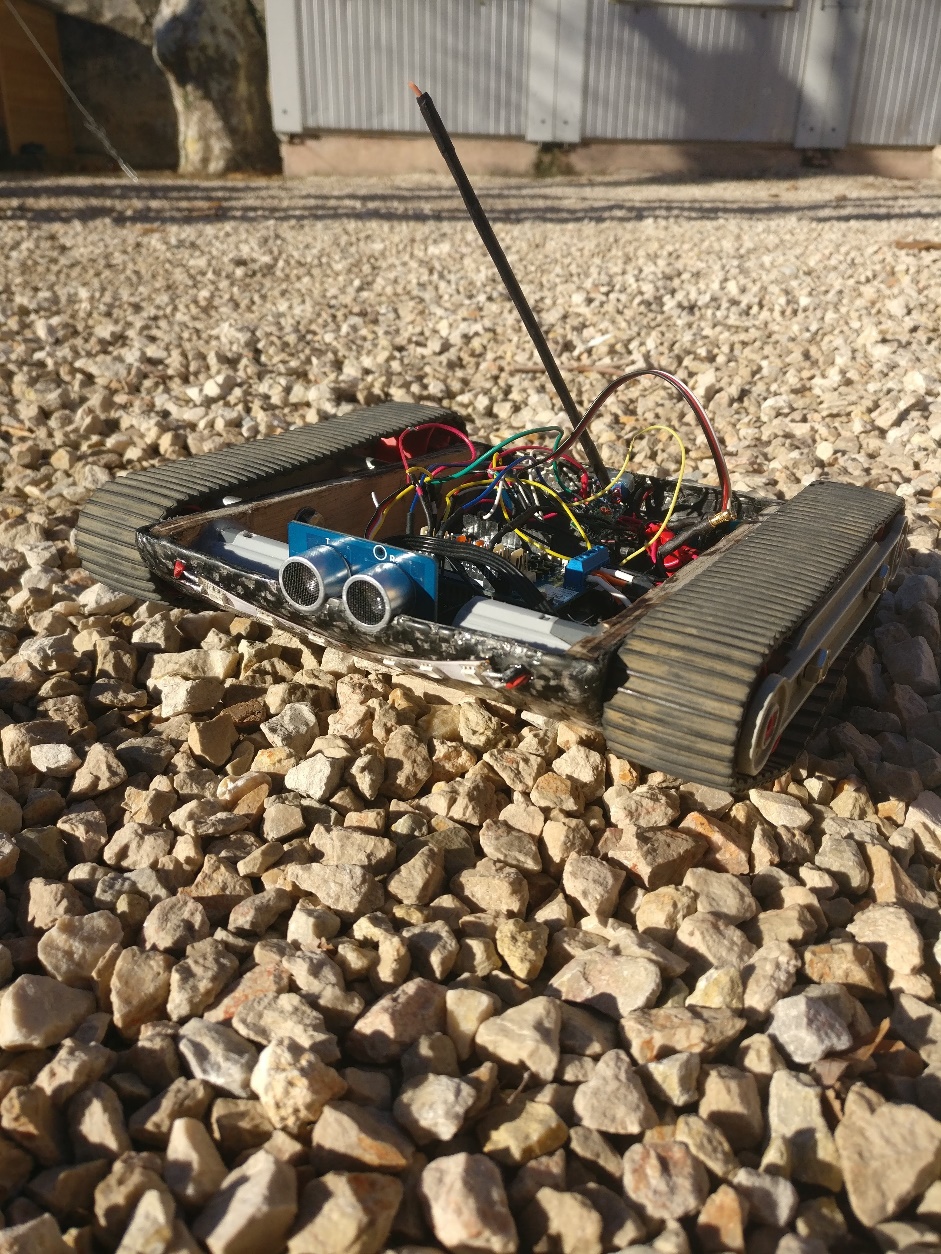
**Mini Projet STI2D 2017-2018**

**Spécialité : système de l’information numérique**



**Robot monté sur chenilles automatisé**

Paul– Commande des moteurs

Léo– Télécommande

Simon– Bluetooth et température

Lucas– Capteur ultrason

Table des matières

[1 Introduction 4](#_Toc500917663)

[2 Présentation générale 5](#_Toc500917664)

[2.1 Brainstorming et planification 5](#_Toc500917665)

[2.2 Bon de commande 5](#_Toc500917666)

[2.3 Sysml et algorigrammes 5](#_Toc500917667)

[2.3.1 Diagramme de cas d’utilisation 5](#_Toc500917668)

[2.3.2 Diagramme d’exigences 6](#_Toc500917669)

[2.3.3 Diagramme séquences 8](#_Toc500917670)

[2.3.4 Diagramme d’états 8](#_Toc500917671)

[2.3.5 Diagramme définitions block 9](#_Toc500917672)

[2.3.6 Diagramme block interne 9](#_Toc500917673)

[2.4 Montage électrique 10](#_Toc500917674)

[2.5 Algorigramme 11](#_Toc500917675)

[3 Répartition des tâches 12](#_Toc500917676)

[3.1 Télécommande 12](#_Toc500917677)

[3.1.1 Les modules radio 12](#_Toc500917678)

[3.1.2 Principe du système 13](#_Toc500917679)

[3.1.3 Transmission des informations 13](#_Toc500917680)

[3.1.4 Composition du signal : les trames 17](#_Toc500917681)

[3.1.5 Composition des messages 18](#_Toc500917682)

[3.1.6 Description du programme 19](#_Toc500917683)

[3.1.7 Améliorations possibles 20](#_Toc500917684)

[3.2 Commande des moteurs 20](#_Toc500917685)

[3.2.1 Shield moteur 21](#_Toc500917686)

[3.2.2 Moteur 21](#_Toc500917687)

[3.2.3 Pont en H 21](#_Toc500917688)

[3.2.4 L298P 22](#_Toc500917689)

[3.3 Capteur ultrason 23](#_Toc500917690)

[3.3.1 Les ondes ultrason 23](#_Toc500917691)

[3.3.2 Le capteur à ultrasons 24](#_Toc500917692)

[3.3.3 Le code 25](#_Toc500917693)

[3.4 Capteur de température et Bluetooth 25](#_Toc500917694)

[3.4.1 Le capteur de température 25](#_Toc500917695)

[3.4.2 Transmission des mesures : 31](#_Toc500917696)

[3.4.3 Description du programme : 35](#_Toc500917697)

[4 Annexe 1 : 38](#_Toc500917698)

[5 Bibliographie 39](#_Toc500917699)

[6 Remerciements 40](#_Toc500917700)

# Introduction

Le robot monté sur chenilles automatisé est un robot amovible, fonctionnant sur une base de carte Arduino Uno. Le robot est télécommandé par une radiocommande à la fréquence de 433 MHz. Le robot possède un capteur de température, un capteur ultrasons pour éviter les obstacles, et une liaison Bluetooth pour la transmission de la température sur un ordinateur.

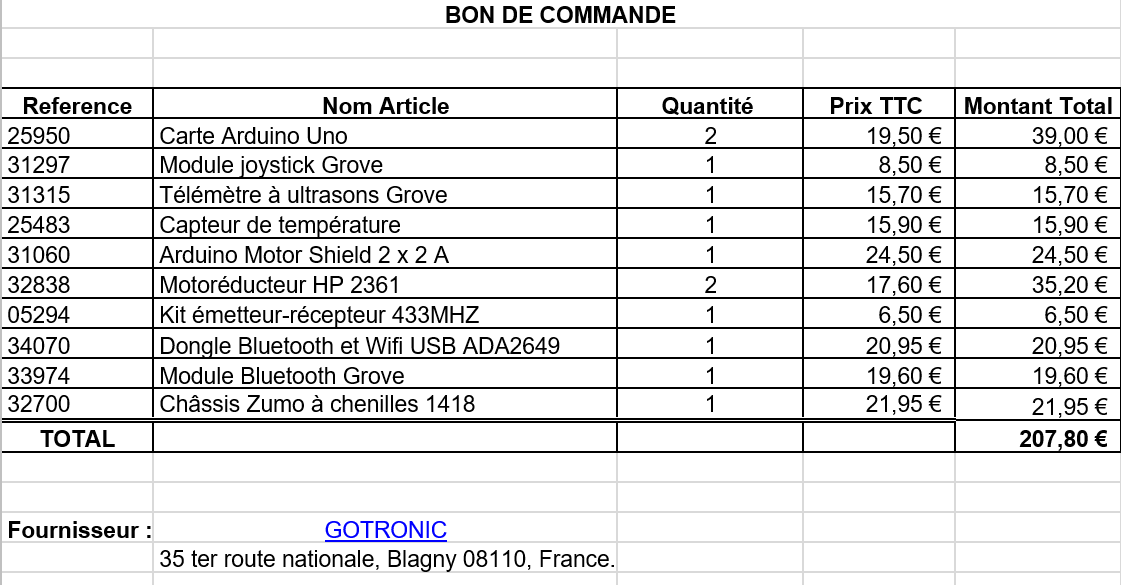
# Présentation générale

## Brainstorming et planification

Une image contenant capture d’écran

Description générée avec un niveau de confiance élevéPlanification gant Project :

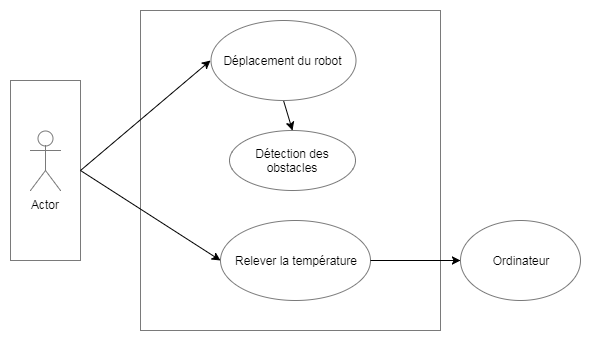
## Bon de commande



## **Sysml et algorigrammes**

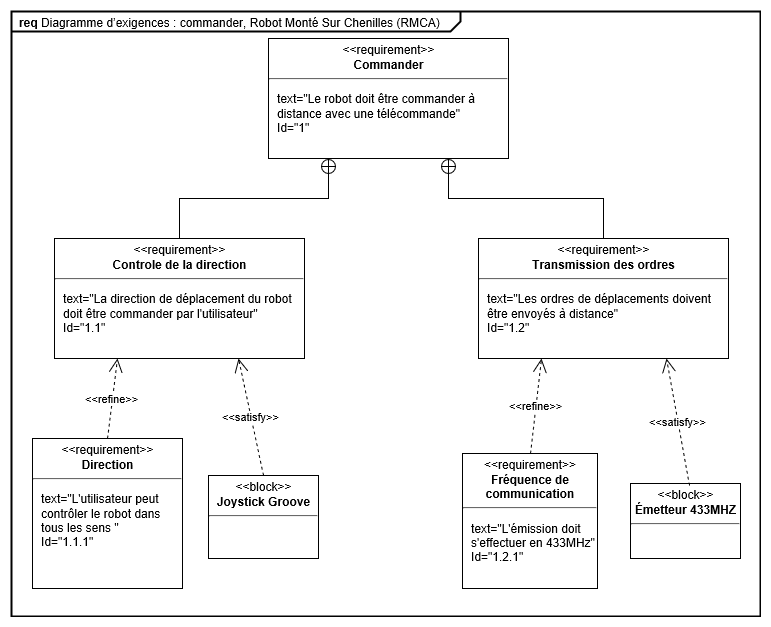
### Diagramme de cas d’utilisation

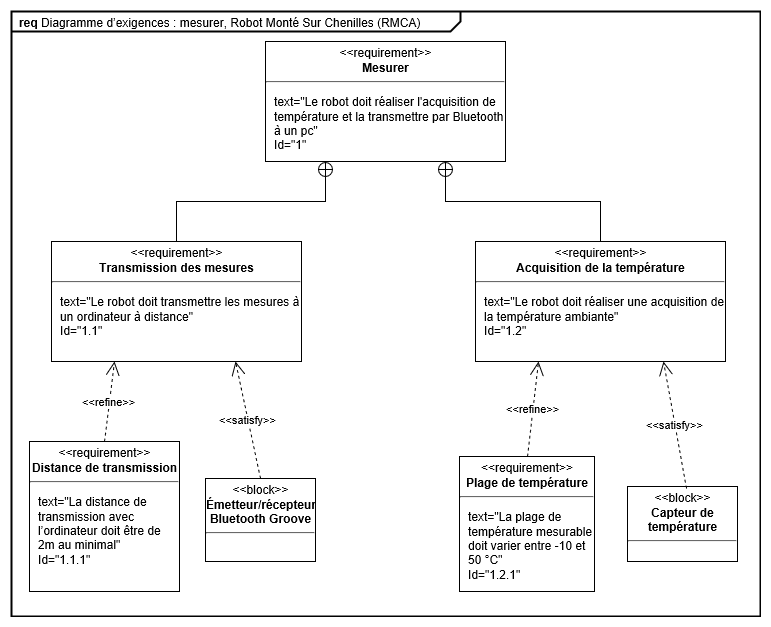
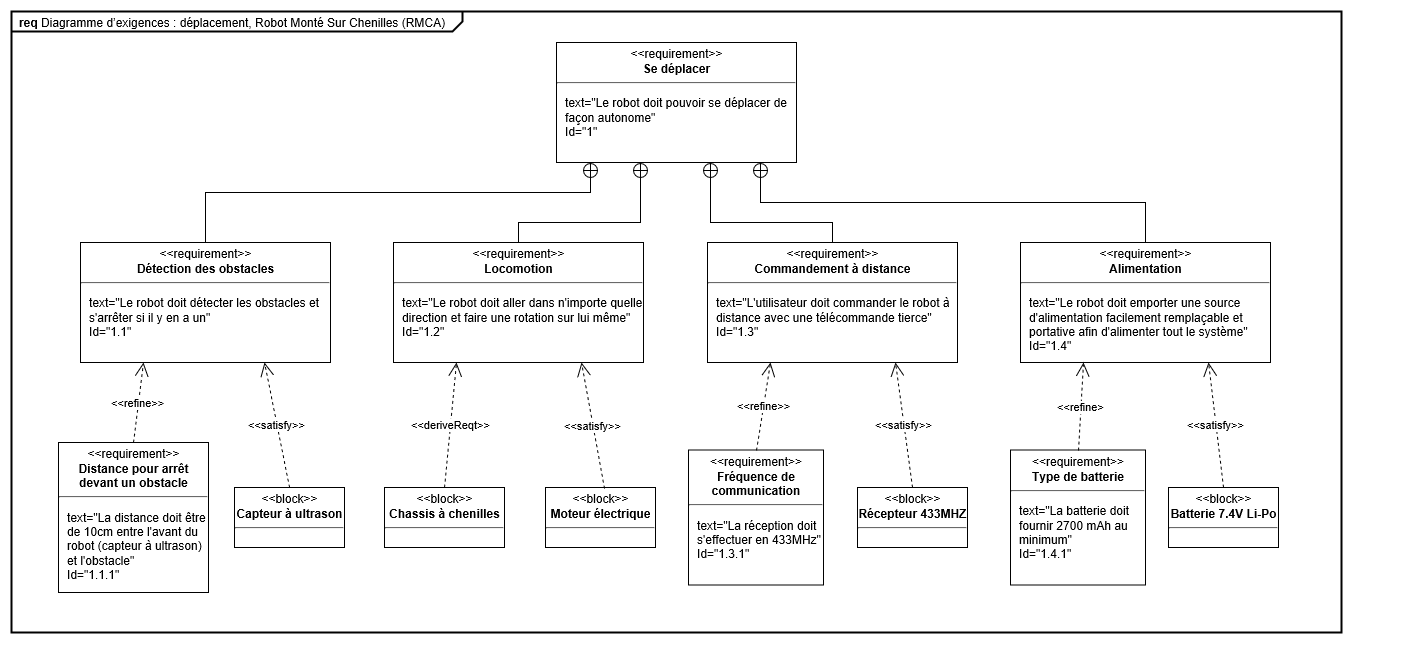
Le schéma suivant présente le diagramme de cas d’utilisation du système



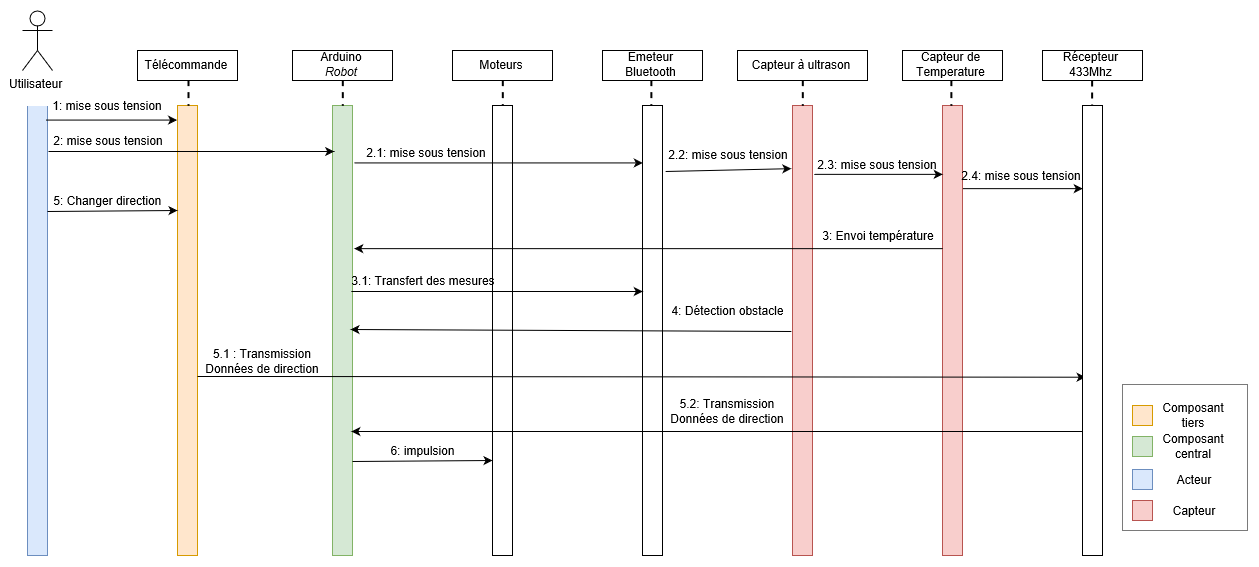
### Diagramme d’exigences

Les schémas suivants présentent les diagrammes d’exigences du système

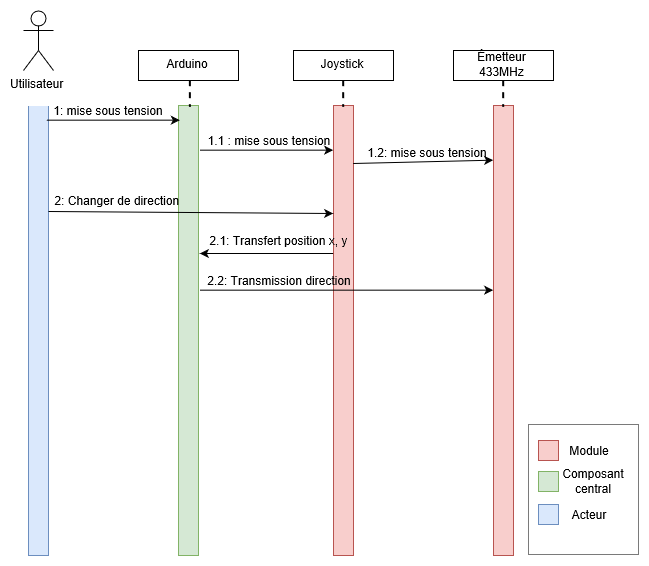




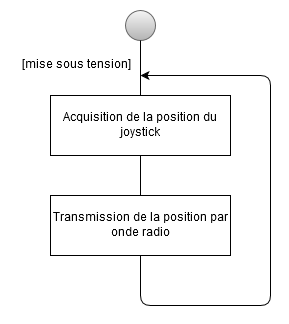
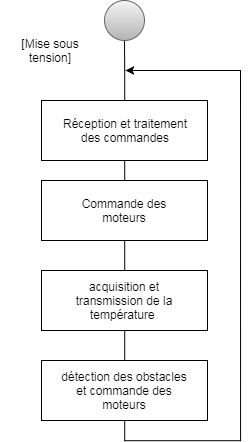
### Diagramme séquences

Le diagramme de séquences du robot est le suivant :

Télécommande :

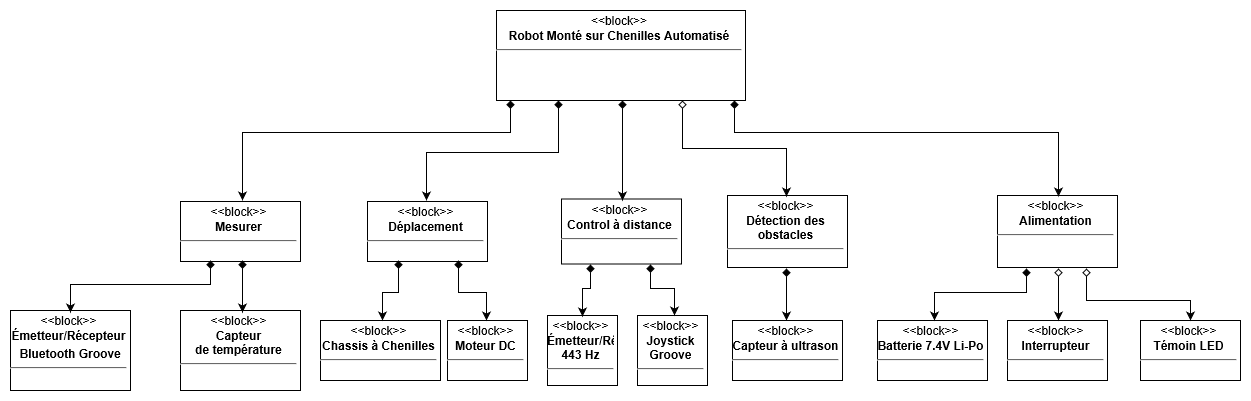


### Diagramme d’états



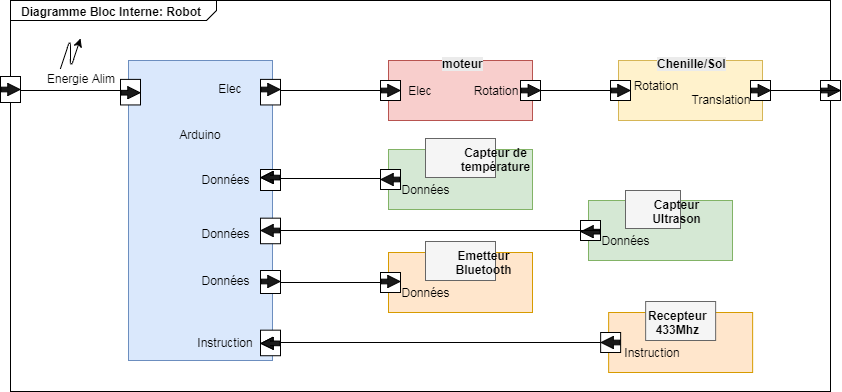
Digramme d’état robot  digramme d’état télécommande

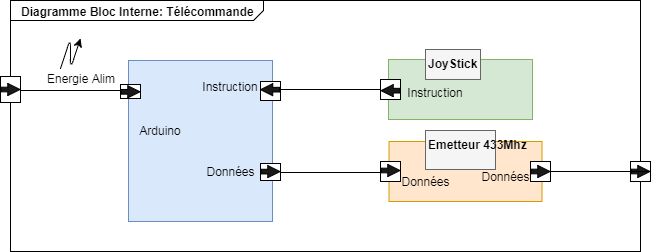
### Diagramme définitions block



### Diagramme block interne

Diagramme du robot :



Diagramme de la Télécommande :

## Montage électrique

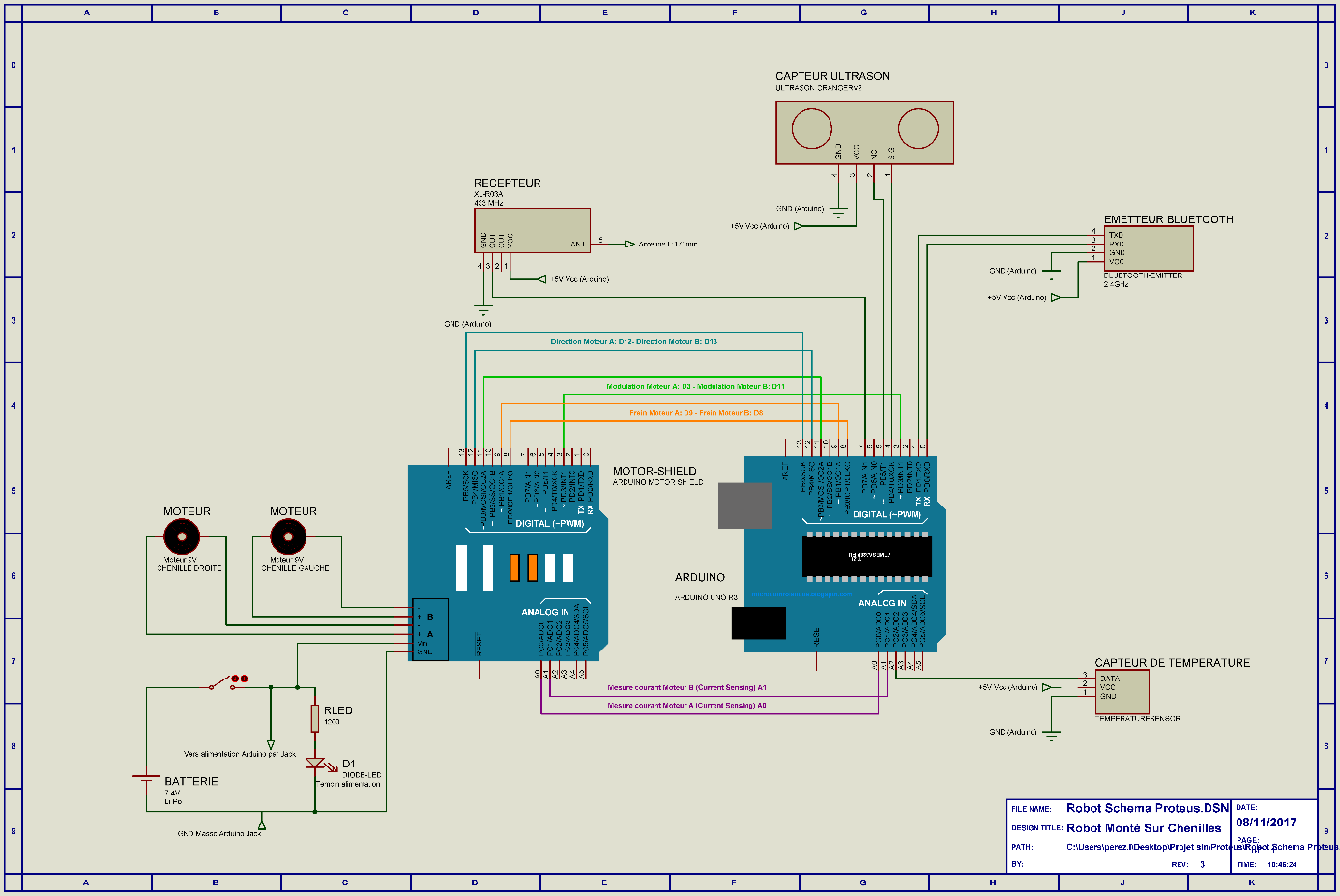
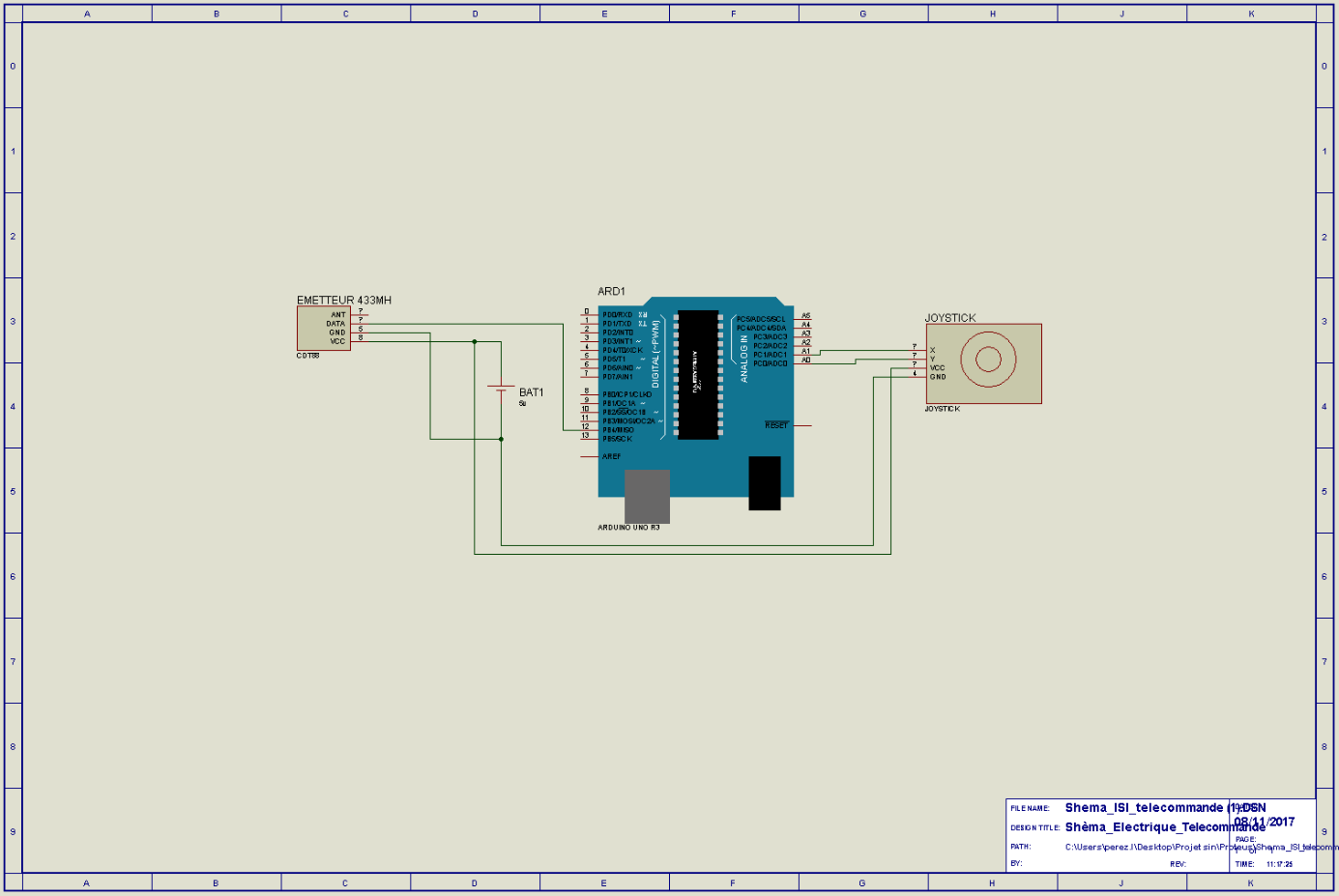
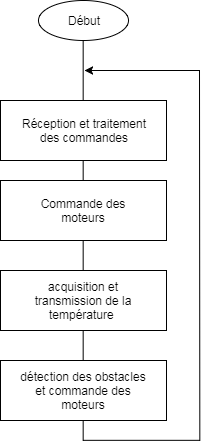
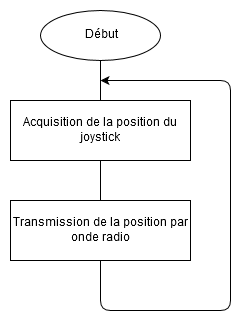
Schéma du montage électrique du robot :  

Schéma du montage électrique de la télécommande : 

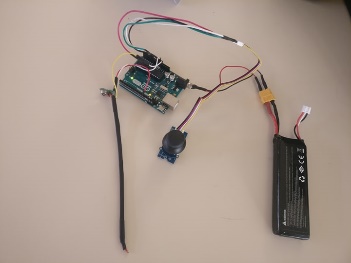
## Algorigramme

Robot : télécommande :



# Répartition des tâches

## Télécommande



Le robot est contrôlé à partir d’une télécommande utilisant des ondes radio de fréquence 433 Mhz et piloté par un joystick pour ses mouvements.

Ensemble de la télécommande

### Les modules radio

Pour établir une communication entre les deux systèmes une modulation du signal en signal radio est nécessaire. Cela consiste en la transformation du signal de sa forme originale en une forme adaptée au canal de transmission en faisant varier les paramètres d’amplitudes, la phase et la fréquence d’une onde sinusoïdale ; ici une transformation de filaire à hertzien et inversement, selon une modulation d’amplitude seulement, appelé ASK (amplitude shift keying).

Dans ce cas, j’ai à ma disposition deux modules communicants, un émetteur et un récepteur tous les deux de fréquence 433 Mhz. Les deux modules communiquent grâce à deux Arduino Uno et une librairie, nommé « Virtual Wire ».

L’ensemble RF est constitué d'un émetteur CDT88 et d'un récepteur CDR03A dont les caractéristiques sont les suivantes :

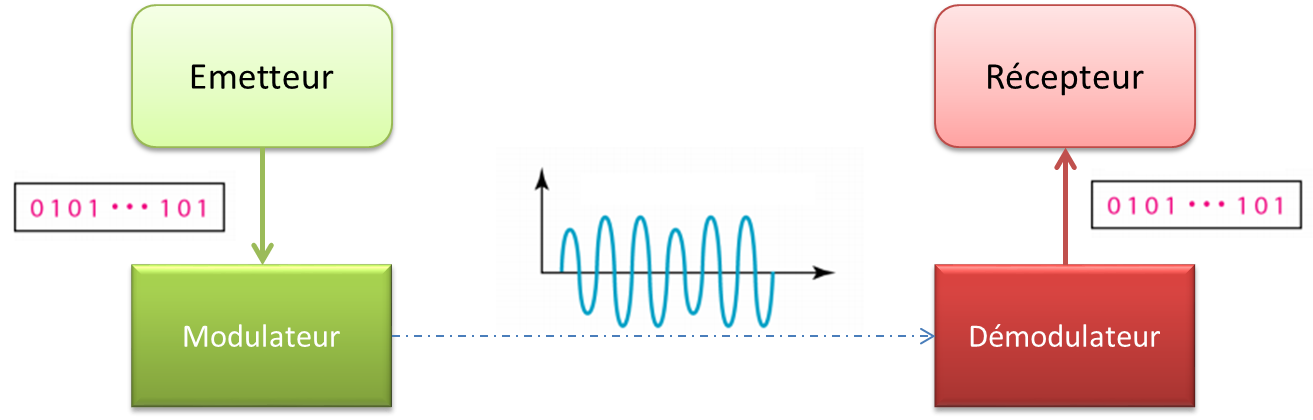


|  |  |
| --- | --- |
| Alimentation Emetteur | 3 à 12 Vcc |
| Alimentation Récepteur | 3,3 à 6 Vcc |
| Fréquence | 433 MHz |
| Modulation | ASK |
| Sortie récepteur | Etat Haut : ½ VCC  Etat Bas : 0,7 V |
| Portée maximale | 30m intérieur |

### Principe du système

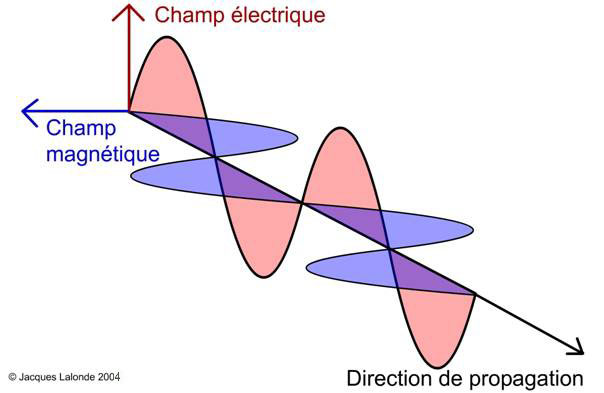
Le schéma suivant présente le principe d’émission réception :

L’émetteur, relié à la pin 12 de la carte Arduino prend en entrée un signal numérique qui est traduit en signal analogique pour la transmission. De son côté le récepteur prend le signal analogique et le transforme en un signal numérique interprétable par la carte Arduino localisée dans le char.



### Transmission des informations

#### Le signal transmis



Le signal analogique transmis est une onde électro magnétique de fréquence 433 MHz. Cette onde est donc constituée d’un champ électrique et d’un champ magnétique tous les 2 perpendiculaires au sens de propagation de l’onde :

Le champ électrique se mesure à l’aide d’un champmètre en volt. La mesure donne : XXXX V

Ainsi grâce à la mesure du champ électrique je suis en mesure de retrouver la distance max d’émission et la puissance du signal.

#### L’antenne

L’antenne joue un rôle important pour maximiser la transmission du signal. Il faut noter que la bande de fréquence de 433 MHz est une bande très utilisée dans les applications domestiques à faibles performances ou débits et cette bande de fréquence est donc souvent saturée (télécommandes, thermomètres extérieurs, alarmes sans fils, jouets).

Il existe différentes tailles d’antenne reposant toutes sur la longueur d’onde du signal, les antennes d’onde complète, les demi-ondes et les quarts d’onde. C’est cette dernière que j’ai retenue car je n’avais pas besoin de disposer d’une très grande portée.

Dans le cas de ce Montage, l’émetteur et le récepteur ont été équipés avec des antennes dites « quart d’onde » de 17,3 cm.

Calcul de la longueur de l’antenne :

Longueur d’onde :

Célérité =

Fréquence =

Conversation en quart d’onde :

Longueur d’onde = 69,28

#### Les différents types de modulation

La modulation permet de transmettre une information à distance, allant de quelques mètres à plusieurs kilomètres. La modulation permet d’adapter le signal au canal de transmission (filaire, hertzien, fibre optique).

Il existe deux grandes familles de modulation :

* Les modulations analogiques
* Les modulations numériques

Un signal est de forme : Pp(t) = Pp.cos (2.π.fp.t + β)

Il existe 3 catégories de modulations :

* modulation **d'amplitude** : on varie le paramètre Pp
* modulation de **fréquence** : on varie le paramètre fp
* modulation de **phase** : on varie le paramètre β

La modification de l’amplitude produira une onde plus ou moins haute :



La variation de la fréquence produira une onde plus ou moins compressée :



La variation de phase produira une onde plus ou moins décalée dans le temps.

#### Les modulations analogiques

Les modulations analogiques permettent de transmettre une information analogique telle que voix, musique et image avec une qualité suffisante dépendant du canal de transmission.

Les modulations existantes sont :

|  |  |  |
| --- | --- | --- |
| Nom | Type | Description |
| Modulation d'amplitude (AM) | Amplitude | C’est une onde utilisée en pour les radio, et l’onde est obtenue en multipliant directement la porteuse par le signal |
| Modulation d'amplitude à bande latérale unique (BLU) | Amplitude | C’est une modulation d’amplitude comportant une seule raie spectrale. Elle est utilisée en radiotéléphonie HF (Haute Fréquence) et VHF (Très Haute Fréquence) et en modulation de multiplex hertziens |
| La modulation de fréquence ( FM ) | Argument | L’onde résultante garde la même amplitude mais sa fréquence varie en fonction du signal. L’onde est moins influencée par les perturbations de type bruits ou interférence. Elle est utilisée en radio. |
| La modulation de phase ( PM ) | Argument | C’est une modulation de phase entraînant une modulation de fréquence, très utilisée en radiotéléphonie VHF et UHF |

#### Les modulations numériques

Les modulations numériques permettent de transmettre des informations numériques. Elles correspondent à une variation de **fréquence** de l’onde porteuse d’une information numérique.

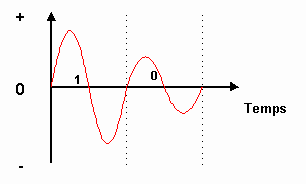
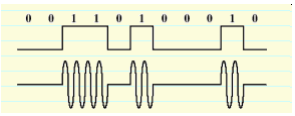
Les modulations existantes sont :

|  |  |
| --- | --- |
| Nom | Description |
| La modulation en tout-ou-rien (OOK ou CW) | OOK signifie On Off Keying. Elle présente des durées variables et est utilisée en télégraphie (code Morse). Elle présente l’avantage d’être reconnaissable auditivement par un opérateur (si couplée à une fréquence auditive). |
| La modulation par commutation d'amplitude (ASK) | L’amplitude est dépendante de plusieurs valeurs discrètes, représentant les 1 et les 0 **(Voir plus loin)** |
| Modulation en FSK | FSK signifie Frequency-Shift Keying. Cette modulation permet de distinguer le symbole zéro de l’absence d’information ce qui permet la transmission asynchrone. En revanche, la bande occupée est importante et ne permet pas de transmettre des informations avec un débit important. |
| Modulation en APK | APK signifie Amplitude Phase Keying. On fait donc varier en même temps l’amplitude et la phase. |

#### La modulation ASK

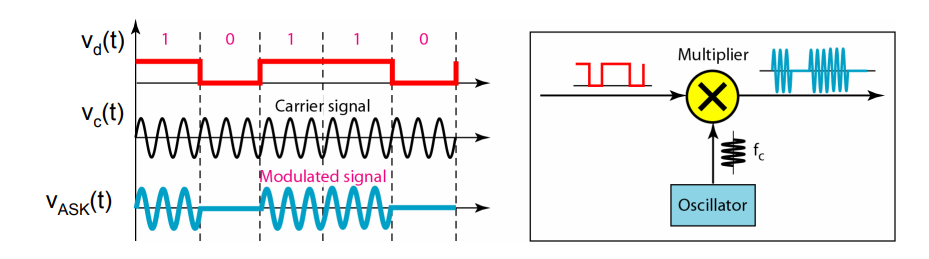
La modulation d’amplitude s’applique en faisant varier l’amplitude du signal en fonction des bits à coder. Ainsi l’amplitude du signal de transmission d’un « 1 » sera différente de l’amplitude de transmission d’un « 0 ».

Par exemple :

 ou 

Dans la suite nous prendrons le cas où l’amplitude est positive pour transmettre un « 1 » et nulle pour transmettre un « 0 » (2nd cas ci-dessus).

Le schéma suivant présente le principe d’encodage du signal, qui est dans ce cas en tout ou rien. En entrée, on retrouve un signal numérique donc chaque bit à une durée égale correspondant à une période, ce signal étant « multiplié » par une sinusoïde représentant le signal numérique.

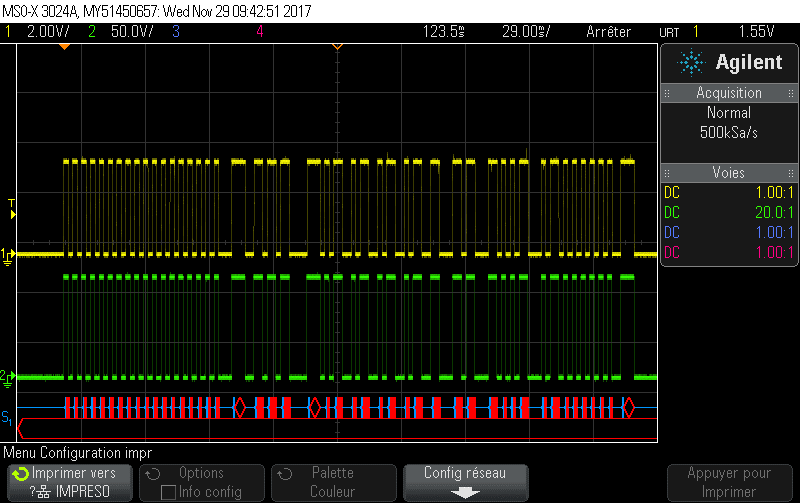


A noter que la modulation ASK est très sensible aux bruits dûs aux interférences d’autres signaux.

### Composition du signal : les trames

De façon à transmettre un message numérique, il est nécessaire de coder ce message composé d’octets en une suite de bits. Pour ce faire, j’ai utilisé la librairie VirtualWire disponible pour Arduino. Cette librairie offre des API (fonctionnalités) permettant d’envoyer de courts messages via des transmissions sans fil comme ASK.

Cette librairie fabrique des trames de signaux

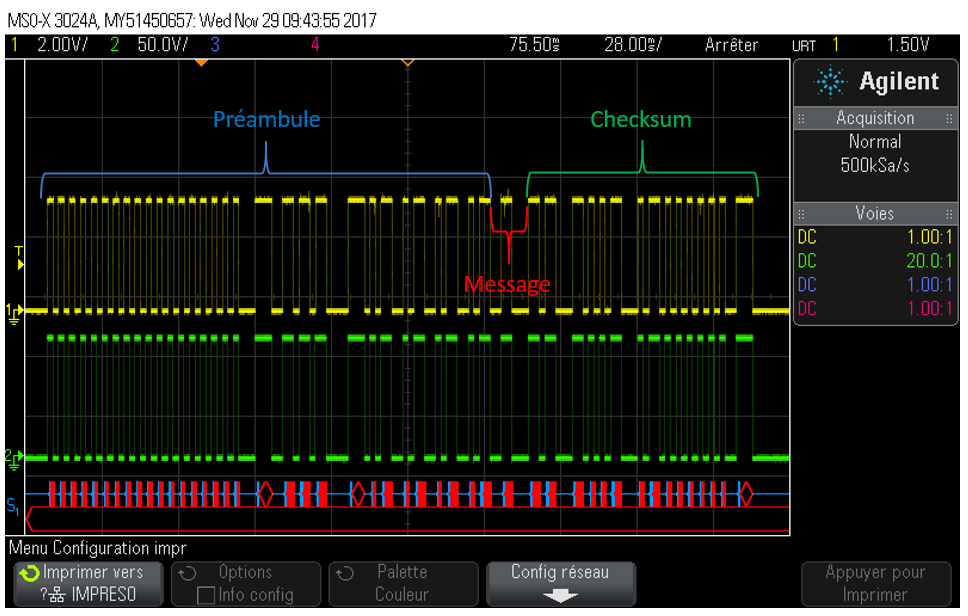


Trame des émetteur récepteur (émetteur jaune, récepteur vert)

Une trame est composée :

* D’un entête
* Du message
* D’un checksum permettant de vérifier l’intégrité du message envoyé.

Voici donc le même exemple de trame, analysé en fonction des trois parties précédentes :



La trame est constituée comme suit, sachant qu’un « byte » est constitué d’un groupe de 2x6 bits

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| bit | 36 | 24 | 12 | 12\*N | 24 |
| byte | 36 bits | 2 bytes | 1 byte | N bytes | 2 bytes |
| contenu | Préambule | Symbole de début 0xb38 | Longueur de message (4 à 30 bytes), incluant ce byte et le checksum | Message, avec un maximum de 27 bytes | Checksum |

Un message a donc une longueur maximum de 408 bits et est transmis en 0,204s à un taux de 2000 bps.

### Composition des messages

Pour le fonctionnement du système de transmission je crée des valeurs de commande 1,2,3 et 4 correspondant respectivement aux ordres avancer, reculer, gauche et droite. Ce sont donc ces quatre digits qui sont transmis comme information entre l’émetteur et le récepteur.

### Description du programme

L’ensemble émetteur récepteur fonctionne avec deux Arduino indépendantes, donc deux programmes différents, grâce à une transmission de valeurs de commande, des chiffres entiers.

Télécommande :

Code complet issu de la télécommande pour la partie émission.

#include <VirtualWire.h>

**int** joyPin1 = 0; // Definition de l'axe x du jostick

**int** joyPin2 = 1; // Definition de l'axe y du jostick

**int** x = 0; //initialisation de la variable de l'axe X

**int** y = 0; //initialisation de la variable de l'axe Y

**float** valeur = 0; //initialisation de la valeur de commande

**void** setup() {

Serial.begin(9600); // definition du baudrate du serial

vw\_setup(500); // initialisation de la librarie virtual wire

}

**void** loop() {

// lecture des entre du joystick

x = analogRead(joyPin1);

y = analogRead(joyPin2);

valeur = 0; //definition de la valeur de commande a 0

delay(200); //Attente pour eviter de surcharge l'emetteur en envoie

//Association des valeur du joystick avec la valeur de commande

//Pour les donnee pour l'axe X (avant, arriere)

**if** (x < 500){

Serial.println("avancer");

valeur = 1;

}**else** **if**( x > 510){

Serial.println("reculer");

valeur = 2;

}**else**{

}

//Pour les donnee pour l'axe y (droite, gauche)

**if** (y < 500){

Serial.println("gauche ");

valeur = 3;

}**else** **if**( y > 510){

Serial.println("droite");

valeur = 4;

}**else**{

}

vw\_send((byte \*) &valeur, **sizeof**(valeur)); // On envoie le message

vw\_wait\_tx(); // On attend la fin de l'envoi

}

Robot :

Dans le robot le code se divise en deux parties, l’initialisation et la réception.

*Extrait de la fonction d’initialisation du récepteur :*

**void** setupReceiver() { //fonction d’initialisation du récepteur pour la télécommande

vw\_set\_rx\_pin(broche\_rx); //On configure la broche de réception (broche 7) pour recevoir les commandes depuis la télécommande, cette broche est relier au OUT du module Récepteur 433Mhz

vw\_setup(500); // Commencer la liaison avec une vitesse de réception de 500 bits par secondes, utilise pour recevoir les commandes depuis la telecommandes

vw\_rx\_start(); // On commence à écouter sur la broche pour recevoir les commandes

}

Extrait de la fonction de réception du robot :

byte taille\_message = **sizeof**(**float**); //initialisation de la variable de la taille de message recues

**float** commandFm; // initisalisatoin de la variable des messages recues

// On attend de recevoir un message

**if** (vw\_get\_message((byte \*) &commandFm, &taille\_message)) {// recuperation du message en fonction son contenue et sa longueur

commandeReceived = (**int**) commandFm; // stockage du message recue dans une variable sous forme de nombre decimale

### Améliorations possibles

Les améliorations suivantes permettraient d’augmenter la portée du signal :

* Augmenter la tension d’alimentation de l’émetteur 🡪 Augmentation de la puissance
* Changer de fréquence
* Changer de module Emission / Réception
* Avoir des antennes plus performantes

## Commande des moteurs

Le robot fonctionne avec deux moteurs à courant continu (motoréducteur connecté à des chenilles) qui sont alimentés par le shield motor qui est lui-même connecté à une carte Arduino qui alimente entre 5 et 12 V. Il est contrôlé par un programme, des ponts en H et avec des pin qui se trouvent sur le microcontrôleur.

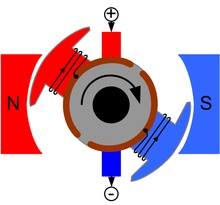
### Shield moteur

Ce type de composant est une carte de commande qui nous permet de contrôler simultanément le sens et la vitesse des moteurs via des pin et deux ponts en H. Ce driver convient pour des moteurs à courant continu, Brushed il est possible de l’alimenter avec 2A max par moteur ou 4A avec une alimentation extérieure comme c’est le cas sur notre montage. Il est aussi composé d’un contrôleur moteur L298.

### Moteur

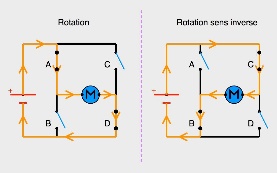
Les moteurs à courant continu sont composés de :

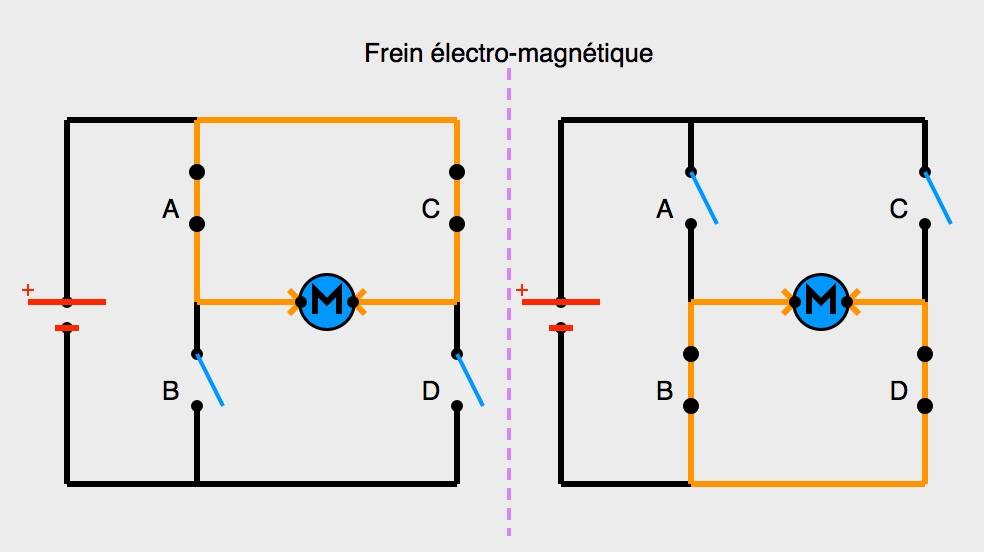
* Un stator qui est la partie fixe (statique) du moteur. Il est constitué d’un électro-aimant alimenté en courant continu dont le rôle est de produire un champ magnétique. Le stator fixé sur la carcasse cylindrique du moteur entoure la seconde partie du moteur qui est le rotor. (Stator est aussi appelé inducteur)



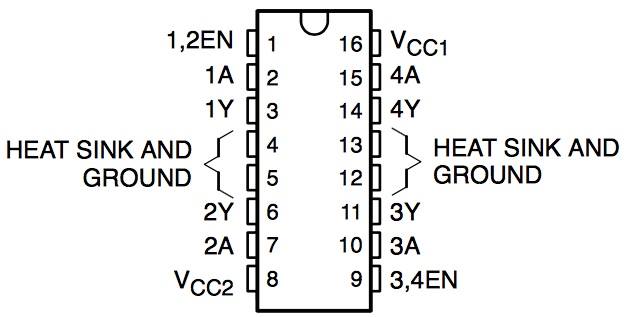
* Un rotor placé à l’intérieur du stator est la partie tournante (rotative) du moteur. Il est constitué d’un cylindre portant des bobinages disposés latéralement sur sa surface. Les extrémités des fils de ces bobinages sont reliées au collecteur qui entoure l’arbre moteur. (rotor est aussi appelé induit)
* Un collecteur qui comporte des contacts métalliques isolés les uns des autres qui assurent la distribution du courant continu aux bobinages du rotor par l’intermédiaire de deux balais.
* Des balais parfois appelés charbons et l’ensemble connecteur-balais est la partie la plus fragile du moteur.

### Pont en H

Grâce à ce pont en H je peux donc gérer le sens des moteurs à courant continu grâce à la connexion des différents interrupteurs ici présents sous forme de schémas.

On peut aussi associer les interrupteurs comme le montre le schéma ci-contre. Il nous montre que le moteur ne reçoit pas de courant, il est donc en roue libre (il tourne donc grâce à la force de l’inertie du robot). Il va donc créé un courant qui va pouvoir être utilisé dans ce pont en H comme frein électro-magnétique. Le moteur s’envoie son propre courant à l’envers cela permet l’arrêt du moteur plutôt que de la laisser en roue libre

### L298P



Ce composant est le contrôleur des moteurs CC. Il possède un double pont en H capable de délivrer 2A par pont fonctionnant jusqu’à 46V.

• pin 1 : active la partie gauche si l'on y envoie un état haut (+5V dans notre cas) et la désactive à l'état bas. On peut donc l'utiliser pour envoyer un signal PWM.

• pin 2 : c'est le pin de commande du demi pont 1 (là où on envoie le courant au transistor pour qu'il laisse ou non passer le courant).

• pin 3 : on branche une patte du moteur ici.

• pin 4 : c'est le GND (et le radiateur, ou dissipateur de chaleur). On y branche l'autre patte d'un moteur si on l'utilise que dans un sens. Il faut le relier au GND de l'Arduino.

• pin 5 : GND pour le demi pont 2 (il est d'ailleurs conseillé de relier tous les GND utilisés ensemble).

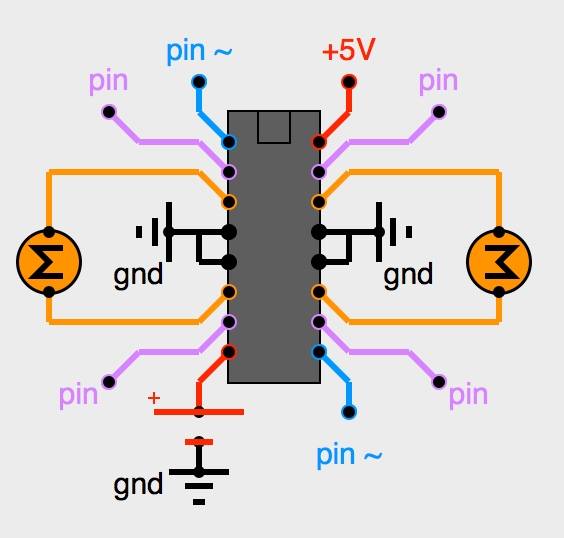
• pin 6 : on connecte l'autre patte du moteur (si utilisation dans les deux sens).

• pin 7 : commande du demi pont 2.

• pin 8 : c'est ici que l'on connecte la source d'alimentation des moteurs (circuit de puissance).

• pin 9 : active la partie droite (donc demie pont 3 et 4). Possible en PWM.

• pin 10 : commande du demi-point 3.

• pin 11 : connexion de l'autre patte.

• pin 12 et 13 : GND.

• pin 14 : connexion d'une patte du second moteur.

• pin 15 : commande du demi pont 4.

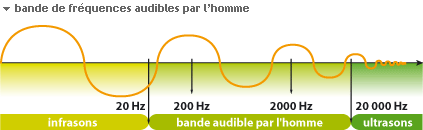
• pin 16 : c'est ici qu'on connecte le +5V de référence (circuit de commande).

## Capteur ultrason

### Les ondes ultrason

Les ultrasons sont des ondes mécaniques se propageant à la vitesse de ~340m.s-1  dans l’air.

La gamme de fréquences des ultrasons se situe entre 16kHz et 10 MHz, trop élevées pour être perçues par l'oreille humaine.



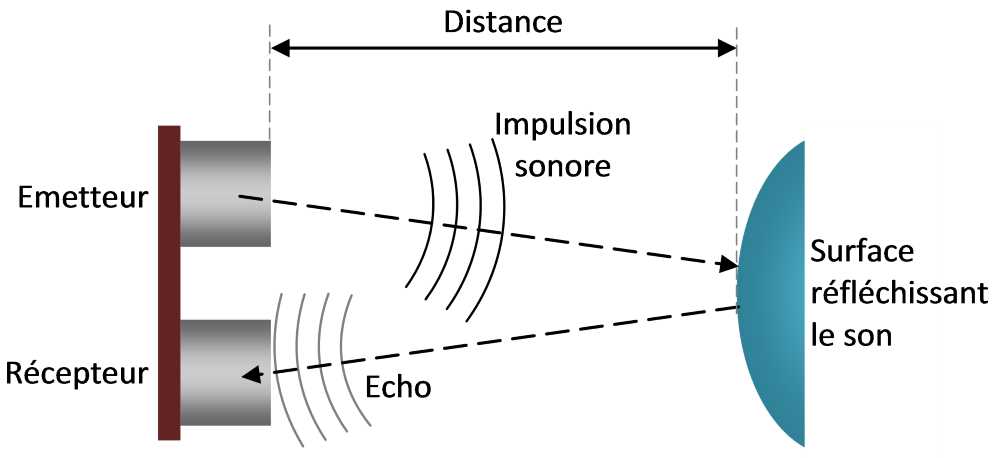
Les ultrasons sont utilisés dans d’autre domaine :

* La médecine avec les échographies
* L’armée avec les sonars

Certains animaux tels que les chauvesouris ou les cétacés émettent des ultrasons pour faire de l’écholocalisation.

### Le capteur à ultrasons

Un capteur à ultrasons émet à intervalles réguliers de courtes impulsions sonores à haute fréquence. Ces impulsions se propagent dans l’air à la vitesse du son. Lorsqu’elles rencontrent un objet, elles se réfléchissent et reviennent sous forme d’écho au capteur. Celui-ci calcule alors la distance le séparant de la cible sur la base du temps écoulé entre l’émission du signal et la réception de l’écho.



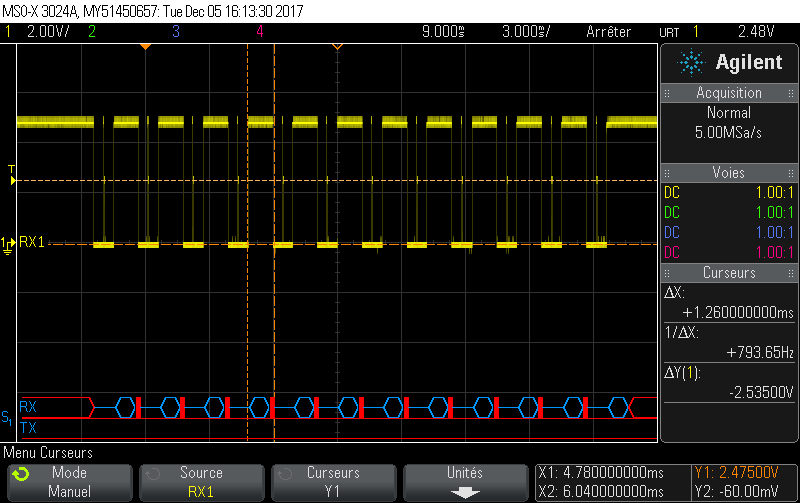


Ce ranger Grove Ultrasonic est un module de mesure de distance sans contact qui fonctionne à 40KHz. Lorsque nous fournissons un signal de déclenchement d'impulsion avec plus de 10uS à travers une broche unique, le Grove Ultrasonic Ranger émettra 8 cycles de 40 kHz et détectera l'écho. La largeur d'impulsion du signal d'écho est proportionnelle à la distance mesurée.

La formule est la suivante :

*Distance = signal d'écho temps élevé \* Vitesse du son (340M / S) / 2.*

Le trig et l'écho singal de Grove Ultrasonic Ranger partagent 1 broche SI



Vitesse du son=340m.s-1

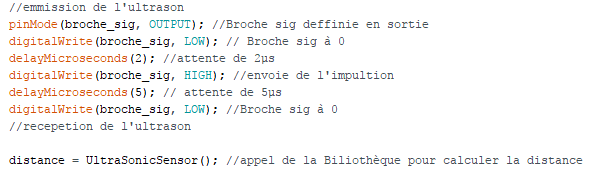
Largeur d’impulsion de l’onde = 1.26ms

D = (v\*t)/2

D = (340\*1.26\*10-3 )/2

D = 0.21m soit 21cm

### Le code

Utilisation de la bibliothèque ''ultrasonic.h''

## Capteur de température et Bluetooth

### Le capteur de température

#### Présentation du capteur P/N 1124:

Le **P/N 1124** est un capteur de température de type **semi-conducteur.**

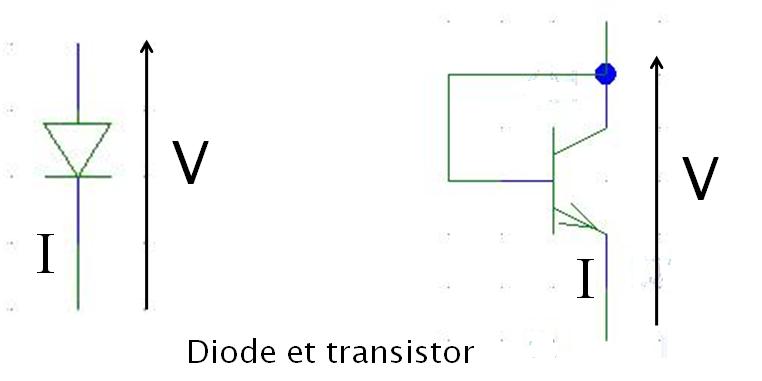
Il est donc un capteur intégré. Le **P/N 1124** transforme directement la température à mesurer en grandeur électrique grâce à un conditionneur. Il est donc nécessairement alimenté.

#### Fiche Technique

P/N 1124

|  |  |
| --- | --- |
| Signal en sortie | Analogique |
| Capteur de température | |
| Température Min | -30 °C |
| Température Max | 80 °C |
| Précision de la mesure | 0.2 °C |
| Marge d’erreur typique (à 25°C) | ± 0.5 °C |
| Margeur d’erreur maximal | ± 2 °C |
| Propriétés électriques | |
| Tension Alim Min | 4 V DC |
| Tension Alim Max | 6.5 V DC |
| Consommation Max | 1 mA |
| Voltage en Sortie Min | 250 mV DC |
| Voltage en Sortie Max | 4.8 V DC |
| Tension par degré | 22,5mV |
| Propriétés physiques | |
| Température Min De Fonctionnement | -40 °C |
| Température Max De Fonctionnement | 105 °C |
| **Référence tension en sortie** | |
| Température mesurée :-50°C | 0,25V |
| Température mesurée : 0°C | 1,37V |
| Température mesurée : +150°C | 4,75V |

#### Principe de fonctionnement du capteur de température :

Comme il est écrit ci-dessus, le **P/N 1124** estun capteur de type semi-conducteur. Ces derniers ressemblent à des transistors, ils utilisent une propriété importante des diodes et des transistors :

La tension à leurs bornes, lorsqu’ils sont traversés par un courant constant, dépend de la température. D’ordinaire, ce phénomène est gênant, ici on en tire parti.

#### Autres principes de capteurs de température disponibles sur le marché

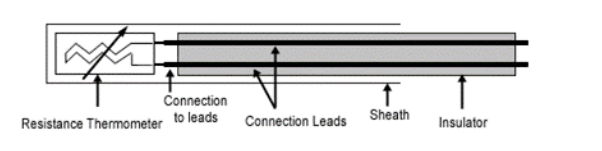
##### Les thermocouples :

Contrairement aux capteurs de températures de type semi-conducteur, les thermocouples n’ont pas besoin de composant électronique pour fonctionner. Ils sont simplement fabriqués à partir de deux câbles assemblés. Grâce à un effet physique (l’effet Seebeck) pour deux métaux de natures différentes assemblés, il y a une mince, mais mesurable tension qui passe entre les deux câbles et qui augmente en fonction de la température. Les types de métaux utilisés affectent la plage de tension mesurée, le coût et la sensibilité, c’est pourquoi il y a plusieurs types de thermocouples.

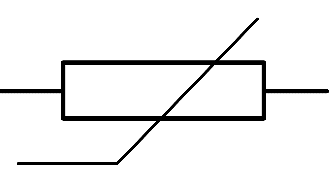
##### Par résistance métallique, les sondes RTD

Les RTD (Resistance Temperature Detectors - capteurs de température à résistance) fonctionnent sur le principe des variations de résistance électrique des métaux purs et se caractérisent par une variation positive linéaire de la résistance en fonction de la température. Concrètement, une fois chauffée, la résistance du métal augmente et inversement une fois refroidie, elle diminue. Les éléments types utilisés pour les RTD incluent le nickel (Ni) et le cuivre (Cu) mais le platine (Pt) est de loin le plus courant, en raison de l’étendue de sa gamme de températures, de sa précision et de sa stabilité. Faire passer le courant à travers une sonde RTD génère une tension. En mesurant cette tension, on peut déterminer la résistance et ainsi, la température.

Les RTD sont utilisées dans les thermomètres médicaux pour mesurer la fièvre par exemple.



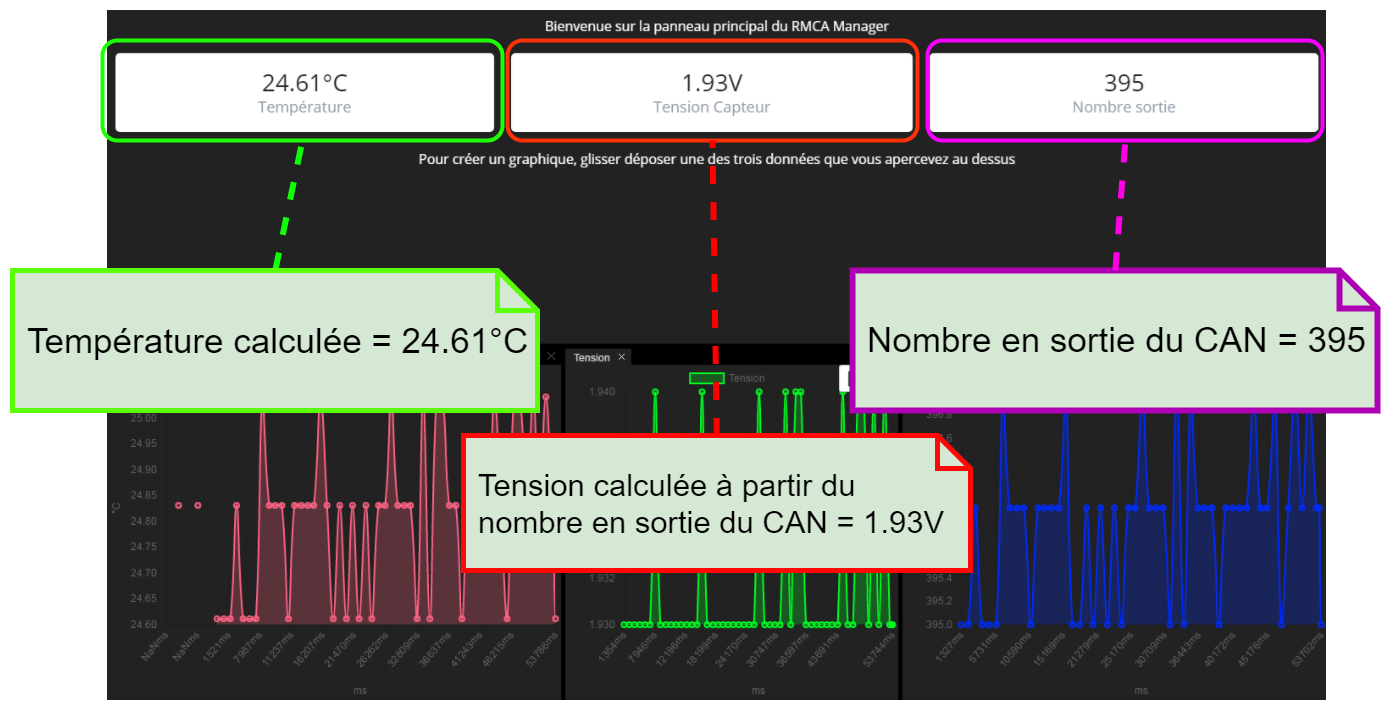
##### Les thermistances :

Les thermistances, comme les capteurs de température à résistance (RTD), sont des conducteurs thermosensibles dont la résistance varie avec la température. Les thermistances sont constituées d’un matériau semi-conducteur d’oxyde métallique encapsulé dans une petite bille d’époxy ou de verre. En outre, les thermistances présentent généralement des valeurs de résistance nominale plus élevées que les RTD (de 2 000 à 10 000 Ω) et peuvent être utilisées pour de plus faibles courants.

Symbole utilisé pour les thermistances

#### Mesure de la température : expression des calculs

La figure ci-dessous présente la visualisation des mesures sur l’ordinateur. L’interface graphique a été développée spécifiquement dans le cadre du projet.

  
Capture d'écran du RMCA Manager

L’interface délivre 3 informations :

|  |  |
| --- | --- |
| **Nom :** | **Valeur :** |
| Température calculée | 24,61°C |
| Tension calculée | 1,93 V |
| Nombre en sortie du CAN | 395 |

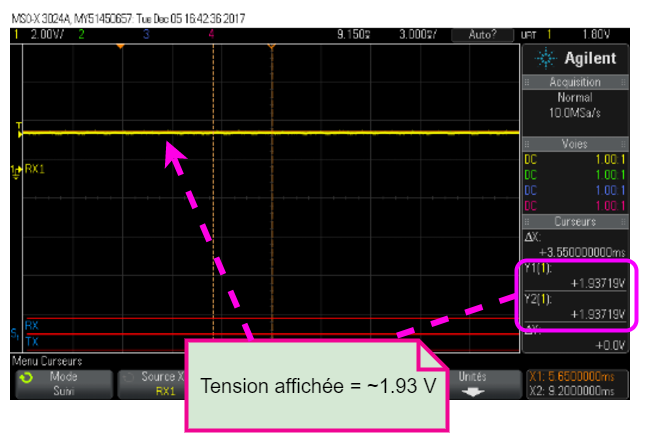
**Expression des calculs permettant d’accéder à la valeur de température :**

La sortie du capteur de température est un signal analogique. On cherche à l’utiliser pour calculer la température.

La première chose à obtenir est le voltage en sortie du CAN (Convertisseur Analogique Numérique) de l’Arduino. La formule est la suivante :

Sachant que le CAN de l’Arduino Uno possède une résolution de **10 bits,** que le **nombre en sortie du CAN est égal à 395** et que la carte est alimentée en **5 volts**:

Relevé de la tension en sortie du capteur avec l’oscilloscope :



La tension obtenue en résultat du calcul est bien celle affichée sur l’oscilloscope et l’ordinateur, elle est donc vérifiée.

On veut déduire la température :

On sait qu’à 0°C la tension en sortie du capteur est de 1,37 V. Par ailleurs, la tension varie de 22,5mV/°C, donc :

La température calculée (24,9°C) correspond pratiquement à celle affichée sur l’ordinateur (24,6°C). La différence de 0,3°C est liée à la précision de mesure des données, et aux arrondis appliqués. **Les mesures affichées sur l’ordinateur sont donc valides.**

### Transmission des mesures :

#### Le module Serial Bluetooth Grove:

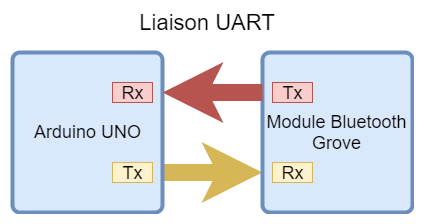
Le module Bluetooth Groove sert à transmettre les mesures du capteur de température à l’ordinateur. Ce module est alimenté en 5V et communique avec l’Arduino en liaison série UART.

Données techniques :

|  |  |
| --- | --- |
| Tension Alimentation | 5,0V DC |
| Puissance de transmission Max | +4dBm |
| Portée | 10 à 20 mètres |
| Sensibilité | -80dBm |
| Version Bluetooth | V2.0 |

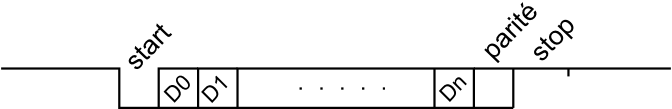
#### Communication entre la carte Arduino UNO et le module Bluetooth.

Comme il est écrit ci-dessus, la communication avec le module s’effectue en liaison UART **(Universal Asynchronous Receiver Transmitter)**. Ce protocole de communication utilise deux broches, une de réception (Rx), pour recevoir des données, et une autre de transmission (Tx), pour transmettre des données.



Une trame UART sur la broche de réception du module est constituée des bits suivants :

* Un bit de *start* toujours à 0 : servant à la synchronisation du récepteur
* Données : la taille des données est comprise entre 5 et 9 bits. (8 bits sur l’Arduino UNO)
* Parité : Paire ou Impaire (optionnel) **(sur l’Arduino UNO la parité n’est pas comprise)**
* Fin : Un bit de stop, toujours à 1.



Constitution d'une trame UART

La vitesse de transmission (le baud) en bit par seconde peut varier. Sur le module Bluetooth Grove, elle est à 9600bps par défaut.

##### Mise en forme des données puis envoi vers l’ordinateur

Afin d’être envoyées à l’ordinateur, les données sont mises en forme pour être interprétées par l’ordinateur. Les données sont structurées en JSON **(JavaScript Object Notation).**

  
Structure en JSON de l’envoi de la mesure de type « Température », encodé en ASCII

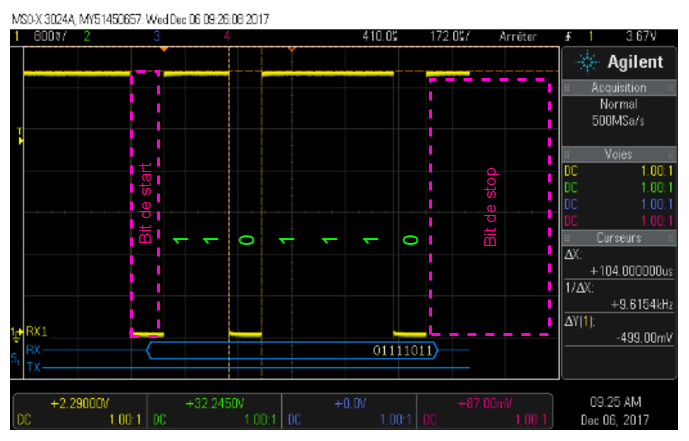
Il y a deux champs envoyés, encodés en ASCII :

* le champ « type » : correspond au type de mesure
* le champ « data » : correspond à la valeur de la mesure

Après avoir été reçu par l’ordinateur, le JSON est transformé afin d’afficher la valeur de la mesure.

##### Capture de la liaison UART avec l’oscilloscope :

Un oscilloscope a été branché sur la **broche Rx** du module Grove afin d’intercepter la communication entre la carte et le module



Décodage d’une trame UART sur la broche Rx. Envoi de 01111011 - 0x7B

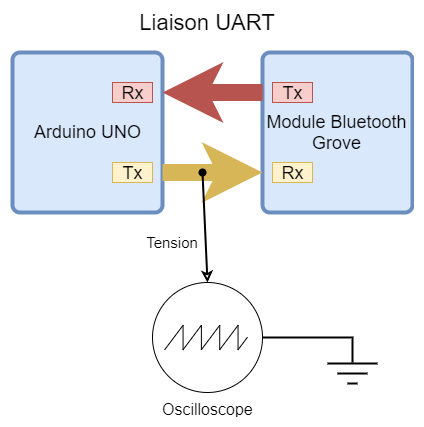
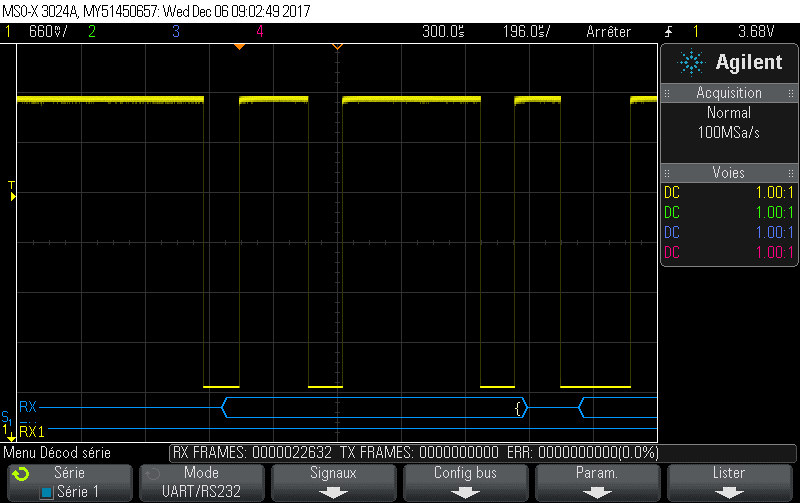


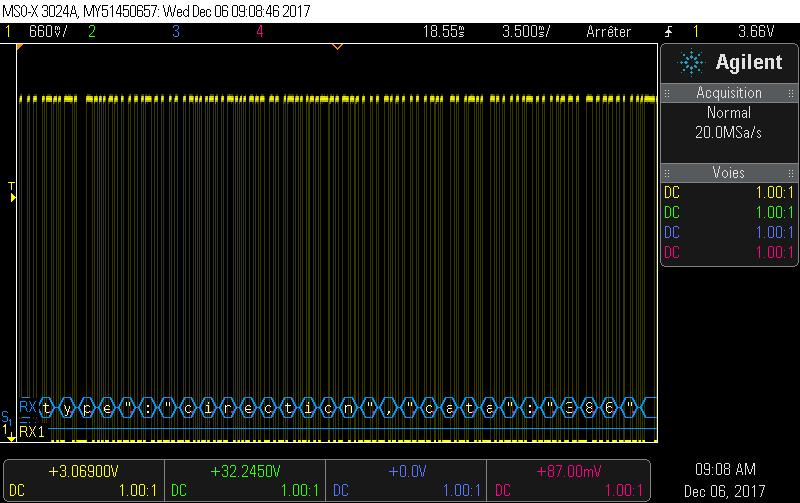
Schéma montage avec oscilloscope

Grâce à la table ASCII, on peut décoder 0x7B en caractère :

  
Caractère affiché: "{"

On en déduit le caractère «{», qui représente le premier caractère de l’objet JSON affiché plus haut.

Si on capture la communication entière :

  
Affiché : {"type": "direction", "data":"386"}

Ceci représente le nombre en sortie du CAN envoyé à l’ordinateur, égal à 386.

### Description du programme :

#### Partie acquisition de la température :

**float** readTemp() { // Fonction utilisée pour mesurer la température

**int** reading = analogRead(sensorPin); // On lit nombre à la sortie du CAN relié au capteur de température

/\* V

                      pe

       V = N x ----------

        e n

                     2 -1

\*/

sendInfo("direction", String(reading)); // On envoie par Bluetooth avec la fonction sendInfo() la mesure du nombre en sortie du CAN afin de l'afficher sur l'ordinateur

// On multiplie par la tension max qui peut être convertie, ici 5v

**float** voltage = reading \* 5.0;

// On divise par 2^(10)-1= 1023, 10: nombre de bit du CAN sur l’Arduino Uno. On obtient donc le voltage :

voltage /= 1023.0;

sendInfo("battery", String(voltage)); // On envoie par Bluetooth avec la fonction sendInfo() la mesure du voltage obtenue à partir du calcul afin de l'afficher sur l'ordinateur

**float** temperatureC = (voltage - 1.37)/(0.0225) //Conversion avec la formule

**if** (DEBUG) { // Si debug = 1

Serial.print(temperatureC); // Envoyer la température au moniteur série Arduino IDE

Serial.println(" degrees C"); // Envoyer l'unité et insérer une nouvelle ligne au moniteur série Arduino IDE

}

**return** temperatureC; // On retourne la valeur en °C qui a été mesurée

}

Le code C++ ci-dessus est un extrait du code Arduino du robot. C’est la fonction readTemp() qui convertit puis retourne la température au format **float.**

Cette fonction est appelée dans la boucle du programme Arduino afin d’envoyer la mesure:

//envoi de la température

sendInfo("temperature", String(readTemp()));

On appelle la fonction sendInfo() avec 2 paramètres : « temperature » ce qui correspond au type de mesure et une conversion en texte (String) du résultat de la fonction readTemp() (ce qui est notre valeur de la mesure).

#### Partie transmission des mesures

Intéressons-nous maintenant à cette mystérieuse fonction sendInfo() :

**void** sendInfo(String key, String data) { // Fonction utilisée pour envoyer les mesures par Bluetooth. 2 paramètre: ((String)key: le type de mesure (temperature, battery, direction), (String)data: la mesure à envoyer)

**int** placeArray = 0; // Cette variable est un nombre entier utilisé pour determiner dans quel index du tableau lastSendTime il faut conserver le temps en ms quand la mesure a été envoyée, permet de limiter l’envoi des mesures par Bluetooth à 1 mesure toute les 1 secondes

**if** (key == "temperature") // si le type de donnée est égal à "temperature", l'emplacement dans le tableau lastSendTime sera à 0

placeArray = 0;

**else** **if** (key == "battery") // si le type de donnée est égal à "battery", l'emplacement dans le tableau lastSendTime sera à 1

placeArray = 1;

**else** **if** (key == "direction") // si le type de donnée est égal à "direction", l'emplacement dans le tableau lastSendTime sera à 2

placeArray = 2;

**if** ( millis() - lastSendTime[placeArray] > rateLimitMs) { // Si le temps en ms depuis le démarrage du programme retourné par la fonction millis() moins le dernier temps d’envoi pour un certain type de donné est supérieur au dernier à 1000ms, alors :

StaticJsonBuffer<200> jsonBuffer; // On crée une nouvelle mémoire tampon de taille 200 caractères pour structurer en JSON les mesures à envoyer. Cela permet de stocker une valeur fixée de caractères dans la mémoire, et donc de fonctionner avec des appareils possédant peu de mémoire RAM.

JsonObject& root = jsonBuffer.createObject(); // On crée un nouveau objet JSON, utilisé pour structurer les mesures

root["type"] = key; // On ajoute la clef type, pour le type de mesure, et on lui affecte la variable key

root["data"] = data; // On ajoute la clef data, pour la valeur de la mesure, et on lui affecte la variable data

root.printTo(hc06); // On envoie l'objet « root » en texte au module Bluetooth par liaison UART et donc à l'ordinateur

lastSendTime[placeArray] = lastSendTime[placeArray] + rateLimitMs; // Afin de limiter l’envoi de données, on affecte au tableau lastSendTime à l'index du type de donnée: le dernier temps d’envoi + 1000ms (variable rateLimitMs), afin de limiter à 1 mesure par seconde.

}

}

Cette fonction permet l’envoi d’une mesure au format JSON d’un certain type et permet de limiter son envoi à 1 mesure par seconde. Le tableau lastSendTime est déclaré globalement et est de type int.

La variable hc06 représente la liaison série avec le module Bluetooth. Elle est définie avec :

SoftwareSerial hc06(2, 6); //TX RX, Déclare une nouvelle instance SoftwareSerial, sur les broches TX: 2, RX:6, utilisé pour communiquer avec le module Bluetooth

La liaison UART est établie avec la fonction setupBluetooth(), qui est elle-même appelée au démarrage de la carte Arduino.

**void** setupBluetooth() { // setupBluetooth(), pour initialiser le Bluetooth

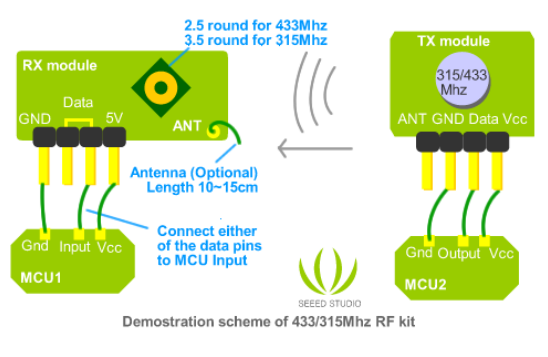
hc06.begin(9600); // Initialiser la liaison avec une vitesse de transmission de 9600 bits par seconde, utilisé pour communiquer avec le module Bluetooth en liaison UART et donc envoyer les mesures à l’ordinateur

hc06.print("AT+NAMERMCA"); // On envoie au module Bluetooth le message "AT+NAMERMCA", ce message est une commande pour le module Bluetooth (tout message qui commence par AT est une commande pour configurer le module), cette commande est NAME, elle configure le nom afficher au autres péripheriques Bluetooth, ici "RMCA". Il y a d'autre type de commande, comme PIN, VERSION etc.

}

# Annexe 1 :

Schéma de principe de l’émetteur et du récepteur



# Bibliographie

<https://web.stanford.edu/class/ee102b/contents/DigitalModulation.pdf>

<http://www.airspayce.com/mikem/arduino/VirtualWire.pdf>

<https://fr.wikipedia.org/wiki/Modulation_du_signal>

<http://denis.michaud.free.fr/cd_meteosat/pages_HTML/Electronique/demodAM/hf/modulation/modulation.htm>

# Remerciements

Nous remercions M. BO\*\*\* pour ses conseils tout au long des différentes séances.