Institut Supérieur des Études Technologiques de Bizerte

AU : 2023-2024 | **L3-S5 : Dép. GE (ElnI)**
DS | Électronique de commande | **Enseignant : A. Mhamdi**
Janvier 2024 | **Durée : 1h½**

Ce document comporte **8** pages numérotées de **1/8** à **8/8**. Dès qu'il vous est remis, assurez-vous qu'il est complet. Les **3** exercices sont indépendants et peuvent être traités dans l'ordre qui vous convient.

Les règles suivantes s'appliquent :

❶ **Aucun document** n'est autorisé.

❷ **L'usage** de tout matériel électronique, sauf calculatrice, est strictement interdit.

❸ **La rigueur** de la rédaction entrera pour une part importante dans la notation.

❹ **Arrondissez** chaque nombre au millième près *(i.e., millième le plus proche)*.

Exercice N°1

⌛ 20mn | (5 points)

Répondez aux questions suivantes :

(a) (1 point) En quoi diffère le codage NRZ du NRZI en terme de transition de signal ?

> NRZ utilise des niveaux de signal constants pour représenter les bits, tandis que NRZI utilise des transitions de signal pour représenter les bits logiques '1' et l'absence de transition pour représenter les bits logiques '0'.

(b) (1 point) Comment le codage AMI permet-il de transmettre des données binaires tout en maintenant une synchronisation du signal ?

> Le codage AMI maintient la synchronisation du signal en utilisant les transitions de polarité pour représenter les bits logiques '1' tout en évitant les transitions pour les bits logiques '0'.

(c) (1 point) Quelle est la principale caractéristique du codage Manchester en terme de transition de signal ?

> Il utilise une transition de signal au milieu de chaque période pour représenter chaque bit de données.

(d) (1 point) Quelle est la principale différence entre le codage Manchester et le codage Manchester différentiel ?

> Le codage Manchester utilise les transitions de signal pour représenter directement les bits de données, tandis que le codage Manchester différentiel utilise les transitions de signal pour représenter les changements de bits.

(e) (1 point) On se propose d'envoyer un signal tétravalent[1] à une vitesse de 1000 bauds. Recherchez le débit en bauds et le débit binaire. Nous rappelons la formule :

$$\mathcal{R} = \frac{\mathcal{D}}{\log_2(\mathcal{V})} \tag{1}$$

> baud rate $\mathcal{R} = 1000\text{bauds}$ (2)
>
> bit rate $\mathcal{D} = 2000\text{bps}$ (3)

Exercice N°2 ⌛ 40mn | (5 points)

La chaîne de caractères suivante " :) " est transférée sur une liaison série. La communication est configurée de la façon suivante :

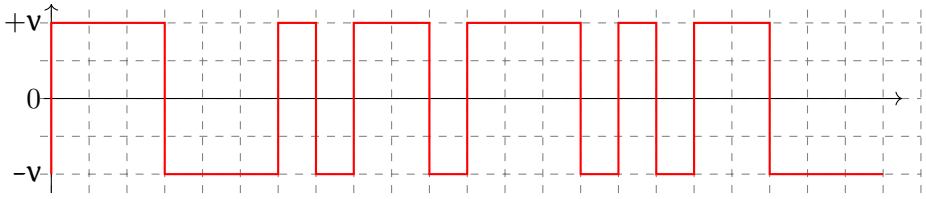
Stop 1 seul bit ;

Parité paire.

Convertissez chaque caractère en sa représentation binaire ASCII et codez la séquence de bits correspondante à la transmission série en utilisant :

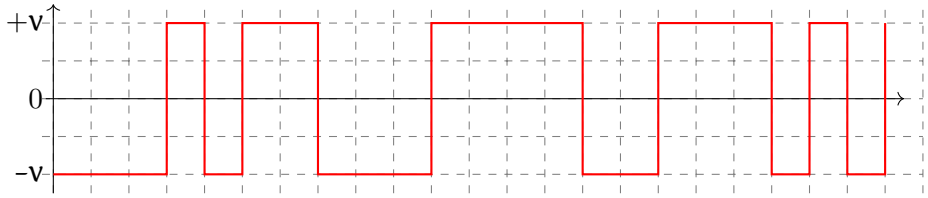
> La séquence binaire à envoyer est : $\underbrace{0}_{\text{Start}}\ \underbrace{00111010}_{:}\ \underbrace{0}_{\text{Parité}}\ \underbrace{1}_{\text{Stop}}\ \underbrace{0}_{\text{Start}}\ \underbrace{00101001}_{)}\ \underbrace{1}_{\text{Parité}}\ \underbrace{1}_{\text{Stop}}$

(a) (1 point) le codage NRZ ;

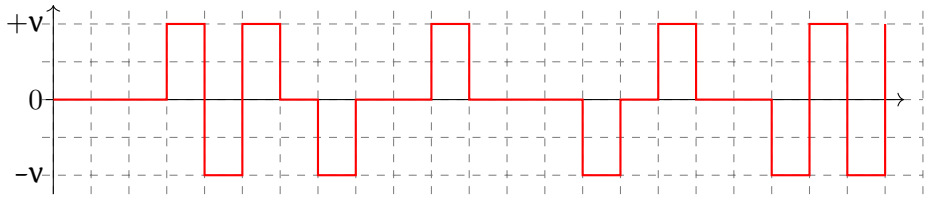


1. C'est un signal à 4 états, c.-à-d. 2 bits par temps élémentaire.

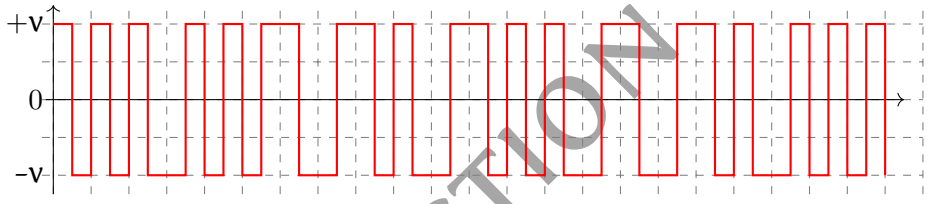
(b) (1 point) le codage NRZI *(en supposant que le niveau précédent était -v)* ;



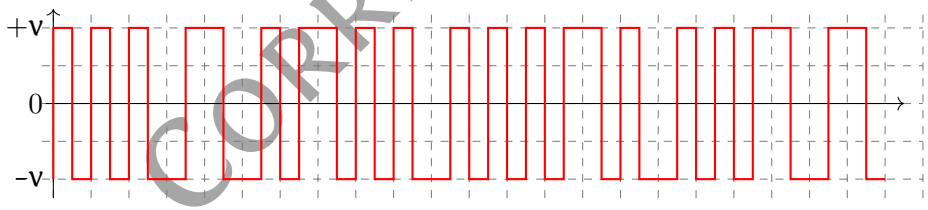
(c) (1 point) le codage bipolaire « AMI » *(en supposant que le niveau précédent était -v)* ;



(d) (1 point) le codage Manchester ;



(e) (1 point) le codage Manchester différentiel *(en supposant que l'impulsion précédente était un front descendant)*.



Exercice N°3

⌛ 30mn | (10 points)

Les questions suivantes concernent l'usage du paquetage `turtlesim` de ROS2. Veuillez consulter FIG. 1, p. 7, FIG. 2 et FIG. 3, p. 8 avant d'y répondre.

(a) (1 point) Qu'est-ce que le paquetage `turtlesim` de ROS2 ?

> Le paquetage `turtlesim` est un simulateur de robot graphique inclus dans le framework ROS2. Il permet de contrôler un robot virtuel en forme de tortue dans un environnement 2D.

(b) (1 point) Comment lancer ce paquetage ?

Pour lancer le paquetage `turtlesim` de ROS2, nous pouvons exécuter la commande suivante dans un terminal :

```
ros2 run turtlesim turtlesim_node
```

Cela démarrera le simulateur `turtlesim` et affichera la fenêtre graphique de la tortue virtuelle.

(c) (2 points) Quels sont les nœuds et les topics importants ?

Le paquetage `turtlesim` de ROS2 comprend principalement deux nœuds :

- `turtlesim_node` représente la tortue virtuelle et gère son état, ses mouvements et ses interactions avec l'environnement ;
- `turtle_teleop_key` permet de contrôler la tortue à l'aide du clavier en utilisant les touches fléchées.

Les topics importants du paquetage `turtlesim` incluent :

- `/turtle1/cmd_vel` permet de publier les commandes de mouvement pour la tortue virtuelle ;
- `/turtle1/pose` fournit la position et l'orientation actuelles de la tortue virtuelle.

(d) (1 point) Comment contrôler la tortue virtuelle ?

Nous pouvons utiliser le nœud `turtle_teleop_key`. Il faut garder le simulateur `turtlesim` en cours d'exécution, puis dans un autre terminal, nous exécutons la commande suivante :

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args
--remap cmd_vel:=/turtle1/cmd_vel
```

On peut aussi utiliser l'exécutable `turtle_teleop_key` du paquetage `turtlesim` comme suit :

```
ros2 run turtlesim turtle_teleop_key
```

(e) (2 points) Comment savoir quel type de message publié ?

Afin de pouvoir afficher la liste des topics avec le type de messages qu'ils supportent, il faut taper la commande :

```
ros2 topic list -t
```

Dans le cas de `turtlesim`, les messages les plus couramment utilisés sont :

- `geometry_msgs/msg/Twist` est utilisé pour contrôler le déplacement de la tortue virtuelle. Il contient des informations sur la vitesse linéaire et angulaire de la tortue. Nous pouvons publier un message

de ce type pour déplacer la tortue en avant, en arrière ou pour la faire pivoter.

`turtlesim/msg/Color` est utilisé pour spécifier la couleur du tracé effectué par la tortue virtuelle. Il contient des informations sur les composantes RVB (rouge, vert, bleu) de la couleur.

(f) (2 points) Comment tracer un cercle avec `turtlesim`?

Nous utilisons `ros2 topic pub` pour publier un message sur le topic `/turtle1/cmd_vel`, qui est utilisé pour contrôler la vitesse et la direction de la tortue. Le message est de type `geometry_msgs/Twist` :

```
ros2 topic pub -r 1 /turtle1/cmd_vel geometry_msgs/msg/Twist ``{linear:
{x: 1.0}, angular: {z: 1.0}}''
```

Dans cet exemple, nous définissons une vitesse angulaire « angular » de 0.8 autour de l'axe z, ce qui fera tourner la tortue à une vitesse de rotation d'un 0.8 radian par seconde. Nous ajustons aussi la vitesse linéaire « linear » à 1 par exemple afin de créer finalement un cercle.

(g) (1 point) Comment vérifier si les nœuds sont correctement connectés dans ROS2, et visualiser les flux de données entre les différents composants?

`rqt_graph` est un outil graphique interactif qui facilite la compréhension de l'architecture d'un système ROS en affichant les nœuds et les connexions sous forme de graphique. La commande qui permet de lancer cet outil est : `rqt_graph`

DOCUMENT ANNEXE

TABLE 1 – Table des codes ASCII et leur correspondance (0→127).

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL	16	10	DLE	32	20	␣	48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
1	01	SOH	17	11	DC1	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	02	STX	18	12	DC2	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	03	ETX	19	13	DC3	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	04	EOT	20	14	DC4	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	05	ENQ	21	15	NAK	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	06	ACK	22	16	SYN	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	07	BEL	23	17	ETB	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	08	BS	24	18	CAN	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	09	HT	25	19	EM	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	0A	LF	26	1A	SUB	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	0B	VT	27	1B	ESC	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	0C	FF	28	1C	FS	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
13	0D	CR	29	1D	GS	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	0E	SO	30	1E	RS	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	0F	SI	31	1F	US	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

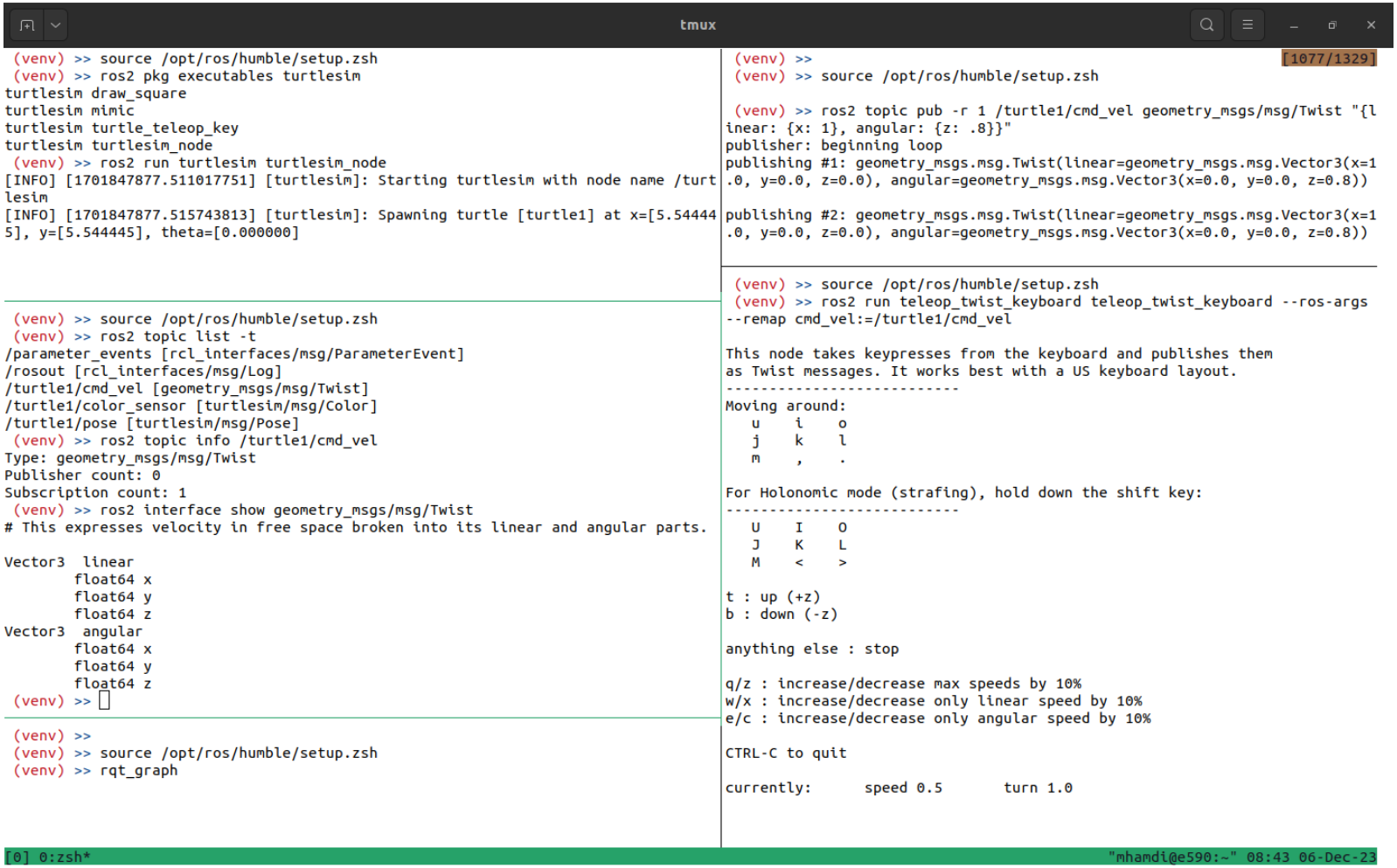


FIG. 1. Terminal – ROS2

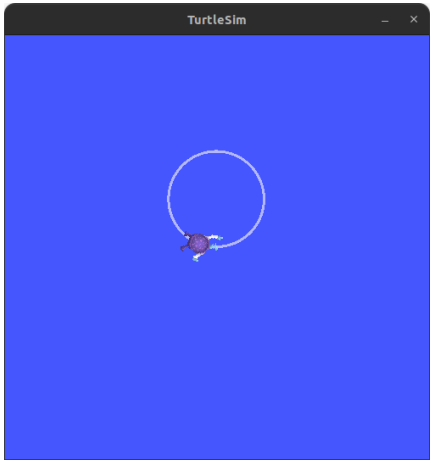


FIG. 2. Nœud `turtlesim_node`

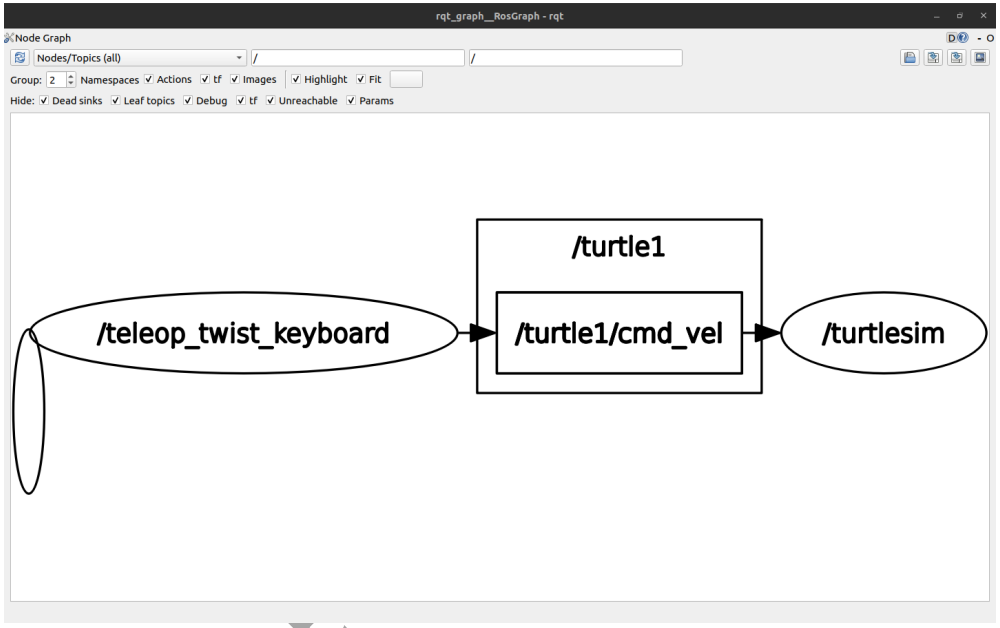


FIG. 3. Nœuds et « topics » en cours d'exécution