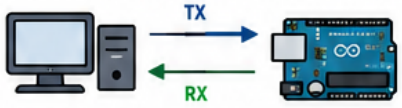
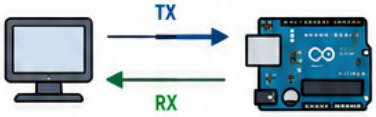
LA LIAISON SÉRIE RS232 / UART

Une liaison série permet à deux équipements d'échanger des informations numériques bit par bit sur un nombre réduit de fils.



1 PRINCIPE

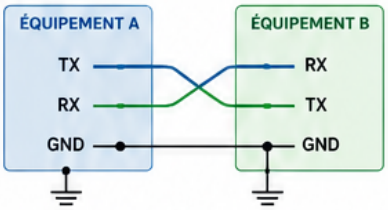
- Communication numérique bit par bit.
- Exemples :
  - Ordinateur ↔ Arduino
  - Arduino ↔ Capteur
  - Arduino ↔ Arduino



2 CÂBLAGE MINIMAL

Une liaison série nécessite au minimum 3 fils.

Fil	Fonction
GND	Masse commune (0 V)
TX	Transmission des données
RX	Réception des données

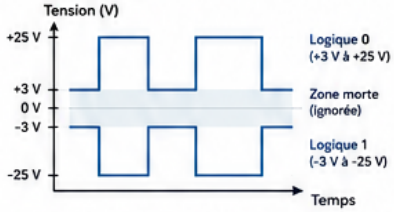


Le TX d'un équipement doit être connecté au RX de l'autre, et inversement. Les masses (GND) doivent être communes.

3 RS232 ET UART

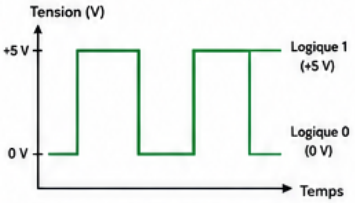
RS232 (ordinateurs)	
Niveau logique	Tension
0	+3 V à +25 V
1	-3 V à -25 V

Les tensions entre -3 V et +3 V sont ignorées afin de limiter les erreurs dues aux parasites.



UART (microcontrôleurs)	
Niveau logique	Tension
0	0 V
1	+5 V

Le protocole est identique à la RS232 mais les tensions sont adaptées aux microcontrôleurs.



4 STRUCTURE D'UNE TRAME SÉRIE (ASYNCHRONE)

Aucune horloge commune : l'émetteur et le récepteur doivent avoir la même vitesse (même nombre de bauds).



Exemple de trame :
Rouge : bit START (0)
Bleu : bits de DONNÉES
Vert : bit STOP (1)

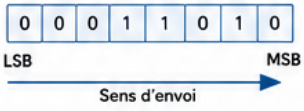
- Le bit START (0) indique le début de transmission.
- Le bit STOP (1) indique la fin de transmission.

5 TRANSMISSION DES DONNÉES

Les caractères sont codés en ASCII.
Exemple :

Caractère	Décimal	Binaire (8 bits)
P	80	01010000

Transmission UART : du poids faible (LSB) vers le poids fort (MSB)



6 VITESSE DE TRANSMISSION

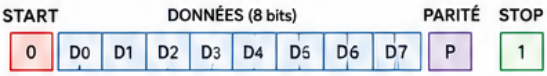
La vitesse de transmission s'exprime en bauds (bit/s). L'émetteur et le récepteur doivent être configurés à la même vitesse.

Valeurs courantes					
1200 bauds	2400 bauds	4800 bauds	9600 bauds	19200 bauds	115200 bauds

Exemple : 9600 bauds = 9600 bits transmis par seconde.

7 GESTION DES ERREURS : BIT DE PARITÉ

On peut ajouter un bit de parité après les données et avant le bit STOP. Son rôle : détecter certaines erreurs de transmission en vérifiant la cohérence des données reçues.



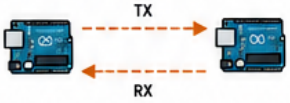
Modes :
- Parité paire : le nombre de "1" (données + parité) est pair.
- Parité impaire : le nombre de "1" (données + parité) est impair.

Si la parité est incorrecte à la réception → erreur détectée.

8 FULL-DUPLEX ET HALF-DUPLEX

Half-Duplex

Un seul sens de communication à la fois.
Exemple : émettre puis recevoir.



Full-Duplex

Émission et réception simultanées.
La liaison série UART est généralement Full-Duplex.



À RETENIR

- 3 fils minimum : TX, RX et GND.
- TX ⟷ RX obligatoirement croisés.
- Trame : START → DONNÉES → STOP.
- Communication asynchrone (pas d'horloge commune).
- Bits envoyés du LSB vers le MSB. (LSB → MSB)
- Vitesse exprimée en bauds.
- UART : version microcontrôleur de la liaison série.

COMMENT L'INFORMATION CIRCULE ?

SUPPORTS DE COMMUNICATION, DONNÉES NUMÉRIQUES ET PROTOCOLES

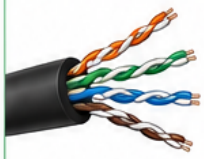
TERMINALE STI2D SIN

1. DES SUPPORTS DIFFÉRENTS... MAIS TOUJOURS DES INFORMATIONS NUMÉRIQUES !

Pour communiquer, les systèmes utilisent différents supports physiques.

Mais dans tous les cas, ce sont des informations numériques (des 0 et des 1) qui circulent.

FIL DE CUIVRE (CÂBLE ÉLECTRIQUE)

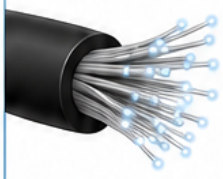


- Les informations sont transmises sous forme de signaux électriques.
- Exemples : câble téléphonique, Ethernet RJ45, USB, UART...

Exemple de signal électrique (niveau logique)



FIBRE OPTIQUE (LUMIÈRE)



- Les informations sont transmises sous forme de signaux lumineux.
- Exemples : réseau fibre, Internet très haut débit...

Exemple de signal lumineux (présence ou absence de lumière)

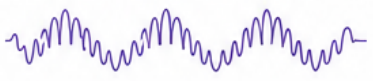


ONDES ÉLECTROMAGNÉTIQUES (SANS FIL)



- Les informations sont transmises par des ondes radio dans l'air ou dans l'espace.
- Exemples : Wi-Fi, Bluetooth, 4G/5G, LoRa, GPS...

Exemple de signal radio (variation d'amplitude / fréquence)



Quelle que soit la technologie, l'information est **codée en binaire (0 et 1)**.

Exemple de données numériques :

1	0	1	1	0	1	0	1	1	1	0	...

Ces données pourront ensuite être interprétées par le récepteur (texte, image, commande, mesure...).

2. DES PROTOCOLES POUR ORGANISER LES ÉCHANGES

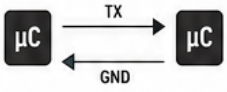
Pour que les équipements compremnent entre eux, ils suivent des règles communes : ce sont les **protocoles**.

Un protocole définit le **format** des données, l'ordre des échanges, les contrôles d'erreurs, etc.

EXEMPLES DE PROTOCOLES

UART

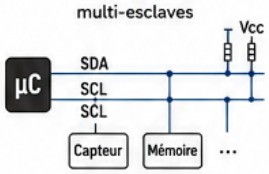
Communication série asynchrone point à point



Simple, utilisé pour la console, le GPS, le Bluetooth...

I2C

Bus série synchrone multi-maîtres multi-esclaves



Utilisé pour les capteurs, mémoires, horloges...

SPI

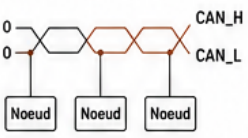
Bus série synchrone maître / esclave



Rapide, utilisé pour les écrans, cartes SD, mémoires...

CAN

Bus multi-maîtres différentiel robuste



Utilisé dans l'automobile, l'industrie, la robotique...

3. LA STRUCTURE GÉNÉRALE D'UN PROTOCOLE

Quel que soit le protocole, un échange suit toujours une logique générale.



Un équipement souhaite prendre la parole sur le bus. Il vérifie que le canal est libre (ou arbitre).

Il envoie un signal pour indiquer qu'il va émettre. Les autres savent qu'il doit écouter.

Les données sont envoyées selon un format défini par le protocole (trame, bits, adresses, contrôle...).

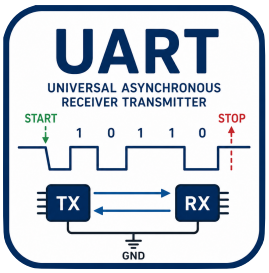
Le destinataire accuse réception (ACK) ou signale une erreur (NACK).

L'émetteur libère le bus. Les autres équipements peuvent à leur tour communiquer.

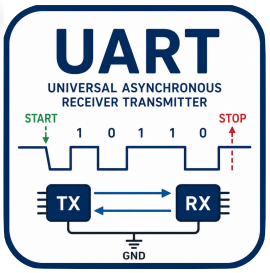
À RETENIR

- Les supports peuvent être électriques (cuivre), lumineux (fibre) ou radio (ondes).
- Dans tous les cas, ce sont des données numériques (0 et 1) qui circulent.
- Les **protocoles** définissent les régles du dialogue entre équipements.
- Un échange suit toujours une logique : demander à parler, parler, envoyer, recevoir, terminer.





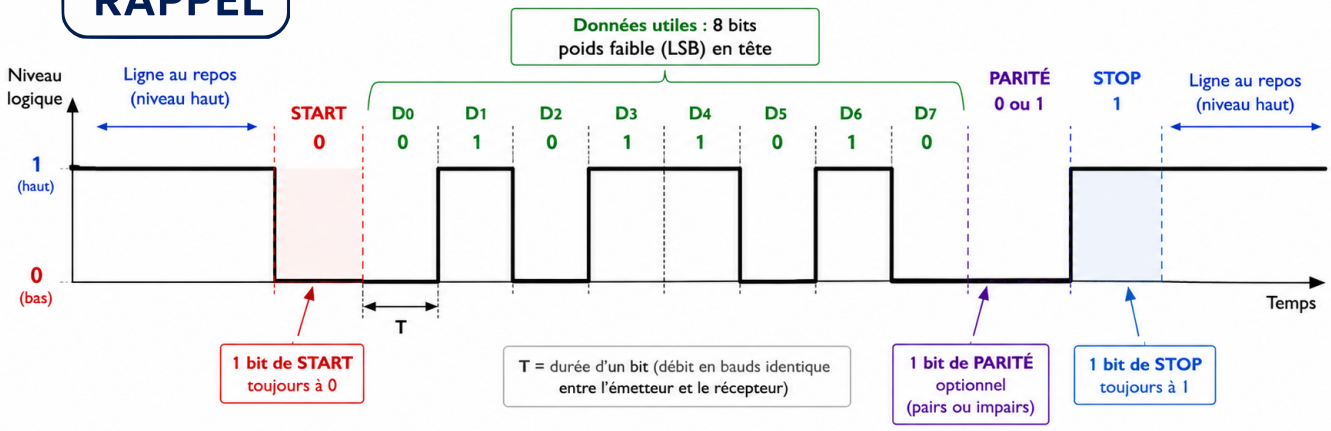






TRAME UART (8N1) – TRANSMISSION ASYNCHRONE

8 bits de données, sans parité (N), 1 bit de stop



En résumé :

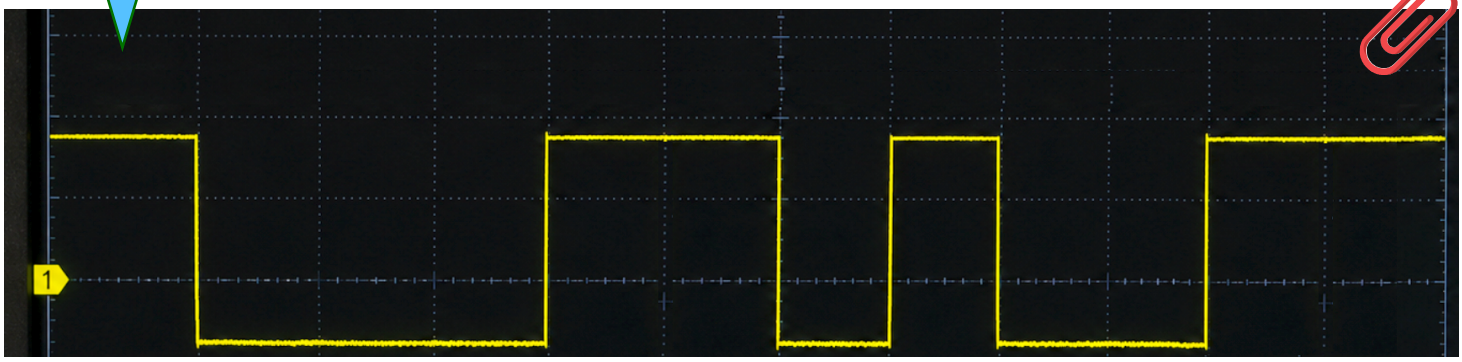
- La ligne est au repos à 1 (niveau haut).
- Le bit START = 0 indique le début de la trame.
- Les 8 bits de données sont envoyés du poids faible (LSB) vers le poids fort (MSB).
- Le bit de PARITÉ est optionnel (0 ou 1).
- Le bit de STOP = 1 indique la fin de la trame.
- Après le STOP, la ligne reste à 1 en attente d'une nouvelle trame.

Exemple de trame (valeurs ci-dessus) :

START	D0	D1	D2	D3	D4	D5	D6	D7	P	STOP
0	0	1	0	1	1	0	1	0	0	1



Le chronogramme suivant est observée sur un oscilloscope. Il s'agit d'une transmission avec le protocole UART (8E1) d'un caractère codé en code ASCII



Sensibilité horizontale : 500ms/div
Sensibilité verticale : 2v/div

Combien de bits ont été transmis en tout dans cette trame ? .....

Combien de bits utiles ont été transmis ? .....

Quelle est la tension correspondant à un 1 logique ? .....

Quelle est la valeur binaire de l'octet transmis dans la trame ? .....

En observant le document du codages ASCII des caractères, combien de caractères peuvent être codés ?

.....

Bits b_4 b_3 b_2 b_1	Column → / Row ↓	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FC	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Combien de bits sont nécessaires pour coder chaque caractères du code ASCII ? .....

Par rapport à l'octet transmis dans la trame quels sont les 7 bits du caractères ? .....

A quel caractère correspondent ils ? .....

Codez 2 caractères en code ASCII.

1er caractère : .....

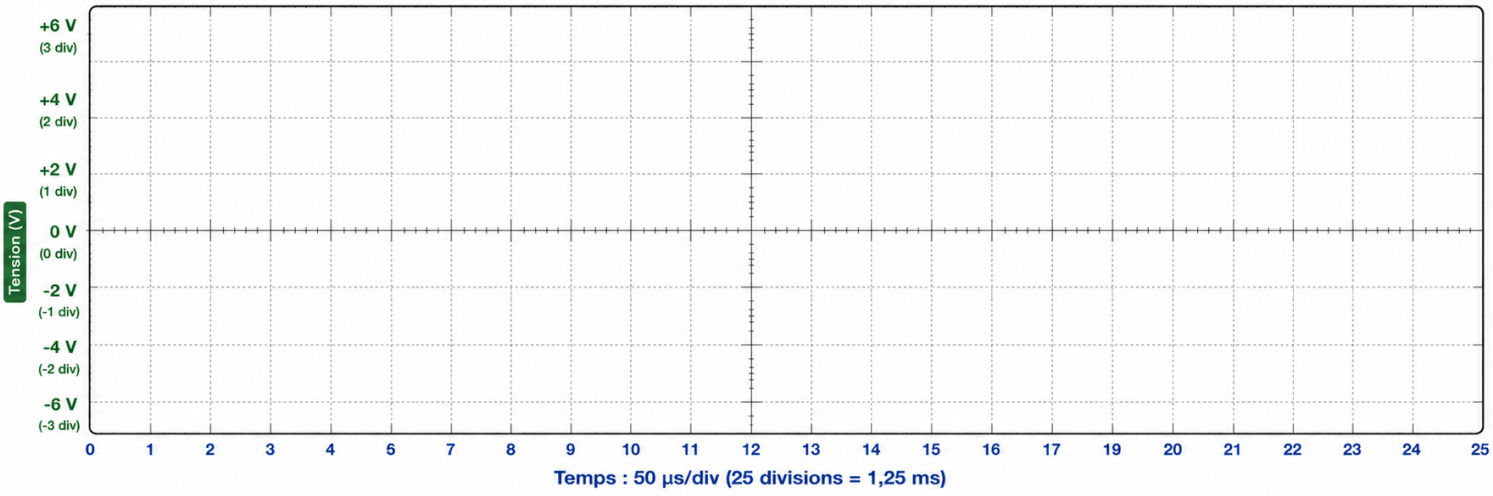
2eme caractère : .....

TRAME UART – 9600 bauds – 8E1

Débit : 9600 bauds
Durée d'un bit : 104,17 µs

Format : 8E1
(8 bits de données, parité paire, 1 bit de stop)
LSB (D0) transmis en premier

Base de temps : 50 µs/div
(1 division = 50 µs)
25 divisions horizontales



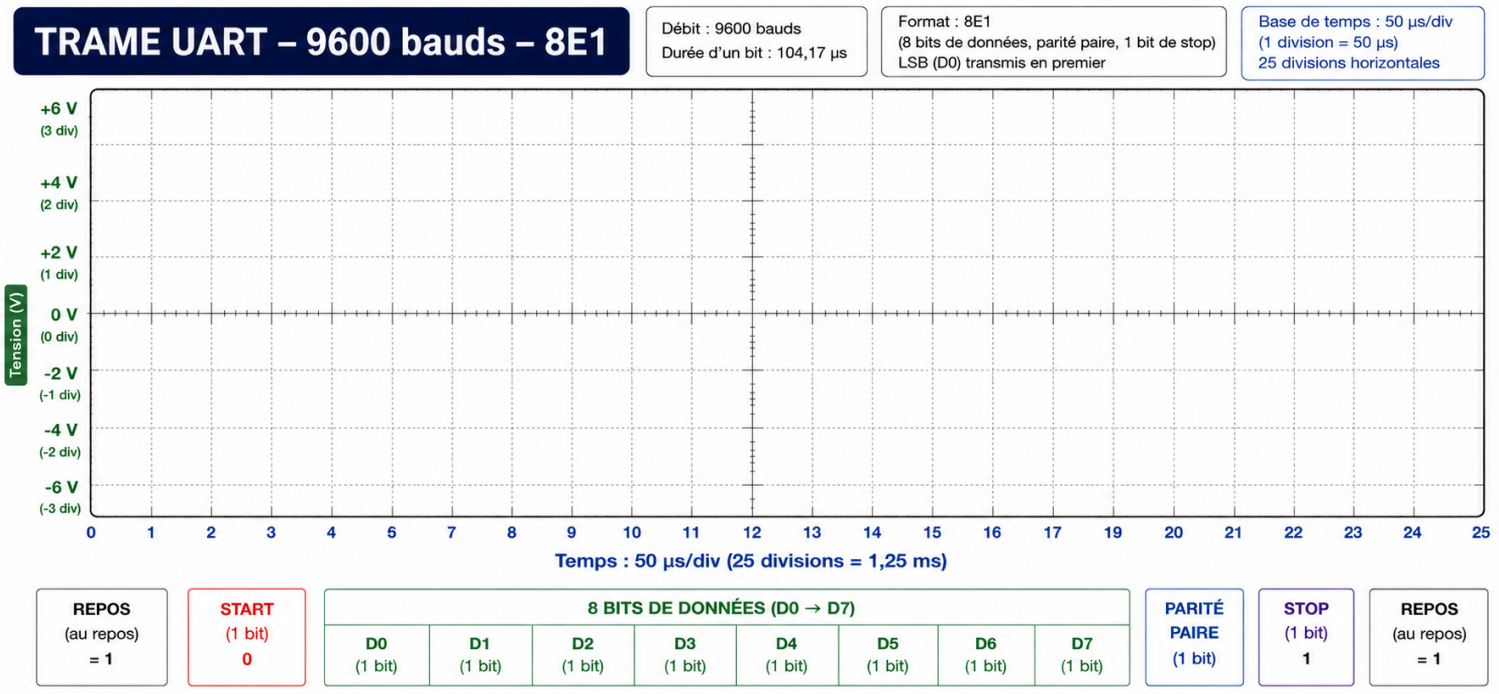
REPOS (au repos) = 1	START (1 bit) 0	8 BITS DE DONNÉES (D0 → D7)								PARITÉ PAIRE (1 bit)	STOP (1 bit) 1	REPOS (au repos) = 1
		D0 (1 bit)	D1 (1 bit)	D2 (1 bit)	D3 (1 bit)	D4 (1 bit)	D5 (1 bit)	D6 (1 bit)	D7 (1 bit)			

Consigne : Tracer ci-dessus la trame UART correspondante en respectant les niveaux logiques (0 V / +V) et la durée d'un bit.

NOM : Groupe SIN :

CARACTERE : ..... **Code ASCII :** ..... **Code Trame :** .....

Tracez la trame et décodez la BIT par BIT

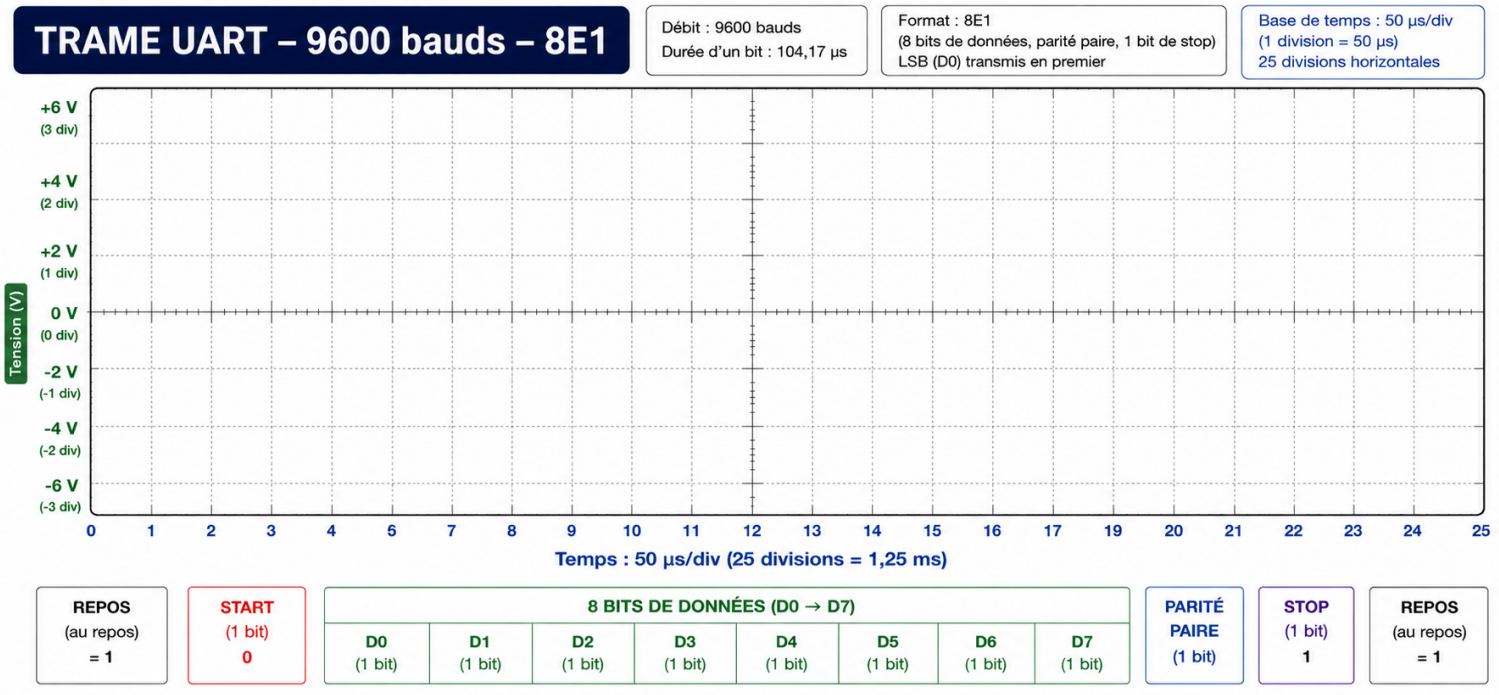


Consigne : Tracer ci-dessus la trame UART correspondante en respectant les niveaux logiques (0 V / +V) et la durée d'un bit.

CARACTERE : ..... **Code ASCII :** ..... **Code Trame :** .....

CARACTERE : ..... **Code ASCII :** ..... **Code Trame :** .....

Tracez la trame et décodez la BIT par BIT



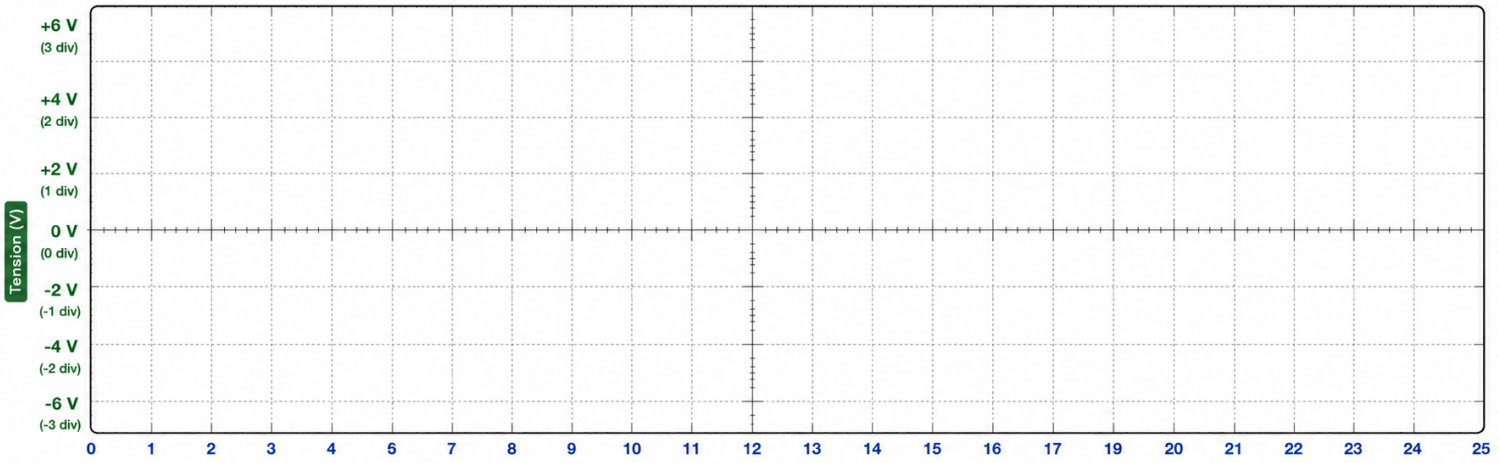
Consigne : Tracer ci-dessus la trame UART correspondante en respectant les niveaux logiques (0 V / +V) et la durée d'un bit.

<u>EXERCICE 3 :</u> tapez le code suivant dans vittascience, observez la sortie D7 à l'oscilloscope et complétez le tableau associé

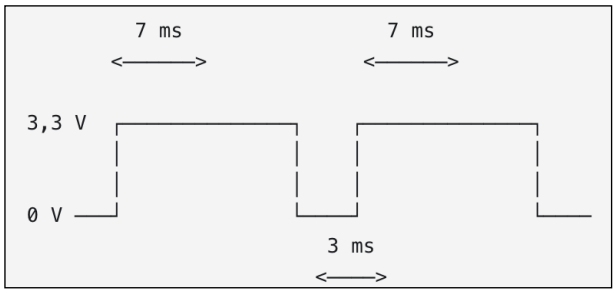
```
from machine import Pin
import time

sortie = Pin("D7", Pin.OUT)

while True:
    sortie.value(1)
    time.sleep_ms(5)

    sortie.value(0)
    time.sleep_ms(5)
```

Grandeur	Mesure
Tension état haut	V
Tension état bas	V
Durée état haut	.....ms
Durée état bas	.....ms

<u>EXERCICE 4 :</u> tracez le signal observé en précisant la tension et le temps de chaque état du signal



<u>EXERCICE 5 :</u> modifiez le programme pour observer le signal ci-dessous et complétez les tableaux



Durée état haut	..... ms
Durée état bas	..... ms

<u>PROGRAMME</u> → **<u>SIGNAL OBSERVÉ</u>**

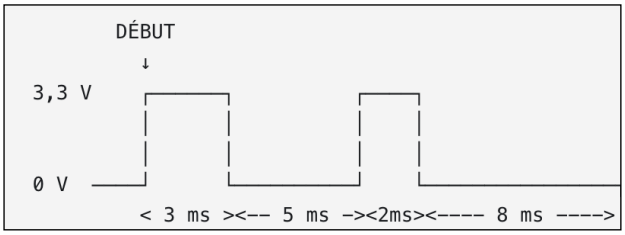
`sortie.value(.....)` → `tension = ..... V`

`time.sleep_ms(.....)` → `durée = ..... ms`

`sortie.value(.....)` → `tension = ..... V`

`time.sleep_ms(.....)` → `durée = ..... ms`

<u>EXERCICE 6 :</u> modifiez le programme pour observer le signal ci-contre





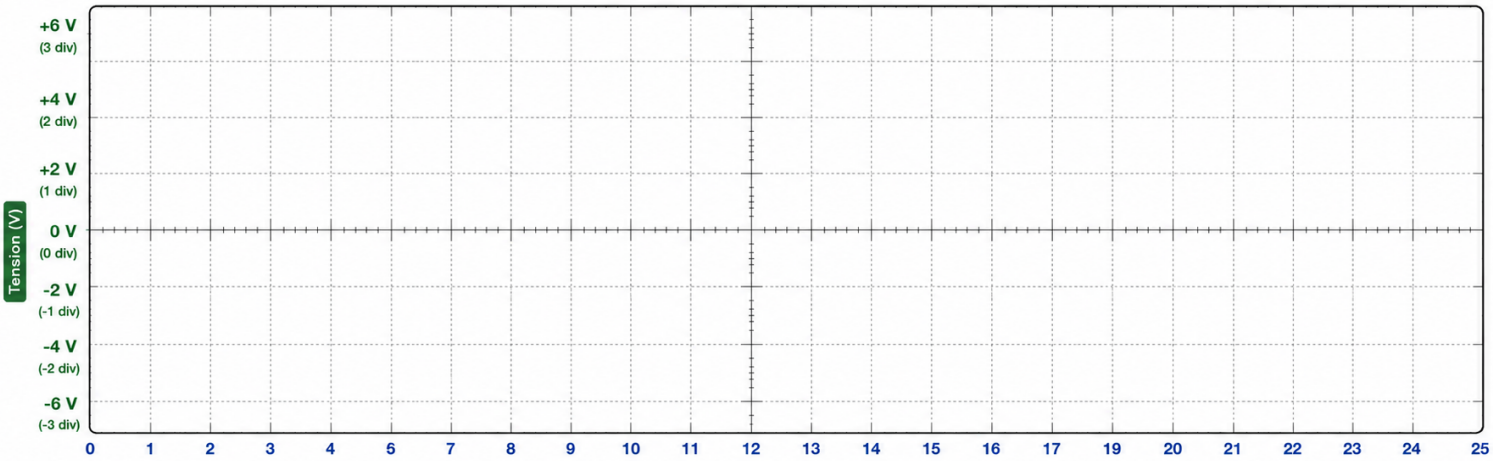
OBSERVATION DE TRAMES UART À L'OSCILLOSCOPE

Les trames observées ne contiennent pas de BIT de parité

<u>EXERCICE 1 :</u> tapez le code suivant dans Vittascience, tracez la trame et décodez la BIT par BIT

```
from machine import UART
import time

uart = UART(2, 1200)

while True:
    uart.write(b'…')
    time.sleep ms(20)
```

<u>À la place des pointillés notez la lettre défini par votre professeur.</u>



<u>EXERCICE 2 :</u> tapez 2 lettres donnez par votre professeur (à la place des pointillés), tracez la trame et décodez la BIT par BIT

