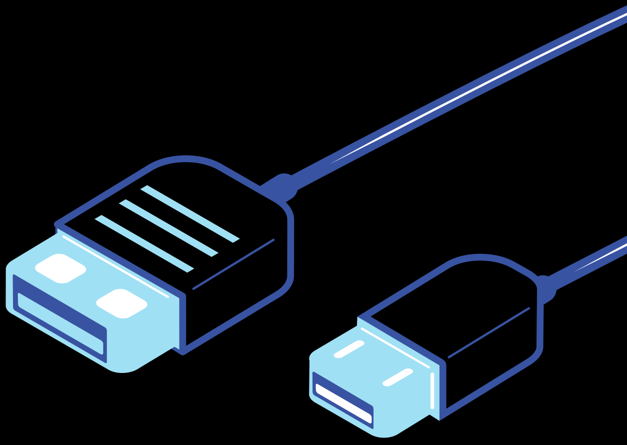
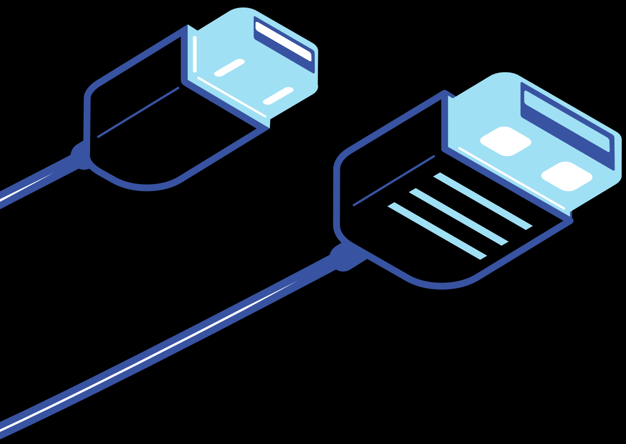


UART

Universal Asynchronous Receiver / Transmitter

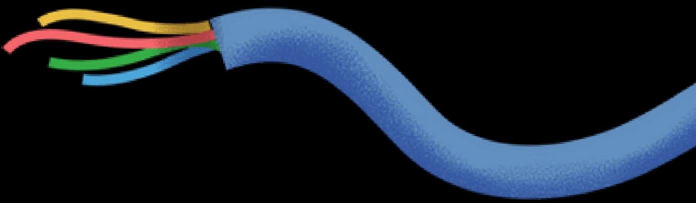


UART

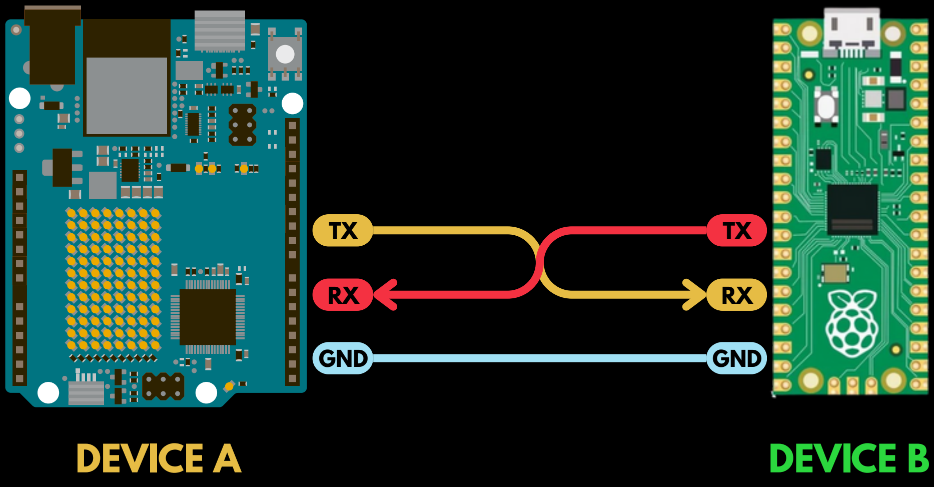
(Universal Asynchronous Receiver / Transmitter)

"Universal" means the protocol can be applied to any compatible transmitter and receiver. "Asynchronous" indicates that it does not use a clock signal for data communication.

UART is a commonly used hardware communication protocol that enables serial data exchange between devices using just two wires, one for transmitting and one for receiving. It's called "Universal" because it can work with various devices, and "Asynchronous" because it doesn't need a shared clock signal for synchronization. UART is ideal for point-to-point communication, such as between microcontrollers and peripherals, and it can operate in half-duplex or full-duplex modes for one-way or two-way data transmission.

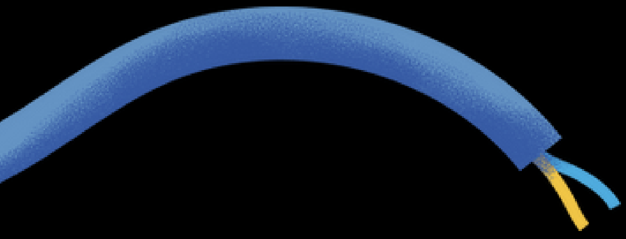


UART Connection



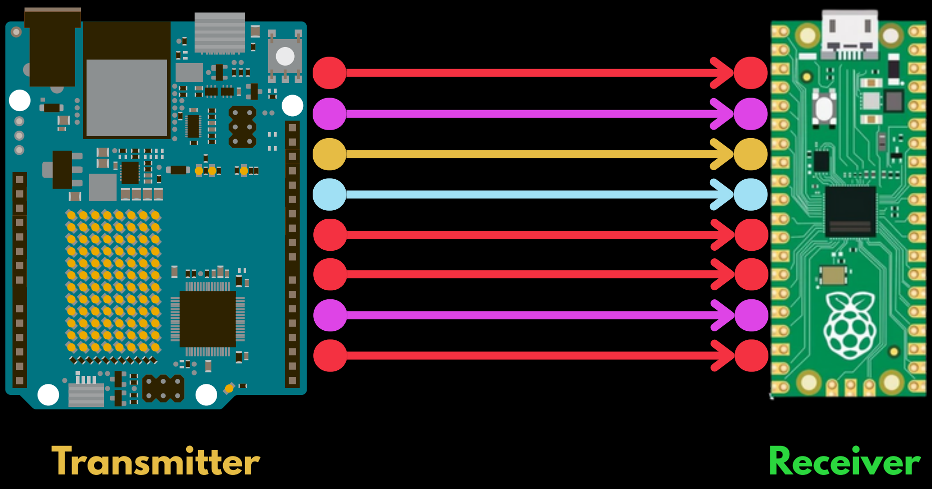
Here, DEVICE A has a Transmitter (TX) pin and a Receiver (RX) pin, and DEVICE B also has a Receiver (RX) and Transmitter (TX) pin. The TX of DEVICE A should be connected to the RX of DEVICE B, and the TX of DEVICE B should be connected to the RX of DEVICE A. Only two wires are needed to establish this communication..

If DEVICE A wants to send data, it transmits through its TX pin, which is received by the RX pin of DEVICE B. Similarly, if DEVICE A wants to receive data, it does so through its RX pin, which gets the data sent from the TX pin of DEVICE B.



A common ground is essential in UART communication to define high and low signals; without it, devices may misinterpret the transmitted data.

Parallel Communication



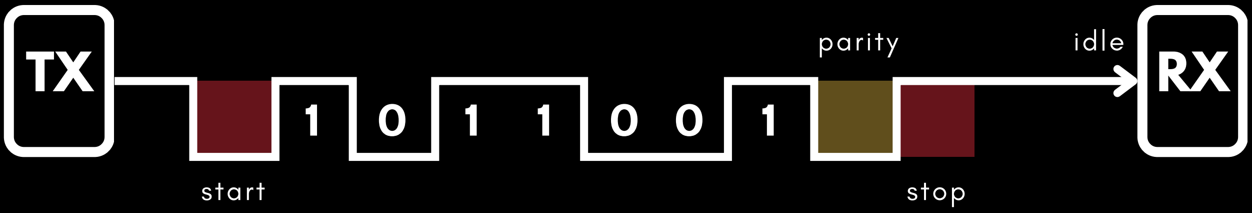
Compared to parallel communication, UART serial communication requires only two lines, making it less complex and more efficient in terms of wiring. In contrast, parallel communication needs multiple data lines (buses), which increases complexity but offers higher data transfer speeds.

Therefore, when high-speed data transfer is required, parallel communication is preferred due to its faster performance. However, for low-speed applications where simplicity and minimal wiring are important, UART serial communication is a better choice as it reduces bus complexity.

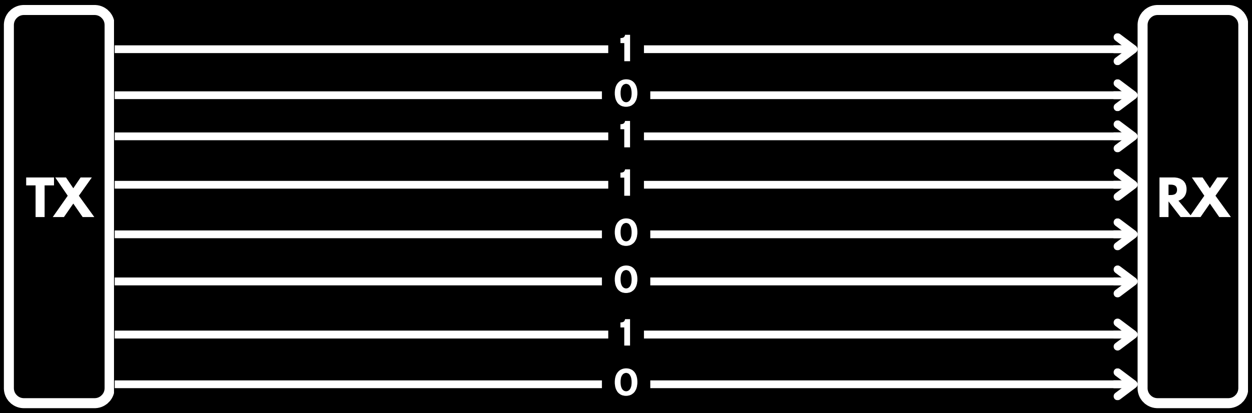
How UART Works

UART operates by transmitting data one bit at a time in a specific sequence, starting with a start bit, followed by data bits, an optional parity bit, and one or more stop bits. Unlike parallel communication, where multiple bits are sent simultaneously, UART sends bits serially. As the name suggests, it is asynchronous, meaning it doesn't use a shared clock signal; instead, it relies on predefined baud rates to synchronize the timing of data transmission.

Serial Communication



Parallel Communication

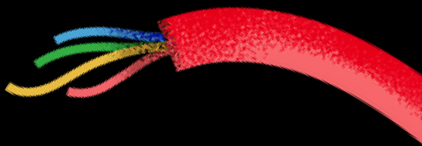


In parallel communication, an 8-bit message requires eight separate data lines...
Serial communication simplifies this by using just one wire for sending and one for receiving.

Baud Rate

Baud Rate is a key parameter in UART communication, defining the number of bits transmitted per second (bps). Both transmitting and receiving devices must use the same baud rate to communicate correctly. A higher baud rate enables faster data transfer but requires precise timing, while a lower baud rate offers more tolerance in timing but slower communication.

eg: In Arduino example,



```
Serial.begin(9600);
```

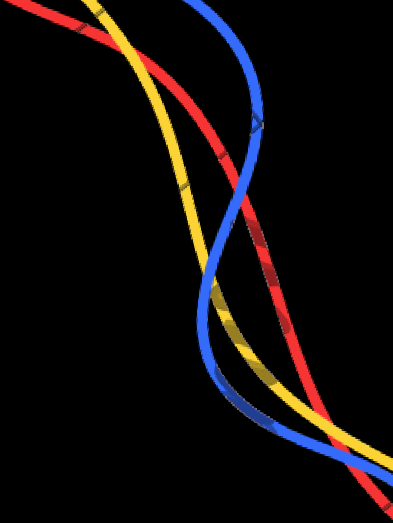
The baud rate is simply the number of symbol changes (bits) per second.

Bit time (T_{bit})

$$\text{baud rate} = \frac{1}{T_{\text{bit}}} \quad \Rightarrow \quad T_{\text{bit}} = \frac{1}{\text{baud rate}}$$

For 9600 bps:

$$T_{\text{bit}} = \frac{1}{9600 \text{ bits/s}} \approx 0.00010417 \text{ s} = 104.17 \ \mu\text{s}$$



Frame duration

Each UART frame typically consists of:

- 1 start bit
- 5–8 data bits (commonly 8)
- 0 or 1 parity bit
- 1 or 2 stop bits

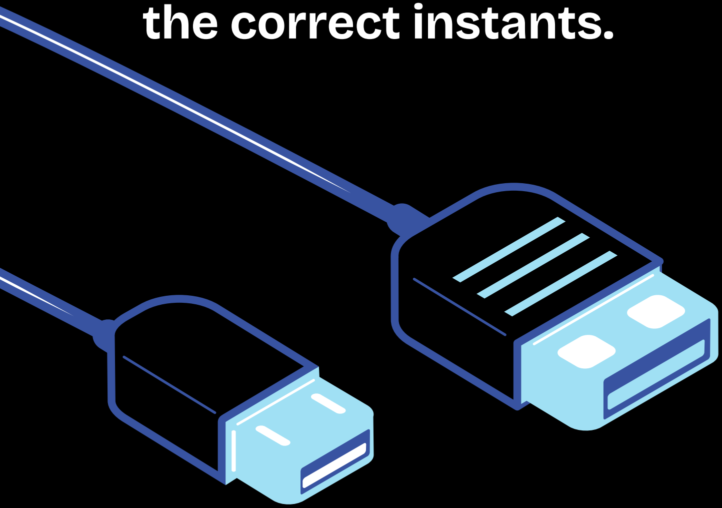
If you use 8 data bits, no parity, 1 stop bit, then a frame is 10 bits long. The time to send one frame is:

$$T_{frame} = (\text{start} + \text{data} + \text{parity} + \text{stop}) \times T_{bit}$$
$$= 10 \times 104.17\ \mu s \approx 1.0417\ \text{ms}$$

Implications

- Higher baud → smaller T_{bit} → faster data but tighter timing tolerances.
- Lower baud → larger T_{bit} → slower data but more forgiving timing.

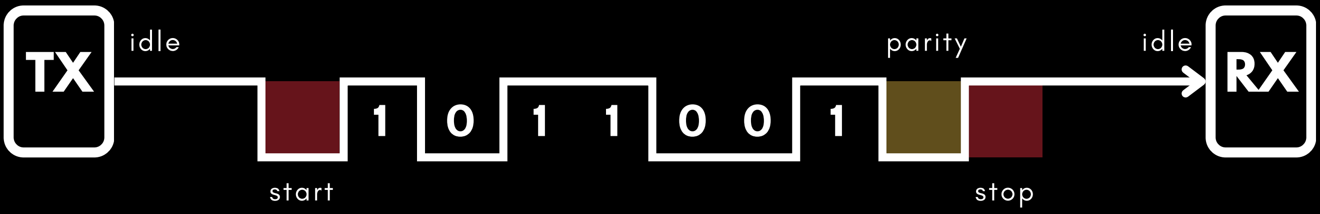
By calling Serial.begin(9600);, you ensure both Arduino TX and RX agree on T_{bit}=104.17 µs, so bits are sampled at the correct instants.



UART Frame Format

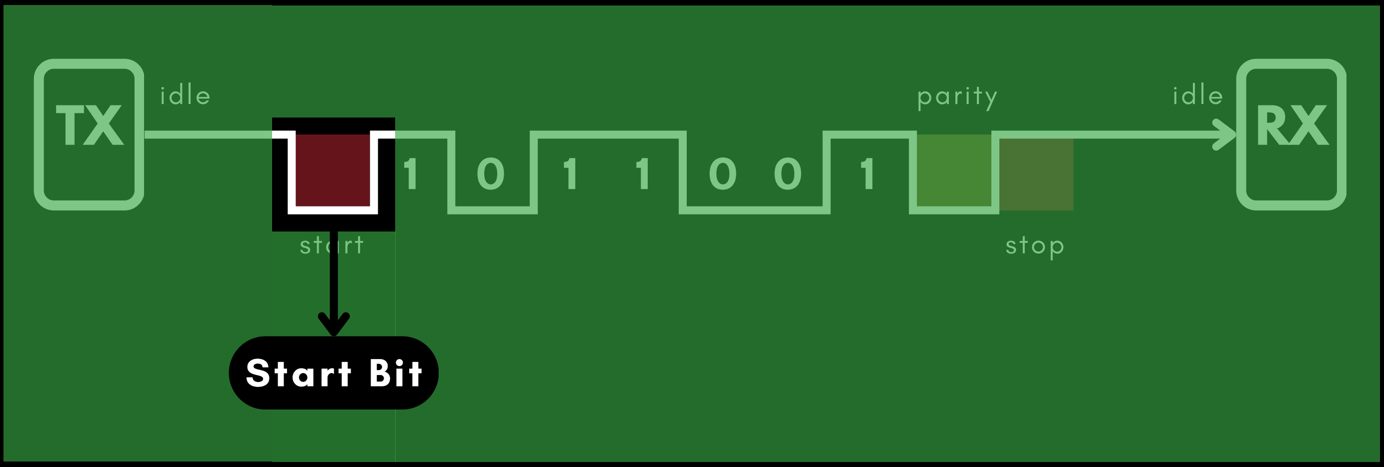
Start and stop bits mark the beginning and end of each data frame, helping the sender and receiver stay synchronized.

UART Frame Format



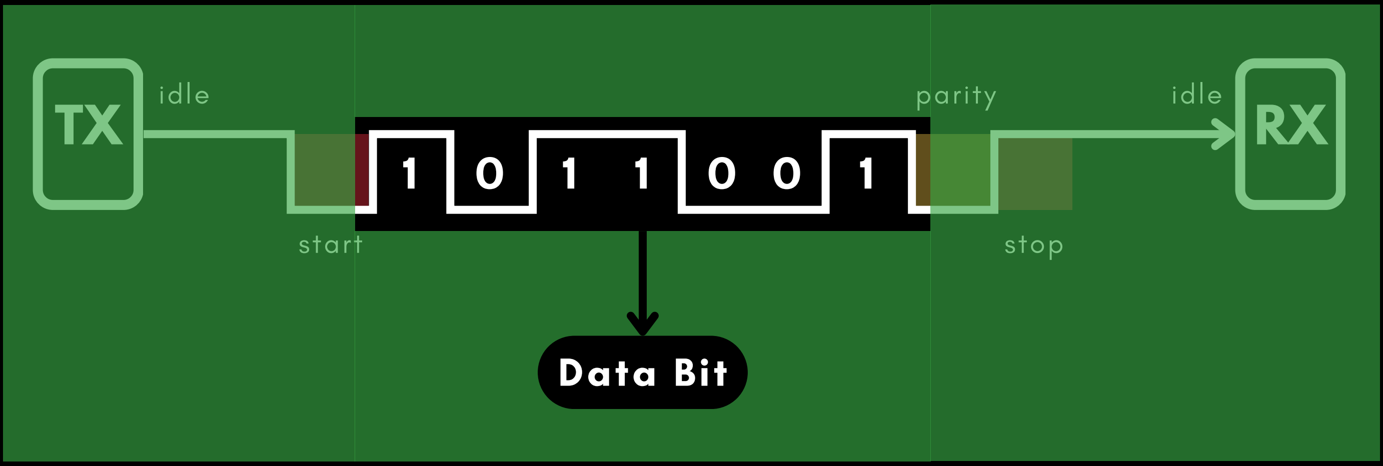
Start Bit

In UART communication, each data frame starts with a single start bit. The start bit is always logic low (0), indicating the beginning of data transmission. This low signal alerts the receiver to prepare for incoming data. Once the receiver detects the start bit, it begins timing the reception of the following bits in the frame.



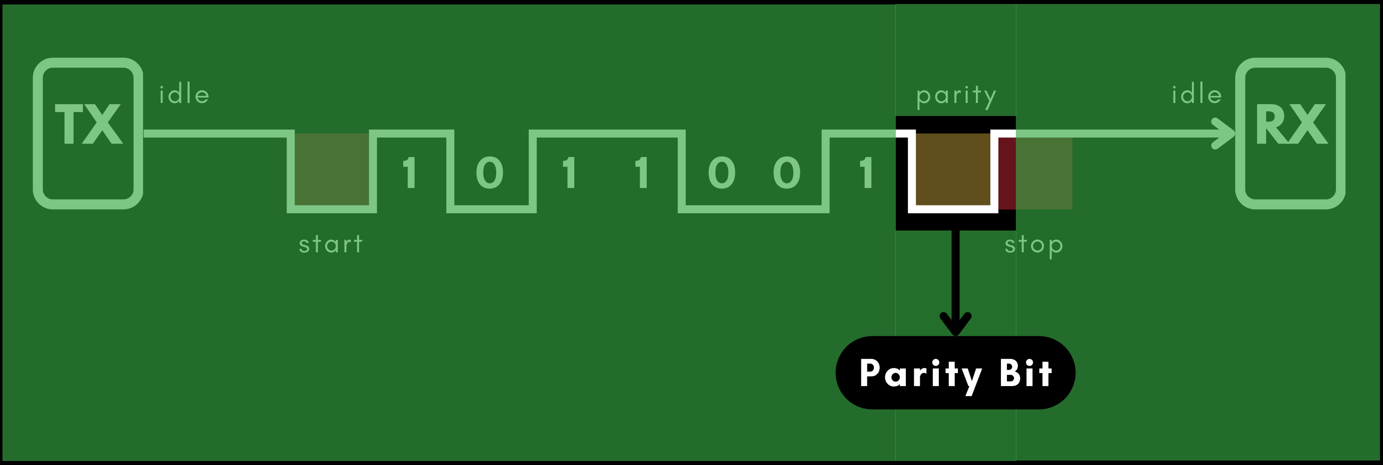
Data Bit

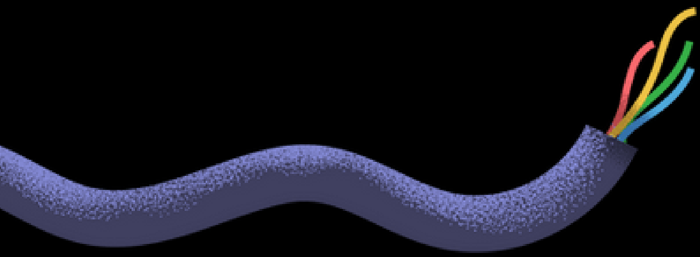
Data bits carry the actual information being transmitted. While the most common configuration is 8 data bits, UART also supports other sizes like 7 or 6 bits, depending on the application's needs.



Parity Bit

It's an error-checking feature added to the data frame. It helps detect transmission errors by ensuring the total number of bits set to logic "1" is either even or odd, based on the selected parity type. If the received data doesn't match the expected parity, the receiver flags it as an error, indicating possible data corruption.

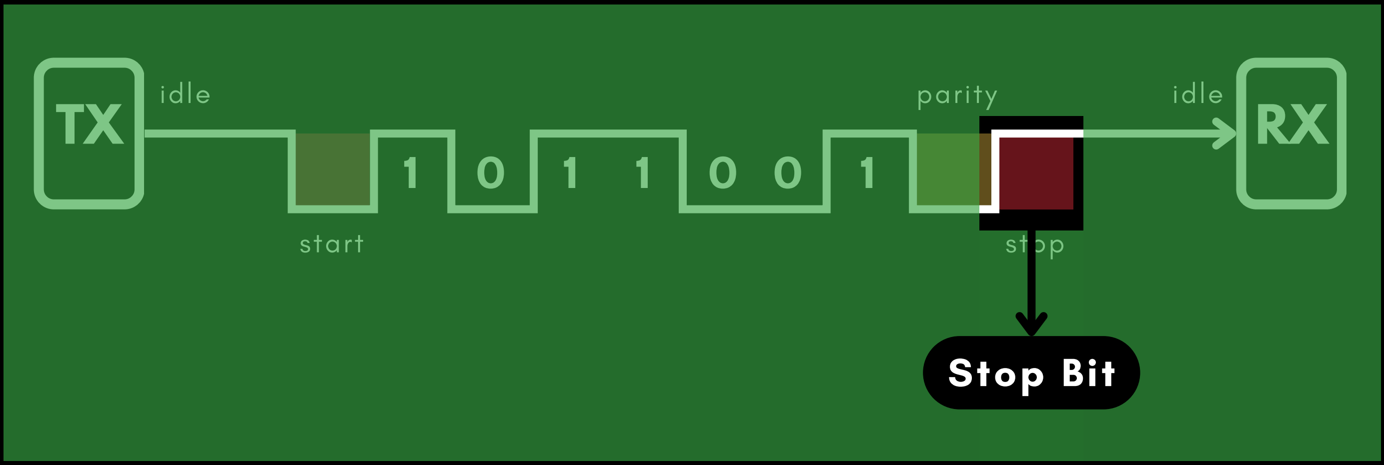


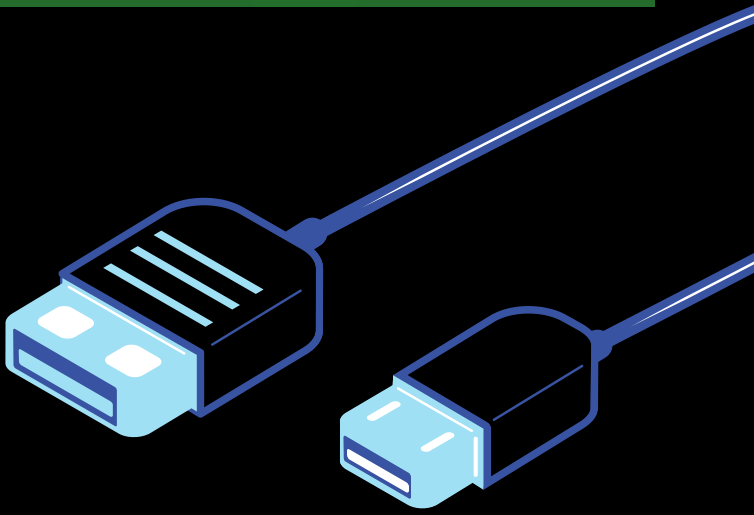


If even parity is used, the total number of 1s in the data (including the parity bit) must be even. If odd parity is used, the total must be odd.

Stop Bit

Stop bit(s) are sent after the data bits to indicate the end of a data frame. The most common setup uses one stop bit, but two stop bits can be used for added reliability. Stop bits are usually at logic high (1), though their polarity may vary depending on the system's configuration.





USART

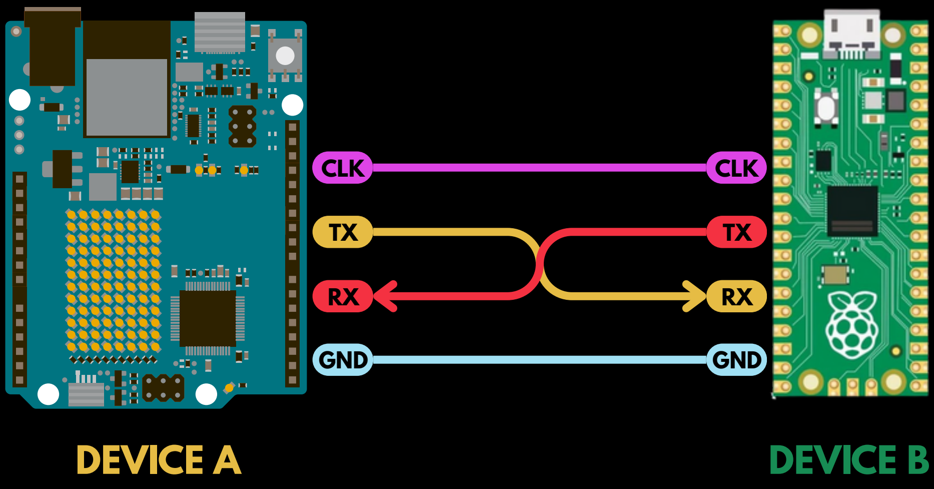
Universal Synchronous/Asynchronous Receiver/Transmitter

UART supports only asynchronous communication (no clock line).

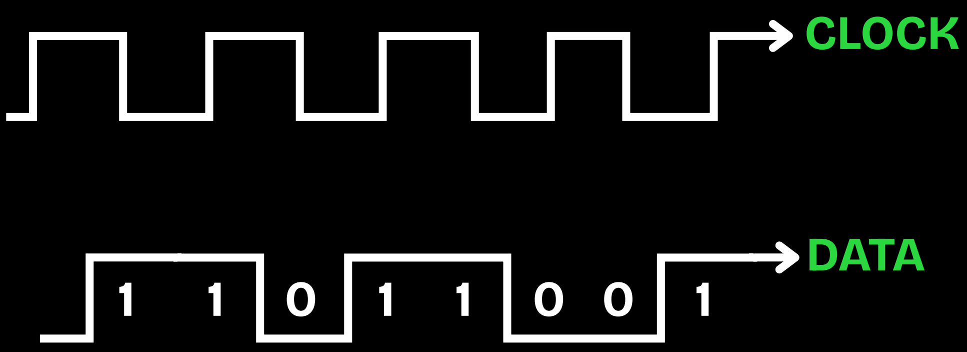
USART supports both:

- Asynchronous mode (like UART).
- Synchronous mode, where a clock signal is used along with data for faster and more reliable communication.

USART = UART + Synchronous mode



Synchronous mode



Kaviselvan