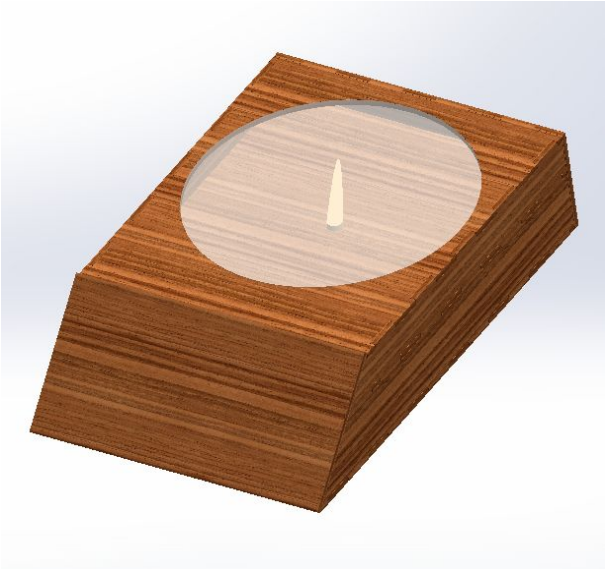
Cahier des charges

-

Boussole

Electronics Project - École 42



Création d'une boussole électronique qui indique l'endroit où l'on désire se rendre.

Auteurs:

Valentin Omnès - vomnes

Félix Herbinet - fherbine

Doriel Chiche - dchiche

Tom Ktorza - tktorza

Date de création:

Février 2018

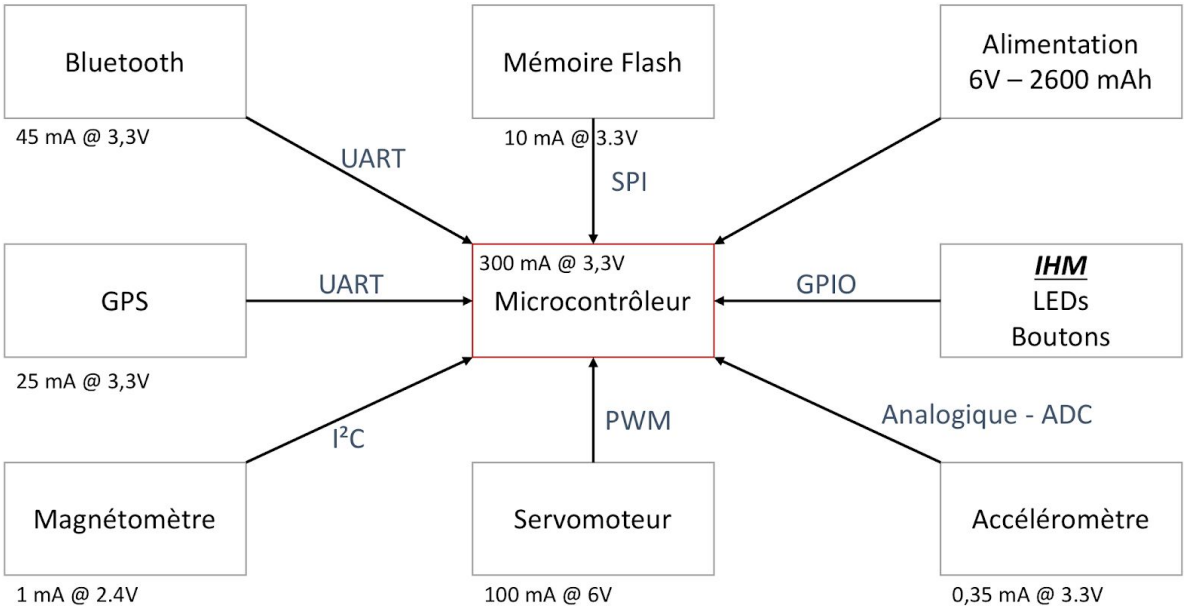
Introduction

Dans le cadre de notre formation à l'École 42, nous avons l'opportunité de développer nos compétences en électronique au travers de la réalisation d'un produit électronique en partant de presque zéro.

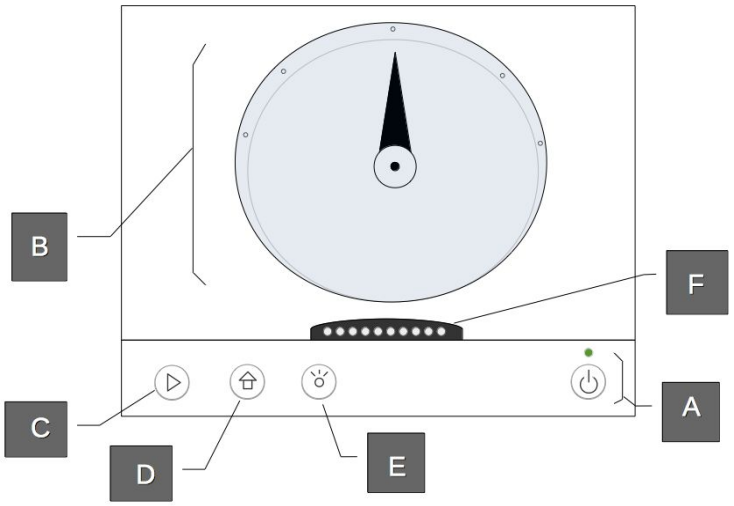
Nous avons décidé de réaliser comme projet une boussole électronique indiquant la direction du lieu où l'on souhaite se rendre.

Ainsi, un système permettra d'envoyer les coordonnées géographiques à la boussole et grâce aux coordonnées géographiques actuelles, la direction du nord et quelques calculs mathématique, l'électronique pourra déterminer et indiquer visuellement cette direction.

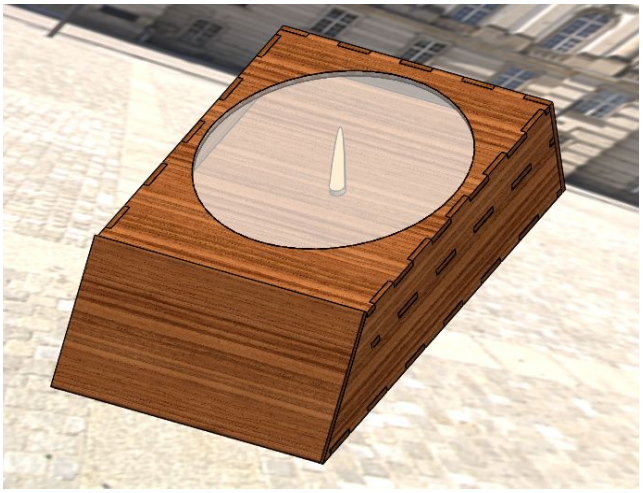
Schéma fonctionnel



Mode d’emploi



La boussole s’utilise de la manière suivante:
On l’allume avec le bouton Power (A), pour appareiller un téléphone portable à la boussole, on appuie longtemps sur le bouton de destination (C) jusqu'à ce que les LEDs de distances (F) clignotent.
Une fois la boussole appareillée, on peut commencer le trajet en appuyant brièvement sur le bouton de destination (C). A tout moment, on peut retourner au point de départ en appuyant sur le bouton home (E).
Durant notre trajet, la boussole mécanique (B), nous indique la direction à suivre. Elle peut être rétroéclairé en appuyant sur le bouton prévu à cette effet (E). A tout moment, l'état de la batterie est donné par la LED de batterie (A).
Une fois arrivé à destination, la couleur de rétro-éclairage change (exemple: rouge) et le pointeur de la boussole se fige sur une position définie. Plus l’on se rapproche de notre destination, plus il y a de LEDs de distances (F) allumés.



Nous allons nous même fabriquer le boîtier de la boussole en bois. Ci-dessus une première version sommaire (incomplète) de la boussole en 3D (dessiné avec Solidworks).

Liste des composants

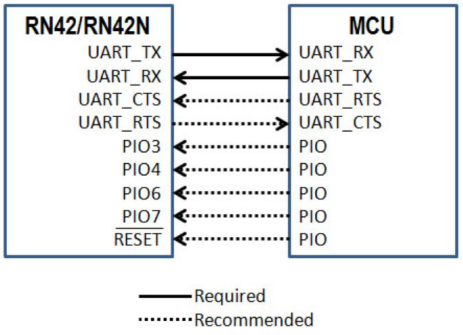
1. Bluetooth

Le module bluetooth sélectionné est un ***RN42***, fabriqué par Microchip, possède un ***antenne intégrée*** et est Bluetooth Special Interest Group (SIG). Il correspond à la ***classe 2*** des modules radio Bluetooth, ce qui correspond à une puissance de ***2,5 mW*** pour une portée de 10 mètres.

***Attention*: Pins de 0.80mm de diamètre distancés de 0.40mm, il n'y a pas de pins à souder dessous le module.**

- *Protocoles*

Il utilise le protocole Universal Asynchronous Receiver Transmitter ***(UART)*** (cf. Partie sur les protocoles de communication). Nous allons utiliser le protocole bluetooth Serial Port Profile (SPP) data rate présent sur le module afin d'avoir un débit de données de 240 Kbps en 'Slave' mode ou 300 Kbps en 'Master' mode. Le SPP est le protocole le plus standard dans le domaine des profils bluetooth, celui-ci utilise les lignes RX et TX.
Pour commander le bluetooth les informations sont dans le guide intitulé 'Bluetooth Data Module Advanced User's Guide', celui-ci est disponible en PDF sur internet.



Connection du RN42 au Microcontrôleur

Sur ce schéma ci-dessus, les pin qui sont intéressant sont l'UART_TX pour la transmission et l'UART_RX pour la réception. L'utilisation de l'UART_CTS et UART_RTS est recommandée uniquement lorsqu'il y a besoin d'envoyer en continue des données.

- *Caractéristiques électrique*

La consommation de ce module est de 26µA au repos, 3mA lorsqu'il est connecté et 30mA en mode transmission.
Le module nécessite une tension d'alimentation comprise entre 3,0 (min) et 3,6 (max) Volts (V) avec une tension conseillée (typical) de 3,3V.
L'activité radio (Discovery or Inquiry Window Time) nécessite 40mA.
La connection avec transfert de données entre 40mA (minimum) et 50mA (maximum) avec une valeur typique de 50mA.

- *Utilisation*

Le RN42 possède 2 modes :

➔ **Command mode** : Permet de configurer le module en choisissant *mode 'Master' ou 'Esclave'*, le *Baud rate* (bps), *Serial Port Flow Control*, *nom de l'appareil* ou encore le Pin code.

Il est nécessaire de configurer le module avant d'utiliser le mode data sinon les configurations par défaut sont utilisées.

➔ **Data mode**

2. Global Positioning System (GPS)

Le module GPS, que nous allons utiliser est le *Breakout GPS de Microstack* équipé du GPS L80-M39 de Quectel. Il est impossible pour nous d'utiliser directement un module GPS (sans breakout) car les modèles disponible nécessitent de souder des pads qui se situent dessous le module ce qui est impossible avec l'équipement dont nous disposons.

Ce module est un module dit hautement intégré ce qui nous permet de récupérer directement la position GPS de notre module avec une latitude et une longitude.

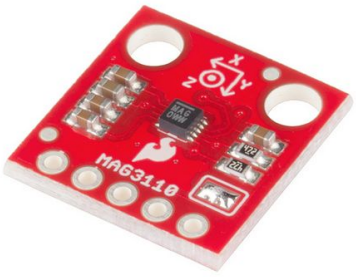
Ce module est utilisable entre 3V et 4.3V toutefois il est conseillé dans la datasheet, d'alimenter ce module avec une tension de 3.3V et avec un courant de 25mA.

Le protocole utilisé pour transmettre les coordonnées GPS au microcontrôleur est le protocole UART (cf. Partie sur les protocoles de communication), utilisant les pins TX0, et RX0. Le message transmis (chaîne de caractères) par le module avec le protocole UART est standardisé il s'agit du NMEA, pour National Marine Electronics Association.

Le message est formaté de la manière suivante:

Début	**Time**	**Latitude**	**Longitude**	**Satellites**
$GPRMC,	*235316.000, A,*	*4003.9040, N,*	*190512.5792, W,*	*009*
Code de début du message	Heure de réception : 23h53min16sec (GMT + 0.00)	Latitude en degrés décimaux Nord.	Longitude en degrés décimaux Ouest.	Nombre de stellite avec lesquelles le module a communiqué.

3. Digital Magnetometer



Nous avons sélectionné le 'Digital Magnometer' Xtrinsic **MAG3110** Three-Axis capable de mesurer des champs magnétiques avec un **Output Data Rate (ODR) allant jusqu'à 80 Hz**.

Cependant le composant MAG3110 en lui même est trop petit, de chaque côté, il y a 5 pins de 0.2mm d'épaisseur sur 2mm, ce qui est impossible à souder avec le matériel que nous disposons. Nous allons donc utilisé la *Breakout Board de SparkFun* intégrant le MAG3110.

Le MG3110 permet de détecter **3 canaux de champ magnétiques**, il nécessite une alimentation électrique comprise entre **1.95V et 3.6V**, possède 5 pins.

- *Communication*

Le magnétomètre utilise le **protocole de communication I²C** (cf. Partie sur les protocoles de communication) en retournant comme données de sorties les **valeurs des abscisses x, y et z qui oscillant entre -180 et 180**, lorsque x retourne 0 cela signifie que le magnétomètre pointe vers le nord.

- *Utilisation des pins*

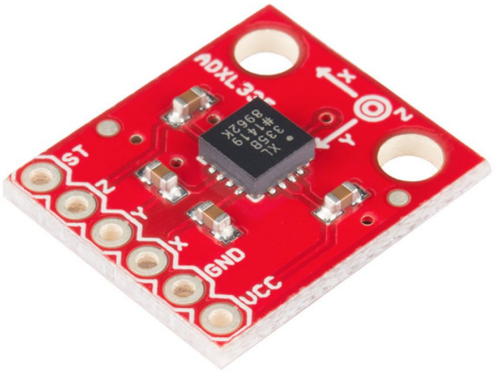
Pin	Utilisation
VCC	Alimentation électrique
GND	Connexion électrique à la masse
SDA - Data	Serial Data pin pour la communication I²C
SCL - Clock	Serial Data pin pour la communication I²C
INT - Interrupt	Pin d'interruption - 'High' quand de nouvelles données sont prêtes

- *Informations complémentaire*

La calibration et la détection du nord magnétique fonctionne seulement lorsque le magnétomètre à un niveau d'orientation avec l'abscisse pointant en l'air ou en bas. Pour détecter le nord magnétique indépendamment de l'orientation de la boussole, il est nécessaire d'utiliser en plus un accéléromètre qui permet de recueillir plus d'informations permettant de corriger ce problème.

Il est important de notifier que de temps en temps l**e magnétomètre nécessitera une calibration** qui faudra intégrer à l'utilisation de la boussole. Pour calibrer la boussole, il faut la poser sur une surface plane et la faire tourner sur elle même.

4. Accéléromètre



Nous avons sélectionné le **ADXL335** d'Analog Devices capable de nous retourner les valeurs des positions x, y et z correspondant aux mouvements. L'accéléromètre sera complémentaire au magnétomètre et permettra d'utiliser les valeurs de ce dernier et cela peut importe son sens.

Le ADXL335 seul nécessite la soudure d'un pad sous le composant ce qui est impossible avec notre équipement, nous avons donc décidé d'utiliser une breakout board intégrant le composant. Il nécessite une alimentation électrique comprise entre **1.8V et 3.6V**.

La bande passante peut être adapté par rapport aux besoins, les axes X et Y peuvent varier de 0.5 Hz à 1600 Hz, l'axe Z quand à lui peut varier de 0.5 Hz À 550 Hz.

- *Communication*

Le ADXL335 communique avec le microcontrôleur en transformant les données analogique en digital (cf. Partie sur les protocoles de communication).

- *Utilisation des pins*

Pin	Utilisation
VCC	Alimentation électrique
GND	Connexion électrique à la masse
ST	Built-in self-test
X	Sortie analogique des valeurs de l'axe X
Y	Sortie analogique des valeurs de l'axe Y
Z	Sortie analogique des valeurs de l'axe Z

5. Mémoire flash

La mémoire flash, que nous allons utiliser est une LE25U40CMDTWG, fabriqué par On Semiconductor. Ce modèle dispose d'un stockage total de **4Mbit (512K x 8bits)** et la limite maximum de la fréquence est de **40 MHz** (inférieur à celui de notre microcontrôleur), dispose du **protocol de communication SPI** (cf. Partie sur les protocoles de communication) et de 8 broches (facilement soudable) et est sous une tension de 2.3 a 3.6V.

Cette mémoire flash permettra de stocker ces éléments la position de départ, la destination, la position actuelle et les valeur X, Y et Z par rapport à l'orientation par rapport au nord.

Type de donnée:

- Double: 8 octets
- Int: 4 octets

Contenu	Taille en octets	Bytes
Position GPS - Float	8 octets x 2	16
Orientation X,Y, Z - Int	4 octets x 3	12

Pour stocker les positions GPS nous avons donc besoin de 48 bytes et pour stocker l'orientation de la boussole nous avons besoins de 12 bytes, ce qui donne un total de 60 bytes (480 bits). Une mémoire flash de 4Mbit est donc suffisance et permet d'avoir de l'espace libre pour stocker d'autre données.

Pour interagir avec ce module mémoire, il faut envoyer un opCode en premier lieu: read, program, erase, read/write status registers, etc. suivi des 24 bit d'adresse avec laquelle nous voulons commencer, à savoir par exemple l'adresse de stockage de la position Home.
Il sera nécessaire d'utiliser les commandes lire, écrire et effacer. Il est impossible de réécrire par dessus des données, il nécessite au préalable de les écraser. L'écriture sur la mémoire flash n'est pas directe, il est indispensable de stocker l'information dans un buffer puis pour réellement écrire il faut stimuler le pin d'événement qui va activer l'événement associé à l'action.

6. Servomoteur

Le Servomoteur miniature sélectionné est un **D2S51**, fabriqué par Magic RCM. Ce produit offre la possibilité de contrôler la vitesse et direction de rotation sans angle limite (360 degrés).

La largeur des impulsions varie entre 1000µs et 2000µs;

- Entre 1000µs et 1500µs, le servo tourne en avant, 1000us en vitesse maximum et 1500µs stop.
- Entre 1500µs et 1500µs, le servo tourne en sens inverse, 2000µs en vitesse maximum.

Cette caractéristique réponds donc parfaitement à nos besoins de plus sa petite taille est un atout.

- *Caractéristiques techniques*
  - ➔ Dimensions: 19 x 8 x 17 mm
  - ➔ Poids: 2,9 gr
  - ➔ Alimentation: 4,8 à 6 Vcc
  - ➔ Vitesse (sans charge): 0.10sec/60°@4.8V; 0.09sec/60°@6.0V
  - ➔ Angle: 360°
  - ➔ Couple: 0,45 à 0,6 kg.cm
  - ➔ Fréquence d'impulsion : Période de 20ms, 1000us-2000us

Un servo moteur a trois câbles:

- Alimentation électrique
- Masse électrique
- Signal de contrôle

Il est contrôlé par le signal de contrôle, les impulsions électrique sont envoyées avec un interval de 20 millisecondes (50Hz) et varient entre 1000 et 2200 nanosecondes (en durée). Le signal est généré par un modulateur de largeur d'impulsion (Pulse-width modulation (PWM) - type de signal digital). Le PWM est intégré au microcontrôleur.

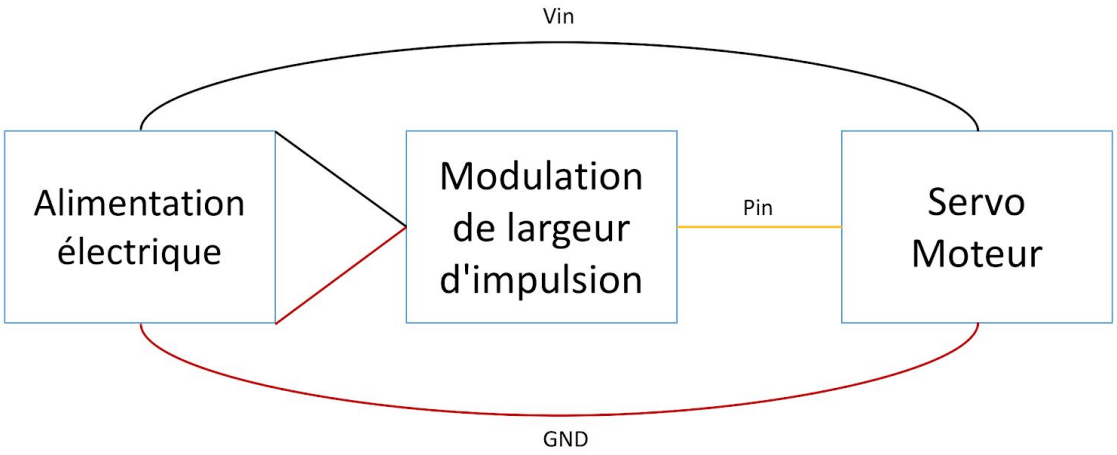
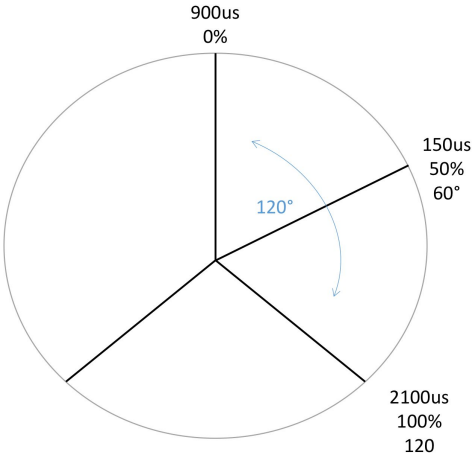


Schéma électrique



Variation du rapport cyclique - Duty cycle - Exemple 900us - 2100us

7. IHM (Interactions homme-machine)

- *LED*

Nous avons selectionné les LED SMP4-RGY de BIVAR avec les couleurs Rouge, Vert et Jaune. Elles sont du type composant monté en surface (CMS), éclairent sur un angle de 120 degrés pour 2.4mm de diamètre.

- *Boutons*

Nous avons sélectionné le commutateur tactile B35-1000 de OMRON. Il est en CMS, du type SPST (Single-pole, single-throw), très simple d'utilisation; soit l'interrupteur est fermé, soit il est complètement déconnecté; très utilisé pour les boutons On-Off. Il peut supporté de 1 mA à 50 mA et 5 VDC à 24 VDC.

Nous avons sélectionné le commutateur à bouton-poussoir PVB4 EE 300 NS LFS de C&K. Il est en CMS, du type SPDT, supporte au maximum 14 VDC et 100mA.

8. Microcontrôleur

Par rapport au composant que nous avons préalablement sélectionné et détaillé nous avons besoin d'un microcontrôleur disposant d'au moins deux UART, un I²C, un SPI et un PWM, avec une fréquence d'au moins 40MHz.

Nous avons ainsi sélectionné le **PIC32MX250F128B-I/SO** qui dispose de suffisamment de composant pour gérer les protocoles de communication utilisé et d'une fréquence maximum de **50MHz**.

Bilan électrique

Nous pouvons déterminer la consommation d'un composant électrique grâce à la relation suivante "***Puissance (Watts) = Intensité (Ampères) x Tension (Volts)***".
Les données permettant les calcules ci-dessous proviennent des 'datasheets' des différents composant, les valeurs collectées proviennent des situations de haute consommation des composants.

Composant	Intensité @ Tension	Puissance (en Watts)
Bluetooth	50 mA @ 3,6V	0,18
GPS	25 mA @ 4,3V	0,1075
Magnétomètre	1 mA @ 3,6V	0,0036
Mémoire flash	10 mA @ 3.6V	0,036
Servomoteur	100 mA @ 6V	0,6
Microcontrôleur	300 mA @ 3,6V	1,08
Accéléromètre	0.35 mA @ 3.6V	0,00126
Total	486,35 mA	2,00836 Watts

- *Alimentation électrique*

Pour calculer l'efficacité d'une batterie, il faut regarder son voltage (V) et sa capacité estimée en ampere-hours (Ah). Il y a la possibilité d'installer plusieurs batteries en '**Série**' afin d'additionner les volts ou en '**Parallèle**' pour additionner la capacité des batteries.

Dans notre cas, nous allons les installer en série car même si la capacité est importante, il nécessite beaucoup de voltages en particulier pour alimenter le servomoteur.

Nous avons sélectionné le lot de 4 piles rechargeable LR06 AA de VARTA avec un voltage de 1.5V et une capacité de 2600 mAh.

$$\frac{BatteryCapacity(Amphours)}{CurrentDraw(Amps)} = BatteryLife(Hours)$$

$$\frac{2600\ mAh\ /\ 1000}{486{,}35\ mA\ /\ 1000} = 5{,}35 = 5h\ 21min$$

2600 mAh → Piles - 486,45 mA → Tableau bilan électrique

Cette configuration nous permettra d'avoir une tension total de 6V avec une capacité de 2600 mAh pour une autonomie optimale de 5 heures et 21 minutes.
Afin d'alimenter les composants nécessitant moins de voltage, il sera nécessaire d'intégrer un convertisseurs DC-DC ou un 'Logic Level Converter'.

Liste des principaux composants

Composant	Réf. Fabricant	Fabricant	Réf. Fournisseur
Microcontrôleur	PIC32MX250F128B-I/SO	Microchip	Farnell - Code 2097772
Bluetooth	RN42-I/RM	Microchip	Farnell - Code 2143310
GPS	MICROSTACK GPS L80-M39	Microstack Quectel	Farnell - Code 2434228
Digital Magnetometer	MAG3110FCR1	Freescale Sensors	Sparkfun - SEN-12670
IHM - LED	SMP4-RGY	Bivar	Farnell - Code 2293501RL
IHM - Bouton	B35-1000	Omron	Farnell - Code 177807
IHM - Bouton On/Off	PVB4 EE 300 NS LFS	C&K	Farnell - Code 2435330
Servomoteur	D2S51	Magic RCM	Gotronic - Code 33508
Mémoire flash	LE25U40CMDTWG	On Semiconductor	Farnell - Code 2627898
Accéléromètre	EVAL-ADXL335Z	Analog Devices	Farnell - Code 1699046

- *Datasheets*

Bluetooth:
http://www.farnell.com/datasheets/2182405.pdf?_ga=2.178919462.347971440.1520186285-643474983.1519298976&_gac=1.24276552.1519298993.CjwKCAiA8bnUBRA-EiwAc0hZkywiyH4MyRGYYK3wvaHkhp7hQFUsBCKMg6mEmk4xGn_3zsGZ0vH-sxoCGjcQAvD_BwE
GPS:
https://www.quectel.com/UploadImage/Downlad/L80_Hardware_Design_V1.1.pdf
Magnétomètre:
https://cdn.sparkfun.com/datasheets/Sensors/Magneto/MAG3110_v9.2.pdf
Mémoire flash:
http://www.farnell.com/datasheets/2118264.pdf?_ga=2.100591134.1400266483.1520346149-643474983.1519298976&_gac=1.223987305.1519298993.CjwKCAiA8bnUBRA-EiwAc0hZkywiyH4MyRGYYK3wvaHkhp7hQFUsBCKMg6mEmk4xGn_3zsGZ0vH-sxoCGjcQAvD_BwE
Servomoteur:
http://digitalmeans.co.uk/shop/360_degree_micro_servo-450gcm
Microcontrôleur:
http://www.farnell.com/datasheets/2244305.pdf?_ga=2.38717603.1042700240.1520354140-2095494544.1519307294

Accéléromètre:
http://www.farnell.com/datasheets/2258652.pdf?_ga=2.259329930.1400266483.1520346149-643474983.1519298976&_gac=1.229224174.1519298993.CjwKCAiA8bnUBRA-EiwAc0hZkywiyH4MyRGYYK3wvaHkhp7hQFUsBCKMg6mEmk4xGn_3zsGZ0vH-sxoCGjcQAvD_BwE

Protocoles de communication

- *Universal Asynchronous Receiver Transmitter - UART*

A l'intérieur de l'électronique embarqué différent circuit individuel ont besoin de s'échanger des informations, de ce fait il nécessite d'avoir un protocole de communication commun. Il en existe des centaines de différents incluant la transmission série dont fait partie l'UART.

La transmission série permet d'envoyer une continue un flux de données, **un seul bit à la fois**. Ce type d'interface peut fonctionner avec seulement un fil électrique mais habituellement jamais plus de quatre.

A l'intérieur du protocole série, il y a plusieurs série, il y a d'autre type tel que le **synchrone** (with clock) ou **l'asynchrone** (without clock).

L'UART est du type asynchrone, ce type nécessite un certain nombre de **règles** ou mécanismes qui aident à assurer un **transfert de donnée éfficace et sans erreurs**. Le protocole est grandement configurable, ainsi la partie critique est d'être sur que les deux éléments utilisant la transmission série sont configuré pour utiliser exactement le même protocole.

- **Baud rate** : Détermine la rapidité à laquelle les données sont envoyées au travers de la ligne, habituellement exprimé en *bits-per-second* (bps). Le baud rate le plus courant est 9600 bps, celui-ci est particulièrement utilisé lorsque la vitesse de transmission n'est pas primordiale.
- **Trame des données** : Chaque bloc de données est envoyé dans un paquet de bits (1 ou 0) standardisé.

Trame	Début	Données	Parité	Fin
Taille	1	5-9	0-1	1-2
Détails	Synchronisation	Données	Check d'erreurs	Synchronisation

- ➢ *Données:* C'est la donnée qui est transportée, ça taille n'est pas fixe elle peut varier entre 5 et 9 bits. Il faut aussi configurer la **Endianness** des données, big-endian (most significant bit first) ou little-endian (least significant bit first), par défaut elle est du type little-endian.
- ➢ *Bit(s) de synchronisation:* Permet de déterminer le début et la fin de la trame.
- ➢ *Bit(s) de parité:* C'est une forme de système très simple (bas-niveau) pour la vérification des erreurs. Pour déterminer le bit de parité, tous les bits (5 à 9) du byte de donnée sont additionnées, et la parité de zéro permet de déterminer si le bit est activé ou pas.

Il y a par exemple le standard **9600/8N1** :

- 9600 est le Baud rate
- 8 pour le nombre de bit de données
- N car il n'y a pas de parité
- 1 car il y a un seul bit de fin

La transmission des caractères ASCII 'O' et 'K' (2 paquets de données) donnerait donc "**0 11110010 1 . 0 11010010 1**":

- 'O' => 79 => 0b01001111
- 'K' => 75 => 0b01001011

Câblage

Il nécessite seulement 2 fil électrique, un pour envoyer (TX) les données et l'autre pour les recevoir (RX).

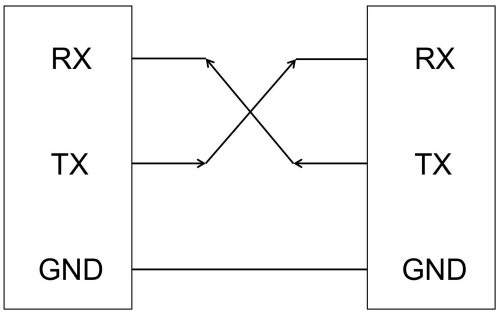


Schéma de transfert, réception
pour la communication en série

Implémentation matériel

L'implémentation de série la plus populaire est le Logic-level TTL (Transistor-transistor logic). Dans ce cas, le signal série qui exists a un voltage qui varie sur une échelle de 0 à 3,3V (ou 5V).

- Le signal au niveau **Vcc** (3,3 ou 5V) corresponds soit :
  - Une ligne idle
  - Un bit de valeur 1
  - Un bit d'arrêt
- Le signal au niveau **GND** (0V) corresponds soit :
  - Un bit de valeur 0
  - Un bit de début

Ainsi, l'UART est une implémentation de la transmission en série. Il agit principalement comme un intermédiaire entre les interfaces parallèle et séries.
Dans notre cas, l'UART est disponible directement à l'intérieur du microcontrôleur PIC32.
Le côté de *transmission* permet de :

- **Créer** de paquet de données
- **Ajouter les bits de synchronisation et de parité**
- **Envoyer** ce paquet au travers de la ligne TX avec un **timing précis** (Baud Rate)

Le côté de *réception* permet de :

- Récupérer les données sur la ligne TX en fonction du baud rate spécifié
- Supprimer les bits de synchronisation
- Transférer les données

Il est important de notifier que aujourd'hui les UARTs les plus avancés peuvent recevoir les données au travers d'un buffer (First In First Out) qui peut aller de quelques bits à plus plusieurs milliers de bytes.

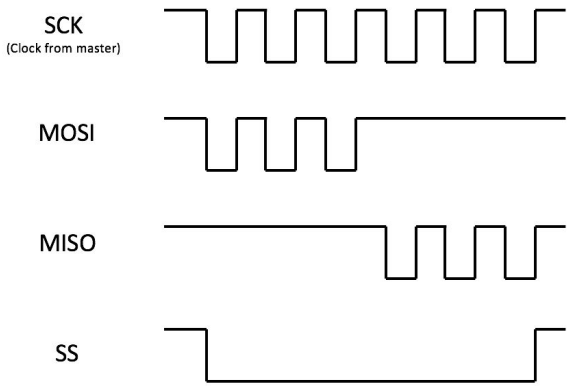
- *<u>Serial Peripheral Interface (SPI)</u>*

Le SPI est un bus informatique est un communément utilisé pour envoyer des données entre un microcontrôleur et de petits périphériques tel que de la mémoire flash, c'est précisément dans ce cas là que nous allons l'utiliser.

Ce bus est un système de transfert de données synchrone, ce qui signifie qu'il utilise différentes lignes pour les données et une horloge (clock) qui permet de garder les deux côtés en parfaite synchronisation. Cette horloge est un oscillateur qui envoie un signal périodique qui permet au récepteur de savoir exactement quand récupérer les bits sur la ligne de données.

- *Réception des données*

Le côté qui génère le signal périodique (*SCK*) est appelé ***the master***, l'autre côté est appelé ***the slave***, il peut y avoir plusieurs 'slaves'.
Quand les données sont envoyés d'un 'master' à un 'slave', c'est la ligne de données *MOSI* pour "Master Out / Slave In" qui est utilisée.
Quand les données sont envoyés d'un 'slave' à un 'master', c'est la ligne de données *MISO* pour "Master In / Slave Out" qui est utilisée.
La ligne Slave Select (*SS*) permet de dire à un 'slave' qu'il doit se réveiller et envoyer ou recevoir des données. Cette ligne est utilisée lorsqu'il y a plusieurs 'slaves' afin de sélectionner le 'slave' avec lequel on souhaite communiquer, la sélection est effective lorsque ligne est 'low'. Chaque 'slave' possède sa propre ligne SS.



- *<u>Inter-integrated Circuit (I²C) Protocole</u>*

L'objectif du I²C est de permettre à plusieurs 'slaves' intégrés à un circuit électrique de communiquer avec un ou plusieurs éléments 'master'.
Contrairement au SPI, le I²C ne nécessite que de deux fils électrique. De plus, il peut supporter jusqu'à 1008 composants 'slaves' et est simple à implémenter.
Dans le bus I²C, il y deux signaux:

- SCL: Le signal périodique qui est généré par l'actuel 'master'
- SDA: Le signal qui contient les données

Quelques appareils 'slave' peuvent forcer le signal périodique à être à bas niveau afin de ralentir la vitesse d'envoie des données par le 'master', on appel cela 'clock stretching'.

Les buc I²C sont dit “open drain” ou open collector, c’est à dire qu’ils peuvent pousser le signal d’une ligne à niveau faible (low) mais ne peut pas faire l’inverse c’est pousser le signal haut (high). Pour se faire chaque ligne de signal possède une résistance pull-up pour mettre le signal haut lorsque aucun appareil ne le pousse à bas niveau (low).
Un système où un appareil à un voltage plus élevé que l’autre pourrait être problématique, cependant le I²C permet de les connecter sans utiliser de ‘level shifter’, simplement en connectant une résistance pull-up sur seulement sur la ligne de signal ayant le plus faible des deux voltage. Attention, certaines conditions doivent être réunis pour que ce soit possible sinon il faut utiliser un level shifter.

Le signal doit suivre un certain protocole/trame pour être considéré comme un signal I²C valide. La plupart des appareil s’occupent de formater le signal suivant cette trame.
Il y a deux types de trame :

- Adresse: le master indique à quel slave le message est entrain d’être envoyé.
- Data (Une ou plusieurs) : un message de 8 bits de données envoyé du master au slave ou inversement.

Les données sont placées sur la ligne SDA après que la ligne SCL soit basse (low) et est échantillonné après que la ligne SCL soit haute.
La durée entre deux ‘clock’ est déterminé par l’appareil sur le bus et change d’une puce à l’autre.

- *Condition de début*

Pour démarrer la trame d’adresse, l’appareil master laisse SCL haut et pousse SDA bas (cf. Schéma I²C trame). Cela permet d’informer tous les appareils qu’une transmission est sur le point de commencer.
Lorsque plusieurs masters souhaite contrôler le bus en même temps, c’est l’appareil qui pousse SDA à bas niveau le premier qui gagne la course et donc le contrôle.

- Trame de l’adresse

La trame comportant l’adresse est tout le temps la première lors d’une nouvelle communication.
Il existe deux types d'adressage, le 7-bit et 10-bit adresse, le PIC32 permet de gérer les deux types. Nous allons nous attarder sur le type 7-bit qui est plus simple car ne nécessite pas de transmettre l’adresse du slave au préalable.

Trame Adresse	Most Significant Bit First	Read (1) / Write (0)	NACK/ACK (9ème bit)

Une fois que le premier bit de la trame est envoyé, l’appareil de réception à le contrôle à travers la ligne SDA. La ligne SDA devient basse avant l’envoie du 9ème signal périodique.

- Trame des données

Le master continue de générer le signal périodique a un interval régulier. Les données sont envoyées sur la ligne SDA soit par le master ou le slave, cela dépend du bit Read/Write.

- Condition de stop

Une fois que toutes les données ont étaient transmises, il faut envoyer le signal de fin.
Cette condition corresponds à un SDA qui passe de faible à haut juste après que SCL soit passé de bas à haut (cf. Schéma I²C trame). C'est en quelque sorte l'inverse de la condition de début.
Durant les opérations d'écriture, la valeur sur la ligne SDA ne doit pas changer quand la ligne SCL est haute afin d'éviter de simuler une fausse condition d'arrêt.

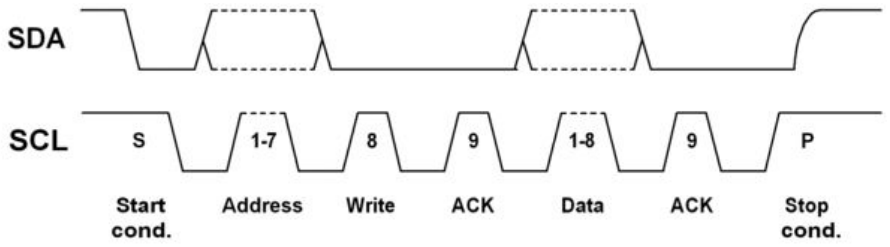


Schéma I²C trame

- *General-purpose input/output (GPIO)*

Les GPIO sont des pins standard qui sont simplement utilisés pour allumer ou éteindre des dispositifs (LEDs, boutons…).

- *Signal analogique*

Pour rendre le signal analogique utilisable par le monde digital, il est nécessaire de le transformer le voltage analogique du pin en une valeur digital, pour se faire il est nécessaire d'utiliser l'**Analog to Digital Converter** (ADC) intégré au microcontrôleur.
ADCs changent beaucoup entre les différents microcontrôleur. Le ADC de notre PIC32 est un 10-bit ADC, ce qui signifie qu'il est capable de détecter 1,024 (210) différents niveau d'analogie.

$$\frac{Resolution\ of\ the\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

Par exemple :

- Resolution of the ADC → 10-bit ADC → 1023
- System Voltage → 6V

$$\frac{1023}{6V} \times \text{Valeur analogique} = \text{Valeur digitale}$$

Ressources

Bluetooth :
https://learn.sparkfun.com/tutorials/bluetooth-basics
https://learn.sparkfun.com/tutorials/using-the-bluesmirf
https://www.bluetooth.com/specifications
http://ww1.microchip.com/downloads/en/DeviceDoc/bluetooth_cr_UG-v1.0r.pdf
GPS :
https://learn.sparkfun.com/tutorials/gps-basics
Magnétomètre:
https://www.mouser.fr/new/nxp-semiconductors/freescalemag3110
https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html
https://learn.sparkfun.com/tutorials/mag3110-magnetometer-hookup-guide-
Servomoteur:
https://learn.sparkfun.com/tutorials/pulse-width-modulation
Communication :
https://learn.sparkfun.com/tutorials/serial-communication
https://learn.sparkfun.com/tutorials/i2c
http://www.i2c-bus.org/
https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
Mémoire flash :
http://www.eeherald.com/section/design-guide/esmod16.html
https://rheingoldheavy.com/at25sf081-tutorial-01-functionality-overview-01
Microcontrôleur :
http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=30063
http://umassamherstm5.org/tech-tutorials/pic32-tutorials/pic32mx220-tutorials
http://ww1.microchip.com/downloads/en/DeviceDoc/32bitPeripheralLibraryGuide.pdf