tuş takip edilebilir:

```
25     0x95, 0x01, /* Report Count (1) */
26     0x75, 0x08, /* Report Size (8), */
27     ...
32     0x81, 0x00, /* Input (Data, Array), */
```

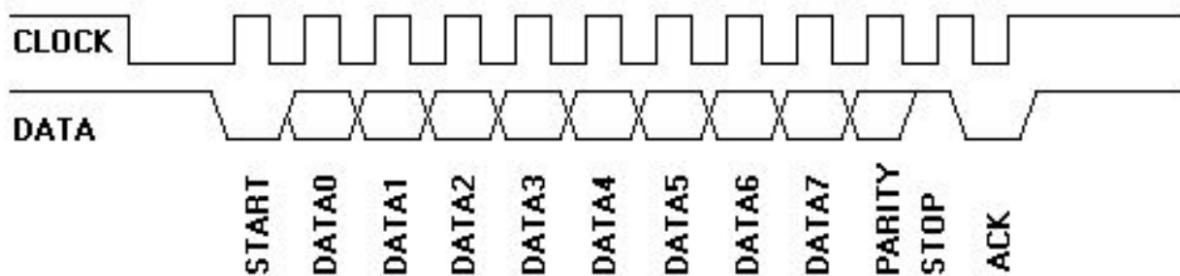
2.4 PS/2 Klavye Protokolü

PS/2, USB'den önceki ilkel bir klavye ve fare protokolüdür. Clock ve veri olmak üzere iki pin ile 10.0–16.7 kHz frekanslarında seri haberleşme desteklediği için herhangi bir mikrodenetleyicide kolayca sürülebilir. Bluetooth kodunu test etmek amacıyla bu tür klavye kullanılmıştır.



Görsel 4: PS/2 portu

Hem clock hem veri sinyali ilk başta 1 düzeyindedir. Clock'un çıkan kenarında veri kaydedilmektedir. Senkronizasyon için başlangıç biti her zaman 0, bitiş biti her zaman 1, ve bitişten önceki bit tek eşlik bitidir. Aralarındaki 8 bit tuş kodlarını taşımaktadır. Cihaz-sunucu haberleşmesi böylelikle 11 bit ile gerçekleşmektedir. Sunucu-cihaz haberleşmesi için aşağıdaki resimde gösterildiği gibi ACK 12. bit olarak eklenebilir, ancak uygulamada kullanılmamıştır.



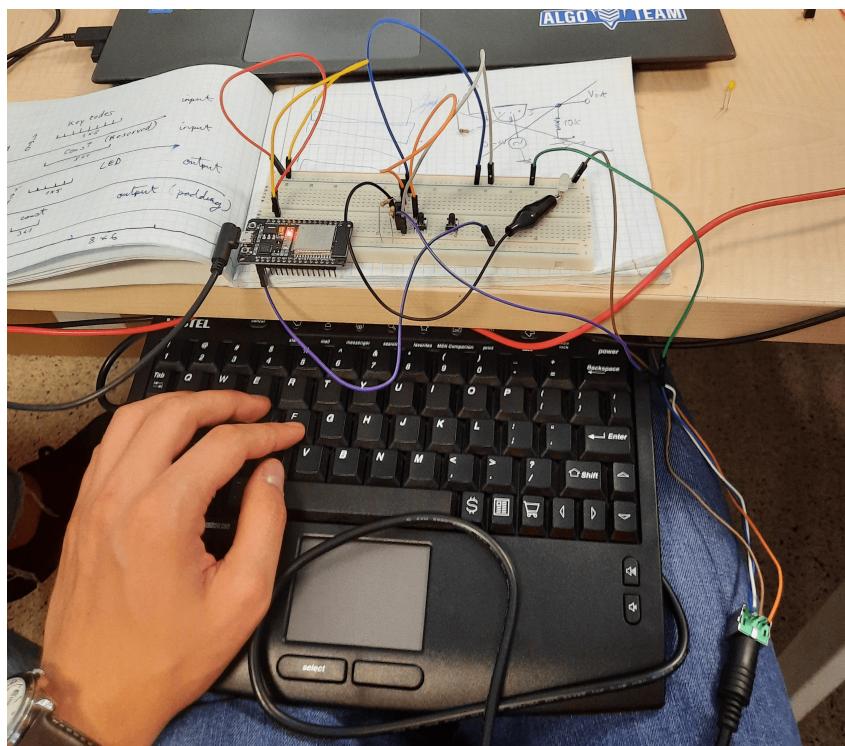
Görsel 5: PS/2 cihaz-sunucu haberleşmesi [7]

3 Uygulama

Teorik bilgilerden istifade edilerek Espressif ESP32 mikrodenetleyicisi ile Bluetooth uygulamasının donanım ve yazılımı hazırlanmıştır. Bu kısımda çalışan prototipin yapımı detaylı açıklanmaktadır.

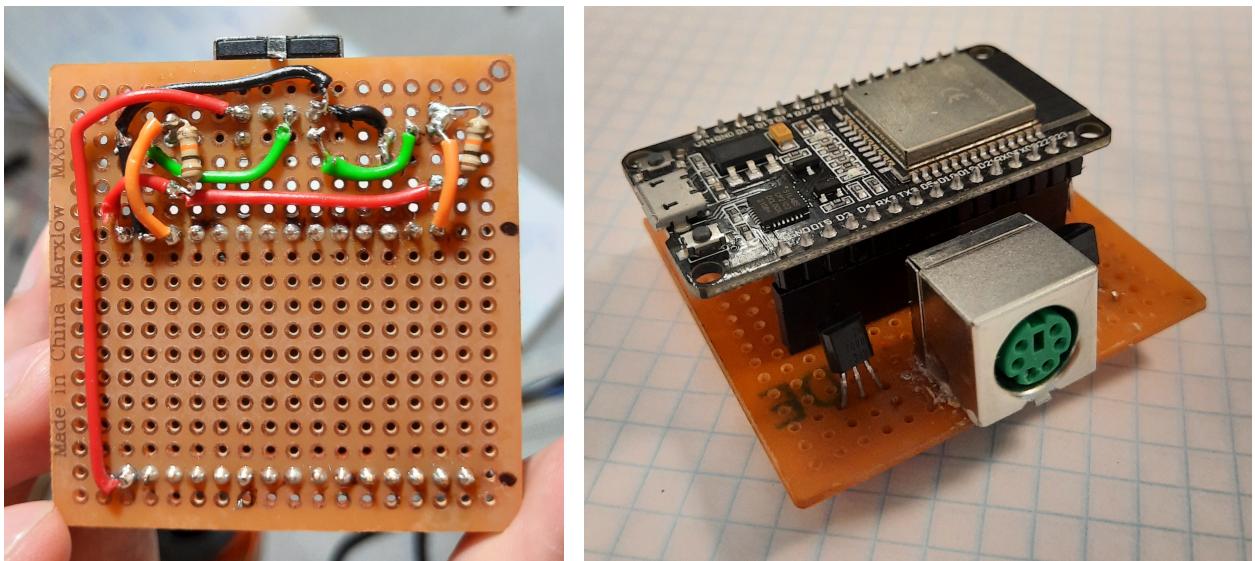
3.1 Donanım prototipi

PS/2 klavyesi ile ESP32 mikrodenetleyicisi mantık voltajı farklarından dolayı uyumsuzdur: ilki 5 volt ile çalışırken diğer 3.3 volt ile çalışmaktadır. Çözüm olarak clock ve veri sinyalleri için birer tane 2N7000 mosfeti kullanılmıştır. Sinyaller gate'e, 3.3 volt ile drain arasında $10\text{ k}\Omega$ resistör, 0 volt source'a bağlanmıştır. Breadboard denemesinde düşürülmüş sinyal LED'e bağlanmıştır ve klavye tuşları basıldığında LED'in yanması gözlemlenmiştir.



Görsel 6: Breadboard denemesi

Başarılı bir denemeden sonra daha rahat kullanım için devre, delikli pertinaksa lehimlenmiştir.



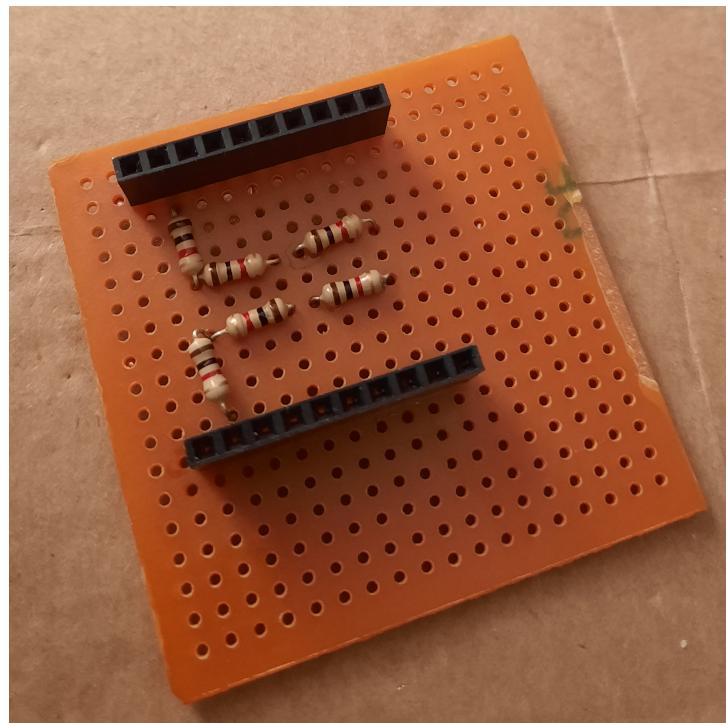
Görsel 7: Birinci prototip

Kullanılan PS/2 girişi kalitesiz çıktılarından dolayı temassızlık oluşturmaktaydı ve klavyenin kablolarının direkt plakete lehimlenmesine karar verilmiştir. Bir şüpheli transistör de değiştirilmiştir.



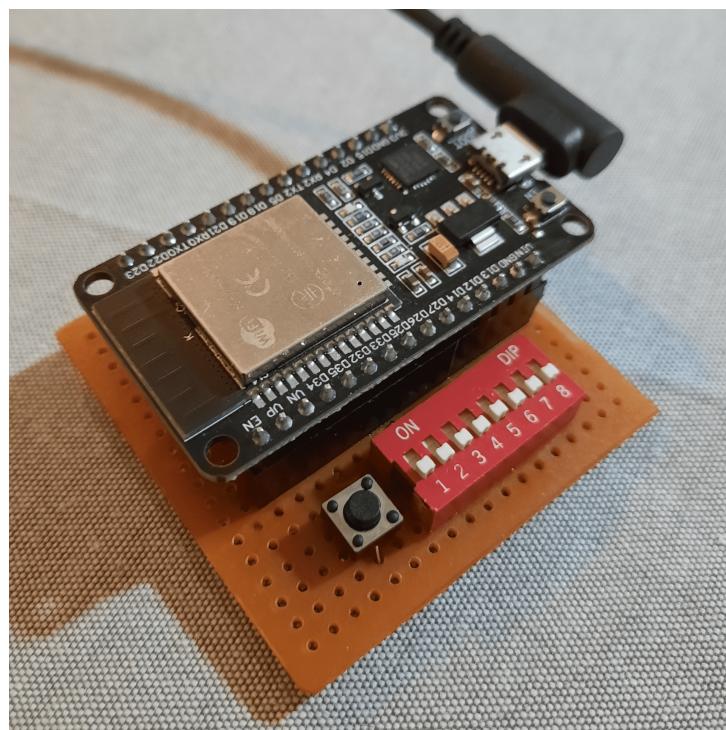
Görsel 8: İkinci prototip

İkinci prototip bir süre iş görmekteydi, hatta bozulmaması adına bağlantılar bir tekniker yardımıyla silikon ile sabitlenmiştir. Bir süre daha çalışıp bozulmuştur. Üçüncü prototipte mosfet yerine resistörlerle voltaj bölücü devresi kurulmuştur, fakat bir fark olmamıştır.



Görsel 9: Üçüncü prototip

Son prototip olarak gerçek klavye fikrinden vazgeçilip 8'li switch ve buton konulmuştur. Switch'te bir ASCII karakteri seçildikten sonra butona basılır. Klavye olarak elverişli olmasa da Bluetooth modülünün çalıştığını göstermek için yeterlidir.



Görsel 10: Prototipin son hali

3.2 ESP-IDF kütüphaneleri

ESP32 çipinin önerilen programlanma aracı olan Espressif IoT Development Framework (ESP-IDF), birçok kütüphane ve örnek kodu içermektedir. Bluetooth İnsan Arayüz Ci-hazı profili de bunların arasındadır, lisanslanmamış Bluetooth faresi örnek kodu bulunmaktadır. Geliştirilen uygulama Espressif'in sunduğu aşağıdaki kütüphanelerden bağımlıdır [8] [9]:

- "freertos/event_groups.h"
- "freertos/FreeRTOS.h"
- "freertos/task.h"
- "freertos/event_groups.h"
- "freertos/queue.h"
- "esp_system.h"
- "esp_wifi.h"
- "esp_event.h"
- "esp_log.h"
- "nvs_flash.h"
- "esp_bt.h"
- "esp_bt_defs.h"
- "esp_bt_main.h"
- "esp_bt_device.h"
- "esp_hidd.h"
- "esp_hid_gap.h"
- "driver/gpio.h"

3.3 Gömülü veriler

Uygulama için azımsanmayacak miktarda tablo hazırlanmıştır. PS/2 tuş kodları, USB tuş kodları ile ASCII karakterlerinin arasında bağlantı kurmak amacıyla HID Rapor Haritası yanı sıra program koduna aşağıdaki tablolar gömülümüştür:

ps2_scancodes tablosu, PS/2'dan okunan bayt ile indekslendiğinde girilen karakterin ASCII kodu elde edilir [6].

```
1 static const char ps2_scancodes[] = {  
2     '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0',  
3     '\0', '\0', '\0', '\0', '\0', 'Q', '1', '\0', '\0', '\0', 'z', 'S', 'A', 'W', '2', '\0',  
4     '\0', 'C', 'X', 'D', 'E', '4', '3', '\0', '\0', '/', 'V', 'F', 'T', 'R', '5', '\0',  
5     '\0', 'N', 'B', 'H', 'G', 'Y', '6', '\0', '\0', '\0', 'M', 'J', 'U', '7', '8', '\0',  
6     '\0', '/', 'K', 'I', 'O', '0', '9', '\0', '\0', '.', '/', 'L', ';', 'P', '\0', '\0', '\0',  
7     '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0',  
8     '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0', '\0',  
9 };
```

`usb_scancodes` tablosuna bir ASCII karakteri indeks olarak verildiğinde Bluetooth'tan gönderilmesi gereken kod elde edilir [5]. Burada verilen değerler, Türkçe Q klavyeye özeldir, fakat ABD İngilizce klavye ayarı da mevcuttur.

```

1 static const uint8_t usb_scancodes[] = {
2 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2a, 0x00, 0x28, 0x00, 0x00, 0x00, 0x00, 0x00
3 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
4 // ! " # $ % & ' ( ) * + , - . /
5 0x2c, 0x1e, 0x35, 0x20, 0x21, 0x22, 0x23, 0x1f, 0x25, 0x26, 0x2d, 0x21, 0x31, 0x2e, 0x38, 0x24
6 // 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
7 0x27, 0x1e, 0x1f, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x38, 0x31, 0x35, 0x27, 0x1e, 0x2d
8 // @ A B C D E F G H I J K L M N O
9 0x14, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x34, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12
10 // P Q R S T U V W X Y Z [ \ ] ^ -
11 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x25, 0x2d, 0x26, 0x20, 0x2e
12 // ` a b c d e f g h i j k l m n o
13 0x34, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x34, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12
14 // p q r s t u v w x y z { | } ~
15 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x24, 0x2e, 0x27, 0x30, 0x00
16 };

```

Bazı karakterler Shift ya da Alt Gr ile birlikte girilmektedir. Her karaktere bir int ayırmak yerine `uint16_t` değişkeninin her biti ile 16 karakterin Shift durumu tutularak bellek tasarrufu sağlanmıştır. Aynı şekilde Alt Gr için de tablo oluşturulmuştur.

```

1 static const uint16_t usb_modifier_shift[] = {
2 // 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
3 0x0020,
4 // 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5 0x0000,
6 // ! " # $ % & ' ( ) * + , - . /
7 // 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1
8 0x47d1,
9 // 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
10 // 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1
11 0x0035,
12 // @ A B C D E F G H I J K L M N O
13 // 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14 0x8000,
15 // P Q R S T U V W X Y Z [ \ ] ^ -
16 // 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
17 0x0003,
18 // ` a b c d e f g h i j k l m n o
19 // 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
20 0x7fff,
21 // p q r s t u v w x y z { | } ~
22 // 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
23 0xffe0
24 };

```

```

1 static const uint16_t usb_modifier_alt[] = {
2 // 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 0x0000,
4 // 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5 0x0000,
6 // ! " # $ % & ' ( ) * + , - . /
7 // 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
8 0x3a00,
9 // 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
10 // 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
11 0x000a,
12 // @ A B C D E F G H I J K L M N O

```

```

13 // 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14 0x8000,
15 // P Q R S T U V W X Y Z [ \ ] ^ -
16 // 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0
17 0x001c,
18 // ` a b c d e f g h i j k l m n o
19 // 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
20 0x8000,
21 // p q r s t u v w x y z { | } ~
22 // 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
23 0x001e
24 };

```

Bir makro ile istenen bite ulaşılabilir:

```
#define HAS_MODIFIER(a, i) (((a)[(int)(i)/16]&(1<<(15-(int)(i)\%16))) != 0)
```

Türkçe'ye özel karakterler char tipinin bir baytına siğmadığı için ek olarak aşağıdaki veri yapısı tanımlanmıştır. İki boyutlu dizinin her satırında tek Türkçe karakterinin iki baytı, USB tarama kodu, Shift ve Alt Gr modlarından null sonlandırıcı ile bölünmektedir. Böyle bir tasarım sayesinde strcmp fonksiyonu bir Türkçe karakterinin dizindeki daha detaylı karşılığını bulabilmektedir.

```

1 static const char usb_scancodes_special[][6] = {
2 // char, char, null, key, shift, alt
3 {0xc4, 0x9f, '\0', 0x2f, 1, 0}, // ğ
4 {0xc3, 0xbc, '\0', 0x30, 1, 0}, // ü
5 {0xc5, 0x9f, '\0', 0x33, 1, 0}, // ş
6 {0xc4, 0xb1, '\0', 0x0c, 1, 0}, // ı
7 {0xc3, 0xb6, '\0', 0x36, 1, 0}, // ö
8 {0xc3, 0xa7, '\0', 0x37, 1, 0}, // ç
9 {0xc4, 0x9e, '\0', 0x2f, 0, 0}, // Ğ
10 {0xc3, 0x9c, '\0', 0x30, 0, 0}, // Ü
11 {0xc5, 0x9e, '\0', 0x33, 0, 0}, // Ş
12 {0xc4, 0xb0, '\0', 0x34, 0, 0}, // İ
13 {0xc3, 0x96, '\0', 0x36, 0, 0}, // Ö
14 {0xc3, 0x87, '\0', 0x37, 0, 0}, // Ç
15 };

```

3.4 Fonksiyonlar

Espressif'in verdiği koduna birkaç fonksiyon eklenerek Bluetooth'tan haberleşme kısmı arka planda bırakılmıştır. Yazılan fonksiyonlar sayesinde bağlanılan cihaza herhangi bir karakter veya uzun metin yazdırılabilir.

Tek char veri tipine siğan karakter yazdırmaktadır. Eğer kısa bir süre sonra 0x00 kodu gönderilmezse, önce girilen karakter hala basılı sayılır ve bağlanılan cihazda sürekli aynı karakter durmadan girilecektir:

```
void hid_send_ascii_char(char);
```

Bir veya daha fazla char'dan oluşan karakter yazdırmaktadır. Bir yukarıdaki fonksiyonda olduğu gibi kısa bir süre sonra karakter alanı sıfırlanmalıdır.

```
int hid_send_nonascii_char(char*);
```

Caps Lock tuş kodunu göndermektedir. Sıfırlanmalıdır.

```
void hid_send_capslock();
```

Metin yazdırın fonksiyon. Her 40 milisaniyede bir karakter göndermekte ve en sonda karakter alanını kendi sıfırlamaktadır.

```
void hid_send_string(char*);
```

PS/2 klavyesinin clock kenarlarında çağrılan kesme fonksiyonu, 11 bit saydıktan sonra buton bilgisini toplamış olup ps2_print fonksiyonunu uyandırmaktadır.

```
void ps2_interrupt_respond(void*);
```

FreeRTOS'un Queue yapısından buton bilgisini okuyup Bluetooth yapılarına iletmektedir.

```
void ps2_print(void*);
```

Kodun final halinde ps2_ ön ekli fonksiyonlar ve değişkenler dahil edilmemiştir.

4 Sonuç

Projenin ilk haftasında Bluetooth ve USB hakkında yoğun *araştırma* yapmak, ikinci haftasında da soyut fikri *geliştirme* denen süreçte hayatı geçirirmek, Ar-Ge bölümünde çalışma tecrübesini yaşamıştır. Gömülü yazılımda iki haftada satışa çıkarılacak bir proje kesinlikle yapılmaz, fakat anlatılan Bluetooth İnsan Arayüz Cihazı modülü, mikrodenetleyiciden veri almanın farklı bir yolunu sunmakta başarılı olmuştur. 8'li DIP switch'ten tek karakter gönderebildiği gibi veri kaybetmeden uzun metin yazdırma testinden de geçmiştir. ENTES'in bazı cihazları, ESP32 çipi tarafından denetlendiği için geliştirilen yazılımın onlara eklenmesi mümkündür.

Kaynakça

- [1] STMicroelectronics (2023), *Guidelines for Bluetooth® Low Energy stack programming on STM32WB/STM32WBA MCUs*, Rev 7.
- [2] Bluetooth SIG, Inc. (2011), *HID over GATT Profile Specification*.
- [3] Bluetooth SIG, Inc. (2023), *Assigned Numbers*.
- [4] USB Implementers' Forum (2001), *Device Class Definition for Human Interface Devices (HID)*, Version 1.11.
- [5] USB Implementers' Forum (2022), *HID Usage Tables for Universal Serial Bus (USB)*, Version 1.3.
- [6] Microsoft Corporation (2004), *USB HID to PS/2 Scan Code Translation Table*.
- [7] Adam Chapweske (2003), *The PS/2 Mouse/Keyboard Protocol*.
- [8] Espressif Systems (Shanghai) Co., Ltd (2023), *ESP-IDF API Reference*.
- [9] Amazon.com, Inc. (2017), *Reference Manual for FreeRTOS*, version 10.0.0, issue 1.