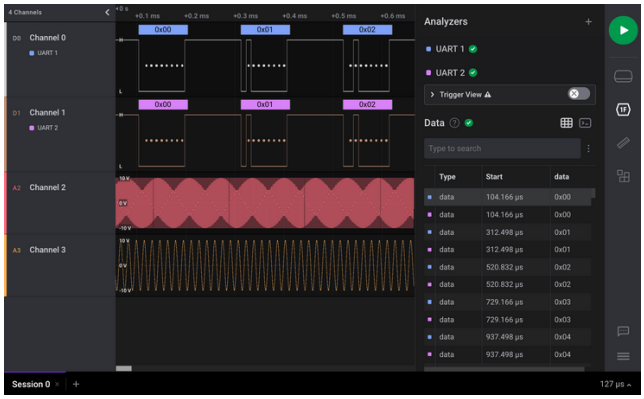
Embedded
Security

b10s HARDWARE

DAY 1

- Installing required Software's.
- FCC id , Basic overview of Communication protocols.
- Handson : UART and Logic analyzer


ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED,
AND SUPPORTED BY ARDUINO.CC AND
THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS
OF ARDUINO.CC on arduino.cc/credits

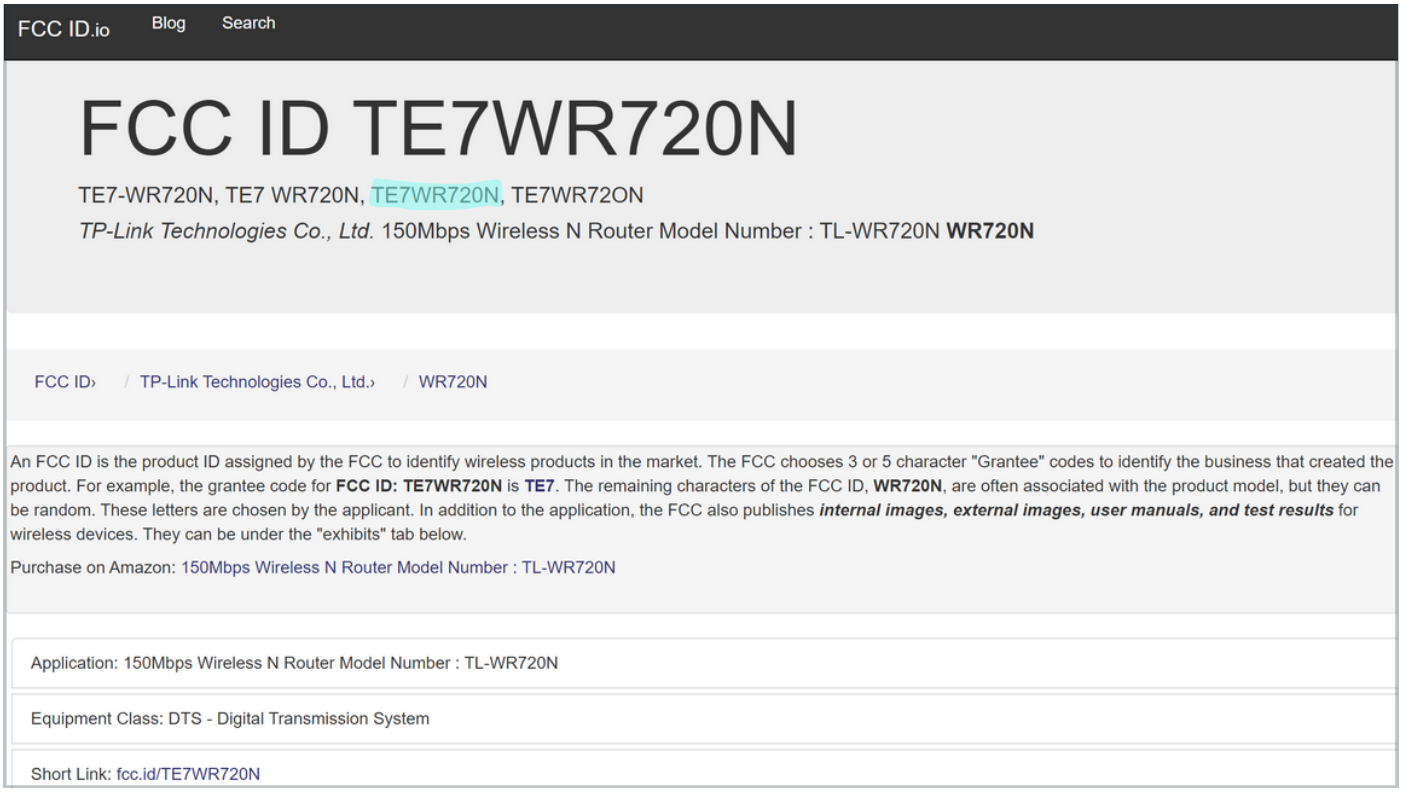

Channel 0
Channel 1
Channel 2
Channel 3
Analyzers
UART 1
UART 2
Trigger View
Data
Session 0

FCC ID

Federal Communications Commission

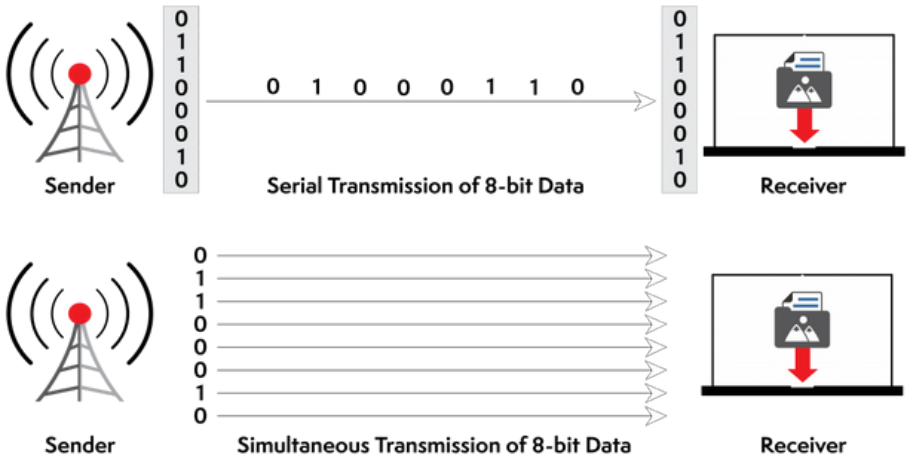
- To get some details about the board we are going to work on.

is a unique identifier assigned to electronic devices that indicates compliance with FCC regulations. It helps regulatory authorities and consumers easily access information about the device's specifications and ensure it meets necessary standards for wireless communication.


FCC ID.io
Blog
Search
FCC ID TE7WR720N
TE7-WR720N, TE7 WR720N, TE7WR720N, TE7WR72ON
TP-Link Technologies Co., Ltd. 150Mbps Wireless N Router Model Number : TL-WR720N WR720N
FCC ID› / TP-Link Technologies Co., Ltd.› / WR720N
An FCC ID is the product ID assigned by the FCC to identify wireless products in the market. The FCC chooses 3 or 5 character "Grantee" codes to identify the business that created the product. For example, the grantee code for FCC ID: TE7WR720N is TE7. The remaining characters of the FCC ID, WR720N, are often associated with the product model, but they can be random. These letters are chosen by the applicant. In addition to the application, the FCC also publishes internal images, external images, user manuals, and test results for wireless devices. They can be under the "exhibits" tab below.
Purchase on Amazon: 150Mbps Wireless N Router Model Number : TL-WR720N
Application: 150Mbps Wireless N Router Model Number : TL-WR720N
Equipment Class: DTS - Digital Transmission System
Short Link: fcc.id/TE7WR720N

What is Communication Protocol?

- Serial and Parallel

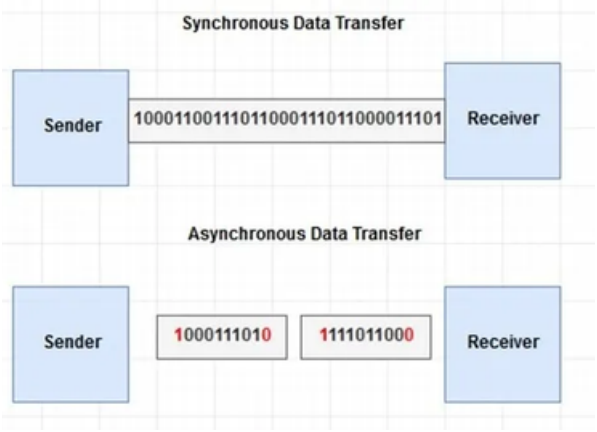


1. Serial and parallel.
->Serial data transmission sends data bits one after another over a single channel.
->Parallel data transmission sends multiple data bits at the same time over multiple channels.

serial ->
1.data bits organized in well.
2.sent if the previous data bit has already been received.

Types of Serial Communication Protocols

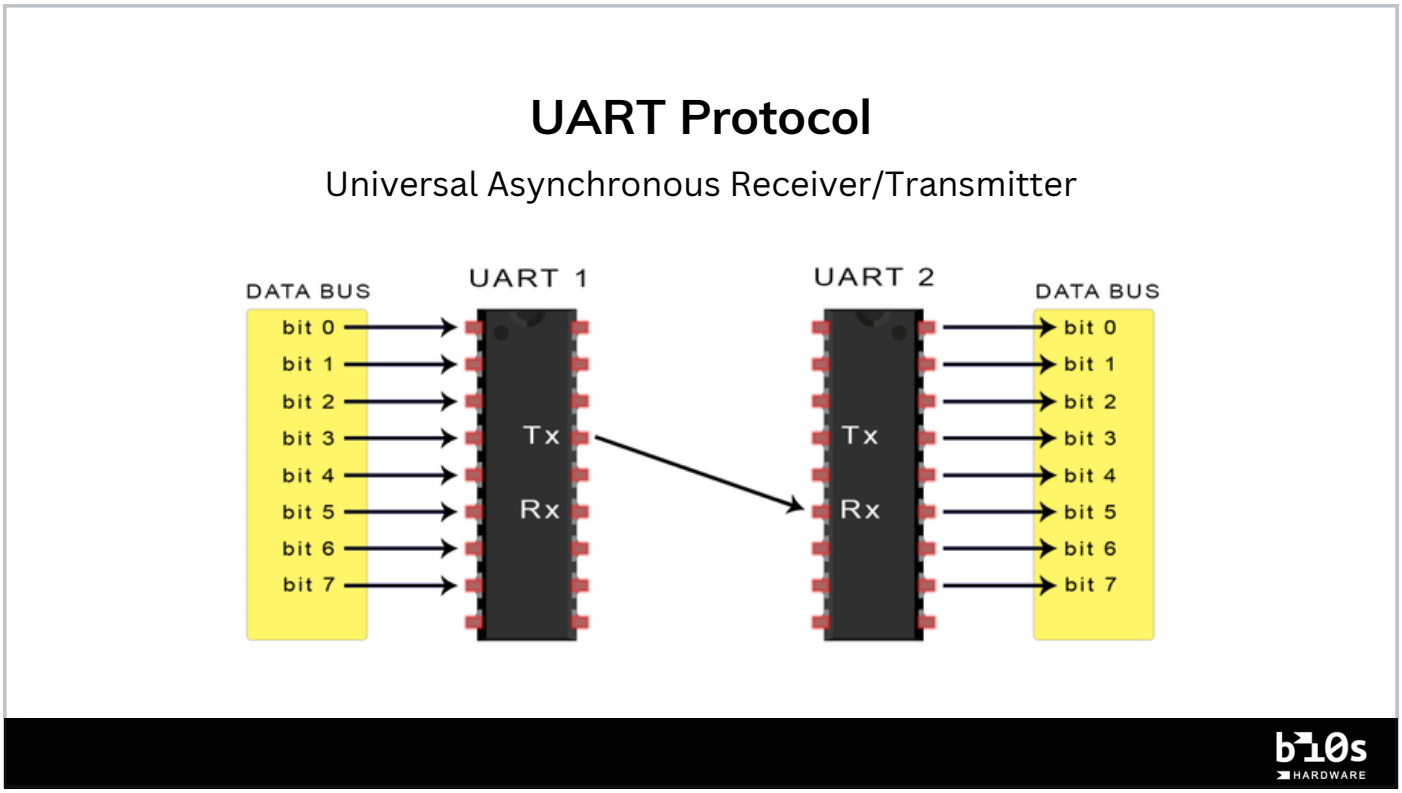
- Synchronous
- Asynchronous



b10s HARDWARE

Asynchronous
->bits sent at any time
->Start bit and Stop bit used btw bytes to synchronize.
->since time not constant btw sending time gaps used.

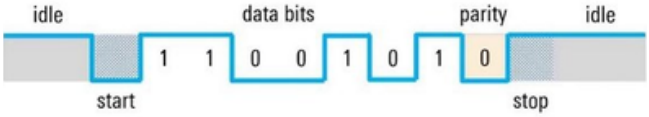
advantage.
-> no synchronization required so any time data can be sent.
disadvantage
->Data transmission is slower(due to gaps).



The UART consists of two main components: a transmitter and a receiver. The transmitter converts parallel data from the device into a serial data stream, and the receiver converts the serial data stream back into parallel data for use by the device.

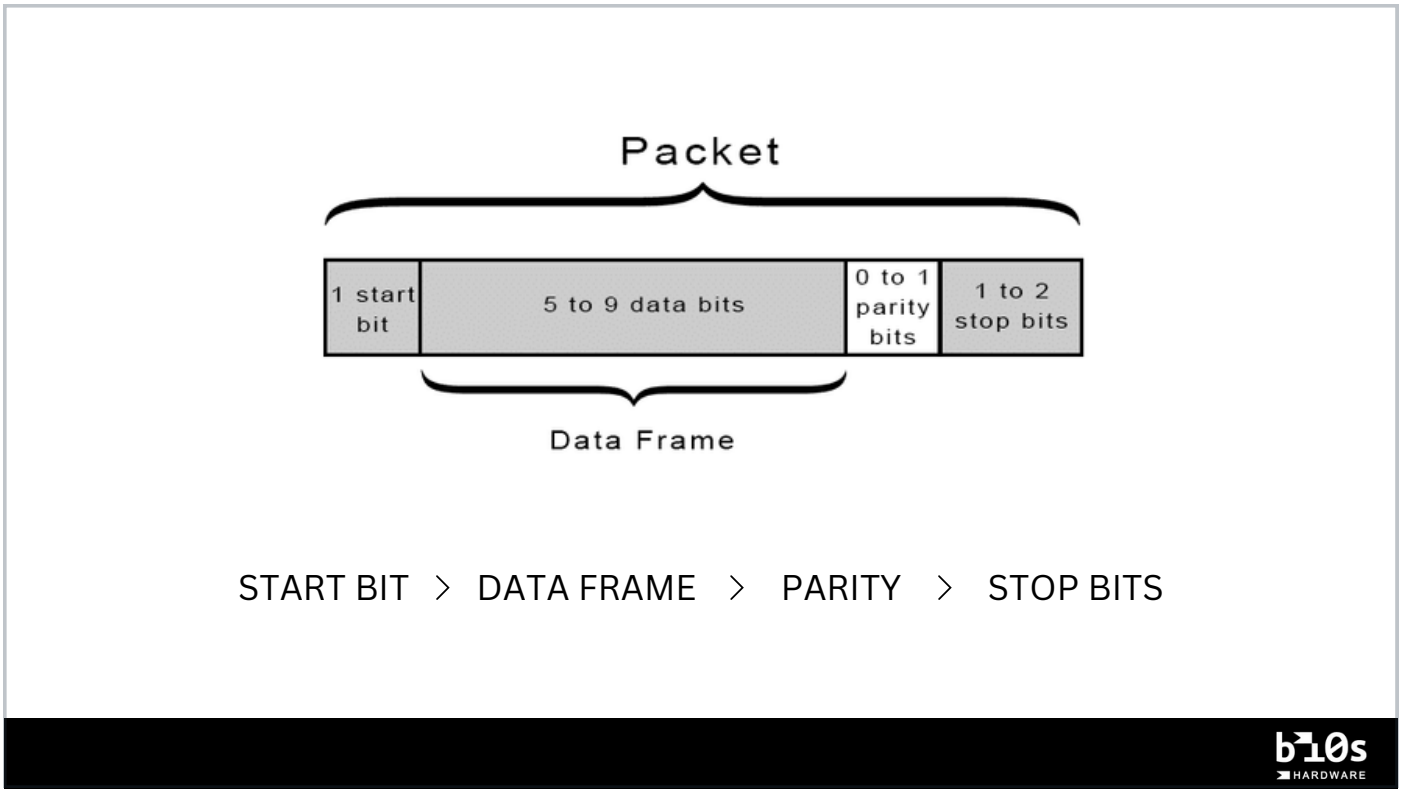
UART Frame Format

- UART frames consist of:
  - Start/Stop bits
  - Data bits
  - Parity but (opt)
- High Voltage (1), Low voltage (0)



b10s HARDWARE

Asynchronous communication in UART means that data is transmitted without a synchronized clock between sender and receiver. It utilizes start and stop bits to frame data, allowing devices to communicate at different speeds without a shared clock.



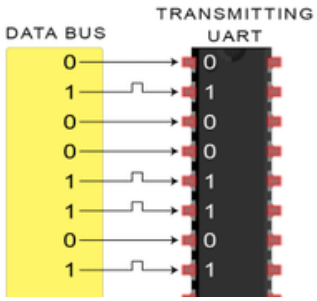
The UART data transmission line is normally held at a high voltage level when it's not transmitting data.

The data frame contains the actual data being transferred.

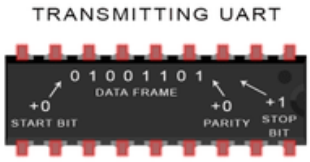
Parity describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission.

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit durations.

STEP 1 The transmitting UART receives data in parallel from the data bus:



STEP 2 The transmitting UART adds the start bit, parity bit, and the stop bit(s) to the data frame:



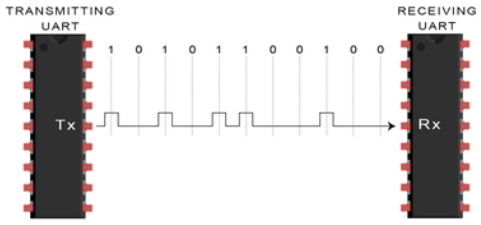
The UART data transmission line is normally held at a high voltage level when it's not transmitting data.

The data frame contains the actual data being transferred.

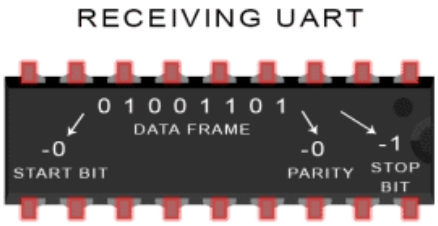
Parity describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission.

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit durations.

STEP 3 The entire packet is sent serially from the transmitting UART to the receiving UART. The receiving UART samples the data line at the pre-configured baud rate:

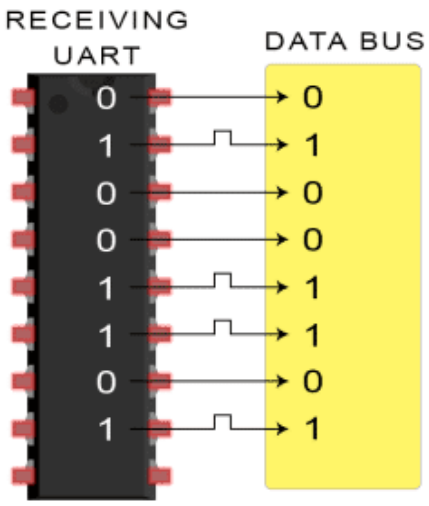


STEP 4 The receiving UART discards the start bit, parity bit, and stop bit from the data frame:



The way i2c works is , there will be a fixed clock pulse, that will set the communication speed for both the master and the slave devices. this need to be the same, because the slave/master device must know when to read the next bit. then the data is transmitted through the sda pin as serial data.

STEP 5 The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end:



The way i2c works is , there will be a fixed clock pulse, that will set the communication speed for both the master and the slave devices. this need to be the same, because the slave/master device must know when to read the next bit. then the data is transmitted through the sda pin as serial data.

ADVANTAGES

- Only uses two wires
- No clock signal is necessary
- Has a parity bit to allow for error checking
- The structure of the data packet can be changed as long as both sides are set up for it
- Well documented and widely used method

DISADVANTAGES

- The size of the data frame is limited to a maximum of 9 bits
- Doesn't support multiple slave or multiple master systems
- The baud rates of each UART must be within 10% of each other

b10s HARDWARE

The way i2c works is , there will be a fixed clock pulse, that will set the communication speed for both the master and the slave devices. this need to be the same, because the slave/master device must know when to read the next bit. then the data is transmitted through the sda pin as serial data.

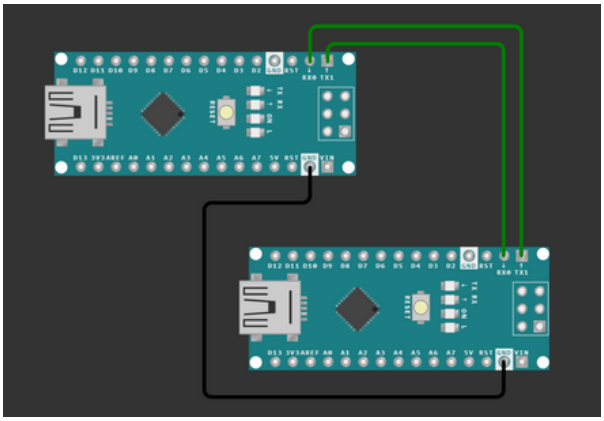
Logic Analyzer

b10s HARDWARE

HANDS-ON

b10s HARDWARE

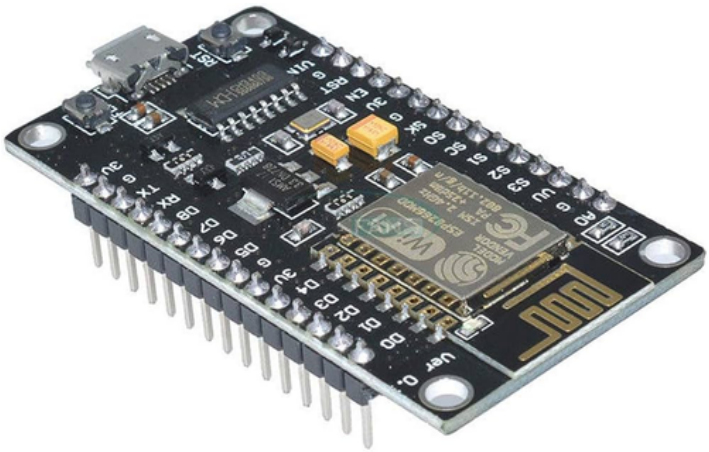
Arduino UART Connections



SAMPLE CODE

DAY 2

ESP8266



b10s HARDWARE

The ESP8266 is a popular, low-cost Wi-Fi module that enables microcontrollers to connect to a Wi-Fi network and communicate with other devices over the internet.

It was developed by Espressif Systems, a Chinese company that specializes in Internet of Things (IoT) solutions.

How to program ESP8266 ?

- Arduino IDE
  - Easy use
  - Rapid-Prototyping
  - Community and Libraries
- ESP-IDF
  - Low-Level Control
  - Memory Management
  - Large Scale and Commercial Projects
- PlatformIO

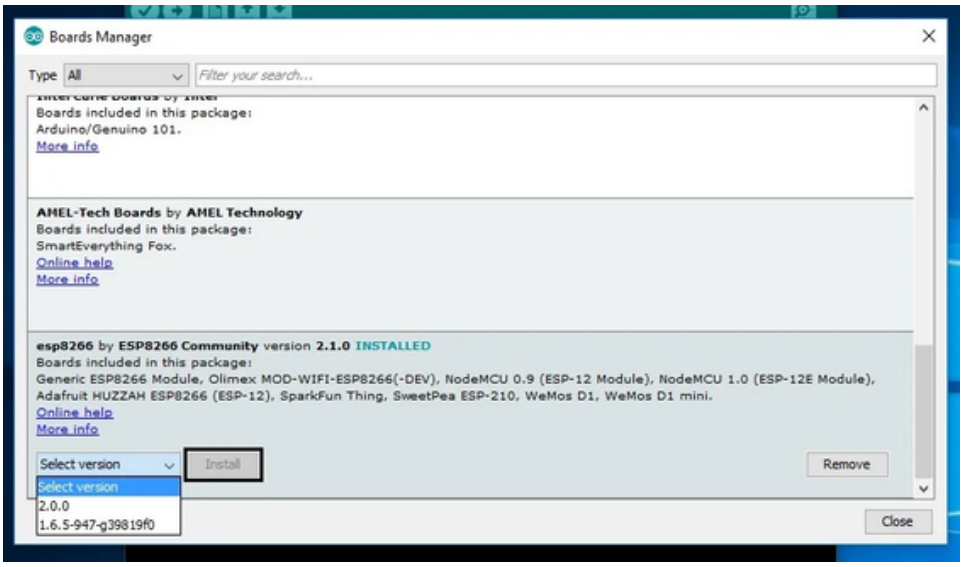
b10s HARDWARE

Arduino IDE

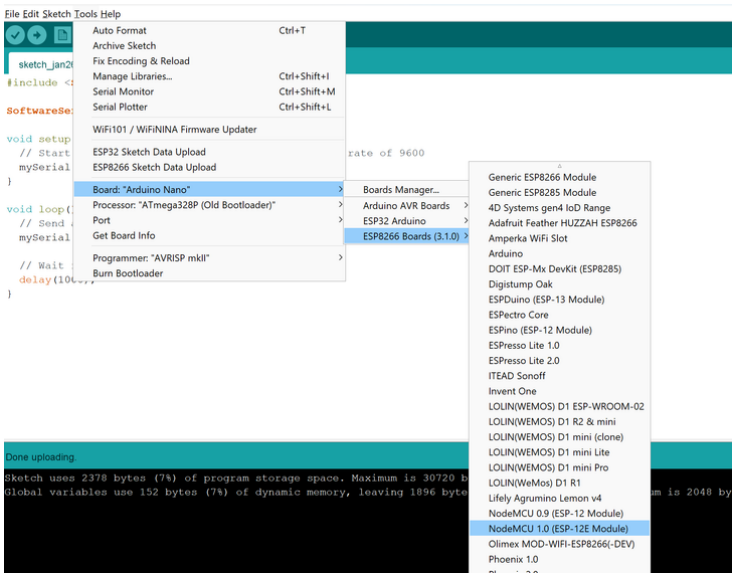
1. Open Arduino IDE
   - FILE -> Preferences.
2. 

http://arduino.esp8266.com/stable/package_esp8266com_index.json

3. Tools ->Board -> Board manager

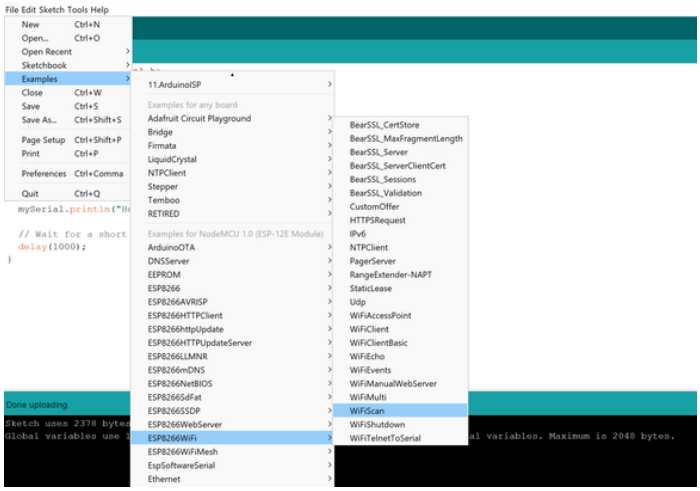


4.


File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP32 Sketch Data Upload
ESP8266 Sketch Data Upload
Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ESP32 Arduino
ESP8266 Boards (3.1.0)
Generic ESP8266 Module
Generic ESP8285 Module
4D Systems gen4 IoD Range
Adafruit Feather HUZZAH ESP8266
Amperka WiFi Slot
Arduino
DOIT ESP-Mx DevKit (ESP8285)
Digistump Oak
ESPDuino (ESP-13 Module)
ESPectro Core
ESPino (ESP-12 Module)
ESPresso Lite 1.0
ESPresso Lite 2.0
ITEAD Sonoff
Invent One
LOLIN(WEMOS) D1 ESP-WROOM-02
LOLIN(WEMOS) D1 R2 & mini
LOLIN(WEMOS) D1 mini (clone)
LOLIN(WEMOS) D1 mini Lite
LOLIN(WEMOS) D1 mini Pro
LOLIN(WeMos) D1 R1
Lifely Agrumino Lemon v4
NodeMCU 0.9 (ESP-12 Module)
NodeMCU 1.0 (ESP-12E Module)
Olimex MOD-WIFI-ESP8266(-DEV)
Phoenix 1.0
Done uploading.
Sketch uses 2378 bytes (7%) of program storage space. Maximum is 30720 b
Global variables use 152 bytes (7%) of dynamic memory, leaving 1896 byte

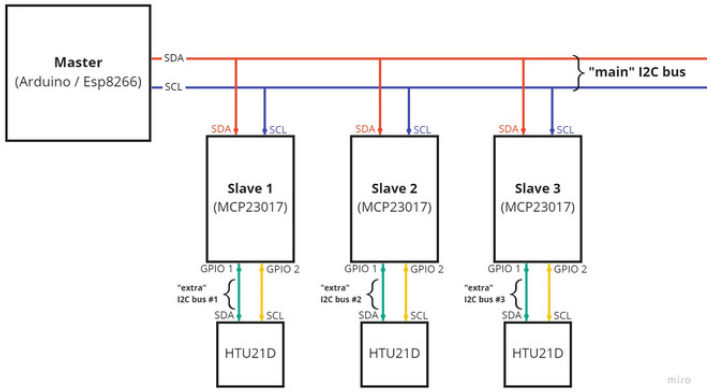
Example code

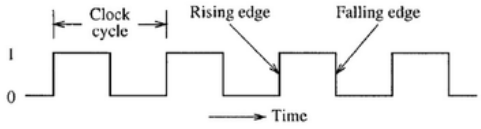
- File -> Examples -> ESP8266WiFi -> WiFiScan



I2C

- It stands for Inter-Integrated Circuit. It uses 2 wires to communicate, namely Serial Data (SDA) and Serial Clock (SCK).

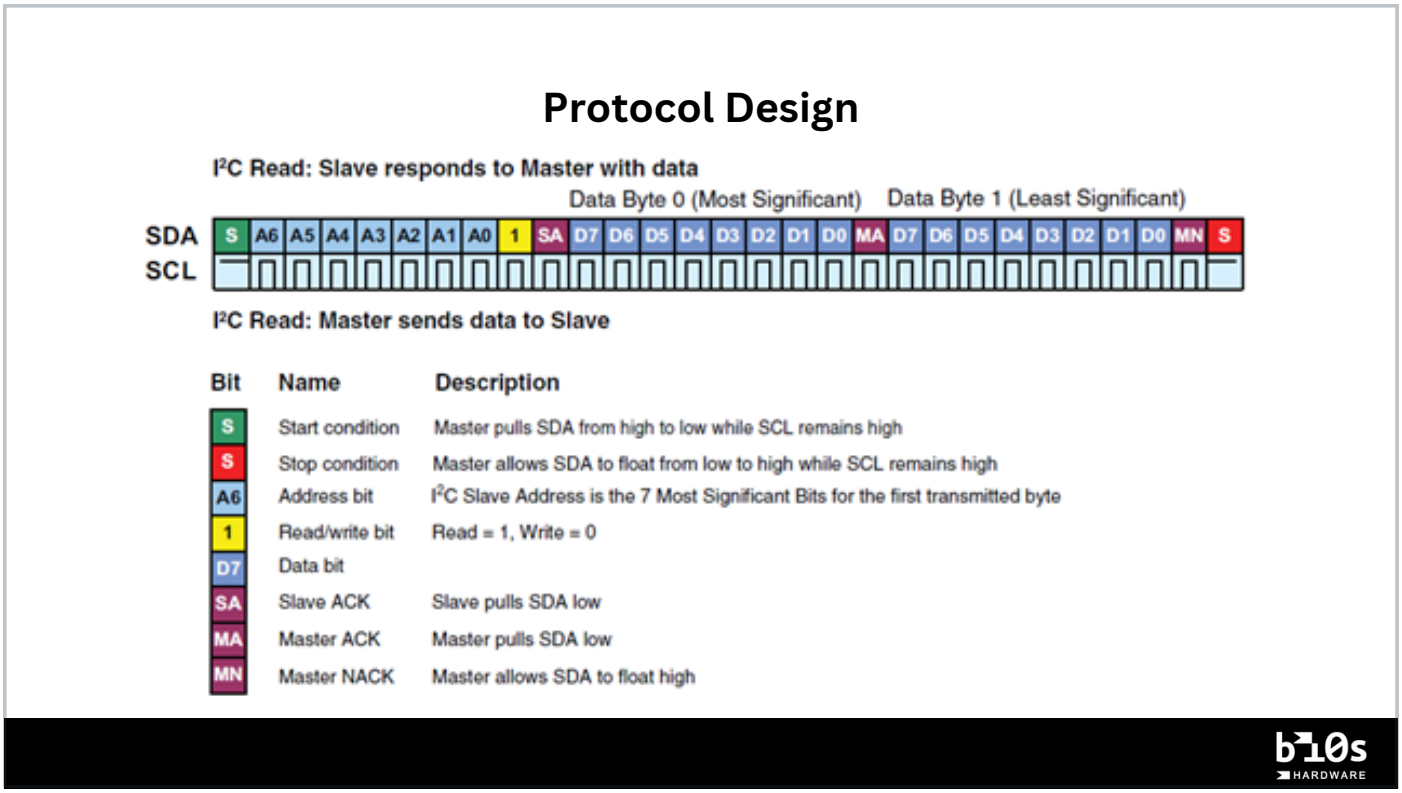




b10s HARDWARE

The way i2c works is , there will be a fixed clock pulse, that will set the communication speed for both the master and the slave devices. this need to be the same, because the slave/master device must know when to read the next bit. then the data is transmitted through the sda pin as serial data.

there are four potential modes of operation for a given I2C bus, they are:
1) master transmit
2) master receive
3) slave transmit
4) slave receive



1. Communication begins with a "Start" condition. This is when the SDA line transitions from high (logic to low (logic 0) while the SCL line is high.

2.After the start condition, the I2C bus expects a 7-bit or 10-bit address (depending on the device) of the target slave device.
The first 7 bits are the device address, and the 8th bit is the Read/Write (R/W) bit.
(0 = write,1 = read)

3. Corresponding slave responds by putting SDA line low.

4. DAta sent.

5. Stop by putting sda
Low->High when SCL is high.

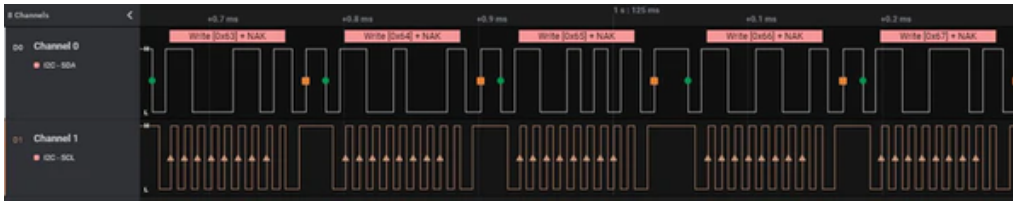
Advantages:

- Only 2 wires required
- Multiple masters can be used.
- Reduced cost and complexity

Disadvantages:

- As there are only 2 wires, there is additional complexity in handling the overhead of addressing and acknowledging.
- It can be inefficient in simple communications and a direct interface such as SPI, might be preferred.

Logic Analyser Output



Hands-on Session

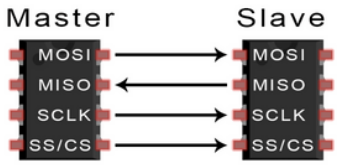
b10s HARDWARE

I2C Example

Day 3

SPI

- Serial peripheral interface



MOSI (Master Output/Slave Input) – Line for the master to send data to the slave.

MISO (Master Input/Slave Output) – Line for the slave to send data to the master.

SCLK (Clock) – Line for the clock signal.

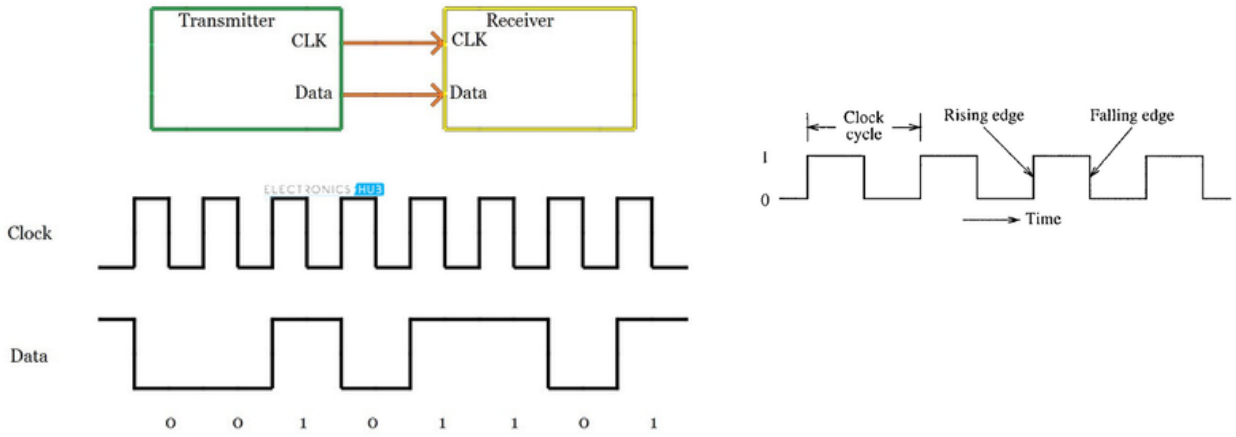
SS/CS (Slave Select/Chip Select) – Line for the master to select which slave to send data to.

b10s HARDWARE

-> benefit of SPI is the fact that data can be transferred without interruption(since no start/stop bit).
->master-slave configuration.

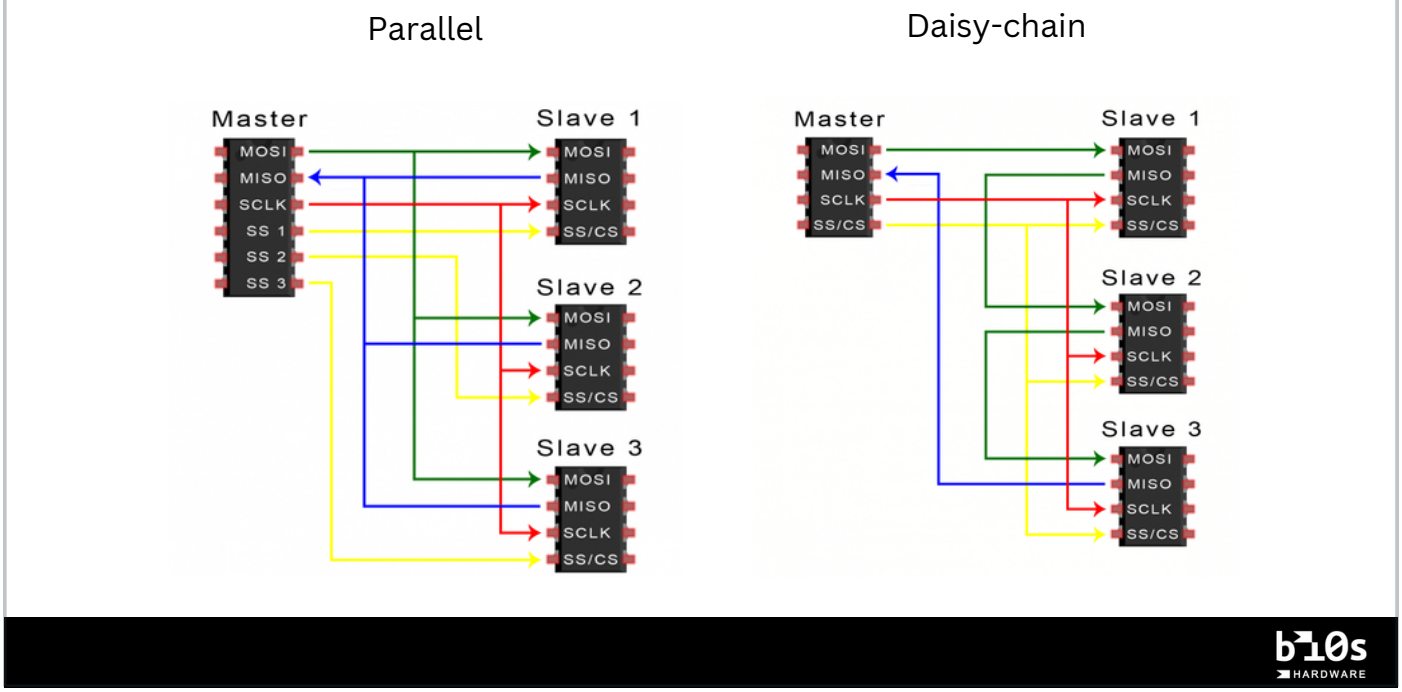
How SPI Works?

1. Clock



->master generates.
->synchronize the communication.
->1 bit transferred in each clock cycle.
->speed determined by frequency of clock.

2. Slave Select



->when SS pulled low then slave selected.
->for multiple slaves each slave should have dedicated slave select lines.

2) Daisy Chain Configuration:

- In a daisy chain configuration, all slaves are connected in series on the same bus.

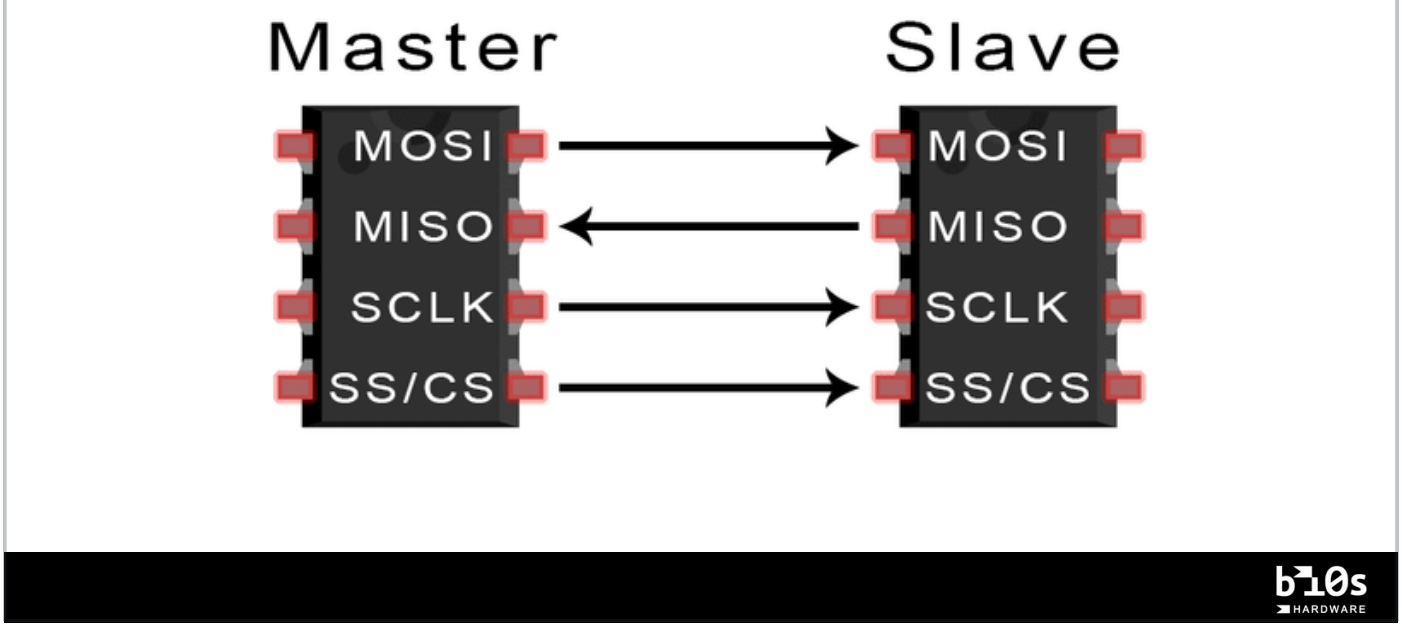
- Each slave has a serial data input (SDI) and a serial data output (SDO) line, and the output of one slave is connected to the input of the next slave.

- The master communicates with the first slave by shifting data into its serial input. The data then cascades through the chain of slaves, and the master receives the output data from the last slave in the chain.

- To select a specific slave in a daisy chain configuration, some extra bits are often sent with the data to indicate which slave is being addressed.

- The master sends the required number of bits to traverse the chain, and each slave examines the bits to determine if the data is meant for it. If not, it passes the data along to the next slave in the chain.

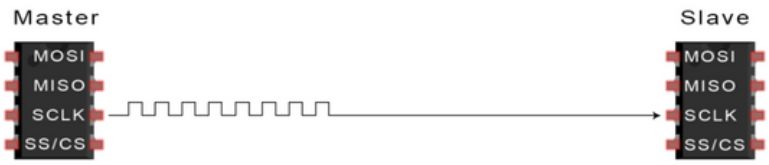
3. MOSI and MISO



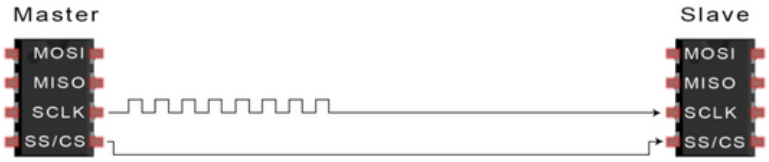
1. mosi and miso.

Steps of SPI Communication

1. The master outputs the clock signal:



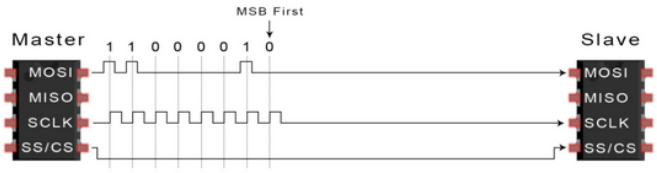
2. The master switches the SS/CS pin to a low voltage state, which activates the slave:



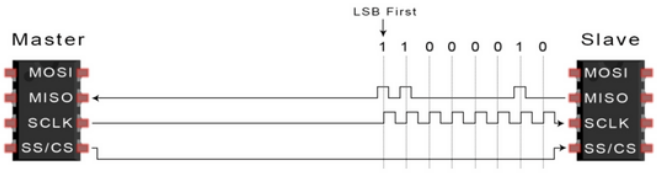
b10s HARDWARE

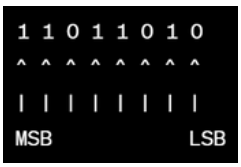
2. master to slave (msb).
   slave to master (lsb).

3. The master sends the data one bit at a time to the slave along the MOSI line. The slave reads the bits as they are received:



4. If a response is needed, the slave returns data one bit at a time to the master along the MISO line. The master reads the bits as they are received:





b10s HARDWARE

2. master to slave (msb).
   slave to master (lsb).

Advantages

- No start and stop bits, so the data can be streamed continuously without interruption
- No complicated slave addressing system like I2C
- Higher data transfer rate than I2C (almost twice as fast)
- Separate MISO and MOSI lines, so data can be sent and received at the same time

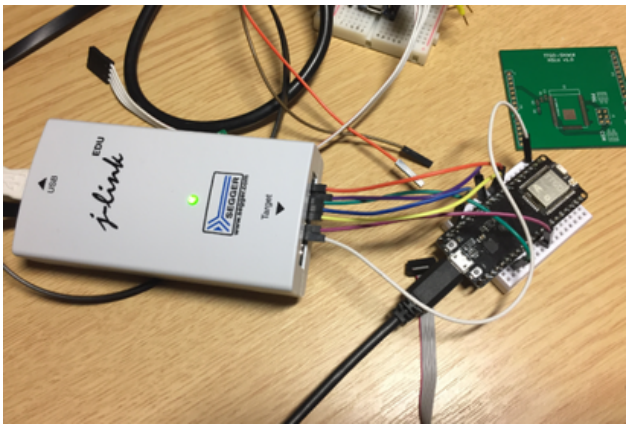
Disadvantages

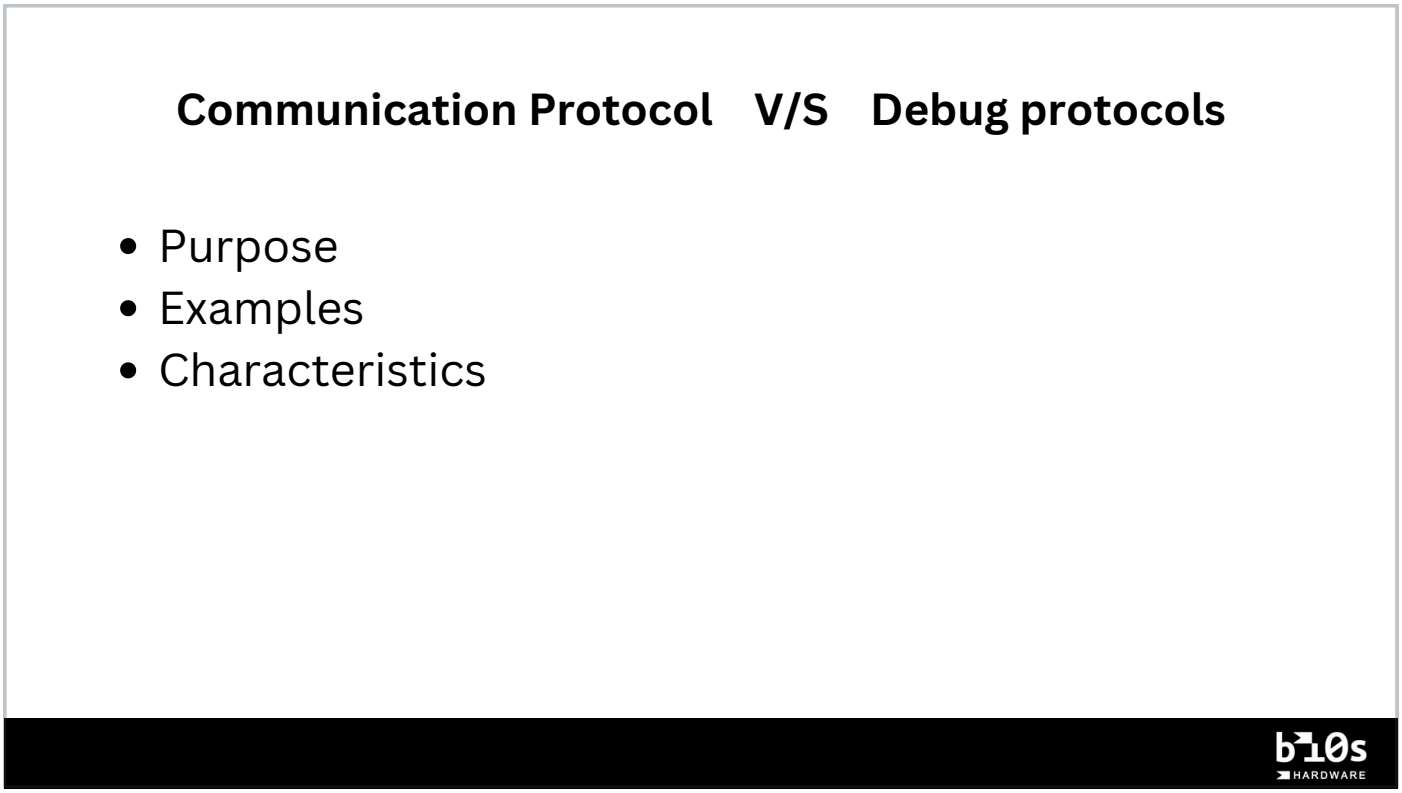
- Uses four wires (I2C and UARTs use two)
- No acknowledgement that the data has been successfully received (I2C has this)
- No form of error checking like the parity bit in UART
- Only allows for a single master

SPI EXAMPLE CODE

Debug Protocols

- JTAG (Joint Test Action Group)
- SWD (Serial Wire Debug)





1. **Communication Protocols:**

- **Purpose:** Communication protocols are designed to facilitate the exchange of data between different components or devices within an embedded system. They define a set of rules and conventions for data transmission, ensuring that devices can understand and interpret the information being exchanged.

- **Examples:** Common communication protocols in embedded systems include I2C, SPI, UART, CAN, and Ethernet. These protocols determine how data is formatted, transmitted, and received, allowing seamless communication between microcontrollers, sensors, actuators, and other peripherals.

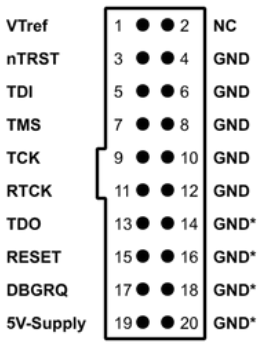
- **Characteristics:**
Communication protocols address issues such as data synchronization, error detection and correction, addressing, and the overall structure of the communication process.

---

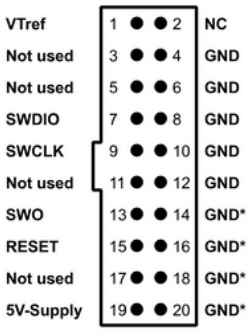
2. **Debugging Protocols:**
- **Purpose:** Debugging protocols are specifically designed to assist developers in identifying and fixing software and hardware issues within an embedded system. They provide tools and interfaces for monitoring the system's behavior, collecting information, and diagnosing problems during the development and testing phases.

- **Examples:** JTAG (Joint Test Action Group) and SWD (Serial Wire Debug) are common debugging protocols used in embedded systems. These protocols enable features such as breakpoints, watchpoints, and real-time debugging, allowing developers to analyze the system's internal state and trace the execution of code.

- **Characteristics:** Debugging protocols focus on providing visibility into the system's operation, allowing developers to inspect variables, step through code, and identify the root causes of issues. They often operate at a lower level than communication protocols and are closely tied to the hardware architecture.



JTAG Pins



SWD Pins

DI: Test Data In, a serial input pin.

DO: Test Data Out, a serial output pin.

CK: Test Clock, a clock pin.

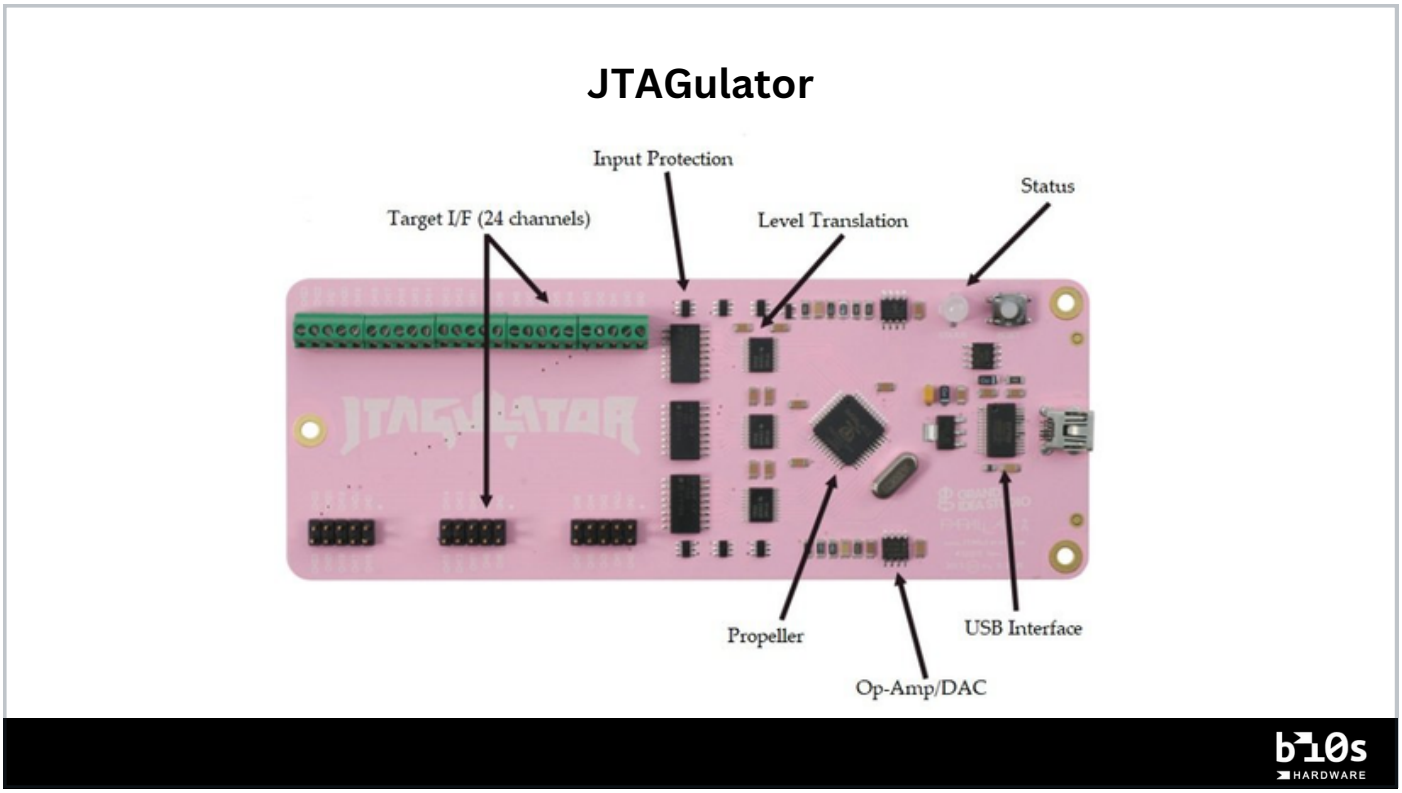
MS: Test Mode Select, a mode selection (control signal) pin

RST: Test Reset, a reset pin.

SWDIO: Serial Wire Data Input Output

SWCLK: Serial Wire Clock

b10s HARDWARE



It's particularly useful for identifying and mapping JTAG interfaces on a circuit board.