PROJET ESE LIDAR :

Datasheet : https://www.slamtec.com/en/Support#rplidar-a-series (à télécharger)

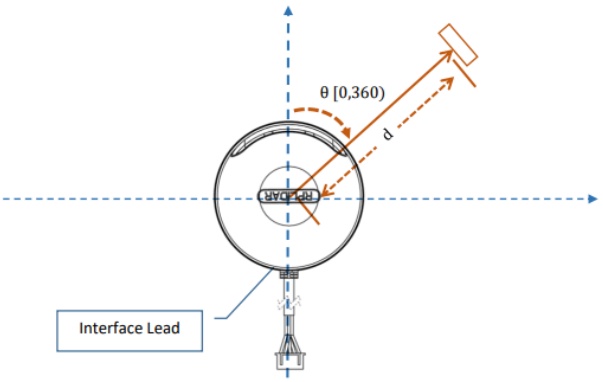

Figure 2-4 RPLIDAR Scanning Data Coordinate System Definition

Communication interface

The RPLIDAR A2 uses separate 5V DC power for powering the range scanner core and the motor system. And the standard RPLIDAR A2 uses XH2.54-5P male socket. Detailed interface definition is shown in the following figure:

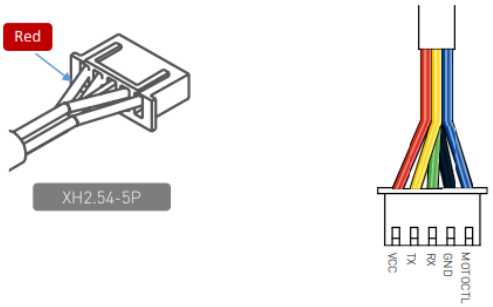

Figure 2-5 RPLIDAR Power Interface Definition

Color	Signal Name	Type	Description	Min	Typical	Max
Red	VCC	Power	Total Power	4.9V	5V	5.5V
Yellow	TX	Output	Serial port output of the scanner core	0V	3.3V	3.5V
Green	RX	Input	Serial port input of the scanner core	0V	3.3V	3.5V
Black	GND	Power	GND	0V	0V	0V
Blue	MOTOCTL	Input	Scan motor /PWM Control Signal (active high, internal pull down)	0V	3.3V	5V

Figure 2-6 RPLIDAR External Interface Signal Definition

Le LIDAR communique en UART 3.3V

Pour contrôler le scan du moteur, regarder la p12-13 de la datasheet.

Pour démarrer le LIDAR, il faut lui dire, il ne le fait pas lui-même.
Quand on envoie une requête, le LIDAR répond, il ne faut pas le surcharger, il faut attendre qu'il réponde avant d'envoyer de nouvelles requêtes.

COMMUNICATION :

Voir p6 rplidar_protocole à télécharger sur le site au début du doc

All request packets sent by a host system share the following common format. Little endian byte order is used.

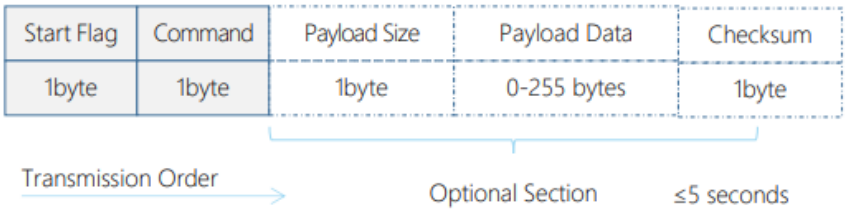

Figure 2-4 RPLIDAR Request Packets' Format

Request Name	Value	Payload	Response Mode	RPLIDAR Operation	Supported Firmware Version
STOP	0x25	N/A	No response	Exit the current state and enter the idle state	1.0
RESET	0x40	N/A		Reset(reboot) the RPLIDAR core	1.0
SCAN	0x20	N/A	Multiple response	Enter the scanning state	1.0
EXPRESS_SCAN	0x82	YES		Enter the scanning state and working at the highest speed	1.17
FORCE_SCAN	0x21	N/A		Enter the scanning state and force data output without checking rotation speed	1.0
GET_INFO	0x50	N/A	Single response	Send out the device info (e.g. serial number)	1.0
GET_HEALTH	0x52	N/A		Send out the device health info	1.0
GET_SAMPLERATE	0x59	N/A		Send out single sampling time	1.17
GET_LIDAR_CONF	0x84	YES		Get LIDAR configuration	1.24

Figure 4-1 The Available Requests of RPLIDAR

Requête à envoyer pour démarrer le scan en mode express :

A5	82	05	M	00	00	00	00	C

Avec C le checksum et M le mode (différent de 0 pour le mode extented, ou 0 pour mode legacy)

En mode legacy ça donne :

A5	82	05	00	00	00	00	00	22

J'ai essayé d'envoyer des trames mais le LIDAR ne répond pas, sur les conseilles de Césair, je vais essayer de brancher une pwm au pin MOTOCTL.

Update :

Dans la doc j'ai vu qu'on pouvait contrôler la vitesse de rotation du LIDAR avec le MOTOCTL, mais si on a juste besoin de contrôler le start et le stop du moteur, on peut seulement envoyer du high level (3.3V ici) au MOTOCTL. Pour stopper la rotation, envoyer du low level (0V).

Update :

Il faut obligatoirement une PWM pour faire fonctionner le LIDAR, j'ai donc créé un timer dans le code pour faire cette PWM sur la broche 2.5. Ainsi, le LIDAR tourne dès qu'il est alimenté + PWM.
Il faut maintenant récupérer les informations qu'il envoie et les traiter.

Pour récupérer les trames, il faut envoyer un start scan pour qu'il commence à scanner.
Cela doit être fait car même si le LIDAR tourne, la PWM ne contrôle que le moteur, donc le LIDAR ne scan encore rien.

Commande du scan normal :

Request Packet:

A5	20

Réponse attendue :

Response Descriptor:

A5	5A	05	00	00	40	81

Response Mode: **Multiple** Data Response Length: **5 bytes**

Format of the Data Response Packets:

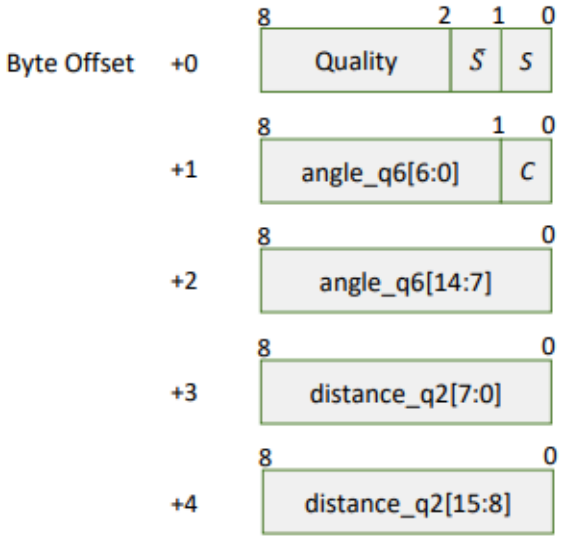

Figure 4-4 Format of a RPLIDAR Measurement Result Data Response Packet

Il faut coder cela.

Tableau des fonctions complété au fur et à mesure :

Fonction	Test valide ?	Utilise autre ft ?	RTOS ?	Commentaire
Lidar_Start_Scan	Oui	Oui	Non	/
Lidar_Stop_Scan	Oui	Oui	Non	/
Send_Lidar_Command	Oui pour no payload	Non	Non	Pas testée pour avec payload
Receive_UART	Non mais le code marche donc oui	Oui	Oui	/
Afficher_LCD	Non mais le code marche donc oui	Non mais lié à Receive	Oui	/
initTimer0	Oui	Non	Non	par Rayan et M
Init_UART0	Oui	Non	Non	/
Temporisation_ms	Oui	Non	Non	/

J'ai créé la fonction Lidar_Start_Scan qui marche très bien après vérification à l'oscillo. La fonction utilise la fonction Send_Lidar_Commande qui ducoup marche également bien pour le case sans payload.

J’observe bien les trames que le LIDAR envoie grâce à l’oscilloscope, maintenant il faut trouver comment les traiter.
Pour cela, comme le LIDAR envoie beaucoup de trames, il ne faut pas surcharger le processeur, donc peut-être baisser la vitesse du LIDAR.

Traitement des données du LIDAR :

Le LIDAR fonctionne en temps-réel, on doit capturer les trames qu’il envoie en temps réel, on va donc mettre en place un RTOS avec un thread qui récupère les trames envoyées et un autre qui les traite, transformant les données et les affichant.

```
196 void Receive_UART(void const * argument){
197
198   char chain[20];
199   while(1){
200
201     osSignalWait(0x0001,osWaitForever);                    //Attente de l'event 1 à 1
202     osSignalClear(ID_Receive,0x0001);                      //On remet l'event à 0
203     Lidar_Start_Scan();                                    //Start le scan
204
205
206     Driver_USART0.Receive(ValeurRecue,5);                  //la fonction remplira jusqu'à 150 cases
207     while (Driver_USART0.GetRxCount() <5);                 //on attend que 150 case soit pleine
208
209     osSignalSet(ID_LCD, 0x0001);                           //Réveil tâche afficher LCD
210
211   }
212
213 }
214
```

```
219 void Afficher_LCD(void const * argument){
220
221   unsigned short SetAngle = 0x0000;
222   float SetAngle_f;
223   unsigned short SetDistance = 0x0000;
224   float SetDistance_f;
225
226   char chain[30];                                          //tableau pour afficher
227
228   while(1){
229
230     osSignalWait(0x0001,osWaitForever);                    //Attente de l'event 1 à 1
231     osSignalClear(ID_LCD,0x0001);                          //On remet l'event 1 à 0
232
233     SetAngle = (ValeurRecue[1] >> 1) + (short)(ValeurRecue[2] << 7);        //on récupère les données de l'angle
234     SetAngle_f = (((float)SetAngle) / 64.0)*0.9;                            //d'après la doc, valeur angle = don
235
236     sprintf(chain,"angle = %04d",(short)SetAngle_f);       //Affichage
237     GLCD_DrawString(0,0,chain);                            //Affichage
238
239     SetDistance = (ValeurRecue[3]) + (short)(ValeurRecue[4] << 8);          //on récupère les données de la dist
240     SetDistance_f = (((float)SetDistance) / 4.0) / 1000;                    //d'après la doc, valeur distance =
241
242     sprintf(chain,"distance = %06f",SetDistance_f);        //Affichage
243     GLCD_DrawString(0,30,chain);                           //Affichage
244
245     osSignalSet(ID_Receive, 0x0001);                       //Réveil tâche receive
246
247   }
248
249 }
```

Ces fonctions complétées par le reste du programme semblent bien fonctionner, le LIDAR tourne et je vois bien l’affichage en temps réel des angles et distances.
Code complet au commit du jeudi 3 avril vers 16h50