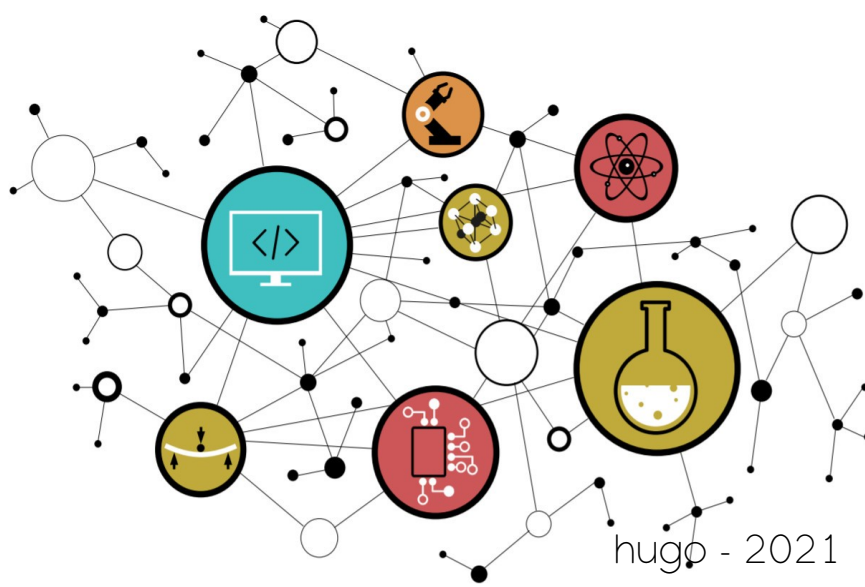


# TRAVAUX PRATIQUES

## MODULE AUDIO BLUETOOTH EXTERNE

---



## SOMMAIRE

### 5. MODULE AUDIO BLUETOOTH EXTERNE

5.1. Introduction : *Bluetooth*

5.2. Configuration du module Audio Bluetooth RN52

# MODULE AUDIO BLUETOOTH EXTERNE

## 5. MODULE AUDIO BLUETOOTH EXTERNE

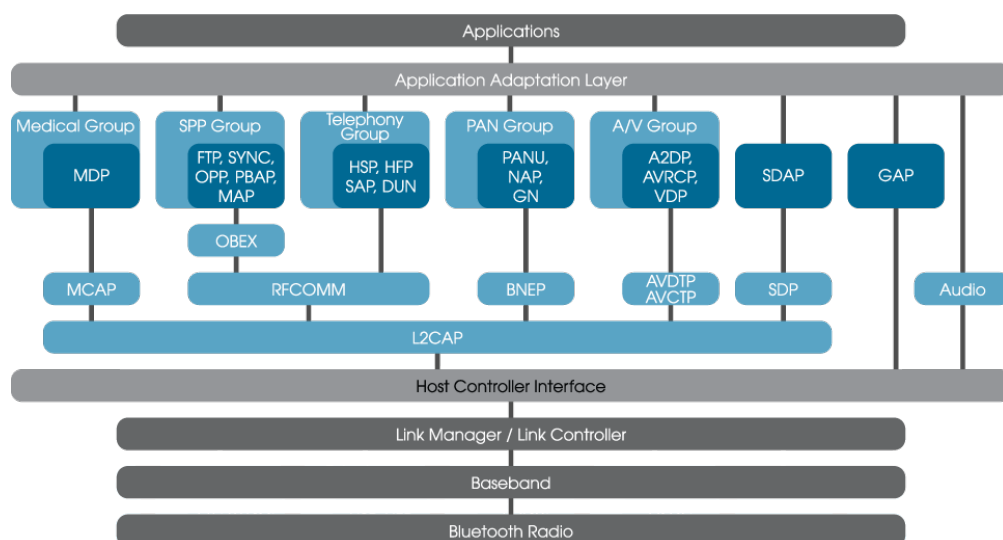
### 5.1. Introduction

Dans l'embarqué, la communication de machine à machine (M2M ou Machine to Machine) est un point souvent central dans la conception d'une application. Il s'agit notamment du cœur des systèmes dits nomades. Les solutions technologiques de communication les plus couramment rencontrées en embarqué à notre époque sont le WIFI, le Bluetooth, les technologies cellulaires (LTE, 4G, 3G et 2G), le NFC (Near Field Communication), LORA, Sigfox, etc. Aucune technologie n'est meilleure qu'une autre. Elles possèdent toutes des avantages et inconvénients. Elles répondent toutes à une famille de besoins parfois très différents.

Par exemple, diffuser un flux vidéo streaming en WIFI est techniquement et physiquement très différent d'une simple lecture de température puis échange toutes les minutes sur un appareillage de production dans une usine. Il existe donc des protocoles et solutions techniques adaptées à chaque besoin. Le tableau ci-dessous synthétise "approximativement" les périmètres d'actions des principales solutions technologiques actuelles du marché.

Protocol	Optimized for Battery Life	Nominal Range Limit	Typical Data Rate	Spectrum
Bluetooth		<10m	2Mbps	ISM 2.4GHz unlicensed
WIFI		<100m	>100Mbps	ISM 2.4GHz/5GHz unlicensed
LORAWAN		>10Km	<50Kbps	ISM 900MHz unlicensed
2G/3G		>30Km	<2Mbps	<i>Licensed cellular</i>
4G		>30Km	>100Mbps	<i>Licensed cellular</i>
NFC		<4cm	100Kbps	ISM 13.56MHz unlicensed

Le Bluetooth est par exemple une norme de communication permettant l'échange bidirectionnel de données sur de courte distance (<100m voire <10cm suivant la classe de fonctionnement) en utilisant des ondes radio dans la bande UHF (bande de fréquence autour de 2,4GHz). Comme la plupart des normes et protocoles de communication, le Bluetooth peut être représenté en couches protocolaires plus ou moins proches du monde physique (couche Radio). Nous parlons souvent de Stack (ou Pile) protocolaire en faisant référence à certaines bibliothèques logiciel. Dans l'exercice de la trame de TP, le module RN52 utilisé intègre et implémente le profil Audio A2DP (groupe A/V ou Audio/Vidéo), utilisant lui-même la couche liaison de multiplexage L2CAP, elle-même interfacée par les couches en bandes de bases et Radio.



Il existe sur le marché des modules de communication plus ou moins intégrés (emport potentiel de couches protocolaires ou stack Bluetooth) avec des échelles de coûts souvent liées à la richesse des fonctionnalités embarquées dans le module de communication (exemple du schéma fonctionnel ci-dessous). Dans le cadre de notre application, nous utiliserons une solution intégrée gérant déjà pleinement une partie du protocole Bluetooth désiré (profil Audio A2DP, couche liaison L2CAP, couche en bande de base et couche radio) et s'interfaçant par simple liaison série asynchrone et périphérique UART. Par exemple, le module Bluetooth RN52 de Microchip supporte déjà les couches protocolaires nécessaires à l'application (cf. schéma de droite ci-dessous). Nous n'aurons qu'à le configurer et le contrôler depuis le MCU par envoi de chaînes de caractères ASCII avec une communication par liaison série asynchrone via UART (cf. tableaux ci-dessous).



Nous pouvons par exemple observer ci-dessous un extrait de la documentation technique du module Bluetooth RN52 (cf. [mcu/tp/doc/datasheets](#)) présentant la synthèse de toutes les commandes de configuration et d'action supportées. Par exemple, si notre MCU envoie par UART les suites de caractères ASCII suivantes (cf. figure de gauche ci-dessus), le module RN52 réalisera les traitements demandés :

- **AV+** : incrémente le volume (Audio Volume +)
- **AT-** : rejoue la dernière piste Audio (Audio Track -)
- **SN, <string>** : change le nom du réseau Bluetooth créé par le module RN52 par <string>
- etc

## SET COMMANDS

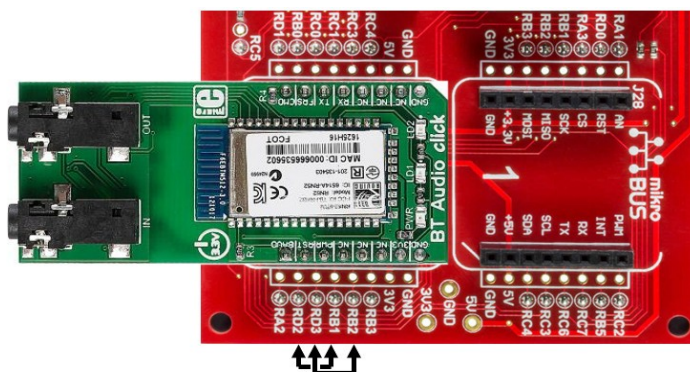
Command	Description
SI, <hex16>	Audio output routing
S-, <string>	Sets the normalized name
S^, <dec>	Automatic Shutdown on Idle
S%, <hex16>	Extended Features
SA, <0,1,2,4>	Authentication enable/disable
SC, <hex24>	Service class
SD, <hex8>	Discovery profile mask
SF, 1	Factory defaults
SK, <hex8>	Connection profile mask
SM, <hex32>	Microphone/LINEIN gain
SN, <string>	Device name
SS, <hex8>	Speaker Level
ST, <hex8>	Tone Level
STA, <dec>	Connection Delay
STP, <dec>	Pairing Timeout
SU, <hex8>	UART Baudrate

## ACTION COMMANDS

Command	Description
+	Toggle the local echo of RX characters in Command mode
@, <flag>	Toggle whether the module is discoverable
#, <0,1>	Accept/reject pairing
\$	Put the module into DFU mode
A, <telephone number>	Initial a voice call to <telephone number>
AD	Retrieve track metadata information
AR	Redial last dialed number
AV+	Increase the volume (AVRCP command)
AV-	Decrease the volume (AVRCP command)
AT+	Play the next track (AVRCP command)
AT-	Play the previous track (AVRCP command)
AP	Pause or start playback (AVRCP command)
B	Reconnect Bluetooth® profiles to the most recently paired and connected device
C	Accept an incoming voice call
E	Terminate an active call or reject an incoming call
F	Release all held calls
I@	Read GPIO configuration
I@, <hex16>	Set GPIO configuration
I&	Reads current GPIO levels for input
I&, <hex16>	Set GPIO levels for output
J	Accept waiting calls and release active calls
K, <hex8>	Kill the currently active connection
L	Accept waiting calls and hold active calls
M, <flag>	Toggle the on hold/mute function
N	Add held call
O	Connect two calls and disconnect the subscriber
P	Activate Voice Command
Q	Query the current connection status
R, 1	Reboot
U	Reset Paired Device List (PDL)
T	Retrieves caller ID information
X, <0,1>	Transfer call between HF and AG

### 5.2. Configuration du module Audio Bluetooth RN52

- Créer un projet MPLABX nommé *rn52\_test* dans le répertoire *disco/bsp/rn52/test/pjct*. Inclure les fichiers *bsp/rn52/test/main.c*, *bsp/common/delay\_200ms.c* et *rn52\_init.c*, *rn52\_cmd.c* présents dans le répertoire *bsp/rn52/src/*. S'assurer de la bonne compilation du projet. S'aider de l'annexe 1.
- Modifier les sources *rn52\_init.c* et *rn52\_cmd.c* afin de configurer le module RN52 (cf. page suivante). La configuration déploiera un réseau bluetooth (nom du réseau à fixer) avec un profil audio A2DP. Par défaut, aucun code d'association ne sera demandé. Le module RN52 sera interfacé par l'UART2 avec un débit de 9600Bd/s et respectant la configuration spécifiée dans le fichier d'en-tête *bsp/rn52/include/rn52.h*. Le module utilisera en tout 4 broches du MCU, respectivement RB1/RB2/RC0-RX2/RC1-TX2 (côté MCU) et RST/PWR\_EN/TX/RX (côté RN52). Afin d'assurer le chemin de l'information, il nous faudra réaliser deux ponts filaires entre les broches :
  - RB1 <-> RD2 via le connecteur J27 de la carte Curiosity HPC (cf. ci-dessous)
  - RB2 <-> RD3 via le connecteur J27 de la carte Curiosity HPC (cf. ci-dessous)



- Développer une application de test implémentant le logigramme suivant en utilisant l'UART2 pour piloter le module RN52. *Ne pas hésiter à interfacer l'UART1 depuis un ordinateur comme console de debug de sortie (non présenté ci-dessous). A l'image du debug par printf souvent fait sur ordinateur :*

