République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Tunis El Manar
École Nationale d'Ingénieurs de Tunis

Département Génie Eléctrique
Cahier des charges Sujet 10

Exploitation de la liaison serie

Elaborée par :

Salsabil JABALLAH

"G11"

Tuteur ENIT :

M. JELASI KHALED

2ème Année GE2.
Année universitaire : 2023/2024

Table des matières

Table des figures

Introduction

Le projet consistera en la mise en œuvre de divers concepts clés tels que **la liaison série (UART)** et **la conversion analogique-numérique (ADC)** à l'aide de la carte STM32F4. Dans ce cahier des charges, je vais expliquer en détail les principaux concepts que je vais utiliser dans le projet. Dans le deuxième chapitre, je fournirai une description approfondie de la carte STM32F4 utilisée ainsi que du logiciel utilisé pour coder et simuler les fonctionnalités du projet.

Chapitre 1

Mini Projet Micro

1.1 Description

Sujet : Exploitation de la liaison serie pour envoyer les donnees de conversion sur canal 1 ADC1 en format ASCII. Il faut envisager un scenarion de verification du bon fonctionnement de programme.

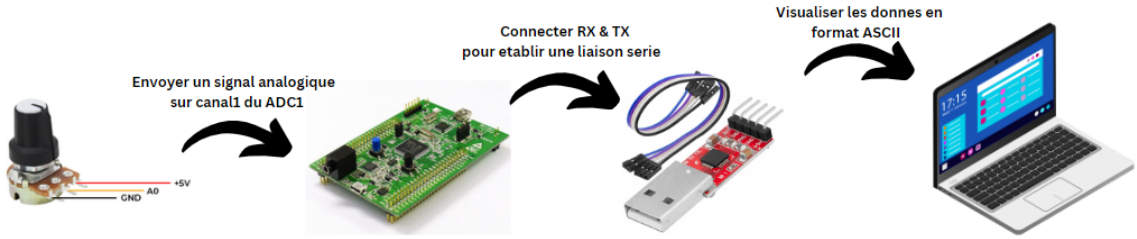


FIGURE 1.1 – Schéma explicatif

Explication du sujet : Le microcontrôleur STM32F407VG lira les données analogiques d'un potentiomètre (J'ai choisi que le signal analogique provienne d'un potentiomètre) sur le canal ADC1(Comme indiqué dans le sujet), convertira ces données en format ASCII, puis les transmettra au PC en connectant la carte à l'ordinateur via un câble USB.Tester la communication en observant les données transmises/reçues dans l'émulateur de terminal Putty Vérifier que les valeurs reçues correspondent aux résultats attendus.

⇒L'objectif principal est de démontrer **la communication série** et **la conversion de données analogiques**.

1.2 Liaison série

Les microcontrôleurs disposent de nombreux moyens pour communiquer avec d'autres systèmes. Des périphériques dédiés aux différents bus industriels sont souvent disponibles. On trouve généralement une liaison série (UART), mais aussi des bus industriels comme l'I2C, le SPI, ou le CAN. D'autres bus plus génériques et communs en informatique sont aussi souvent disponibles tels que l'USB ou l'Ethernet. Nous allons approfondir dans ce chapitre la liaison UART et voir comment on peut la configurer sur un STM32F.

1.2.1 Principes de la liaison série (UART)

Note Dans cette partie, nous allons utiliser la datasheet du microcontrôleurs STM32F411 puisqu'il est similaire à STM32F407 famille (STM32F4xx) .

Une liaison série consiste à envoyer une information bit après bit avec un délai entre chacun. Le bus Universal Asynchronous Receiver Transmitter (UART) est une implémentation de ce principe. La communication via l'UART se fait par l'envoi d'une trame c'est-à-dire un paquet de bits que l'on ne peut pas subdiviser. Le protocole RS232 définit une trame comme étant constituée de :

- un bit de démarrage (start bit) dont le niveau logique est zéro
- entre 7 et 8 bits de données
- éventuellement un bit de parité paire ou impaire (parity)
- et un ou deux bits de stop avec un niveau logique de zéro (stop bits).[7]

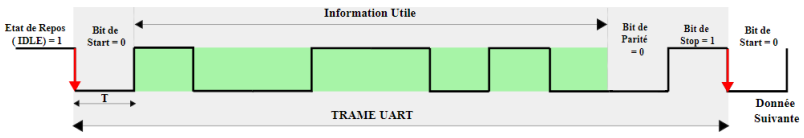


FIGURE 1.2 – Une trame UART

1.2.2 Configuration du périphérique UART

Voici la liaison UART2 dans la carte STM32F411

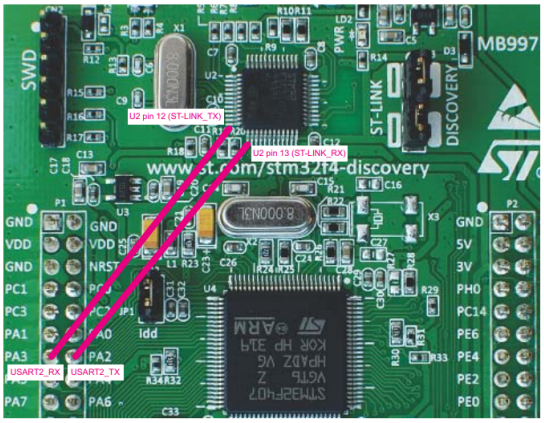


FIGURE 1.3 – ST-LINK VCP connection to USART2

Configuration d'une trame

Le microcontrôleur STM32F4xx présente 3 registres de contrôle/configuration. Dans ces 3 registres, les bits importants pour une utilisation minimale sont :

— UE (CR1 - bit 13) : UART Enable

— M (CR1 - bit 12) : Word length (8 ou 9 bits)

— PCE (CR1 - bit 10) : Parity Control Enable

— TE (CR1 - bit 3) : Transmitter Enable

— RE (CR1 - bit 2) : Receiver Enable

— STOP(CR2 - bits 13 :12) : Nombre de bits de Stop

— RXNEIE (CR1 - bit 5) : RX Not Empty Interrupt Enable : Autorisation des Interruptions en Réception [7]

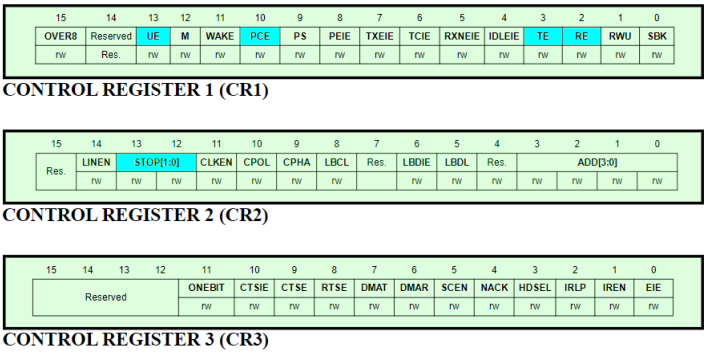


FIGURE 1.4 – Configuration d'une trame

Configuration du baud rate (débit sur la ligne)

[7] Le débit sur la ligne est plus lent que l'horloge du périphérique UART. Il va donc falloir diviser la fréquence afin de définir le baud rate. On a vu dans le chapitre 'liaison série' qu'il fallait attendre T/2 pour faire l'acquisition d'un bit (T étant la durée d'un bit), afin d'être sûr d'avoir un état stabilisé.

Pour garantir ce positionnement par rapport au débit sur la ligne, on divise le temps en quantums de temps 16 fois plus rapides que le débit.

Il faut donc régler dans le registre **BRR** un ratio tel que :

$$USARTDIV = \frac{F_{CK}}{16.baudrate}$$

Le registre BRR contient une valeur au format virgule fixe 12.4

$$\text{soit } F_{CK} = 42MHz$$

Je souhaite baudrate = **9600** J'ai donc :

$$USARTDIV = \frac{F_{CK}}{\text{baudrate}}$$
$$USARTDIV = \frac{42.10^6}{16 * 9600}$$
$$USARTDIV = 273.4375$$

Comme on est au format 12.4, le codage de 273.4375 est le même que celui de $273.7375 * 2^4 = 4375$

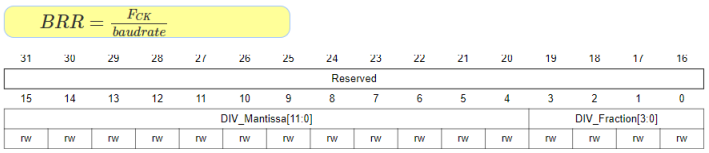


FIGURE 1.5 – Configuration du baud rate

1.2.3 Programmation du périphérique UART

Configuration des Broches

[7] Le STM32F4xx comporte 3 périphériques UART (USART1, USART2, USART6). USART2 est relié au processeur gérant l'interface de debug de la carte (STLINK), de telle sorte qu'il y ait conversion UART -> USB.

Ainsi la liaison UART2 est accessible directement par le cable USB. **Extrait de la documentation de stm32F4xx :**

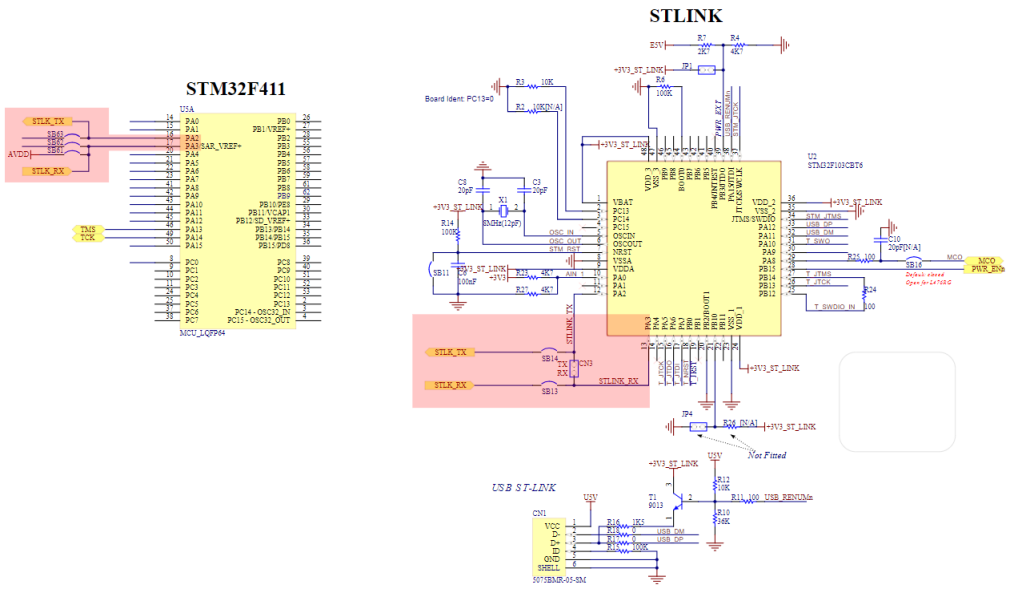


FIGURE 1.6 – Extrait de la documentation de la carte stm32F411

Les broches à configurer sont donc **PA2 (TX) et PA3 (RX)** (Datasheet STM32F411) :

Table 9. Alternate function mapping

Port		AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
		SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1S PI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port A	PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	-	USART2_ CTS	-	-	-	-	-	-	-	EVENT OUT
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	SPI4_MOSI /I2S4_SD	-	USART2_ RTS	-	-	-	-	-	-	-	EVENT OUT
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	I2S2_CKIN	-	USART2_ TX	-	-	-	-	-	-	-	EVENT OUT
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	I2S2_MCK	-	USART2_ RX	-	-	-	-	-	-	-	EVENT OUT
	PA4	-	-	-	-	-	SPI1_NSS/I 2S1_WS	SPI3_NSS/I2 S3_WS	USART2_ CK	-	-	-	-	-	-	-	EVENT OUT
	PA5	-	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_SCK/I 2S1_CK	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA6	-	TIM1_BKIN	TIM3_CH1	-	-	SPI1_MISO	I2S2_MCK	-	-	-	-	-	SDIO_ CMD	-	-	EVENT OUT
	PA7																

DocID026289 Rev 4

FIGURE 1.7 – Alternate function mapping

Configuration du périphérique

[7]
Nous allons utiliser dans notre projet les structures suivantes :

— **UARTHandleTypeDef**

— **USARTTypeDef**

— **UARTInitTypeDef**

et du La fonction **HALUARTInit()** qui permet de configurer les registres du périphérique à partir des éléments définis dans le champ **Init** de la structure **UARTHandleTypeDef**

1.2.4 Domaine d'application

l'UART est toujours utilisé pour des applications faible débit et faible vitesse, car il est très simple, économique et facile à mettre en œuvre.

— Communication entre microcontrôleurs

— Communication entre microcontrôleur et périphériques

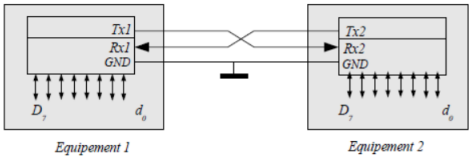


FIGURE 1.8 – EX :Communication entre deux microcontrôleurs

1.3 La conversion Analogique-Numérique

1.3.1 Principe d'un ADC

Les convertisseurs analogique-numérique permettent au microcontrôleur d'accepter une valeur analogique, telle que la sortie d'un capteur, et de convertir le signal dans le domaine numérique. Dans cet section, nous nous concentrons sur le CAN (Convertisseur Analogique-Numérique) STM32.[1]

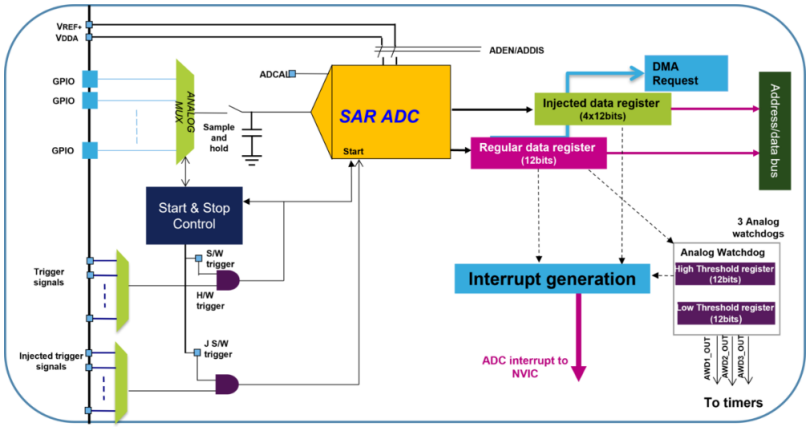


FIGURE 1.9 – Schéma bloc général pour chaque convertisseur analogique-numérique intégré

Rôle : [1] Son rôle est donc de convertir une tension V, généralement comprise entre 0V et +Vref (Vref = 3.3V). La valeur numérique issue de la conversion est comprise entre 0 et 2N -1 avec N le nombre de bits de résolution du convertisseur. Note : Dans notre cas, le CAN serve à convertir les valeurs analogiques entrées par le potentiomètre afin d'agir sur, respectivement, la période du signal MLI et son rapport cyclique. Un CAN est caractérisé par :

- un quatum qui est la valeur en tension d'un incrément de la valeur numérique,
- l'étendue de mesure qui est la plage des tensions en entrée,
- la résolution qui décrit le nombre de valeurs que peut prendre l'ADC sur l'étendue de mesure.

le microcontroleur STM32F407VG posséde 3 ADC (ADC1, ADC2 and ADC3) de 12 bits. Le temps de conversion pour chaque convertisseur est 1 micro seconde

1.3.2 Les caractéristiques d'un ADC

- ADC conversion **rate** 1 MHz and 12-bit resolution
  - 1µs conversion time at 56 MHz (or any X*14Mhz)
  - 1.17µs conversion time at 72 MHz
- Conversion **range**: 0 to 3.6 V
- ADC **supply** requirement: 2.4V to 3.6 V
- ADC **input** range: VREF- ≤ VIN ≤ VREF+ (VREF+ and VREF- available only in LQFP100 package)
- **Dual** mode (on devices with 2 ADCs): 8 conversion mode
- Up to 18 multiplexed **channels**:
  - 16 external channels
  - 2 internal channels: connected to Temperature sensor and internal reference voltage (VREFINT = 1.2V)
- Channels conversion **groups**:
  - Up to 16 channels regular group
  - Up to 4 channels injected group
- Single and continuous conversion modes

FIGURE 1.10 – ADC Features (1/2)
[2]

- **Sequencer**-based scan mode for up to 16 conversion
- External **trigger** option for both regular and injected conversion
- Channel by channel programmable **sampling** time
- **Discontinuous** mode on regular and injected groups
- Left or right Data alignment with inbuilt data coherency
- Analog **Watchdog** on high and low thresholds
- **Interrupt** generation on:
  - End of Conversion
  - End of Injected conversion
  - Analog watchdog
- **DMA** capability (only on ADC1)

FIGURE 1.11 – ADC Features (2/2)
[2]

1.3.3 les modes de conversion d'un ADC

[1]Le convertisseur analogique-numérique (CAN) prend en charge plusieurs modes de conversion :

- **Mode Unique (single) :** Convertit uniquement un canal, en mode unique ou continu.
- **Mode Balayage (scan) :** Convertit un ensemble complet de canaux d'entrée préalablement programmés, en mode unique ou continu.

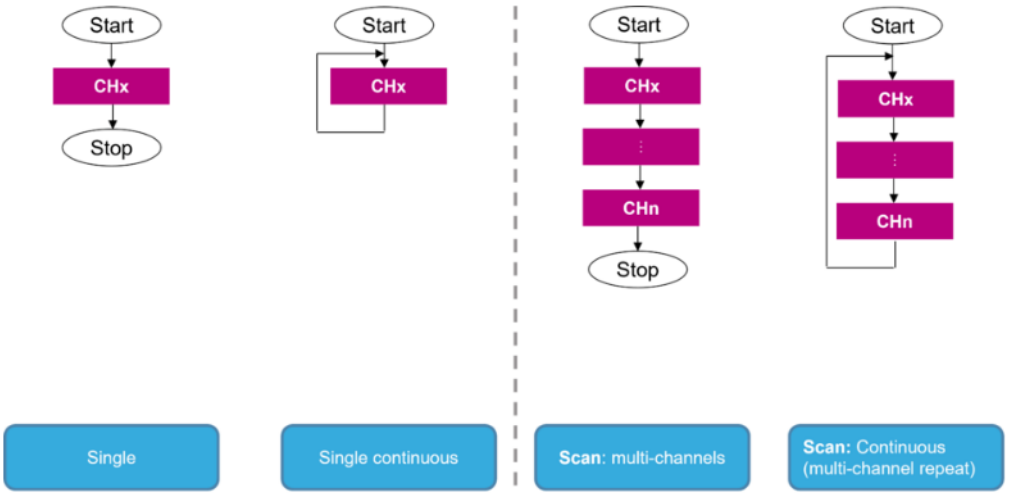


FIGURE 1.12 – les modes de conversion d'un ADC

1.3.4 Étapes de configuration

Comme toute interface programmable, l'ADC doit être bien configuré pour fonctionner correctement.

- Activation du système d'horloge : à travers l'interface RCC (Périphérique APB2)

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADCx, ENABLE)
```

- Configuration de l'interface port (choix des ports à utiliser : Mode Analogique)

Exemple de configuration avec les bibliothèques prédéfinies

```
GPIO_InitTypeDef GPIO_InitStruct;
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AN;
GPIO_InitStruct.GPIO_Pin = GPIO_Pin_x;
GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOx, GPIO_InitStruct);
```

FIGURE 1.13 – lExemple de configuration avec les bibliothèques prédéfinies

- Configuration des paramétres de l'ADC :
  - Diviseur de fréquence (Clock prescaler)
  - Résolution (12 bits/10 bits/ 8bits/6bits)
  - Alignement des données (Data Alignement) Gauche ou Droite.
  - Mode canaux multiples (scan continuous mode / Multiple channels)
  - Mode conversion Continue/Discontinue (Continuous/Discontinuous conversion mode).
  - si le DMA est actif (Direct Memory Access) on peut choisir le mode DMA continuous request.
  - Drapeaux de fin de conversion (End of conversion flag EOC)

Chapitre 2

CHOIX TECHNIQUES : MATERIEL ET LOGICIEL

2.1 Environnement matériel

2.1.1 Description : La carte STM32F4Discovery

La carte STM32F4Discovery, fournie par STMicroelectronics, permet aux utilisateurs de développer facilement des applications avec un microcontrôleur haute performance STM32F4 muni d'un processeur ARM Cortex-M4 32 bits. Elle inclut tout ce qui est nécessaire pour les débutants ou pour les utilisateurs expérimentés pour commencer rapidement à effectuer des développements. Elle est peu coûteuse, son prix n'a rien à voir avec son importance et en plus elle est facile à utiliser, la carte STM32F4Discovery aide à découvrir les fonctionnalités haute performance du microcontrôleur STM32F4 et à développer des applications.

⇒ Cette carte offre à ses utilisateurs les caractéristiques suivantes :

- Un microcontrôleur STM32F407VGTx avec processeur ARM Cortex-M4 32 bits, une mémoire Flash de 1 Mo, une mémoire vive de 192 Ko et une FPU. (M4 : Processeur avec fonction DSP, optimisation mathématiques et virgule flottante simple précision)
- Un ST-LINK/V2 intégré.
- Alimentation de la carte par USB ou par une alimentation externe 3V ou 5V.
- Un accéléromètre à 3 axes.
- Un capteur audio (microphone).
- Un DAC audio avec un amplificateur de classe D intégré.
- Huit LEDs :
  - LD1 (rouge/vert) pour la communication USB

— LD2 (rouge) pour la mise sous tension 3.3 V
— Quatre LEDs Utilisateur
— 2 LEDs USB OTG (USB On-The-Go), LD7 (vert) pour VBUS et LD8 (rouge) pour la surintensité

- Deux boutons-poussoirs (utilisateur et réinitialisation).
- Interface USB OTG avec connecteur micro-AB.
- Connecteurs d'extension pour les entrées/sorties (headers).

[4]

La Figure 2.1 illustre les connexions entre le STM32F407VGT6 et ses périphériques (STLINK/V2-A, boutons-poussoirs, LED, DAC audio, USB, accéléromètre ST-MEMS et microphone, ainsi que les connecteurs).[3]

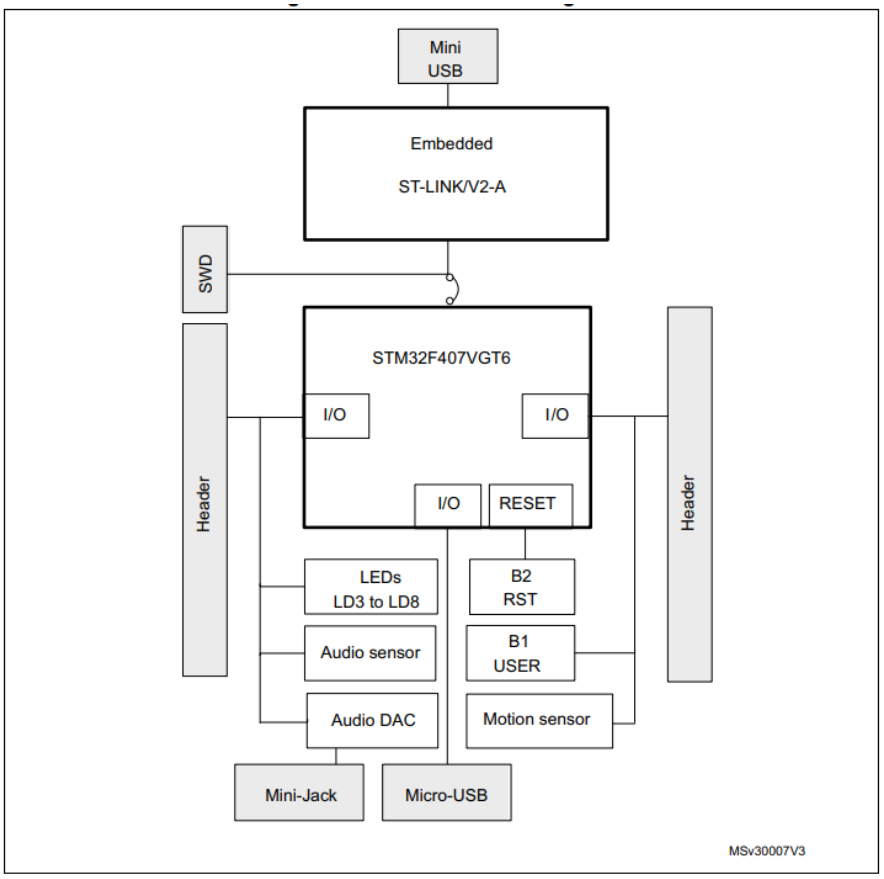
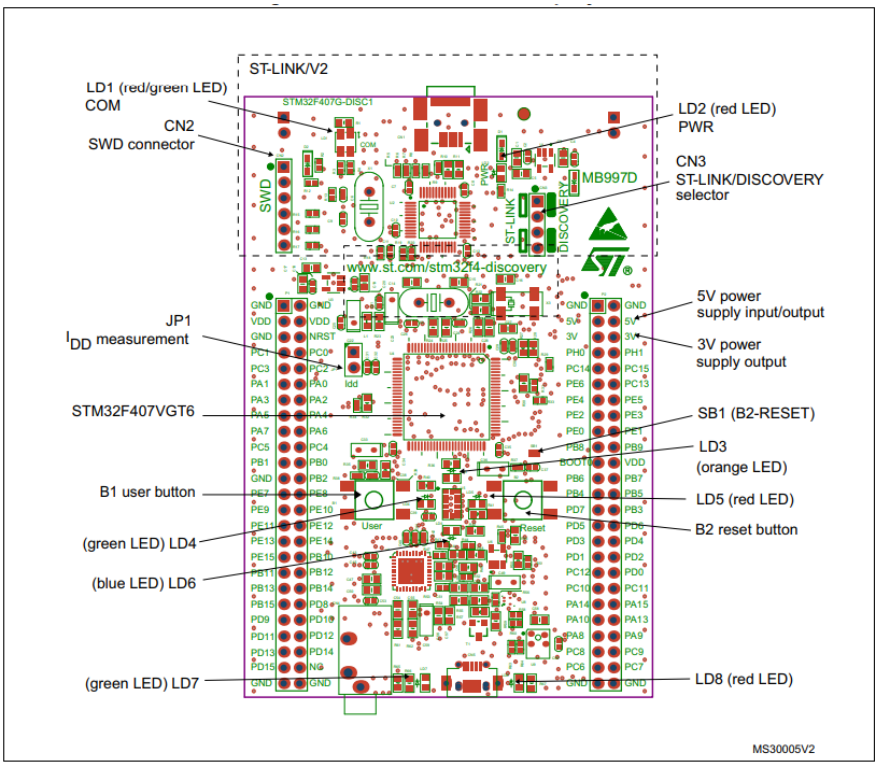


FIGURE 2.1 – Hardware block diagram

2.1.2 Matériel et Disposition

Le **STM32F4DISCOVERY** est conçu autour du microcontrôleur STM32F407VGT6, présenté dans un boîtier LQFP de 100 broches.

Les Figures suivantes aident les utilisateurs à localiser ces fonctionnalités sur la carte STM32F4DISCOVERY.[3]



Note: Pin 1 of CN2, CN3, JP1, P1 and P2 connectors are identified by a red square.

FIGURE 2.2 – STM32F4DISCOVERY top layout

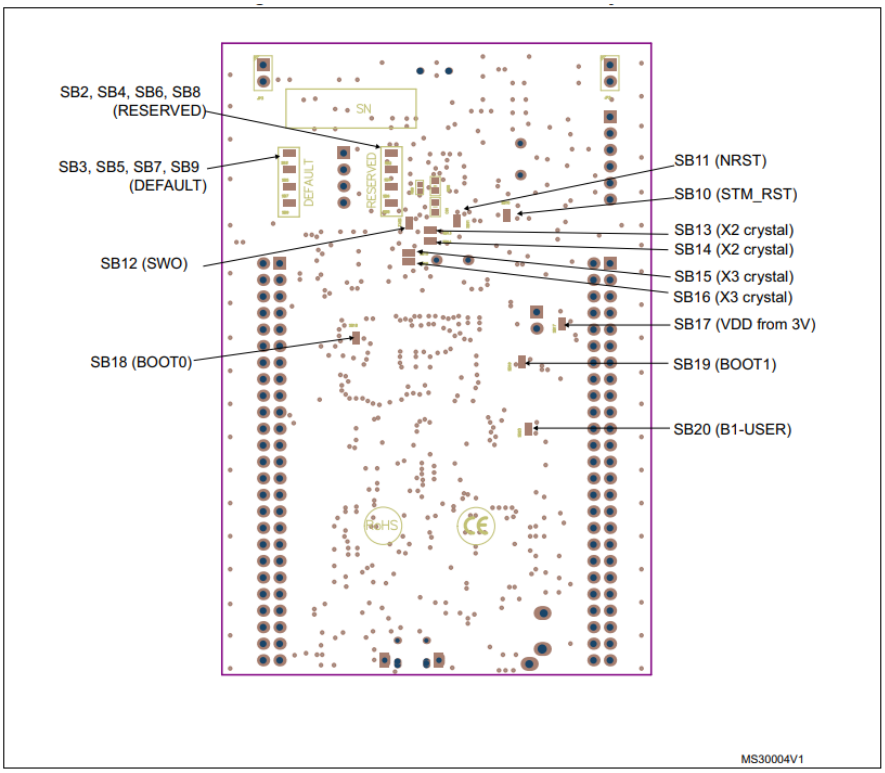


FIGURE 2.3 – STM32F4DISCOVERY bottom layout

2.1.3 Domaine d'application

Les microcontrôleurs STM32F4 proposés par ST permettent de répondre aux attentes des concepteurs de produits électroniques embarqués, connectés et intelligents destinés à des domaines d'application tels que les relevés de compteurs, les énergies renouvelables ou la santé, qui utilisent des systèmes embarqués exigeant de nouvelles caractéristiques, des performances accrues et un rendement énergétique supérieur. « Avec leurs fonctionnalités sophistiquées et leur haut niveau de connectivité, nos tout nouveaux microcontrôleurs STM32 assurent une expérience exceptionnelle aux utilisateurs de produits électroniques courants », a déclaré Michel Buffa, directeur général de la division Microcontrollers de STMicroelectronics. « Nous travaillons d'ores et déjà avec des clients clés pour intégrer des variantes étendues du microcontrôleur STM32 F4, qui fournissent jusqu'à 2 Mo de mémoire Flash en plus, dans des applications telles que les compteurs de relevé intelligents, les contrôleurs de panneaux solaires, les modules sans fil ou les systèmes de surveillance médicale personnelle. [4]

2.1.4 Système d'horloge

Périphériques

[4] Chaque interface doit être pilotée par des signaux d'horloge qui peuvent être identiques ou différents selon leurs configurations. Il est à noter que, au sein du microcontrôleur STM32F4, les interfaces sont réparties en 5 groupes :

- **AHB3 (Advanced High-performance Bus 3) :** FSMCEN (Flexible Static Memory Controller Module Clock Enable).
- **AHB2 (Advanced High-performance Bus 2) :** OTGFSEN (USB OTG FS), RNGEN, HASHEN.
- **AHB1 (Advanced High-performance Bus 1) :** GPIOA, GPIOB, ..., GPIOI, USB, ETHERNET.
- **APB1 (Advanced Peripheral Bus 1) :** TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, USART2, USART3, USART4, USART5, I2C1, I2C2, I2C3, CAN1, CAN2.
- **APB2 (Advanced Peripheral Bus 2) :** TIM1, TIM8, USART1, USART6, ADC1, ADC2, ADC3, SPI1.

2.1.5 Quelques interfaces programmables

Les GPIOx

[5] STM32 GPIO can be used in a variety of configurations. Each GPIO pin can be individually configured by software in any of the following modes :

- Input floating
- Input with pull-up
- Input with pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Le organigramme présenté dans les figures suivantes offre aux utilisateurs une aide rapide pour sélectionner le mode GPIO et la configuration adaptés à leur application.

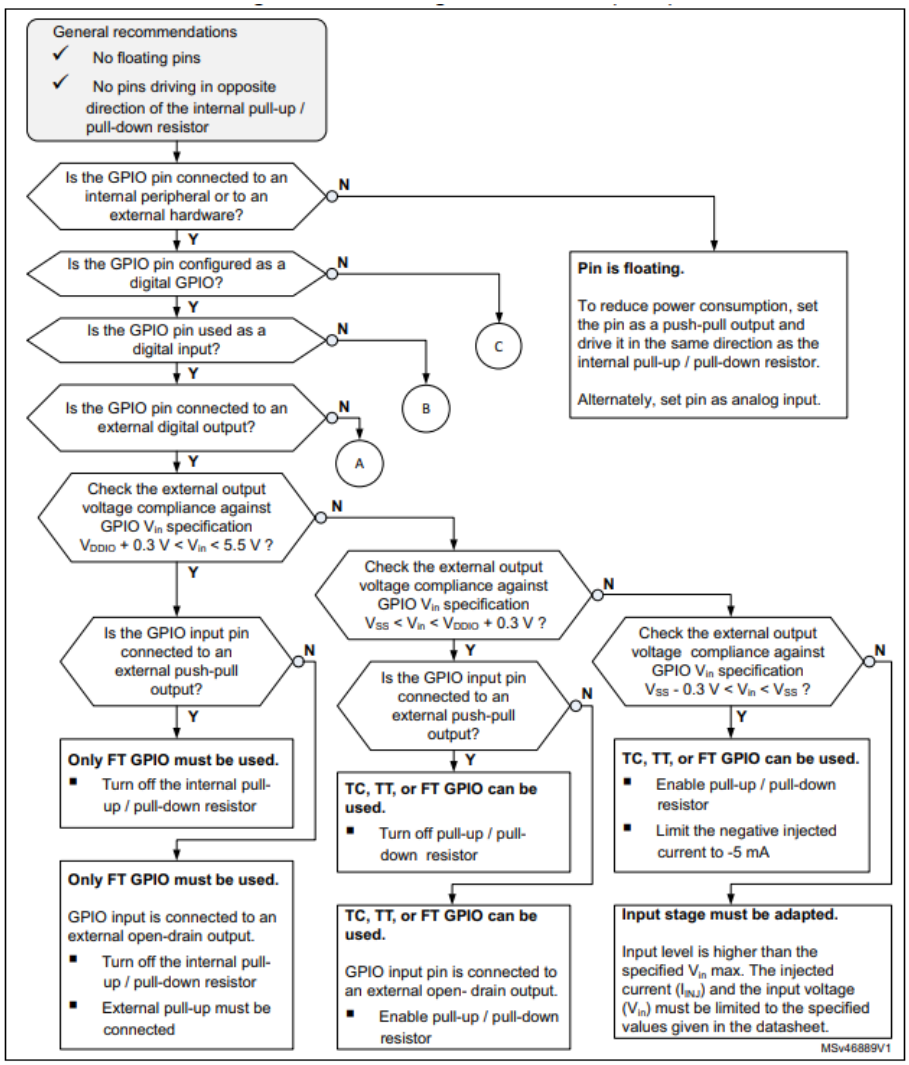


FIGURE 2.4 – GPIO configuration flowchart (1 of 2)

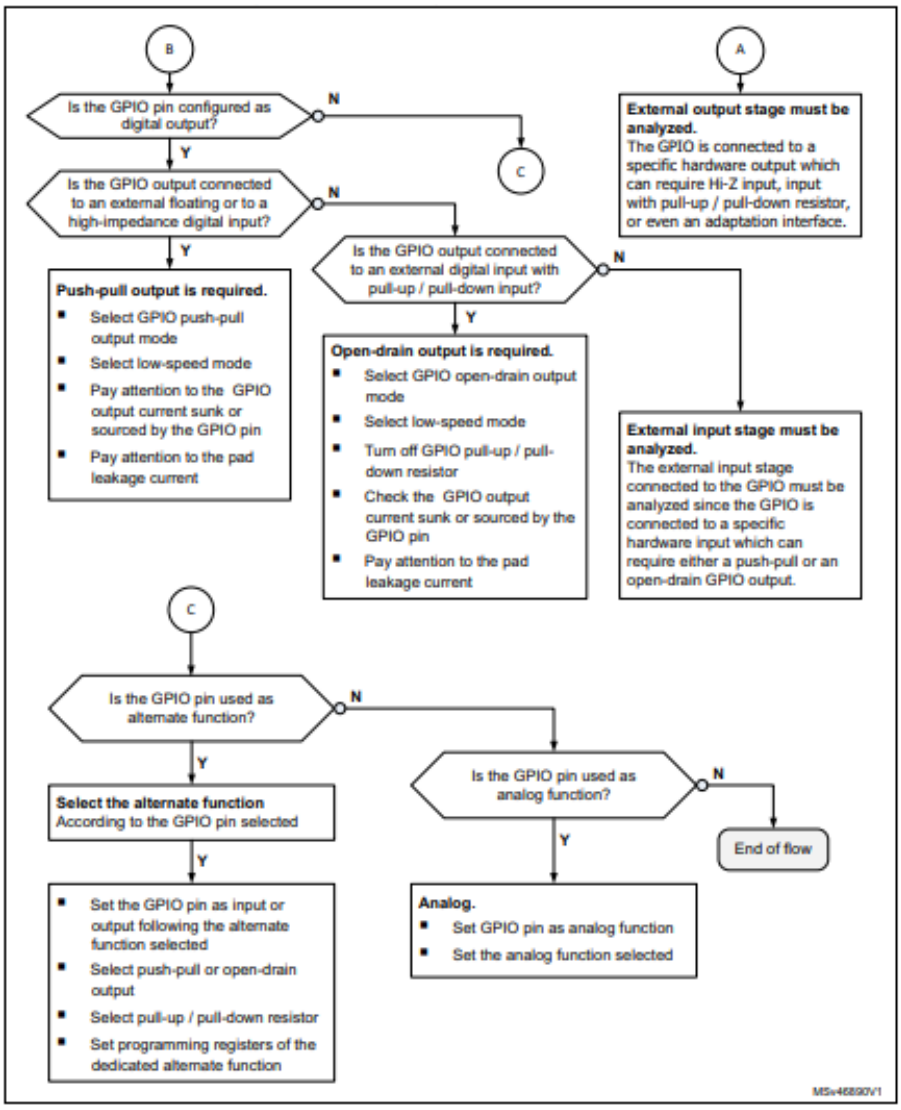


FIGURE 2.5 – GPIO configuration flowchart (2 of 2)

Les Timers et watchdogs

[6] Le dispositif STM32F407xx comprennent deux timers de contrôle avancé, huit timers polyvalents, deux timers basiques et deux timers de surveillance. Tous

les compteurs des timers peuvent être gelés en mode de débogage. Le Tableau suivant compare les caractéristiques des timers de contrôle avancé, des timers polyvalents et des timers basiques.

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen- tary output	Max interface clock (MHz)	Max timer clock (MHz)
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	84	168

FIGURE 2.6 – Timer feature comparison

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen- tary output	Max interface clock (MHz)	Max timer clock (MHz)
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	84	168
	TIM10 , TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	84	168
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	42	84
	TIM13 , TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	42	84
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	42	84

FIGURE 2.7 – Timer feature comparison (continued)

— **Timers de contrôle avancé (TIM1, TIM8) :**
  — Trois-phase PWM générateurs sur 6 canaux
  — Sorties PWM complémentaires avec des temps morts programmables
  — Peuvent être utilisés comme timers polyvalents avec 4 canaux indépendants pour :
    — Capture d'entrée
    — Comparaison de sortie

        - Génération PWM (modes bord ou aligné au centre)
        - Sortie en mode impulsion unique
    - Supportent la génération indépendante de demandes DMA
- **Timers polyvalents (TIMx) :**
    - TIM2, TIM3, TIM4, TIM5 : Timers polyvalents complets avec jusqu'à 16 entrées/sorties PWM
    - TIM9, TIM10, TIM11, TIM12, TIM13, et TIM14 : Timers basés sur un compteur de remise à zéro automatique de 16 bits et un prédiviseur de 16 bits
- **Timers de base (TIM6 et TIM7) :**
    - Principalement utilisés pour le déclenchement du DAC et la génération de forme d'onde
    - Supportent la génération indépendante de demandes DMA
- **Chien de garde indépendant :**
    - Basé sur un compteur de 12 bits et un prédiviseur de 8 bits
    - Horloge indépendante de 32 kHz, peut opérer en modes Stop et Standby
- **Chien de garde de fenêtre :**
    - Basé sur un compteur de 7 bits, peut être utilisé comme un watchdog pour réinitialiser le dispositif
    - Coupure possible du compteur en mode de débogage
- **Minuteur SysTick :**
    - Dédié aux systèmes d'exploitation temps réel
    - Compte à rebours de 24 bits, capacité de remise à zéro automatique, interruption système masquage
    - Source d'horloge programmable

Les ADCs (CANs)

J'ai déjà mentionné et expliqué cela dans le chapitre 1, dans la sous-section intitulée *<La conversion Analogique-Numérique>*.

2.1.6 USB to TTl

L'USB vers TTL est un adaptateur permettant de connecter un ordinateur à un port série TTL via un port USB. En communication, TTL peut signifier Time To Live ou Transistor-Transistor Logic. Le Time To Live détermine la durée de vie d'un paquet de données, tandis que la Transistor-Transistor Logic est un type d'interface série.

TTL et CMOS sont des technologies de puce décrivant les transistors utilisés à l'intérieur des puces, tandis qu'un UART est un type spécifique de puce, un émetteur-récepteur asynchrone universel utilisé pour convertir un caractère parallèle en un flux de bits série. Les câbles série USB TTL fournissent une connectivité entre USB et les interfaces série UART, offrant des niveaux de signal à 5V, 3.3V, ou d'autres niveaux spécifiés par l'utilisateur, avec diverses interfaces de connecteur.
–> Après avoir téléchargé notre programme sur la carte, nous utiliserons ce câble pour visualiser les données transmises par la carte.

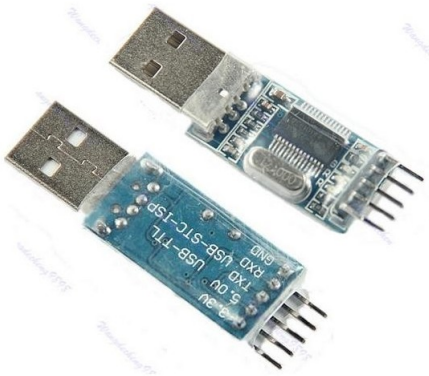


Figure 2.8 – USB to TTL

2.1.7 Potentiometre

Un potentiomètre peut être utilisé pour fournir un signal analogique à la broche PA1 du microcontrôleur STM32F407. Ce signal est ensuite converti en une valeur numérique utilisable par l'ADC1, configuré pour lire les données analogiques de cette broche. **–> on va utiliser un potentiomètre de 1K**

FIGURE 2.9 – Potentiometre

2.1.8 CABLE DATA USB A Vers Mini B

Le câble USB A vers Mini B permet à la fois la programmation du microcontrôleur STM32F407VG depuis un ordinateur, en **téléchargeant le code source compilé dans sa mémoire flash**, et **le débogage** du code en cours d'exécution sur la carte à l'aide d'un IDE comme Keil. Cette connexion permet aux développeurs de **surveiller le comportement du programme**, de faire des pas à pas et de définir des points d'arrêt.

FIGURE 2.10 – Cable USB A-B

2.1.9 Fils de Connection

Nous allons utiliser des fils de connexion femelles pour connecter le potentiomètre et le module USB TTL à notre carte STM32F407VG.

FIGURE 2.11 – Fils de Connection femelles

2.2 Environnement logiciel

2.2.1 keil vision 5

Keil vision 5 est un environnement de développement, qui donne aux utilisateurs un accès facile aux microcontrôleurs ARM® Cortex®-M.Il contient un compilateur en C/C++ ainsi qu'un débogueur. Keil offre la possibilité de simuler en temps réel les tâches effectuées par la carte et d'observer les signaux à la sortie.

FIGURE 2.12 – keil vision 5

2.2.2 Putty émulateur

PuTTY est un émulateur de terminal pour Windows permettant la connexion à une machine distante par protocole ssh. Avec ce logiciel, nous pouvons travailler, depuis notre ordinateur personnel, sur une machine Linux du DMS, en mode ligne de commandes.

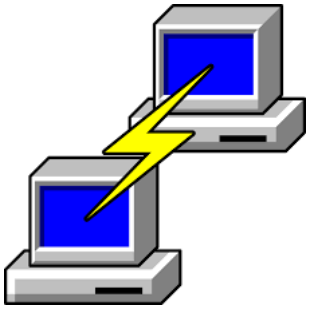


FIGURE 2.13 – Putty

Conclusion

Ce mini projet vise à démontrer une application pratique des microcontrôleurs STM32F4 dans le contexte de la transmission série et de la conversion analogique-numérique.

Références Bibliographiques

[1] *ADC : document de référence.* URL : https://wiki.st.com/stm32mcu/wiki/Getting_started_with_ADC#What_is_an_analog_to_digital_converter_-28ADC-29-3F.

[2] *ADC caractéristiques : document de référence.* URL : 2GE/cours/STM32_peripheriques.pdf.

[3] *les microcontrôleurs STM32 C. HEMDANI : document de référence.* URL : https://www.electronique-mixte.fr/wp-content/uploads/2018/08/Cours-Microcontr%C3%B4leur-microprocesseur-48.pdf.

[4] *Miniprojet MLI ONS JELASI : document de référence.* URL : 2GE/cours/Mini_projet_MLI_ONS___JELASI.pdf.

[5] *stm32 : document de référence.* URL : https://www.st.com/resource/en/application_note/dm00315319-stm32-gpio-configuration-for-hardware-settings-and-low-power-consumption-stmicroelectronics.pdf.

[6] *STM32F405xx STM32F407xx datasheet : document de référence.* URL : https://www.st.com/resource/en/datasheet/DM00037051.pdf.

[7] *UART : document de référence.* URL : https://www.enib.fr/~kerhoas/uart.html.