

Sistemas Operacionais Embarcados

UART

Comunicação serial

Possíveis cenários:

- Aparelhos ligados à internet (IoT)
- Envio de dados de um microcontrolador para um computador pessoal
- Troca de dados entre microcontroladores
- Leitura de sensores (GPS, acelerômetro etc.)
- Leitura e escrita em memória externa

→ USB

→ WiFi

→ Ethernet

→ Bluetooth

→ HDMI

→ VGA

→ UART

→ SPI

→ I2C

→ I2S

→ CAN

→ Etc.

Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ USB

→ WiFi

→ Ethernet

→ Bluetooth

→ HDMI

→ UART

→ SPI

→ I2C

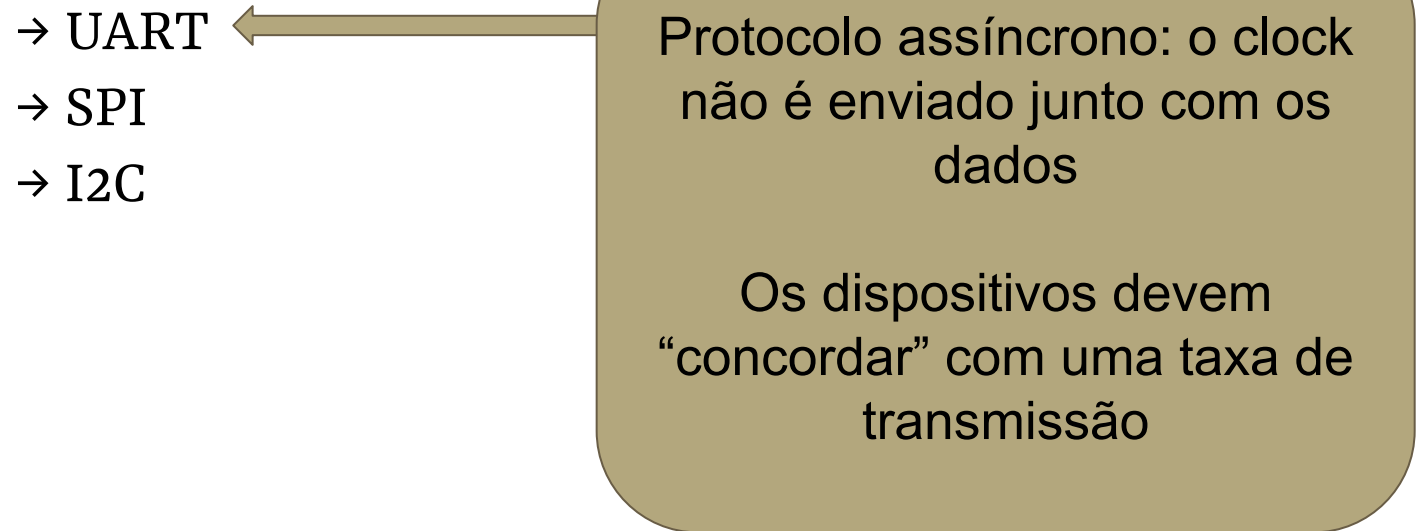
Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C



Protocolo assíncrono: o clock não é enviado junto com os dados

The diagram consists of a light brown rounded rectangle containing text. An arrow points from the left side of this rectangle to the text '→ UART'.

Os dispositivos devem “concordar” com uma taxa de transmissão


Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C



Protocolos síncronos: o clock é enviado junto com os dados

O dispositivo que gera o clock é denominado “mestre”, e os demais dispositivos são denominados “escravos”

Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C

Necessita de 2 fios:
transmissão e recepção

Permite comunicação
full-duplex

O fio de transmissão pode
ser usado para indicar o
endereço do dispositivo
(quando há mais de um
deles)

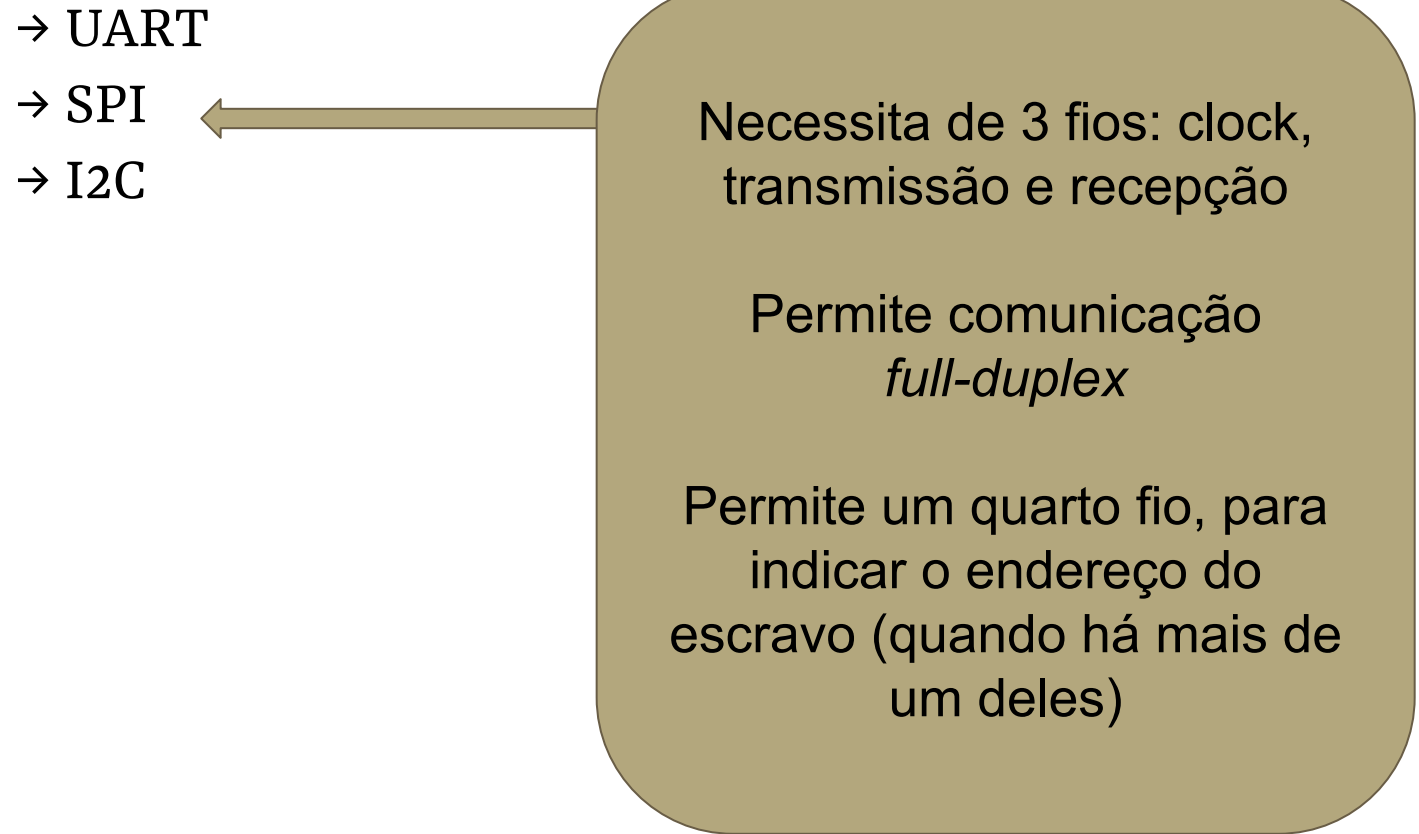
Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C



Necessita de 3 fios: clock, transmissão e recepção

Permite comunicação
full-duplex

Permite um quarto fio, para
indicar o endereço do
escravo (quando há mais de
um deles)

Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C

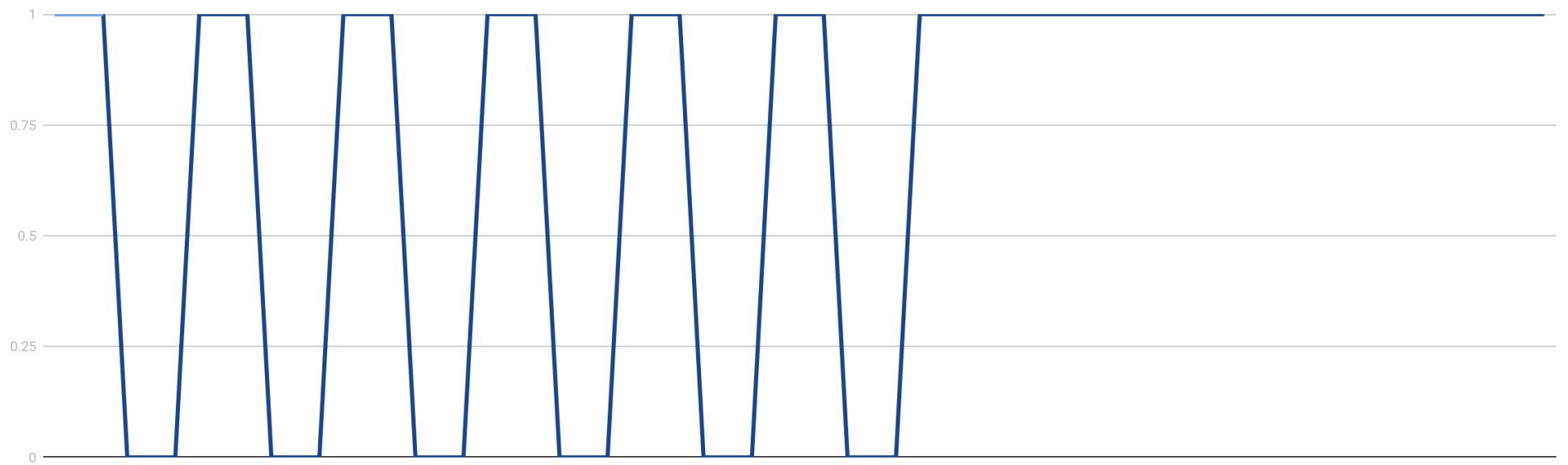
Possui dois fios: clock e dados

Permite comunicação
half-duplex

O fio de dados pode ser usado para indicar o endereço do escravo (quando há mais de um deles)

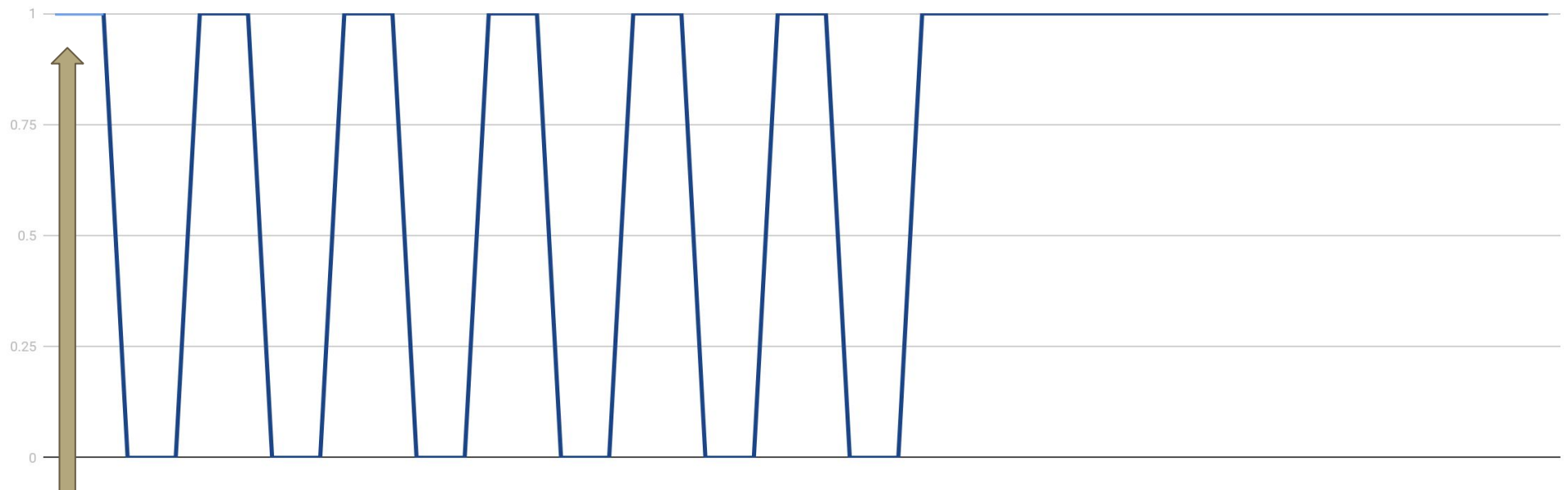
UART

ST	0	1	2	3	4	5	6	7	SP	ST	0	1	2	3	4	5	6	7	SP
----	---	---	---	---	---	---	---	---	----	----	---	---	---	---	---	---	---	---	----



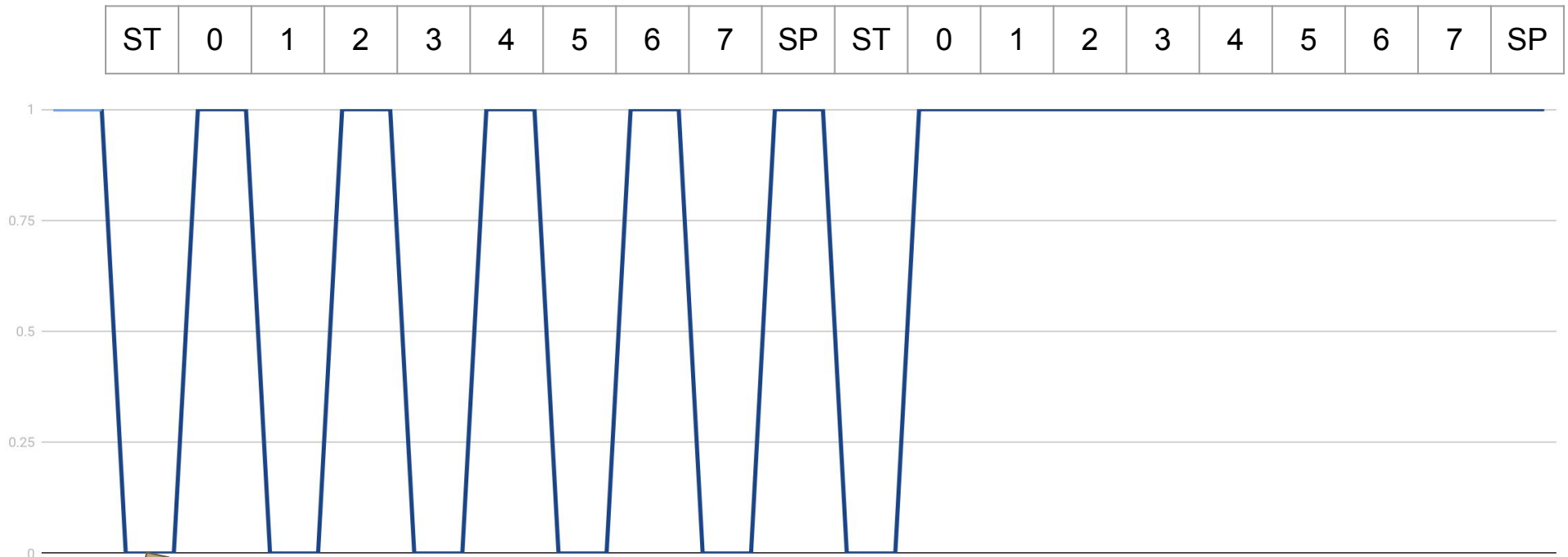
UART

ST	0	1	2	3	4	5	6	7	SP	ST	0	1	2	3	4	5	6	7	SP
----	---	---	---	---	---	---	---	---	----	----	---	---	---	---	---	---	---	---	----



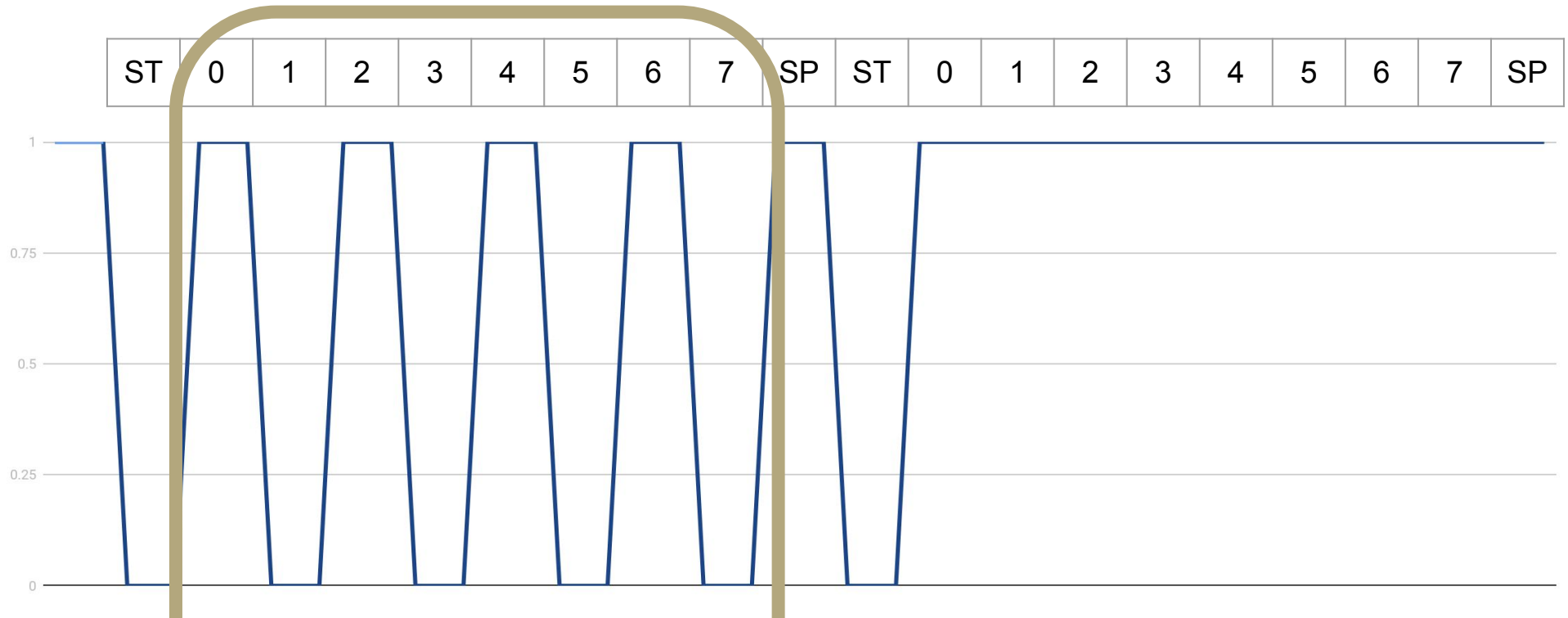
A linha de transmissão (TX) fica em nível alto enquanto não houver dados para enviar

UART



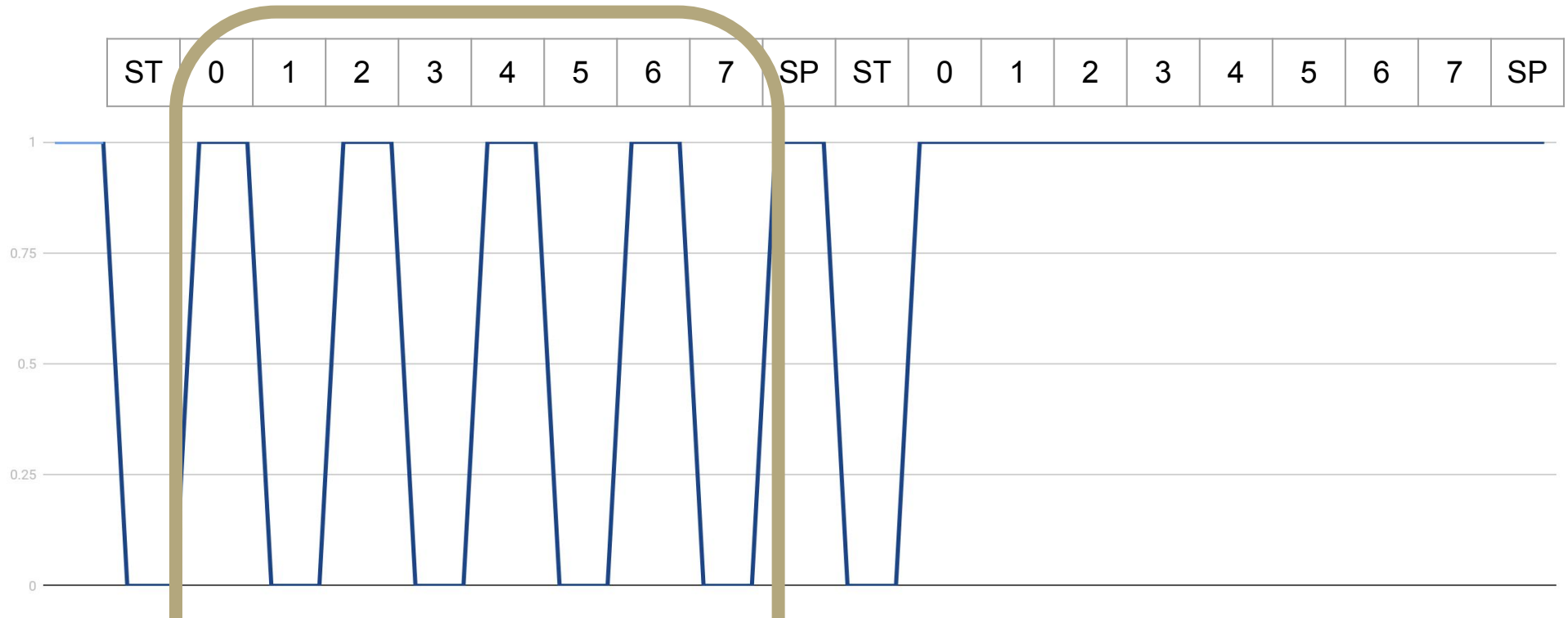
Um bit em nível baixo indica o começo da transmissão (START bit)

UART



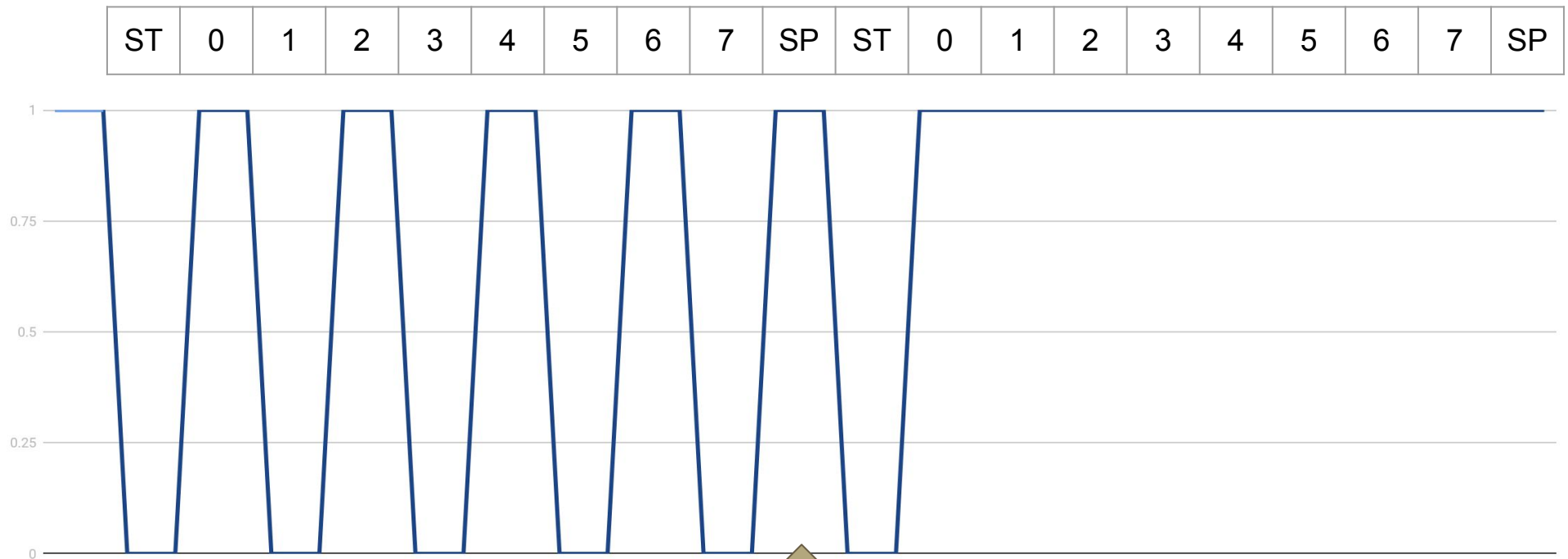
O byte de informação é enviado serialmente
(neste caso, 0b01010101 ou 0x55)

UART



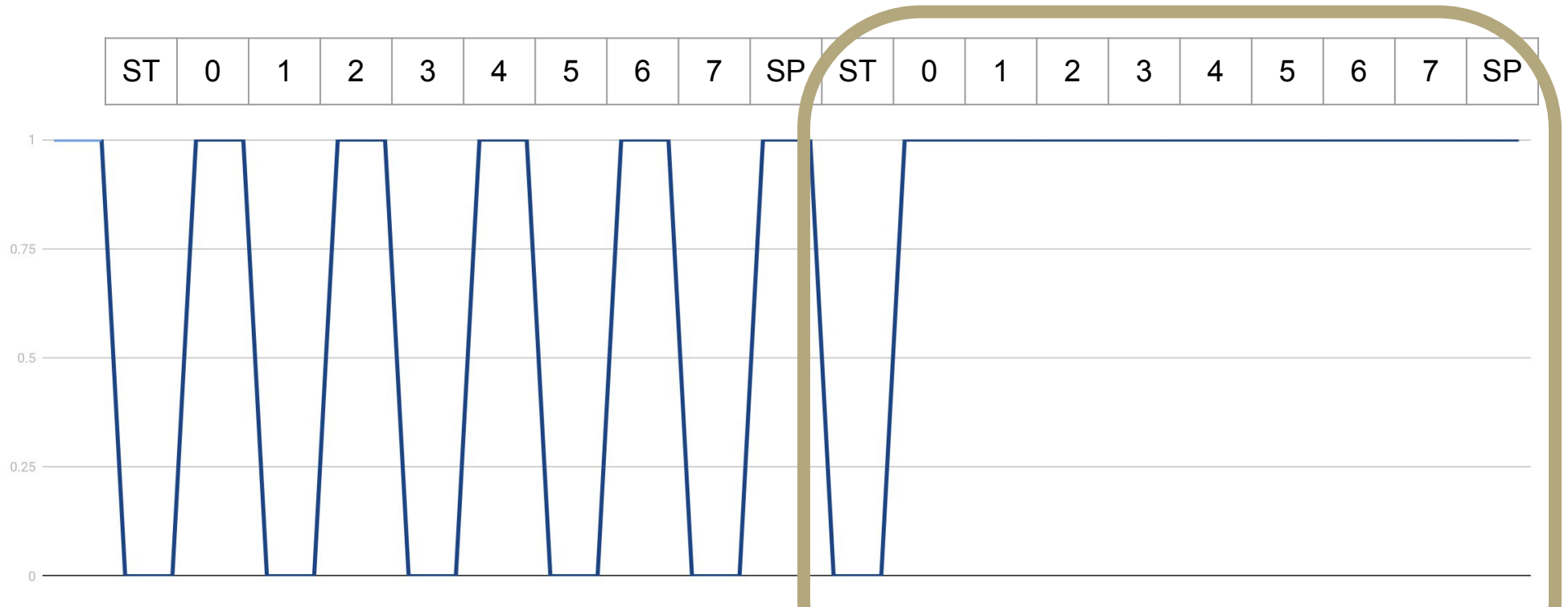
A ordem dos bits enviados deve ser determinada previamente (LSB ou MSB)

UART



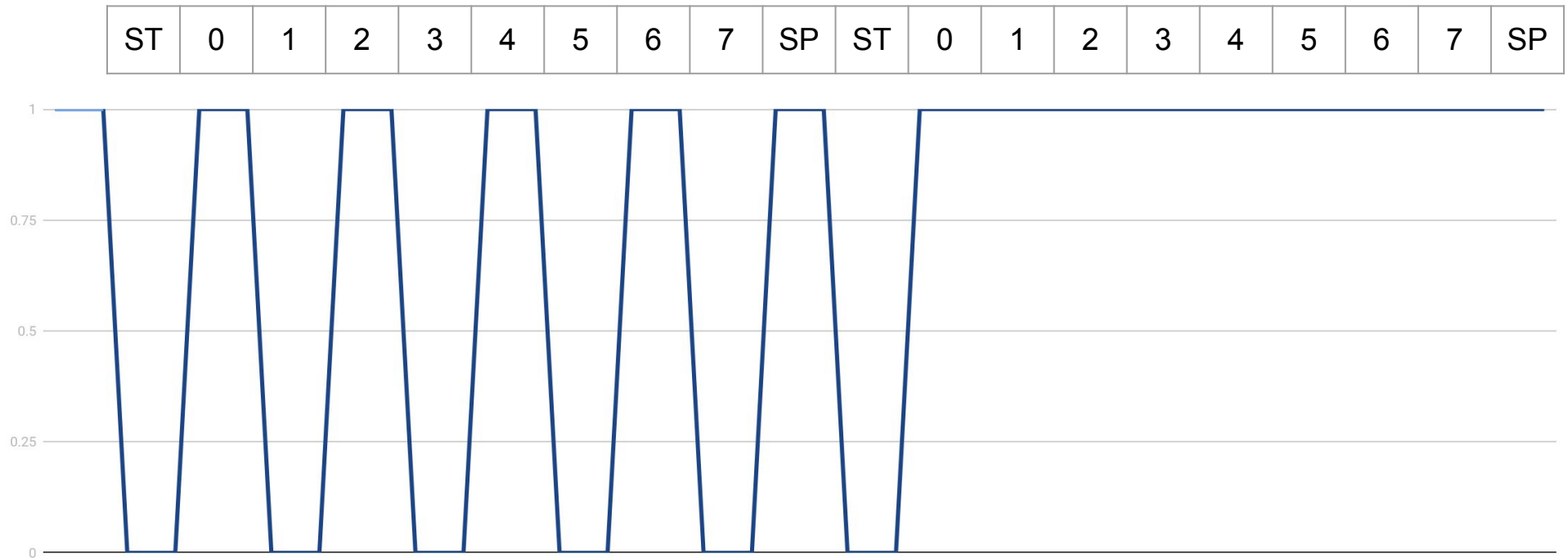
Um bit em nível alto indica o fim da transmissão (STOP bit)

UART



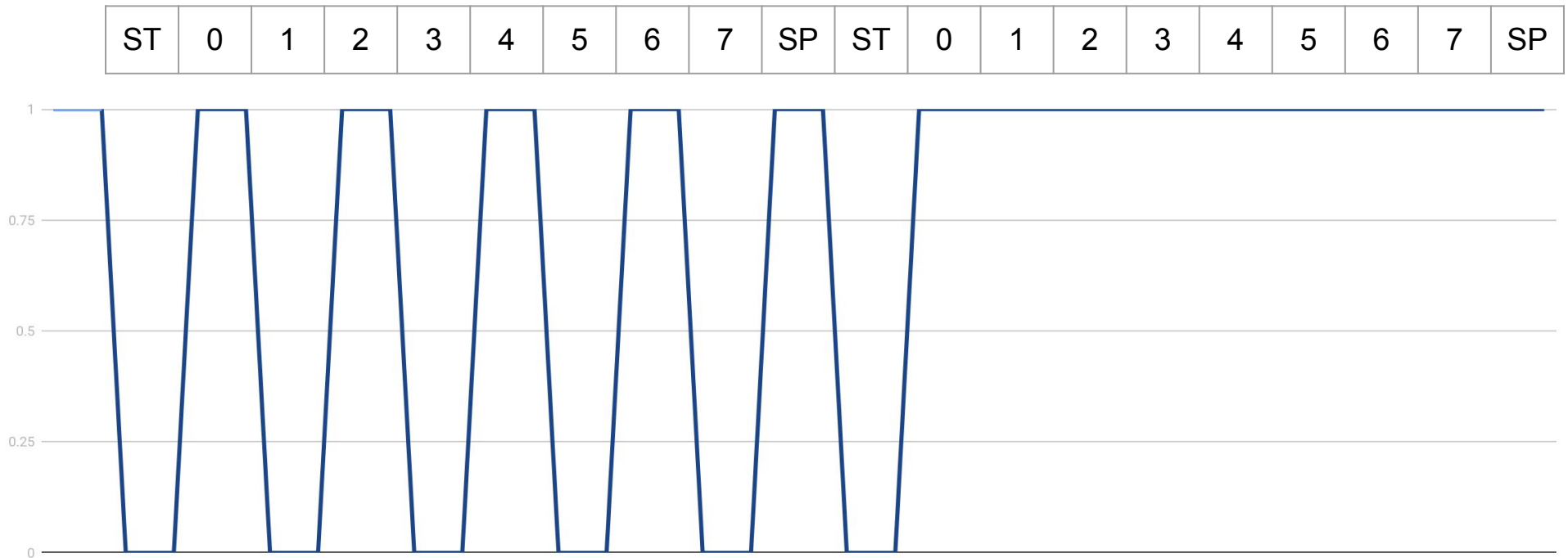
O mesmo protocolo é seguido aqui para enviar o byte 0b11111111, ou 0xFF

UART



Não há sinal de clock. A temporização dos bits deve ser previamente conhecida pelo transmissor e pelo receptor

UART



A taxa de transmissão é chamada de baud rate. Ela é diferente da taxa de dados, já que o protocolo prevê bits extra, como o START e o STOP

UART

ST (0)	D0	D1	D2	D3	D4	D5	D6	D7	AD	PA	SP (1)	SP (1)
--------	----	----	----	----	----	----	----	----	----	----	--------	--------

Pode-se enviar 7 ou 8 bits de informação (caracteres ASCII, por exemplo, precisam de somente 7 bits)

ST (0)	D0	D1	D2	D3	D4	D5	D6	D7	AD	PA	SP (1)	SP (1)
--------	----	----	----	----	----	----	----	----	----	----	--------	--------

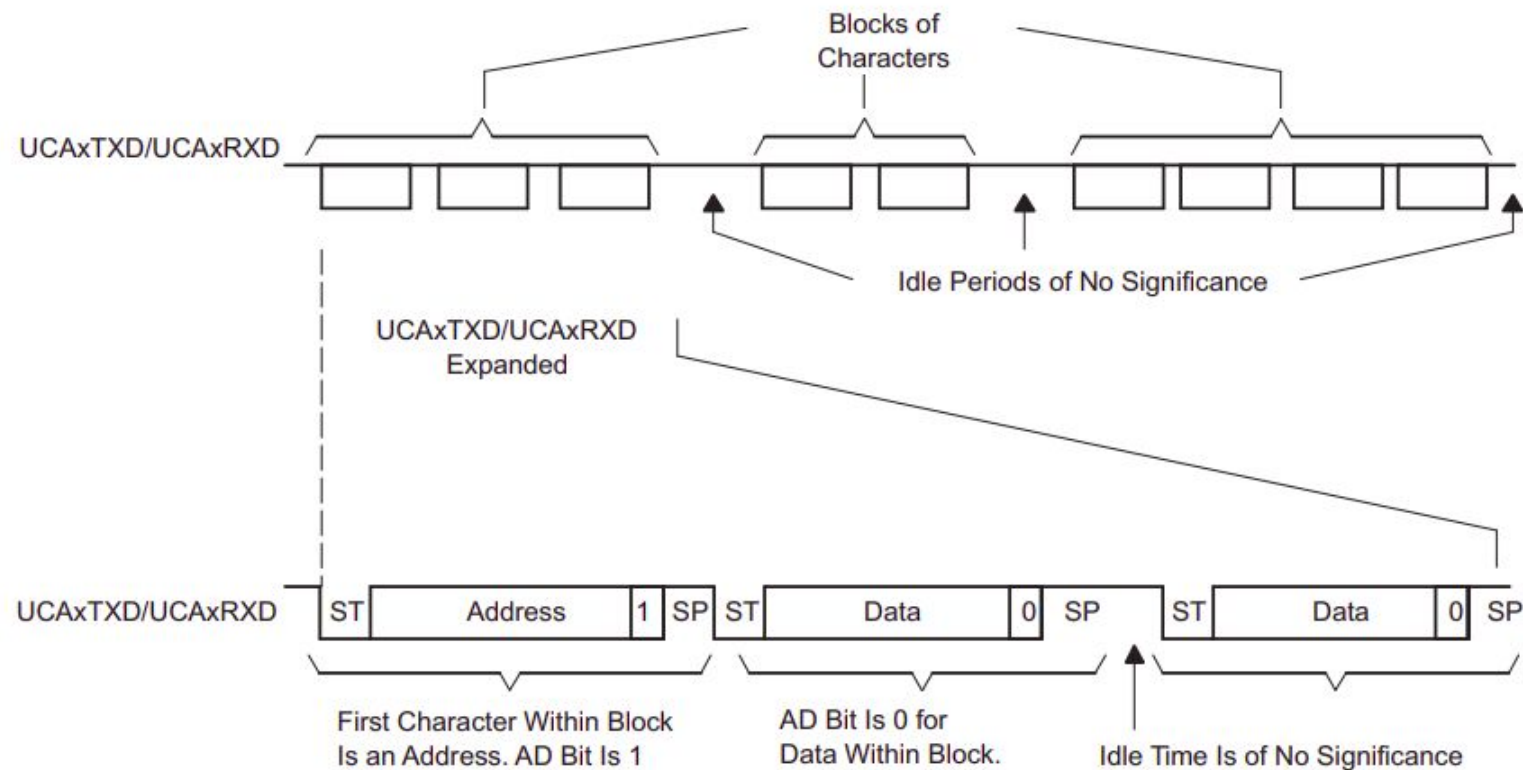
ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

UART

Pode-se acrescentar o envio de endereço, no caso de múltiplos transmissores e receptores

ST (0)	D0	D1	D2	D3	D4	D5	D6	D7	AD	PA	SP (1)	SP (1)
--------	----	----	----	----	----	----	----	----	----	----	--------	--------



UART

Pode-se enviar um bit de paridade, para o receptor conferir se houve erro na transmissão



Byte	Quantidade de 1s	Byte + bit de paridade	
		Par	Ímpar
00000000	0	0 00000000	1 00000000
01010001	3	1 01010001	0 01010001
01101001	4	0 01101001	1 01101001
10111111	7	1 10111111	0 10111111

UART

Pode-se enviar um bit de paridade, para o receptor conferir se houve erro na transmissão

ST (0)	D0	D1	D2	D3	D4	D5	D6	D7	AD	PA	SP (1)	SP (1)
--------	----	----	----	----	----	----	----	----	----	----	--------	--------

Byte	Quantidade de 1s	Byte + bit de paridade	
		Par	Ímpar
00000000	0	0 00000000	1 00000000
01010001			0 01010001
01101001			01101001
10111111			10111111

Se o receptor recebe os bits 100000000, e a paridade é par, ele sabe que houve um erro de transmissão, pois a quantidade de 1s recebida foi ímpar

UART



Pode-se enviar um segundo bit de STOP, para sistemas mais lentos não perderem o sincronismo

UART no RPi

3v3 Power	1			2	5v Power
GPIO 2 (I2C1 SDA)	3			4	5v Power
GPIO 3 (I2C1 SCL)	5			6	Ground
GPIO 4 (GPCLK0)	7			8	GPIO 14 (TXD / Transmit)
Ground	9			10	GPIO 15 (RXD / Receive)
GPIO 17	11			12	GPIO 18 (PCM CLK)
GPIO 27	13			14	Ground
GPIO 22	15			16	GPIO 23
3v3 Power	17			18	GPIO 24
GPIO 10 (SPI0 MOSI)	19			20	Ground
GPIO 9 (SPI0 MISO)	21			22	GPIO 25
GPIO 11 (SPI0 SCLK)	23			24	GPIO 8 (SPI0 CE0)
Ground	25			26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27			28	GPIO 1 (EEPROM SCL)
GPIO 5	29			30	Ground
GPIO 6	31			32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33			34	Ground
GPIO 19 (PCM FS)	35			36	GPIO 16
GPIO 26	37			38	GPIO 20 (PCM DIN)
Ground	39			40	GPIO 21 (PCM DOUT)

UART no RPi

O Raspbian utiliza a porta serial assíncrona (UART) para acesso remoto ao terminal (bash)*

Para utilizar a UART para outros propósitos, você deve desabilitar este acesso remoto, da seguinte maneira:

*[http://elinux.org/RPi_Serial_Connection#Connection to a microcontroller or other peripheral](http://elinux.org/RPi_Serial_Connection#Connection_to_a_microcontroller_or_other_peripheral)

UART no RPi

1 - Execute

```
$ sudo raspi-config
```

vá em Advanced options -> Serial, e desabilite o acesso:

Would you like a login shell to be accessible over serial? No.

Quando sair deste programa, não o deixe dar reiniciar (reboot) o sistema.

UART no RPi

2 - Execute

```
$ grep "enable_uart" /boot/config.txt
```

e verifique se este arquivo contém a linha "enable_uart=1"

Se o arquivo conter a linha "enable_uart=0", abra o arquivo:

```
$ sudo nano /boot/config.txt
```

e troque "enable_uart=0" por "enable_uart=1".

UART no RPi

3 - Reinicie o Raspberry Pi:

```
$ sudo reboot
```

4 - Se você estiver usando o Raspberry Pi 3 ou 4, o arquivo de acesso à porta serial não é `"/dev/ttyAMA0"`, e sim `"/dev/ttyS0"`, pois `"/dev/ttyAMA0"` é a porta serial usada para comunicar com o módulo Bluetooth da placa. Assim, em todos os exemplos desta aula, troque as definições

```
#define TTY /dev/ttyAMA0
```

por

```
#define TTY /dev/ttyS0
```

Hardware para exemplos UART

