PMR3402 - Sistemas Embarcados - 2022

Prof. Dr. André Kubagawa Sato
Prof. Dr. André Cavalheiro
Guilherme Cortez Duran
Prof. Dr. Marcos de Sales Guerra Tsuzuki
Prof. Dr. Rogério Yugo Takimoto

31 de Maio de 2022

PMR-EPUSP

Comunicação Wi-Fi e Bluetooth

Comunicação - Sistemas Embarcados

Requisitos

- Meio
- Protocolo

Protocolos de comunicação são um conjunto de regras que permitem que dois sistemas se comuniquem entre si utilizando um determinado meio físico.

- Por Cabos
  - Disponibiliza mais confiável e mais segura.
  - Menos sujeita a interferência e mais rápida quando comparada a comunicação sem fio.
  - Limitação realacionada a mobilidade dos dispositivos, dificuldade de manutenção, escalabilidade.
  - Exemplos - RS-232, USB, Ethernet, etc..
- Sem Fio
  - Permite uma maior mobilidade e escalabilidade.
  - Mais sujeita a interferência eletromagnética, mais lenta e menos segura.
  - Exemplos - Wi-Fi, Bluetooth, ZigBee, etc

Protocolos de comunicação serial utilizando Arduíno.

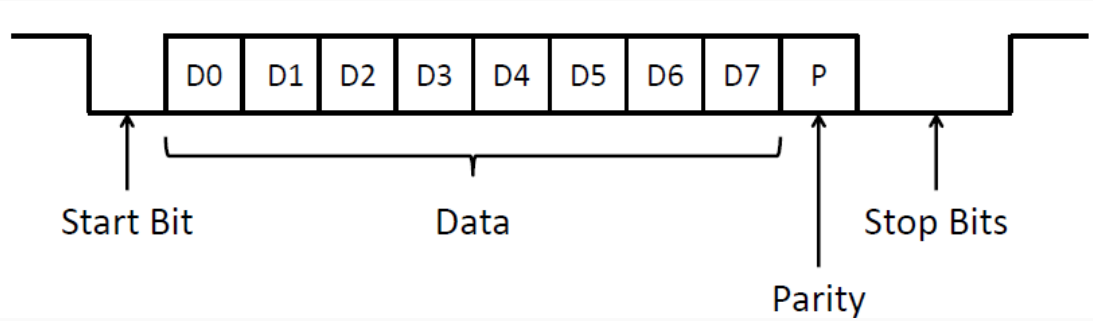
- UART (Universal Asynchronous Receiver Transmittter)
- SPI (Serial Peripheral Interface)
- I2C

UART (Universal Asynchronous Receiver Transmittter)

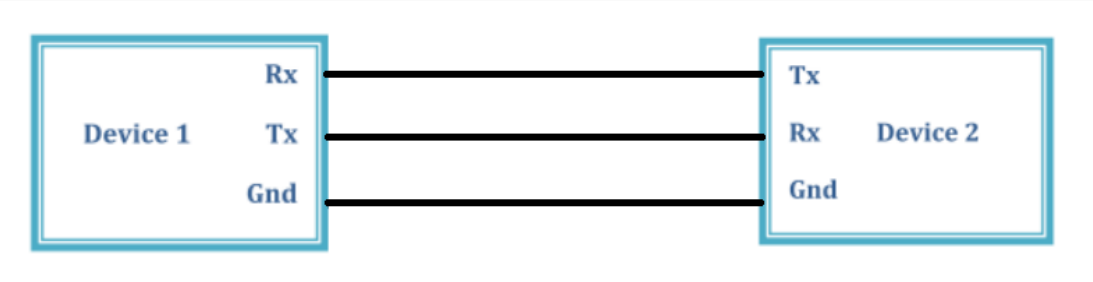
- Protocolo serial, assíncrono, half-duplex.
- Estabelece uma comunicação simples entre dois nós equivalentes.
- Qualquer nó pode iniciar a comunicação
- Como a comunicação é half duplex as duas linhas de comunicação são completamente independentes

UART (Universal Asynchronous Receiver Transmittter)

- Baud Rate
- Formato
  - Bit de Inicio
  - Bit de Dados
  - Bit de Paridade
  - Bit de Parada



Conexão UART e Codificação



- Serial.begin(baud)
- Serial.end()
- Serial.read()
- Serial.print(val), Serial.print(val, formato)
- Serial.available()
- Serial.find(string)

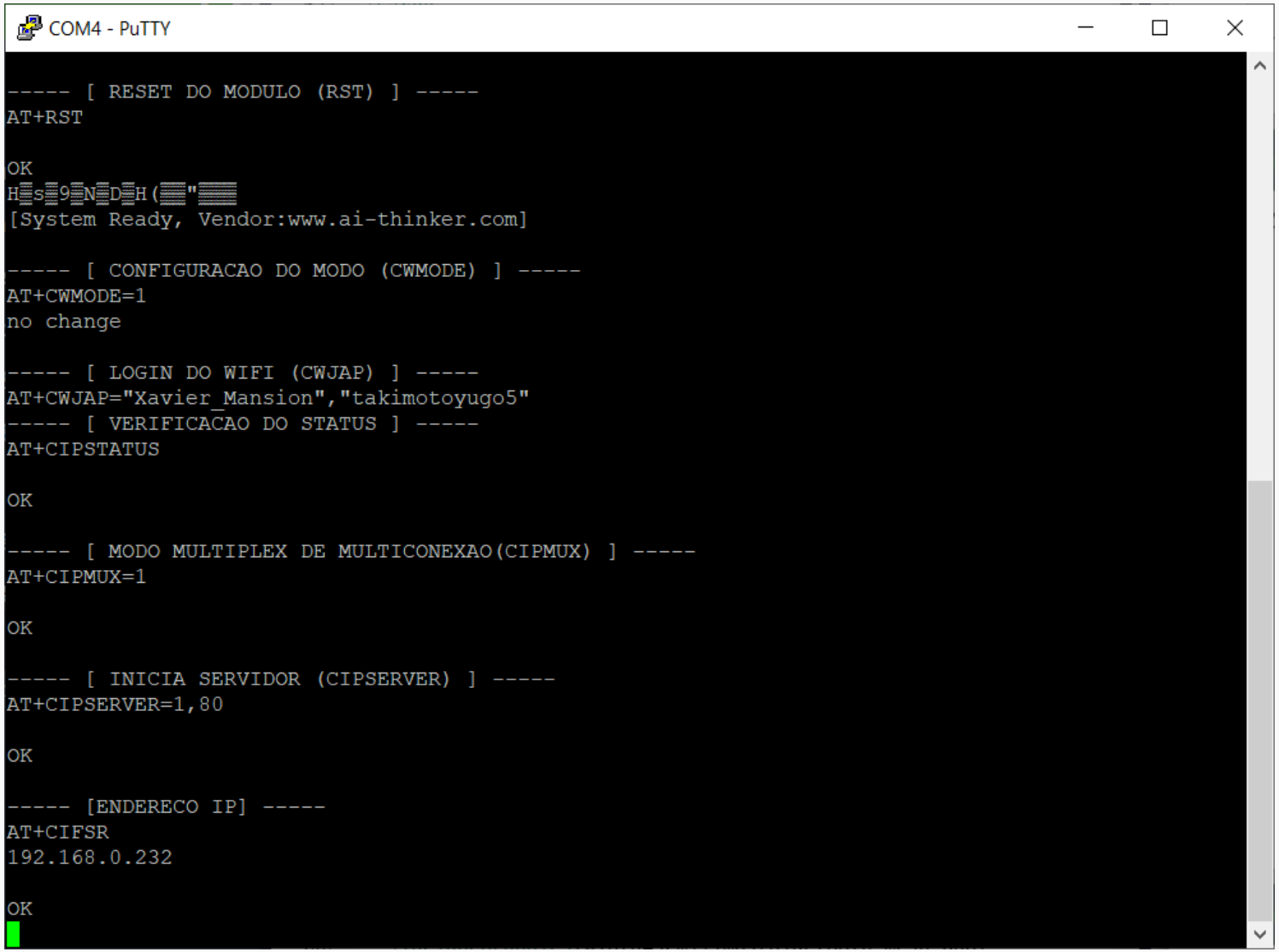
Comunicação - UART

```
#define esp8266 Serial1

void setup() {
  Serial.begin(9600); // monitor Serial 9600 baud.
  esp8266.begin(9600); // comunicacao ESP 9600 baud.

  String SSIDstring = ("\"SSID\"");
  String PASSstring = ("\"senha\"");

  Serial.println("\r\n----- [ RESET DO MODULO (RST) ] -----");
  sendData("AT+RST\r\n", 3000, true);

  Serial.println("\r\n----- [ CONFIGURACAO DO MODO (CWMODE) ] -----");
  // configuracao (1=Station, 2=AP, 3=Station+AP).
  sendData("AT+CWMODE=1\r\n", 3000, true);

  Serial.println("\r\n----- [ LOGIN DO WIFI (CWJAP) ] -----");
  sendData("AT+CWJAP=" + SSIDstring + "," + PASSstring, 12000, true);

  sendData("AT+CIPSTATUS\r\n", 3000, true); //status

  Serial.println("\r\n-- [ MODO MULTIPLEX DE MULTICONEXAO(CIPMUX) ] --");
  sendData("AT+CIPMUX=1\r\n", 3000, true);
```

```
  Serial.println("\r\n----- [ INICIA SERVIDOR (CIPSERVER) ] -----");
  sendData("AT+CIPSERVER=1,80\r\n", 3000, true);

  Serial.println("\r\n----- [ENDERECO IP] -----");
  sendData("AT+CIFSR\r\n", 3000, true);
}

void loop() {
  String webpage;

  if(esp8266.available()) { // recebimento da conexo
    if(esp8266.find("+IPD,")) { // Ler a requisio "IPD"
      delay(1000); // ler todo o dado da serial
      // obter o id da conexo
      int connectionId = esp8266.read() - 48;
      // subtrair 48 pois a funo decimal read() retorna um valor ASCII

     // ----- WEBPAGE -----
     webpage = "<h1>Ola!!</h1>";
     // ----- WEBPAGE -----

      sendData("AT+CIPSEND=" + String(connectionId) + ","
      + webpage.length() + "\r\n", 500, true);
      sendData(webpage, 1000, true); // Enviar a pgina.
```

```
    // Fechar a conexo.
    sendData("AT+CIPCLOSE=" + String(connectionId) + "\r\n", 1000, true);
   }
  }
}

String sendData(String command, const int timeout, boolean debug) {
  String response = "";
  esp8266.print(command); // envia os dados para o esp8266
  long int time = millis();
  while( (time+timeout) > millis()) {
   while(esp8266.available()) { // enquanto houver dados
    char c = esp8266.read(); // ler o proximo caracter
    response+=c;
   }
  }
  if(debug) { Serial.print(response); }
  return response;
}
```

UART (Universal Asynchronous Receiver Transmittter)



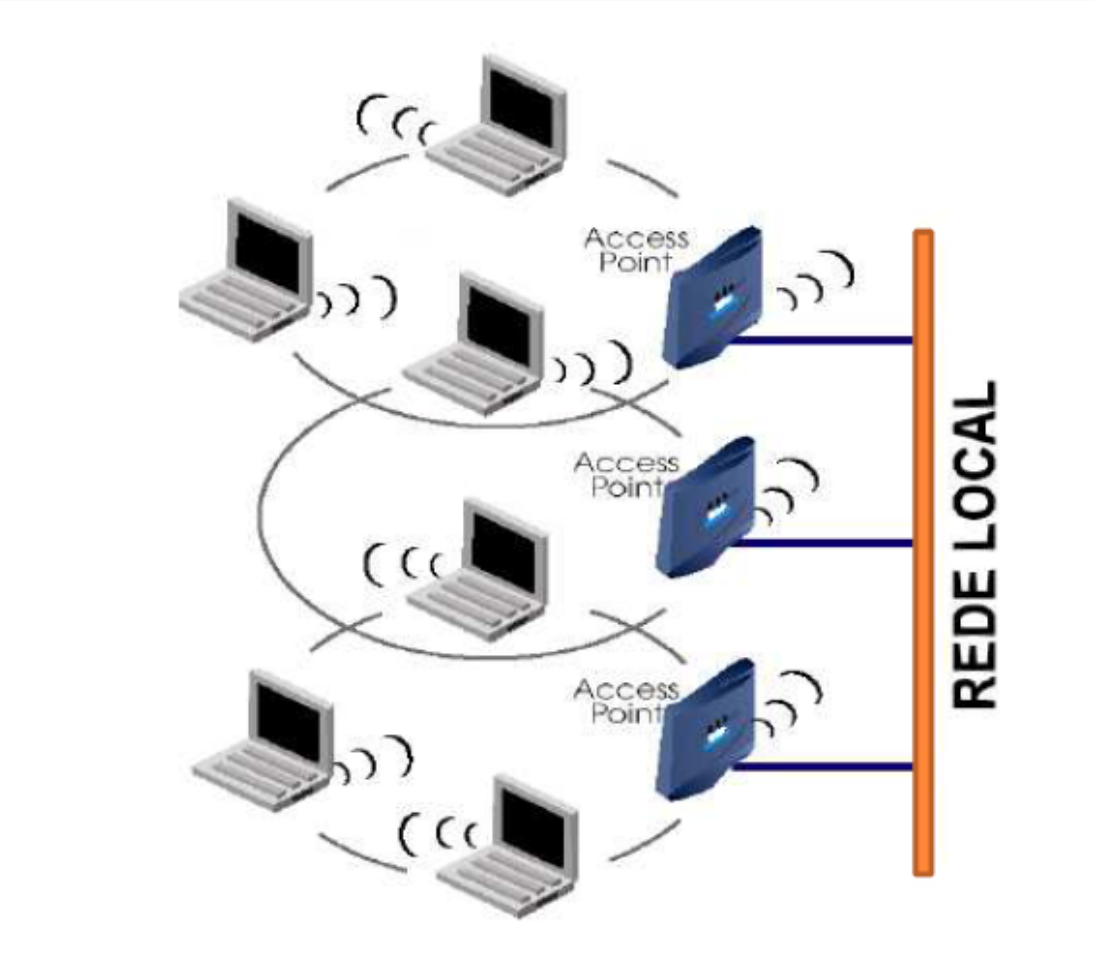
Principais requisitos para a escolha da comunicação sem fio

- Alcance
- Confiabilidade
- Segurança
- Custo
- Consumo de Energia
- Taxa de Transmissão
- Arquitetura da Rede

Wi-Fi

- Contração de Wireless Fidelity
- Nome comum para as diferentes versões do padrão IEEE 802.11 (a/b/g/n/i, etc) que descreve a tecnologia e os protocolos para obter uma rede local sem fio (WLAN)
- A identificação por letras define como as informações são codificadas, a utilização das frequências e as velocidades de transmissão possíveis

Arquitetura Wi-Fi (Estrela)



Padrões Difundidos e características

Padrão	Lançamento	Velocidade	Alcance	Freq de op
802.11	1997	Até 2 Mbps	20 m	2.4 GHz
802.11b	1999	Até 11 Mbps	35 m	2.4 GHz
802.11a	1999	Até 54 Mbps	35 m	5 GHz
802.11g	2003	Até 54 Mbps	70 m	2.4 GHz
802.11n	2009	Até 600 Mbps	70 m	2.4 / 5 GHz
802.11ac	2013	Até 1.1 Gbps	70 m	5 GHz

Comunicação - Wi-Fi

Protocolos que podem ser utilizados com o Arduino e um módulo WI-FI

- HTTP
- FTP
- Telnet
- SMTP
- Modbus
- ...

Comunicação - Wi-Fi

```
#define esp8266 Serial1
int statusLed;

void setup() {
  Serial.begin(9600); // monitor Serial 9600 baud.
  esp8266.begin(9600); // comunicacao ESP 9600 baud.
  pinMode (LED_BUILTIN, OUTPUT);
  statusLed=0;
  digitalWrite(LED_BUILTIN, LOW);
  ......
}

void loop() {
  String webpage;

  if(esp8266.available()) { // recebimento da conexo
    if(esp8266.find("+IPD,")) { // Ler a requisio "IPD"
      delay(1000); // ler todo o dado da serial
      // obter o id da conexo
      int connectionId = esp8266.read() - 48;
      // subtrair 48 pois a funo decimal read() retorna um valor ASCII
      esp8266.find("led="); //procura pela palavra led no sinal enviado pelo esp82
      int led = (esp8266.read()-48);
```

```
 // ----- WEBPAGE -----
webpage = "<html><head><title>LED</title></head>";
webpage = webpage+"<body><p style='line-height:2'><font>Modulo
WiFi ESP8266</font></p>";
webpage = webpage+"<font>ESTADO ATUAL DO LED</font>";

if (statusLed == 1){
 webpage= webpage+"<p style='line-height:0'><font
 color='green'>LIGADO</font></p>";
 webpage= webpage+"<a href=\"/led=0\">APAGAR</a>";
}else{
  if (statusLed == 0){
    webpage= webpage+"<p style='line-height:0'><font
    color='red'>DESLIGADO</font></p>";
    webpage= webpage+"<a href=\"/led=1\">ACENDER</a>";
  }
}
webpage= webpage+"<hr />";
webpage= webpage+"</body>";
webpage= webpage+"</html>";
// ----- WEBPAGE -----
```

```
    if(led==1){
         digitalWrite(LED_BUILTIN, HIGH);
         statusLed = 1;
     }
     else{
       if (led==0) {
        digitalWrite(LED_BUILTIN, LOW);
        statusLed = 0;
        }
     }

     sendData("AT+CIPSEND=" + String(connectionId) + ","
     + webpage.length() + "\r\n", 500, true);
     sendData(webpage, 1000, true); // Enviar a pgina.
     // Fechar a conexo.
     sendData("AT+CIPCLOSE=" + String(connectionId) + "\r\n", 1000, true);
   }
  }
}
```

```
String sendData(String command, const int timeout, boolean debug) {
  String response = "";
  esp8266.print(command); // envia os dados para o esp8266
  long int time = millis();
  while( (time+timeout) > millis()) {
    while(esp8266.available()) { // enquanto houver dados
      char c = esp8266.read(); // ler o proximo caracter
      response+=c;
    }
  }
  if(debug) { Serial.print(response); }
  return response;
}
```

Comunicação - Wi-Fi (Escrevendo dados na Planilha do Google)

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
WiFiClientSecure client;//Cliente seguro (para ter acesso ao HTTPS)
String textFix = "GET path";
const char* ssid = "SSID"; // SSID
const char* password = "password"; // senha
const char* server = "docs.google.com"; // Server URL


void setup()
{
   Serial.begin(9600);//Inicia a comunicacao serial
   Serial.println("Inicio");

   WiFi.mode(WIFI_STA);//Habilita o modo estaao
   delay(100);

   Serial.print("Conectando a SSID: ");
   Serial.println(ssid);
   WiFi.begin(ssid, password);
```

Comunicação - Wi-Fi (Escrevendo dados na Planilha do Google)

```
  //tentativa de conectar na rede
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    // esperar 1 segundos
    delay(1000);
  }
  Serial.println("");
  Serial.print("Connectado a ");
  Serial.println(ssid);

  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  client.setInsecure();
}
void loop()
{
  //conexao ao servidor do Google docs na porta 443 (HTTPS)
  if (client.connect(server, 443))
  {

    String toSend = textFix;//Atribuimos a String auxiliar
    //na nova String que sera enviada
    toSend += random(0, 501);//Adicionamos um valor aleatorio
```

Comunicação - Wi-Fi (Escrevendo dados na Planilha do Google)

```
void loop()
{
      toSend += "&submit=Submit HTTP/1.1";//Completamos o metodo
      //GET para nosso formulario.
      client.println(toSend);//Enviamos o GET ao servidor-
      client.println("Host: docs.google.com");//-
      client.println();//-
      client.stop();//Encerramos a conexao com o servidor
      Serial.println("Dados enviados.");//Mostra no monitor que
      //foi enviado
   }
   else
   {
      Serial.println("Erro ao se conectar");
   }
   delay(5000);
}
```

Comunicação - Wi-Fi (Lendo dados da Planilha do Google)

```
#include <WiFi.h>
#include <WiFiClientSecure.h> // biblioteca ESP32

WiFiClientSecure cl;//cliente seguro para acesso ao HTTPS
String textFix = "GET /path";
String key = "?key=AIzaSyDDIJhiqJNYcHV5bccaSDUGgkuSbMF-PM0";//Chave API
const char* ssid = "SSID";
const char* password = "senha";
const char* server = "sheets.googleapis.com"; // Server URL

//Funcoes para acesso ao arquivo

void setup()
{
  ...
}
```

Comunicação - Wi-Fi (Lendo dados da Planilha do Google)

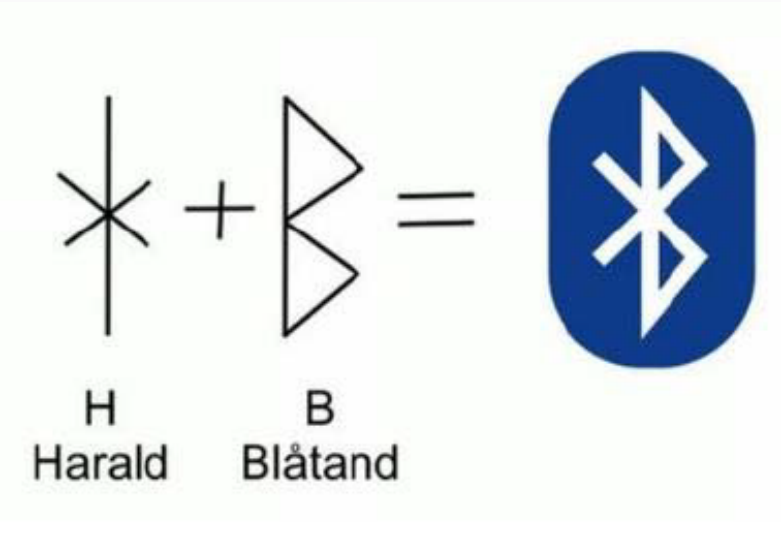
```
void loop()
{
    if (cl.connect(server, 443))//Google APIs (porta 443 - HTTPS)
    {
        String toSend = textFix;
        toSend += "A1:C4";//celulas para leitura
        toSend += key;//chave API
        toSend += " HTTP/1.1";
        cl.println(toSend);//Enviamos do GET ao servidor-
        cl.println("Host: sheets.googleapis.com");
        cl.println();
        Serial.println("Dado recebido:\n");
        String rcv_message=cl.readString();

        cl.stop();//Fecha a conexao
    }
    else
    {
        Serial.println("Erro ao se conectar");
    }

    delay(5000);
}
```

Referência

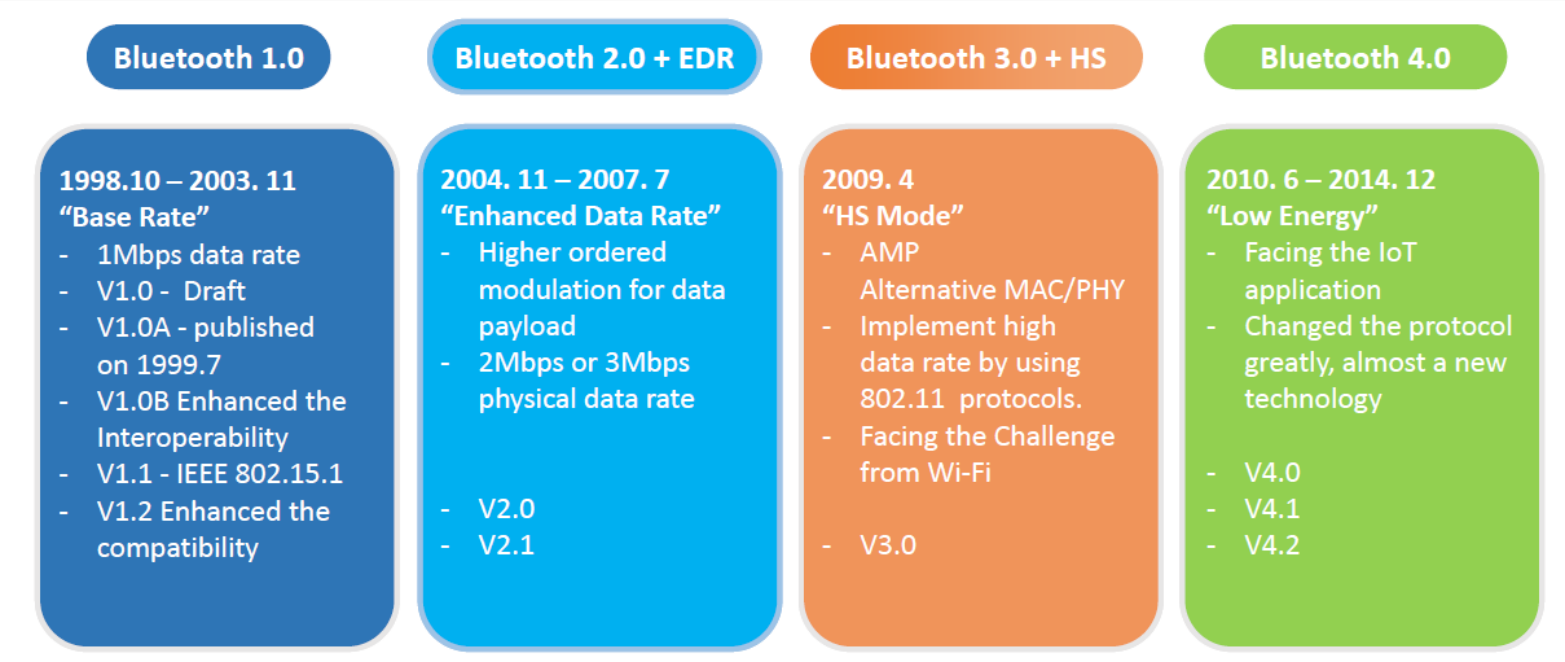
- Sistema de Aquisição e Monitoramento de Dados de Temperatura e Umidade Baseado em Plataforma IoT e Arduino (Induscon 2021)
  - Plataforma IoT - Thingspeak

Bluetooth

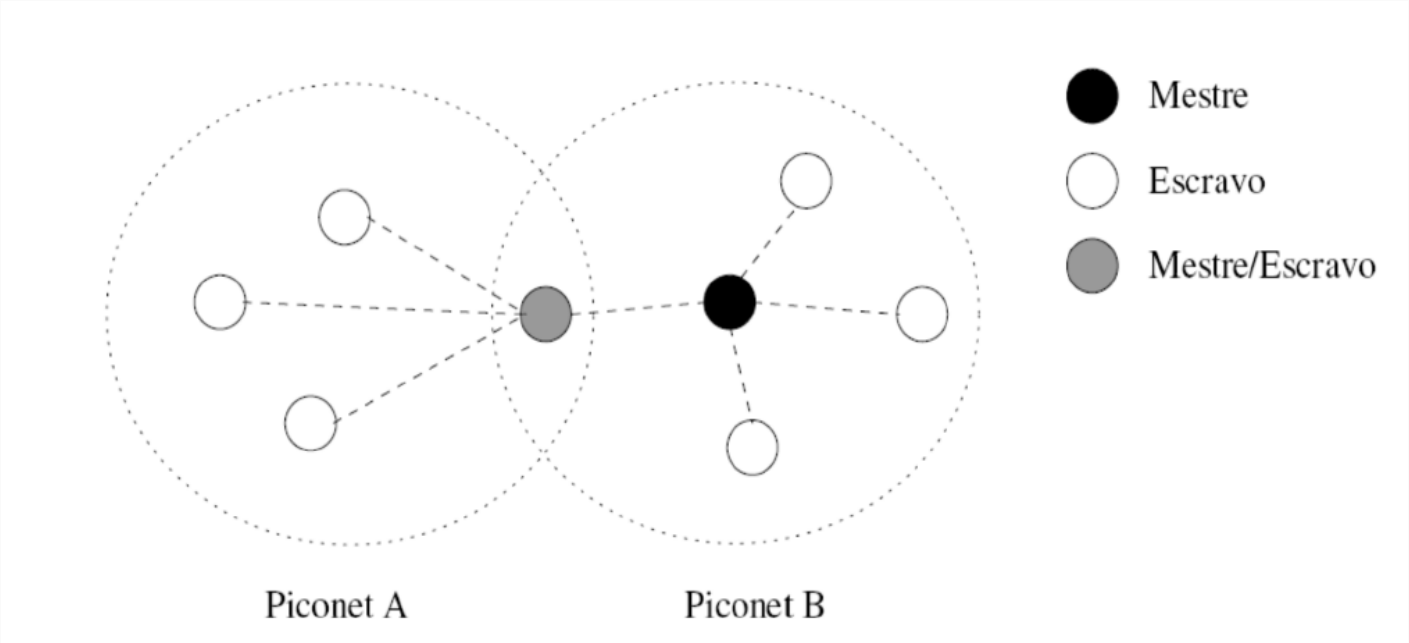
- Um dos padrões de comunicação sem fio de curto alcance mais popular
- Conhecido como IEEE 802.15.1
- Concebido em 1994 na Ericsson Mobile (Suécia) como alternativa sem fio para o RS-232
- Nome inspirado em Harald "Bluetooth"Gormsson (935-985/6) conhecido como Harald Blåtand Gormsen



Evolução do Bluetooth

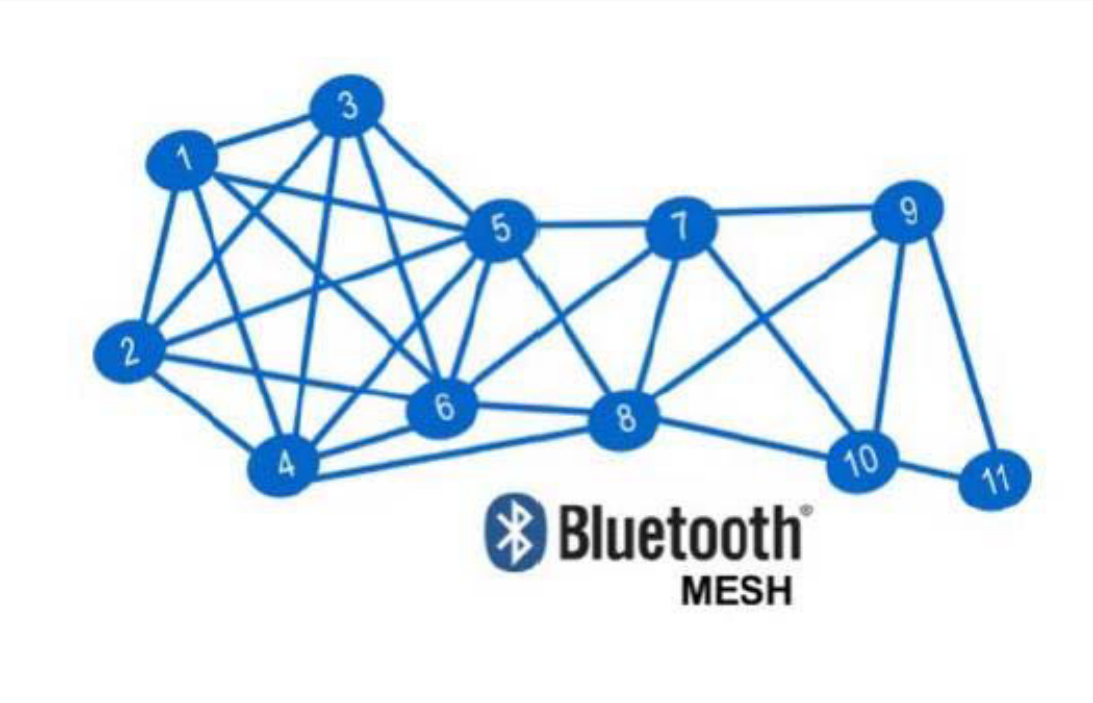


Arquitetura Bluetooth (Ad hoc - Scatternet)

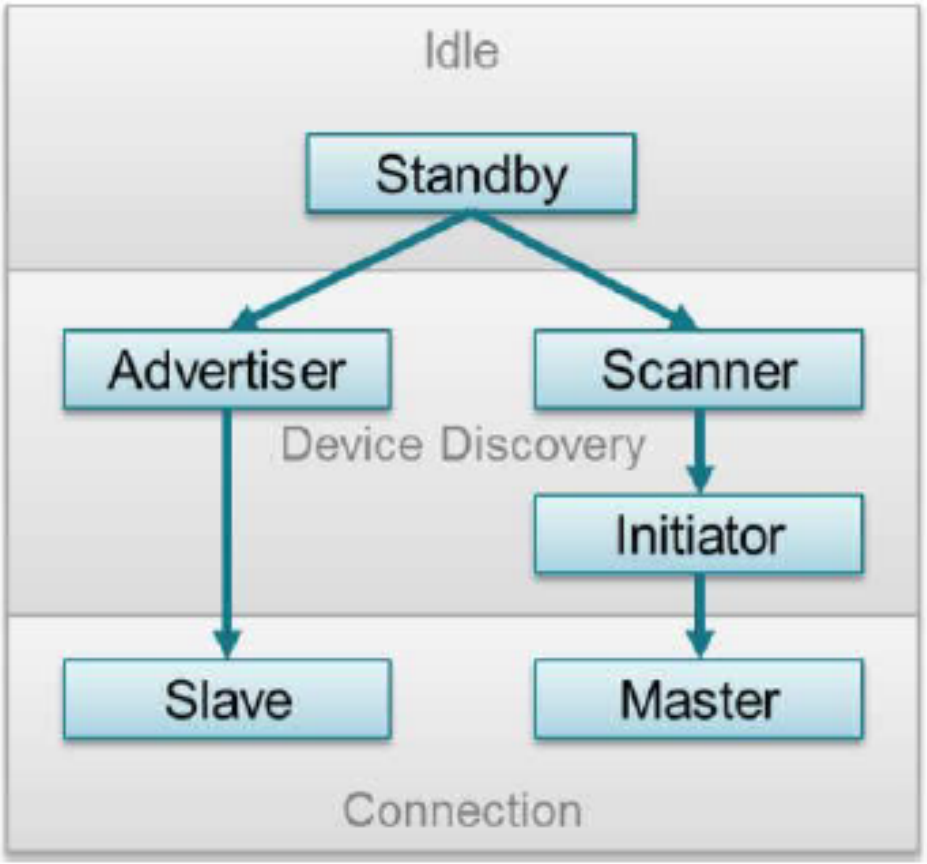


Principais Características

- Bluetooth 4.2
  - Frequência - 2.4 GHz
  - Alcance - 30m
  - Velocidade - 1Mbps
- Bluetooth 5.0
  - Consumo de Energia
  - Velocidade
  - Expansão na conexão - Mesh (IoT)

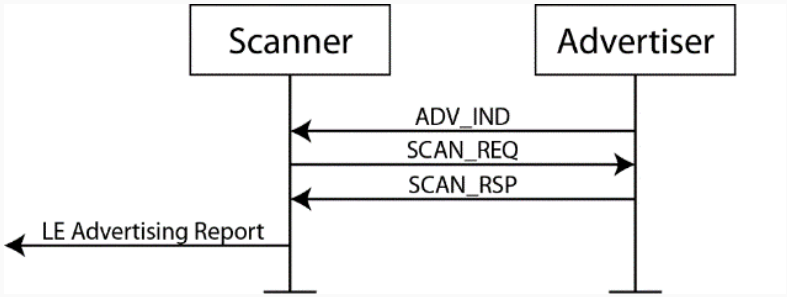


Scaneamento e Conexão Bluetooth

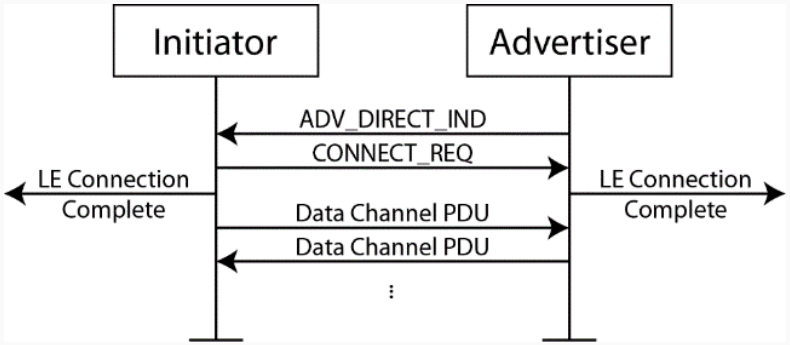


Mensagens Bluetooth (Tudo começa com o anúncio)

- Scan Passivo
- Scan Ativo
  - SCAN_REQ (Mais informação)
  - SCAN_RSP



- Conexão
  - CONNECT_REQ (Ok, vamos conectar)



Comunicação - Bluetooth

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;

void callback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param) {

  if (event == ESP_SPP_SRV_OPEN_EVT) {

    Serial.println("Client Connected has address:");

    for (int i = 0; i < 6; i++) {

      Serial.printf("%02X", param->srv_open.rem_bda[i]);
      if (i < 5) {
        Serial.print(":");
      }
    }
    Serial.println("");
  }
  if(event == ESP_SPP_CLOSE_EVT ){
    Serial.println("Client disconnected");
  }
}
```

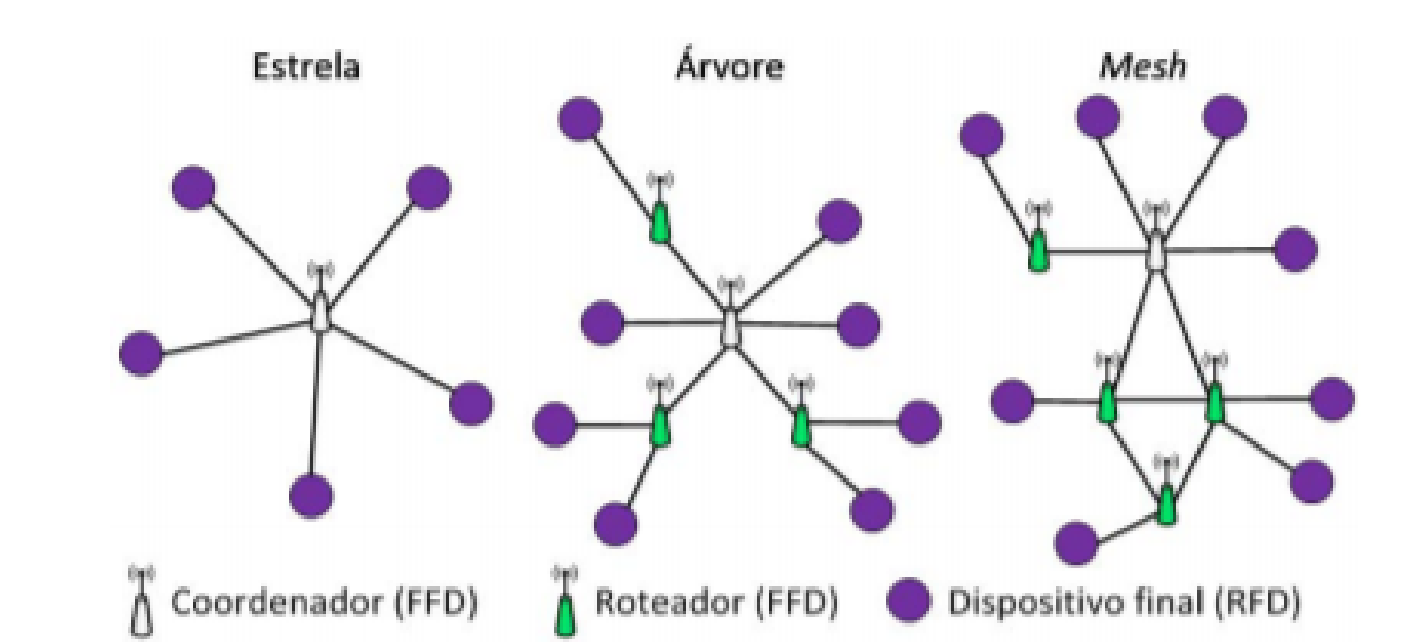
Comunicação - Bluetooth

```
void setup() {
  Serial.begin(9600);

  SerialBT.register_callback(callback);

  if (!SerialBT.begin("BTArduino")) {
    Serial.println("An error occurred initializing Bluetooth");
  } else {
    Serial.println("Bluetooth initialized");
  }
}

void loop() {
  if (Serial.available())
  {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available())
  {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
```

ZigBee

- Construído sobre o padrão IEEE 802.15.4
- Desenvolvimento com ênfase em baixo consumo de energia
- Voltado para aplicações que não necessitam de alta banda, mas sim de baixa latência e baixo nível de consumo de energia

Arquitetura ZigBee (Ad Hoc - Mesh)

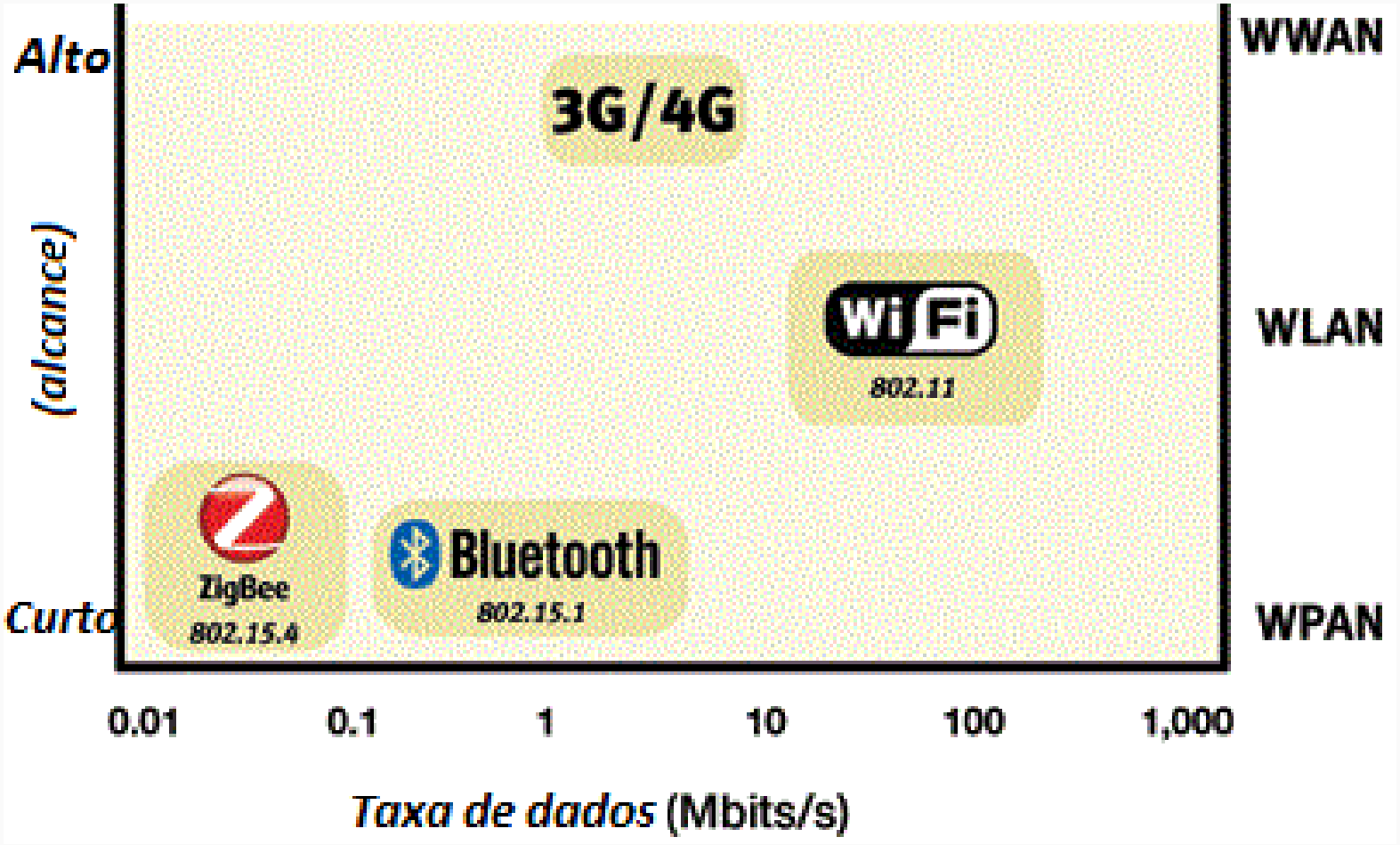


Principais Características

- Frequência - 2.4 GHz
- Alcance (Ambiente Externo) - 100m
- Velocidade - 250Kbps

Utilização com Arduíno

Comparação entre os principais padrões de rede sem fio



Comparação

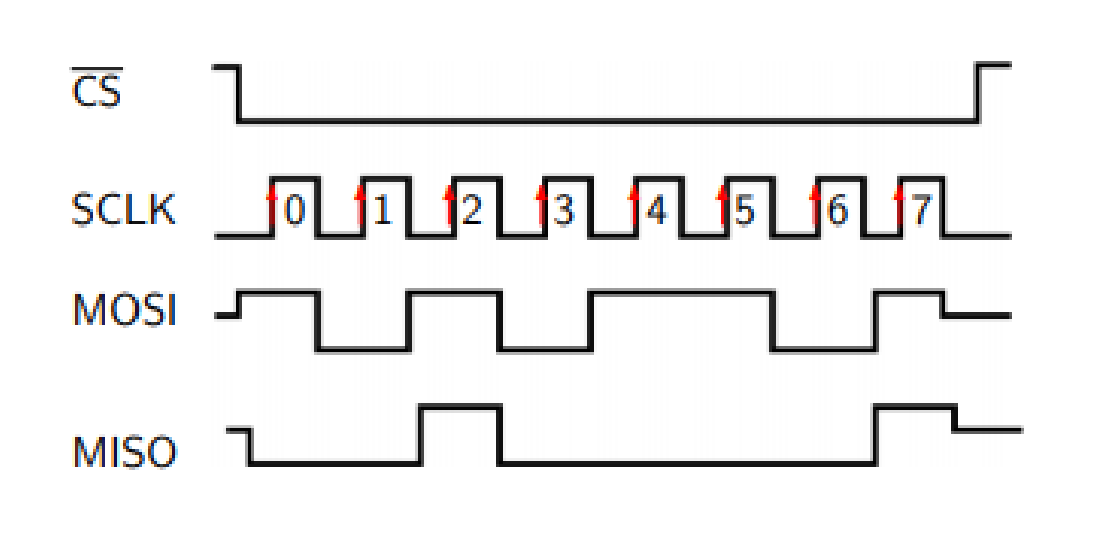
	Wi-Fi	Bluetooth (v 4.2)	Zigbee
Lançamento	1997	2015	2003
Padrão	IEEE 802.11.1g	IEEE 802.15.1	IEEE 802.15.4
Frequência	2.4GHz	2.4GHz	2.4GHz
Alcance	100m	30m	10-100m
Velocidade	54Mbps	1Mbps	250kbps
Arquitetura	Estrela	Scatternet	Mesh
Consumo	Alto	Baixo	Baixo

The End!

SPI (Serial Peripheral Interface)

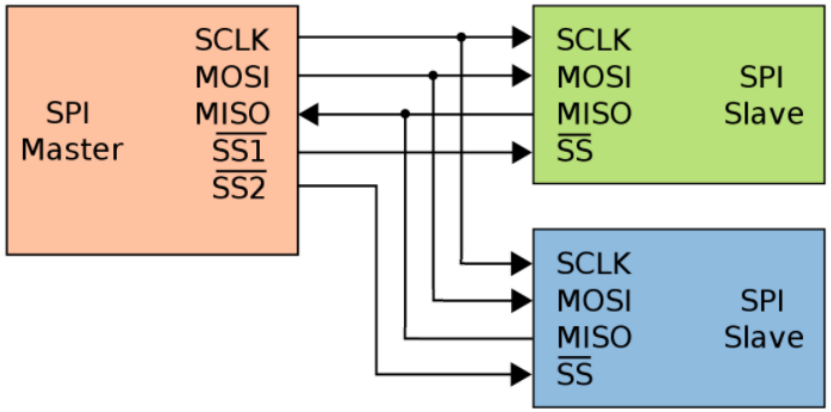
- Comunicação Mestre/Escravo
- Utiliza 3 sinais (e o terra)
  - MISO, MOSI, SCLK
  - e a seleção do chip (CS) para cada dispositivo escravo
- Síncrono, o mestre controla o clock

SPI (Serial Peripheral Interface)



A transferência pode acontecer em ambas as direções simultaneamente (full-duplex)

Conexão SPI e Codificação



Placas	MOSI	MISSO	SCK
Uno	11 ou ICSP-4	12 ou ICSP-1	13 ou ICSP-3
Mega2560	51 ou ICSP-4	50 ou ICSP-1	52 ou ICSP-3

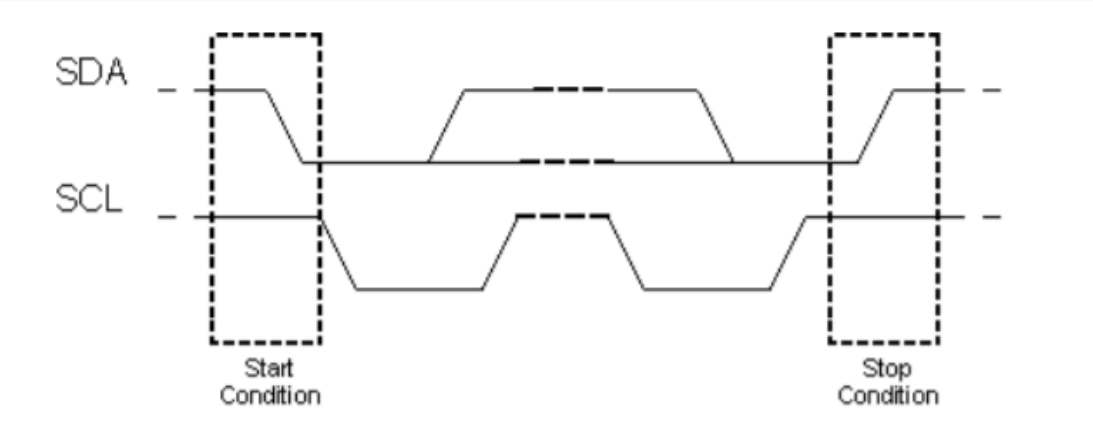
- SPI.begin()
- SPI.transfer(val)
- SPI.end()

```
#include <SPI.h>

void setup() {
  pinMode(10, OUTPUT); // configura pino SC como output
  SPI.begin();    // inicializa o SPI
  Serial.begin(9600);
}

void loop() {

  digitalWrite(10, LOW);    // CS em LOW

  for(byte wiper_value = 0; wiper_value <= 128; wiper_value++) {
    // envia um comando de escrita para o endereco 0x00
    SPI.transfer(0x00);
    SPI.transfer(wiper_value); // escreve o valor
    delay(1000);
  }

  digitalWrite(10, HIGH);   // CS em HIGH
}
```

I2C

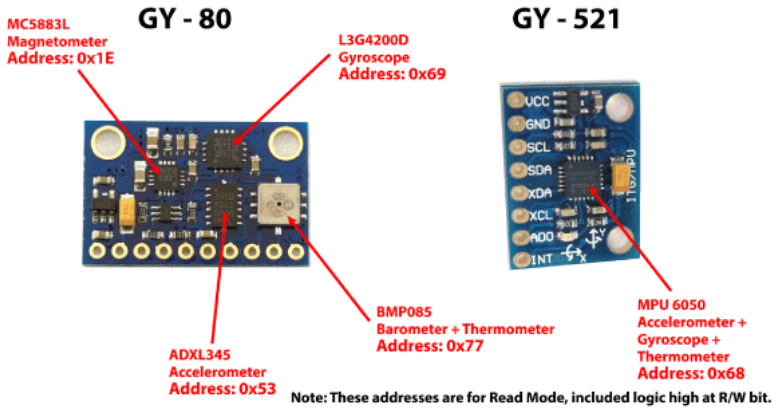
- Comunicação Mestre/Escravo
- Utiliza 2 sinais (e o terra)
  - SDA
  - SCL
- Síncrono, o mestre controla o clock

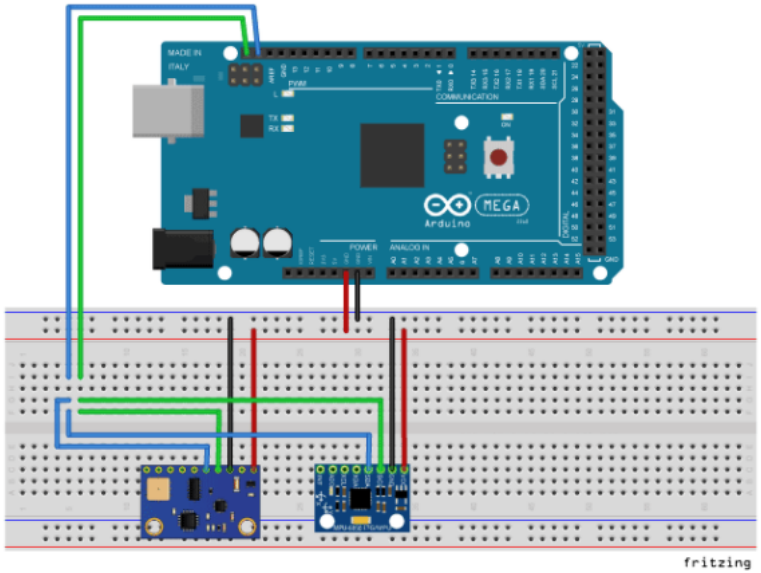
I2C



A transferência não ocorre simultaneamente (half-duplex)

Conexão I2C e Codificação





- Wire.begin()
- Wire.beginTransmission(endereco);
- Wire.requestFrom(endereco, quantidade, stop)
- Wire.write(registrador)
- Wire.read()

```
//Carrega a biblioteca Wire
#include<Wire.h>

//Endereco I2C do MPU6050
const int MPU=0x68;
//Variaveis para armazenar valores dos sensores
int AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
void setup()
{
  Serial.begin(9600);
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);

  //Inicializa o MPU-6050
  Wire.write(0);
  Wire.endTransmission(true);
}
```

```
void loop()
{
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // Solicitacao iniciando em 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  //Solicita os dados do sensor
  Wire.requestFrom(MPU,14,true);
  //Armazena o valor dos sensores nas variaveis correspondentes
  //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcX=Wire.read()<<8|Wire.read();
  //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcY=Wire.read()<<8|Wire.read();
  //0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  AcZ=Wire.read()<<8|Wire.read();
  //0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  Tmp=Wire.read()<<8|Wire.read();
  //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyX=Wire.read()<<8|Wire.read();
  //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyY=Wire.read()<<8|Wire.read();
  //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  GyZ=Wire.read()<<8|Wire.read();
```

```
  //Envia valor X do acelerometro para a serial
  Serial.print("AcX = "); Serial.print(AcX);
  //Envia valor Y do acelerometro para a serial
  Serial.print(" | AcY = "); Serial.print(AcY);
  //Envia valor Z do acelerometro para a serial
  Serial.print(" | AcZ = "); Serial.print(AcZ);
  //Envia valor da temperatura para a serial
  //Calcula a temperatura em graus Celsius
  Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53);
  //Envia valor X do giroscopio para a serial
  Serial.print(" | GyX = "); Serial.print(GyX);
  //Envia valor Y do giroscopio para a serial
  Serial.print(" | GyY = "); Serial.print(GyY);
  //Envia valor Z do giroscopio para a serial
  Serial.print(" | GyZ = "); Serial.println(GyZ);

  //Aguarda 300 ms e reinicia o processo
  delay(300);
}
```