HABERLEŞME PROTOKOLLERİ: UART, SPI, 12C

Haberleşme protokolleri, cihazların birbirleriyle iletişim kurmasını sağlayan standart kurallar ve formatlar setidir. İşte bazı yaygın kullanılan haberleşme protokolleri:

1-) UART (Universal Asynchronous Receiver-Transmitter) – Evrensel Asenkron Alıcı-Verici

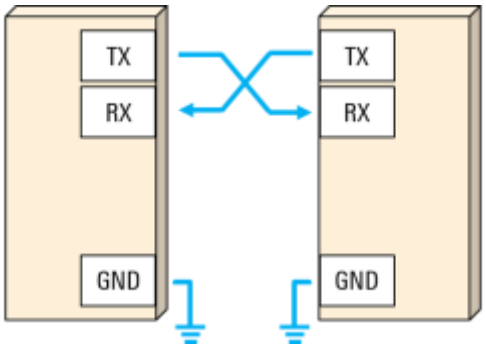
UART, Evrensel Asenkron Alıcı/Verici'nin kısaltmasıdır ve iki cihaz arasında seri veri alışverişi için kullanılan basit ve yaygın bir haberleşme protokolüdür. Bu protokol, asenkron yapısı sayesinde verici ve alıcı arasında ortak bir saat sinyali bulunmadan iletişim sağlar. Asenkron olması, protokolün en büyük avantajlarından biridir, ancak verici ve alıcıya belirli gereksinimler getirir.

UART'ın asenkron yapısı, her iki uç arasında ortak bir saat sinyali olmadığı için, veri iletimi sırasında her iki cihazın da aynı hızda iletim yapması gerektiği anlamına gelir. Bu önceden belirlenmiş hızda gerçekleşir. UART, çok basit bir yapıdadır ve veri iletimi için yalnızca iki kablo kullanır: bir veri iletim hattı ve bir toprak bağlantısı. İletişim tek yönlü, yarı çift yönlü veya tam çift yönlü olabilir. Ayrıca, veriler çerçeveler halinde iletilir, bu da verilerin belirli bir düzen içinde gönderilmesini sağlar.

Sonuç olarak, UART, basitliği, düşük maliyeti ve uygulanabilirliği nedeniyle genellikle düşük hızlarda ve verimlilik gerektirmeyen uygulamalarda tercih edilir. Hem donanım hem de yazılım açısından kolay uygulanabilir olması, bu protokolü popüler kılmaktadır. Ancak, daha yüksek hızlarda ve daha verimli iletişim gereksinimlerinde, UART yerine başka protokoller tercih edilebilir.

Temel Özellikleri:

- **Asenkron** haberleşme: Ortak saat sinyali gerekmez.
- **İki pin yeterli**: TX (veri gönderme), RX (veri alma)
- **Veri iletimi** "çerçeve" şeklindedir: başlangıç biti + veri bitleri + bitiş biti + (isteğe bağlı eşlik biti)
- Her iki uçta da **önceden aynı baud hızı** ayarlanmalıdır (örn. 9600, 115200 baud)
- **Basit**, düşük maliyetli, küçük sistemler için idealdir



Pin Karşılaştırması:

UART Fonksiyonu	Arduino UNO	STM32 (Blue Pill)	Açıklama
TX (Gönderme)	D1	PA9	Veriyi gönderir
RX (Alma)	D0	PA10	Veriyi alır

STM32'de bu pinler genellikle **USART1** içindir, farklı UART modülleri de (USART2, USART3…) mevcuttur. Bu modüller, donanım içinde yer alan bağımsız seri iletişim birimleridir.

STM32'de genelde "USART" adı kullanılır, ancak UART (sadece asenkron) olarak da çalışabilir. Kısaca: **USART(Universal Synchronous/Asynchronous Receiver Transmitter)**, hem senkron hem asenkron haberleşmeyi destekler.

Kullanım Alanları (Roket İçin):

• Bluetooth modülleri (HC-05 gibi) ile kablosuz iletişim

• GPS modülleri ile konum verisi alımı

• Yer kontrol birimi ile seri bağlantı

• Roketteki mikrodenetleyiciden bilgisayara veri aktarımı

Avantajlar:

• Basit kurulum, düşük maliyet

• Donanım gereksinimi az

• 2 kablo ile çalışır

Dikkat Edilmesi Gerekenler:

• Her iki cihazın baud hızı tam olarak aynı olmalı

• Aynı hatta sadece 2 cihaz bağlanabilir (1 gönderici, 1 alıcı)

• Hata kontrolü sınırlı (isteğe bağlı eşlik biti ile)

Kullanım Alanları (Roket İçin):

- **Bluetooth modülleri** (HC-05 gibi) ile kablosuz iletişim
- **GPS modülleri** ile konum verisi alımı
- **Yer kontrol birimi ile seri bağlantı**
- Roketteki mikrodenetleyiciden bilgisayara veri aktarımı

Avantajlar:

- Basit kurulum, düşük maliyet
- Donanım gereksinimi az
- 2 kablo ile çalışır

Dikkat Edilmesi Gerekenler:

- Her iki cihazın baud hızı **tam olarak aynı olmalı**
- Aynı hatta sadece **2 cihaz** bağlanabilir (1 gönderici, 1 alıcı)
- **Hata kontrolü sınırlı** (isteğe bağlı eşlik biti ile)

2. SPI (Serial Peripheral Interface)

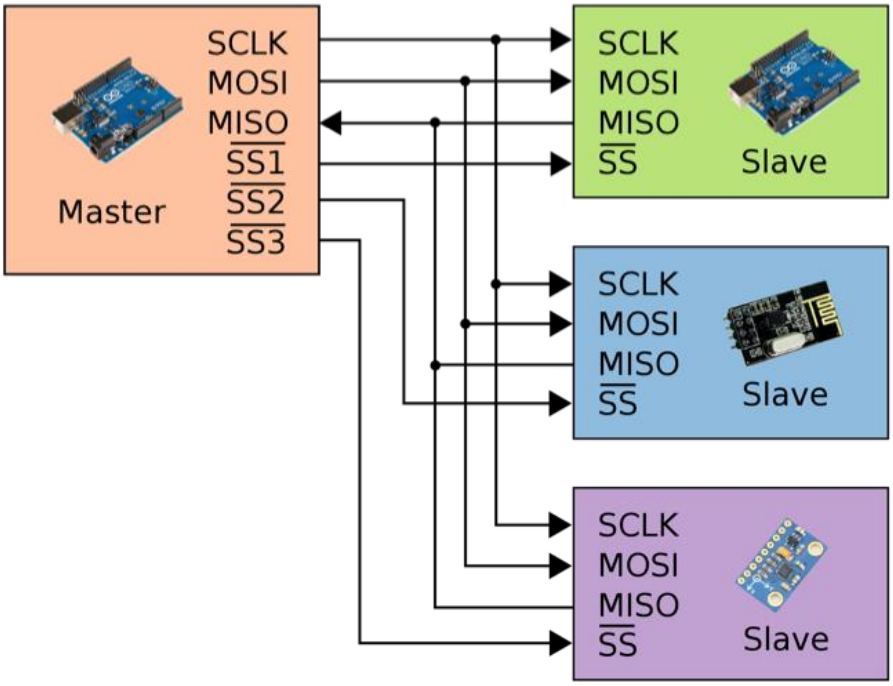
Temel Özellikler:

- Senkron iletişim, saat sinyali master cihaz tarafından sağlanır.
- 4 temel pin kullanılır: MOSI, MISO, SCK, SS
- Full-duplex (aynı anda veri gönderip alınabilir)
- Master-Slave yapısıyla çalışır, her slave için ayrı SS hattı gerekir.

SPI'de Kullanılan Temel Bağlantılar (Pinler)

SPI haberleşmesi 4 adet kablo/pin üzerinden yapılır:

1. **SCLK (Saat Sinyali)/ SCK (Clock):** Ana cihaz tarafından gönderilen zamanlayıcı sinyalidir. Bu sinyal, verilerin ne zaman gönderilip alınacağını belirler.
2. **MOSI (Master Out, Slave In):** Ana cihazın veri gönderdiği hattan slave cihaz veriyi alır. Yani veri masterdan slave'e gider.
3. **MISO (Master In, Slave Out):** Slave cihazın veri gönderdiği hattan master veriyi alır. Yani veri slave'den mastera gider.
4. **SS (Slave Select)/ CS (Chip Select):** Hangi slave cihazın aktif olacağını belirlemek için kullanılır. Birden fazla slave varsa, her birine özel bir SS hattı olur. Aktif olan slave'in SS hattı LOW (0V) seviyesine çekilir.



SPI Pin Karşılaştırması: Arduino UNO vs STM32 (Blue Pill):

SPI Fonksiyonu	**Arduino UNO Pin No**	**STM32 (Blue Pill) GPIO Pin**	**Açıklama**
MOSI	D11 veya ICSP4	**PA7**(A portunun 7. Pini, STM32 üzerinde A7 şeklinde yazar)	Master Output, Slave Input
MISO	D12 veya ICSP1	**PA6**	Master Input, Slave Output
SCK	D13 veya ICSP3	**PA5**	Saat sinyali
SS (Slave Select)	D10	**PA4** *(genelde)*	Slave seçimi pini
SS (Master olarak kullanılırsa)	Kullanıcı tanımlar	STM32'de herhangi bir GPIO (örneğin PB0, PB12 vs.)	Birden fazla slave varsa seçmek için

- STM32'de SPI haberleşmesini kullanmak için bu pinleri **AFIO (Alternate Function)** olarak ayarlamak gerekir.

- Arduino'da SPI kütüphanesi bu pinleri otomatik kullanır, STM32'de genelde **HAL**, **LL** ya da doğrudan register üzerinden ayarlama yapılır.

- STM32'de **SPI2** gibi ikinci SPI modülü de kullanılabilir, fakat başlangıç için **SPI1 (PA5-PA6-PA7)** yeterlidir.

Kullanım Alanları:

- Yüksek hızda veri iletimi gerektiren sistemlerde kullanılır. Örneğin, sensörler Yüksek hızlı sensörler, SD kartlar, ekranlar, sıcaklık veya ivme ölçerler ve Roket sisteminde hızlı veri gerektiren alt sistemlerle haberleşme ile veri alışverişi yaparken kullanılır

Avantajlar:

- Çok hızlıdır (10 Mbps+)
- Aynı anda veri alışverişi yapılabilir

Dezavantajlar:

- Her slave için ayrı SS hattı gerekir
- Kablo sayısı fazladır

Kısaca Özetleyecek Olursak:

- SPI hızlı ve güvenilir bir haberleşme protokolüdür.
- Master-Slave yapısında çalışır.
- 4 temel pin kullanır: SCLK, MOSI, MISO, SS.
- Aynı anda veri gönderip alabilir (full-duplex).
- İsimler cihazlara göre değişse de mantık aynıdır.

3-) I2C (Inter-Integrated Circuit) – İki Telli Seri Protokol

I2C protokolü, kısa mesafeli entegre devreler arası iletişim için kullanılan ve düşük güç tüketimi gerektiren bir iletişim yöntemidir. Bu protokol, bir ana cihaz ve birden fazla bağımlı cihaz veya çoklu ana cihazlar arasında iletişim sağlamak için yaygın olarak kullanılır. I2C'nin temel özelliklerinden biri, her başlatma (START) ve durdurma (STOP) durumu çifti arasında veri yolunun meşgul kabul edilmesidir, bu sırada başka bir master cihaz veri yolunu kontrol edemez.

Eğer bir master yeni bir veri aktarımı başlatmak ister ama veri yolunu bırakmak istemezse, yeni bir START durumu başlatarak "Tekrarlanan Başlatma" (Repeated Start) durumuna geçer. I2C, sadece iki pin kullanır: SDA (Serial Data) veri hattı ve SCL (Serial Clock) saat hattı. Ayrıca, her I2C cihazının bir adresi vardır ve bu adres, cihazlar arasında iletişim için kullanılır.

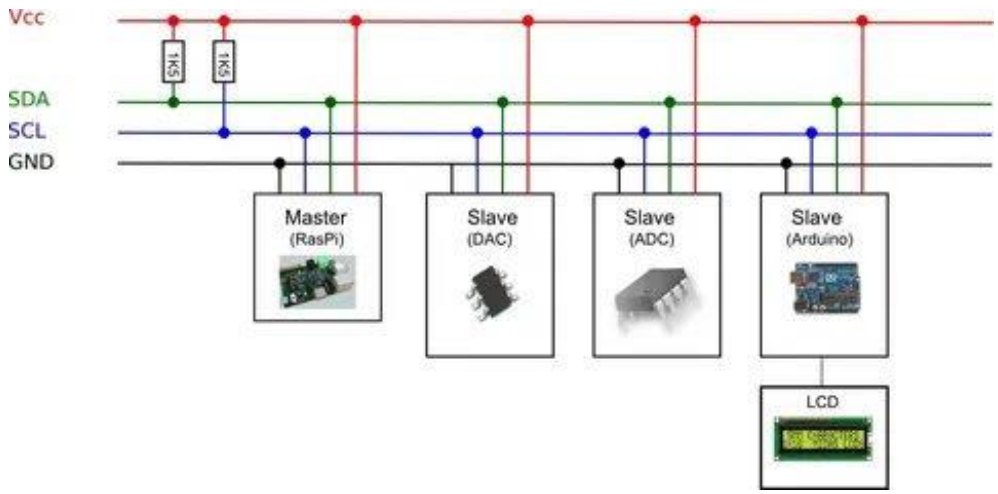
Cihaz adresi genellikle 7 bit veya 10 bit uzunluğundadır ve bu adresin nasıl ayarlanacağı datasheet’te belirtilir. I2C'nin veri iletim hızı ise genellikle 100 kHz (Standard Mode) veya 400 kHz (Fast Mode) olabilir; hangi hızların desteklendiği de datasheet’lerde yer alır. Bu özellikler, I2C protokolünün cihazlar arasında veri iletimini güvenilir ve verimli bir şekilde gerçekleştirmesini sağlar. (Web sitelerinde detaylı teknik bilgiler yer alır.)

İki Temel Pin Kullanımı:

- **SDA (Serial Data):** Verilerin iletildiği ana hat. Veriler bu hat üzerinden iletilir ve alınır.
- **SCL (Serial Clock):** Saat sinyalini taşıyan hattır. Bu hat, verilerin zamanlamasını kontrol eder ve veri iletimi senkronize edilir.

Temel Özellikler:

- **Senkron İletişim:** Veriler, saat sinyali ile senkronize edilir. Saat sinyali, master cihaz tarafından sağlanır.
- **Yarı Çift Yönlü:** Aynı anda veri gönderme ve alma yapılamaz, ancak iki yönlü iletişim mümkündür.
- **Master-Slave Yapısı:** Bir master cihaz ile bir veya daha fazla slave cihaz arasında iletişim kurulur.
- Pull-up dirençler gereklidir (genellikle 4.7kΩ)
- Aynı hatta birçok cihaz bağlanabilir.



I2C Pin Karşılaştırması:

I2C Fonksiyonu	Arduino UNO	STM32 (Blue Pill)	Açıklama
SDA (Veri)	A4	PB7	Veri hattı
SCL (Saat)	A5	PB6	Saat hattı

- **Arduino'da** I2C kütüphanesi SDA ve SCL pinlerini otomatik olarak kullanır.
- **STM32'de** bu pinler AFIO (Alternate Function) olarak ayarlanmalı ve HAL veya LL kütüphaneleri ile kullanılmalıdır.

I2C Protokolünün Özellikleri:

- **Yarı çift yönlü iletişim**: Aynı anda veri gönderme ve alma yapılamaz, ama iki yönlü iletişim mümkündür.

- **Clock Stretching:** Slave cihaz veri almaya hazır değilse, SCL hattını LOW tutarak master'ı bekletir.

- **Arbitraj (Çekişme Çözümü):** Eğer iki master cihaz aynı anda veri iletimine başlarsa, SDA ve SCL hatları izlenerek hangisinin devam edeceği belirlenir.
- **Seri İletim:** Veriler tek bir hat üzerinden sırayla gönderilir.
- **Düşük Hızda İletişim İçin Uygun:** I2C, genellikle düşük hızda veri iletimi gerektiren uygulamalarda tercih edilir.

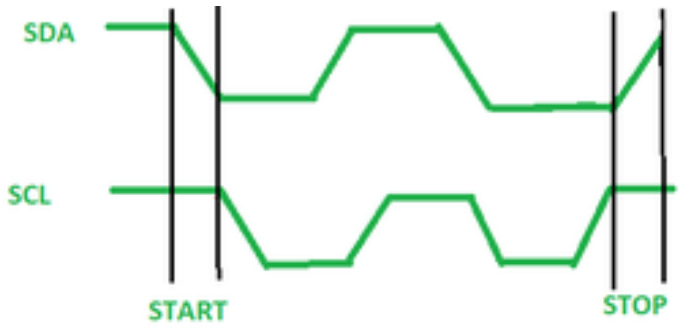
I2C İletişim Adımları:

1. **Başlatma Durumu (START):** Verici, SDA hattını düşükten yüksek seviyeye çekerek başlatma durumu oluşturur.
2. **Adres Çerçevesi:** Verici, köle cihazın adresini gönderir ve okuma/yazma bitini belirler. Cihaz, adresi karşılaştırarak ACK sinyali gönderir.(7-bit veya 10-bit adres)
3. **Okuma/Yazma Biti (R/W):** Hangi işlemin yapılacağını belirler.
4. **Veri İletimi:** Veriler 8-bit olarak gönderilir. Her byte'dan sonra ACK/NACK sinyali alınır.
5. **Durma Durumu (STOP):** Verici, SDA hattını yüksekten düşük seviyeye çeker ve iletimi bitirir.

I2C İletim Formatı:

I2C veri iletimi, 9 bitlik paketler halinde yapılır. Bu paketler aşağıdaki sırayla iletilir:

1. **Başlangıç Durumu (START):** İletimin başladığını gösterir.
2. **Köle Adresi:** 7-bit veya 10-bit adres.
3. **Okuma/Yazma Biti (R/W):** Hangi işlemin yapılacağını belirler.
4. **Veri İletimi:** Veri iletimi gerçekleşir. Her byte'dan sonra ACK/NACK ile yanıt alınır.
5. **Durma Durumu (STOP):** İletim tamamlandığında iletişim sona erer.



Tekrarlanan Başlatma (Repeated Start):

- **Tekrarlanan Başlatma (Repeated Start)** durumu, master cihazının yeni bir veri aktarımına başlamadan önce veri yolunu bırakmadan START durumunu tekrar başlatmasıdır.

- Bu durum, veri yolunun başka bir master tarafından kullanılmasını engeller ve aynı veri yolunda iletişimin devam etmesini sağlar.

ACK/NACK:

- **ACK (Onay) Biti:** Verinin başarıyla alındığını belirtmek için alıcı tarafından gönderilir.
- **NACK (Reddetme) Biti:** Veri alınamadığında alıcı tarafından gönderilir.

I2C Adresleme:

- START bitinden sonra gelen ilk çerçeve adres çerçevesidir.
- Master, haberleşmek istediği slave'in adresini gönderir.
- Her slave bu adresi kendi adresiyle karşılaştırır ve eşleşirse ACK gönderir

I2C Paket Formatı

- I2C'de veri 9 bitlik paketler halinde gönderilir.
- İlk 8 bit **SDA hattı** üzerinden veri olarak gider.
- 9.bit, alıcının **ACK/NACK** durumu için ayrılmıştır.

Bir veri aktarımı: START + adres paketi + veri paketi + STOP şeklinde olur.

I2C BUS İletişim Yapısı:

I2C protokolü, her cihazın benzersiz bir adresi olmasını sağlar. böylece master hangi cihazlarla iletişim kuracağını seçebilir. Bu yönteme Seri Saat (veya SCL) ve Seri Veri (veya SDA) denir. SCL hattı, I2C veriyolundaki cihazlar ile ana cihaz tarafından üretilen veri aktarımı arasında senkronize olan saat sinyalidir.

I2C protokolünde veriler **SDA** (Serial Data) hattı üzerinden taşınır. Verilerin ne zaman gönderileceği ise **SCL** (Serial Clock) hattı üzerinden belirlenir. Bu iki hat iletişim boyunca birlikte çalışır.

Bu iki hatta bağlı cihazların düzgün çalışabilmesi için hatlar genellikle **"boştayken" HIGH (yüksek voltaj)** seviyesinde tutulur. Bunu sağlamak için SDA ve SCL hatlarına **pull-up direnç** adı verilen dirençler bağlanır. Bu dirençler hatları VCC'ye bağlayarak, sinyallerin normalde yüksek kalmasını sağlar.

Pull-up direnci nedir?
Hat üzerindeki voltajın yüksek (1) seviyede kalması için hattı VCC'ye çeken bir dirençtir.

I2C veriyolu **aktif düşük** yapıda çalışır. Yani bir cihaz veri göndermek istediğinde, bağlı olduğu hattı geçici olarak **LOW (sıfır volt)** yapar. Bu şekilde hem birden çok cihaz aynı hatta bağlanabilir hem de veri çakışmaları önlenmiş olur.

Dirençlerin değeri, haberleşme hızına göre seçilir:

Hız	Tipik Direnç Değeri
100 kbps	10kΩ
400 kbps	2kΩ - 4.7kΩ

Eğer pull-up dirençleri doğru bir şekilde kullanılmazsa, I2C iletişimi kararsız çalışır veya hiç çalışmaz.

Bu yapı,STM32 mikrodenetleyicisi için I2C Master ve Slave iletişimini düzgün bir şekilde kurmanıza yardımcı olacaktır. Master cihaz veri gönderirken, Slave cihaz aldığı veriye göre LED'i yakar veya söndürür ve geri yanıt gönderir.

Avantajları:

- **Çoklu Master Desteği:** Birden fazla master cihaz aynı veri yoluna bağlanabilir.
- **Düşük Maliyet:** I2C yalnızca iki hat kullanır (SDA ve SCL).
- **Hata Kontrolü:** ACK/NACK bitleri ile hata kontrolü sağlanır.
- **Birden Fazla Cihaz:** Aynı veri yoluna birden çok cihaz bağlanabilir.
- **Kolay Programlama:** I2C'nin kullanımı, diğer protokollere göre daha basittir.

Dezavantajları:

- **Sınırlı Hız:** SPI'ye göre daha yavaştır.
- **Kısa Mesafede Kullanım:** Uzun mesafelerde iletişim için uygun değildir.
- **Yarı Çift Yönlü İletişim:** Veri alışverişi sırayla yapılır, aynı anda veri gönderilemez.

Kullanım Alanları:

- **Sensörler ve Hafıza Modülleri:** MPU6050, BMP280 gibi sensörlerle iletişim sağlanabilir.
- **RTC, EEPROM, LCD Modülleri:** I2C üzerinden çoklu sensör bağlantısı yapılabilir.
- Tek hatta çoklu sensör bağlantısı

STM32 I2C Protokolü ile Master ve Slave İletişimi:

STM32 mikrodenetleyicisi ile I2C master ve slave cihazları arasındaki iletişim, HAL kütüphanesi kullanılarak yapılabilir. Aşağıda STM32 I2C master ve slave için yazılmış bir örnek kod yapısı bulunmaktadır:

Master Cihaz (STM32) Kodu:

```
#include "stm32f1xx_hal.h"
#include <string.h>

extern I2C_HandleTypeDef hi2c1; // I2C yapılandırması

void master_send_data(void) {
```

```
    uint8_t veri[] = "a";  // Gönderilecek veri
    HAL_I2C_Master_Transmit(&hi2c1, 0x08 << 1, veri, strlen((char*)veri),
HAL_MAX_DELAY);
}

void master_receive_data(void) {
    uint8_t gelen[10];  // Alınacak veri
    HAL_I2C_Master_Receive(&hi2c1, 0x08 << 1, gelen, 7, HAL_MAX_DELAY);
    // Gelen veri ile ilgili işlemler
}
```

Slave Cihaz (STM32) Kodu:

```
#include "stm32f1xx_hal.h"

extern I2C_HandleTypeDef hi2c1; // I2C yapılandırması

void slave_receive_data(void) {
    uint8_t alinan[10];  // Alınan veri
    HAL_I2C_Slave_Receive(&hi2c1, alinan, sizeof(alinan), HAL_MAX_DELAY);

    // Gelen veri ile LED yakma işlemi
    if (alinan[0] == 'a') {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET); // LED yak
    } else if (alinan[0] == 'b') {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);   // LED
söndür
    }
}

void slave_send_data(void) {
    uint8_t yanit[] = "Merhaba";  // Cevap veri
    HAL_I2C_Slave_Transmit(&hi2c1, yanit, strlen((char*)yanit),
HAL_MAX_DELAY);
}
```

Kısaca Özetleyecek Olursak:

- I2C, sadece 2 hat (SDA ve SCL) ile birden fazla cihaz arasında haberleşme sağlayan, senkron ve yarı çift yönlü bir protokoldür.
- **Master cihaz** köle cihazla iletişime başlar, veri gönderir ve alır.
- Veriler **9 bitlik paketler** halinde gönderilir, her byte'dan sonra **ACK/NACK** sinyali alınır.
- Düşük hızda, kısa mesafe veri iletimi için idealdir. Çok sayıda cihazı tek hatta bağlamak için uygundur.
- **I2C**, **SPI'ye** göre daha basit ve ucuzdur ancak biraz daha yavaştır.

I2C ve SPI Protokollerinin Karşılaştırması

Özellik	I2C	SPI
Tel Sayısı	2 (SDA, SCL)	4 (MOSI, MISO, SCK, SS)
İletişim Tipi	Yarı çift yönlü	Tam çift yönlü
Cihaz Sayısı	Adresleme ile çok sayıda	Her cihaz için bir SS hattı
Veri Hızı	Daha yavaş	Daha hızlı
Hata Yönetimi	ACK/NACK ile daha iyi	Daha az sağlam
Maliyet	Daha ucuz (daha az tel)	Daha pahalı (fazla tel)
Karmaşıklık	Daha basit	Daha karmaşık

SPI - I2C - UART Karşılaştırma Tablosu

Özellik	UART	SPI	I2C
Veri İletim Hızı	Orta (~9600 bps - 1 Mbps)	Çok yüksek (10+ Mbps, bazı cihazlarda 50+ Mbps)	Orta (~100 kbps - 3.4 Mbps arası)
Kullanılan Pinler	2 pin: TX, RX	4 pin: MOSI, MISO, SCK, SS	2 pin: SDA, SCL
İletim Yönü	Full-duplex (eş zamanlı çift yönlü)	Full-duplex (eş zamanlı çift yönlü)	Half-duplex (tek yönlü, dönüşümlü)
Bağlantı Sayısı	Sınırlı (genelde 1 master, 1 slave)	Az (Her slave için bir SS hattı gerekebilir)	Çok (adresleme ile 100+ cihaz)
Kullanım Alanları	PC ile seri iletişim, Bluetooth, GPS, WiFi	Flash bellek, sensör, DAC/ADC, ekran	EEPROM, sensör, RTC, LCD, IO genişletici
Avantajları	• Donanım gerektirmez • Basit kurulum	• Çok hızlı • Full-duplex • Basit protokol	• Az tel • Çoklu cihaz desteği • Adresleme var
Dezavantajları	• Cihaz sayısı sınırlı • Senkronizasyon zor olabilir	• Çok tel • Adresleme yok • Master-slave sınırlı	• Daha yavaş • Half-duplex • Karmaşık yazılım

Özellik	UART	SPI	I2C
Hata Kontrolü	Var (start/stop bit, parity bit opsiyonel)	Genellikle yok (manuel yapılır)	Var (ACK/NACK ile)
Maliyet	Çok düşük (sadece 2 pin, donanımsız bile çalışır)	Orta-yüksek (çok pin, karmaşık yapı)	Düşük (2 pin yeterli, basit devre)
Uygulama Alanları	PC bağlantısı, düşük hızlı cihaz haberleşmesi	Hızlı veri aktarımı gereken sistemler	Çoklu sensör ve cihaz içeren sistemler

Ekstra Bilgilerle Kıyas

- **Kablo sayısı**: I2C < UART < SPI
- **Hız**: SPI > UART ≈ I2C
- **Cihaz Sayısı**: I2C > SPI > UART
- **Maliyet**: UART < I2C < SPI
- **Veri Yönü**: SPI & UART (Full-Duplex), I2C (Half-Duplex)

Bağlantılar ile ilgili videolar:

Arduino - Haberleşme protokolleri - SPI - Senkron Haberleşme:

https://youtu.be/PYk3AbUP2a4?si=5IELys_lOO_w8Iij

How to Use SPI Communication in STM32 Microcontroller:

https://youtu.be/-E_MWULHUMQ?si=NEFKbRI9oWMmAr28

UART Haberleşme ile Bilgisayardan Veri Alımı STM32F401:

https://youtu.be/psA6FWEcpok?si=YLmYch4xW9bU6vXK

STM32 İle UART Haberleşme Kesmesi:

https://youtu.be/7cQDsnbvO2o?si=eYgJhjLa4Nf_uNpt

I2C HABERLEŞME UYGULAMASI 1 | ARDUINOLAR ARASI VERI OKUMA VE GONDERME | I2C COMMUNICATION ON ARDUINO

https://youtu.be/gheioBgBa3o?si=ExUprNyd1ugGVoo5