

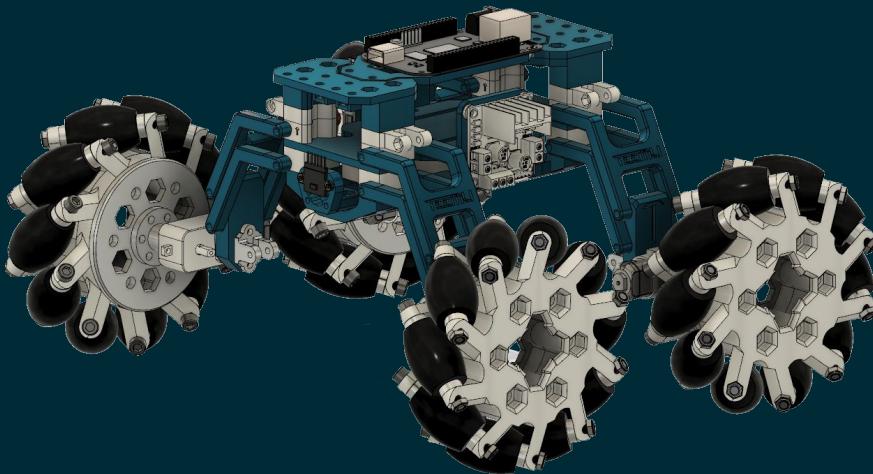
RHO
THETA
PHI

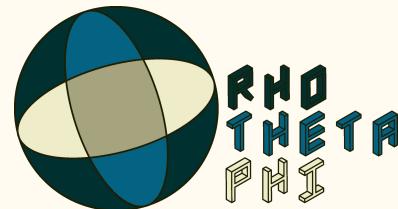


AKKODiS

6 juin 2023

Véhicule Tactique Terrestre -VTT-





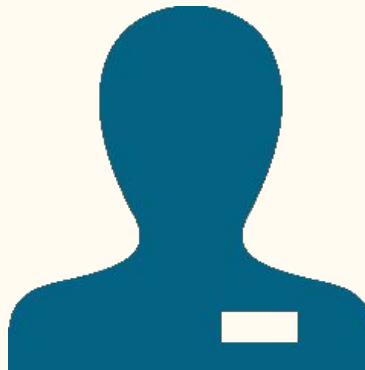
Qui sommes-nous ?



Quentin BLECHET

Bio-Informatique et
Data Science

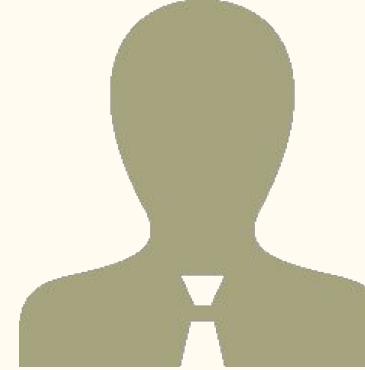
Ingénieur Développeur
Logiciel Embarqué



Bixente BURUCOA

Système électronique

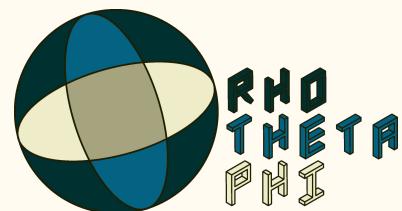
Ingénieur Développeur
Logiciel Embarqué



Sidali ZITOUNI TERKI

Cryptologie et
Sécurité Informatique

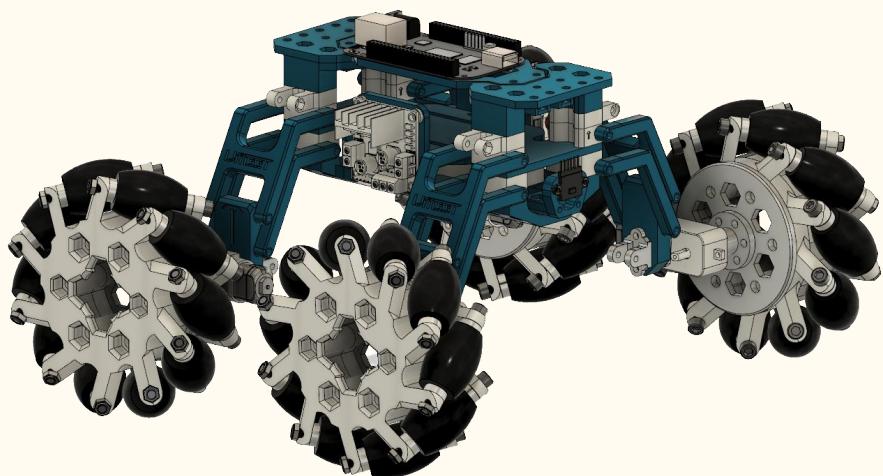
Ingénieur Développeur
Logiciel Embarqué

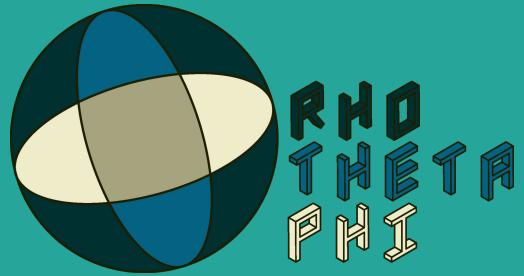


Projet : Véhicule Tactique Terrestre

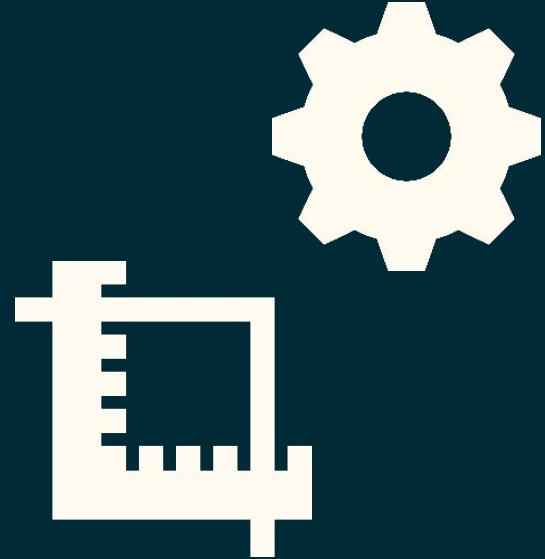
Objectifs :

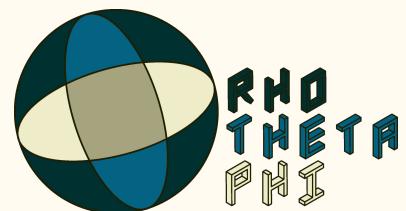
- Se déplacer en zone de combat
- Récupérer et traiter les informations des alliés
- Faciliter le travail du véhicule d'attaque
 - Repérer les adversaires
 - Immobiliser les adversaires
 - Isoler l'adversaire physiquement
 - Couper les communications adverses





Idées de fonctionnalités du VTT





Visualisation du point de vue du rover

Nom de la fonction: Visualisation du point de vue du rover

Entrées: Caméra

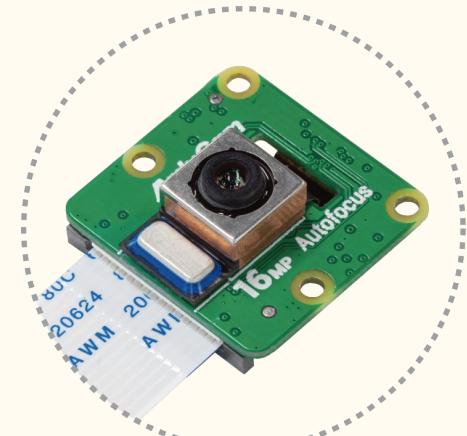
Sorties: Serveur vidéo

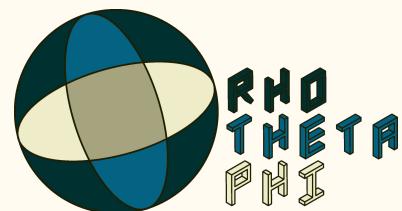
Description de la fonction:

La fonction de visualisation du point de vue du rover utilise une caméra qui filme à l'avant du rover. Le flux vidéo est transmis à un serveur via le Raspberry afin d'être visualisé par l'opérateur.

Comportement attendu:

- La caméra doit être correctement positionnée et suffisamment fixée pour qu'elle ne bouge pas lors du déplacement du rover.
- Le flux vidéo doit être bon pour ne pas avoir trop de latence entre la vidéo et la visualisation par l'opérateur





Contrôle du véhicule à distance

Nom de la fonction: Contrôle du véhicule à distance

Entrées: Coordonnée

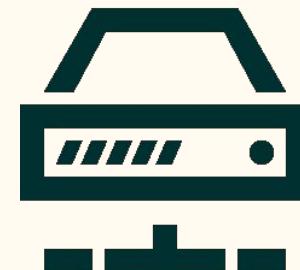
Sorties: Avancement du véhicule

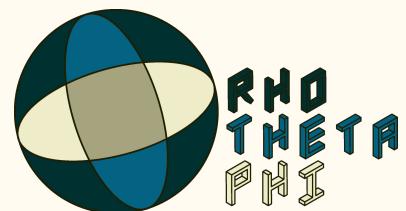
Description de la fonction:

La fonction du contrôle du véhicule utilise les coordonnées envoyées par l'opérateur. Le véhicule avance grâce à la position proposée.

Comportement attendu:

- Le véhicule doit avancer en temps réel pour l'envoi des coordonnées.
- Le véhicule doit pouvoir avancer, reculer, tourner et pivoter.
- Le véhicule doit envoyer sa position exacte.





Détection d'un ennemi ou d'un piège

Nom de la fonction: Détection de la position d'un ennemi, d'un piège ou d'un obstacle.

Entrées: Position d'un ennemi d'un piège ou d'un obstacle.

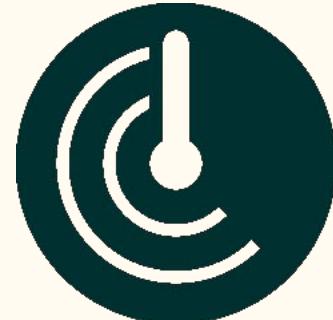
Sorties: Envoie des positions à nos alliés.

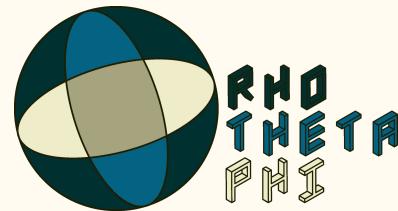
Description de la fonction:

La fonction de la détection de la position d'un ennemi ou d'un piège utilise la caméra. L'opérateur envoie aux alliés la position d'un ennemi et du piège.

Comportement attendu:

- La caméra doit envoyer du flux vidéo en direct.
- L'opérateur doit pouvoir communiquer avec les autres alliés.





Blocage d'une route

Nom de la fonction: Blocage d'une route

Entrées: Position de la route à bloquer

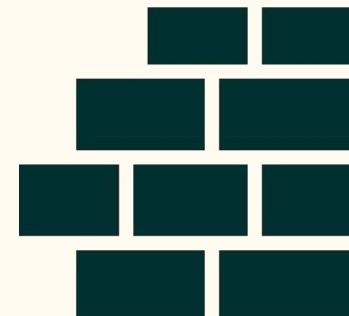
Sorties: Blocage de la route

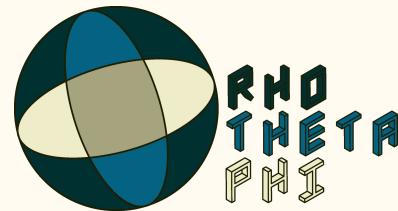
Description de la fonction:

La fonction de blocage d'une route permet après analyse du terrain de bloquer l'accès à un chemin. Le rover dépose un obstacle de la largeur de la route.

Comportement attendu:

- L'obstruction de la route ne doit pas bloquer les véhicules alliés.
- Les véhicules ennemis ne doivent pas pouvoir forcer le passage en roulant.





Déblocage d'une route

Nom de la fonction: Déblocage d'une route

Entrées: Position de l'obstacle à déplacer

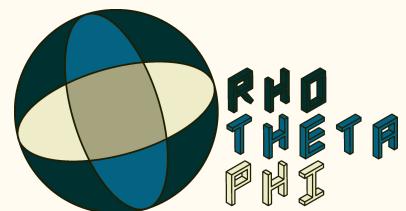
Sorties: Déplacement de l'obstacle

Description de la fonction:

La fonction de déblocage d'une route permet après avoir reçu la position d'un objet, de le déplacer afin de libérer le passage. Le rover s'accroche à l'objet en question et le tire.

Comportement attendu:

- La position doit provenir d'un véhicule allié ou de l'analyse du terrain par lui-même.
- Le déblocage est considéré complet lorsqu'il estime qu'un véhicule peut passer.



Remorquage d'un véhicule allié

Nom de la fonction: Remorquage d'un véhicule allié

Entrées: Position de l'allié

Sorties: Moteur avec crochet

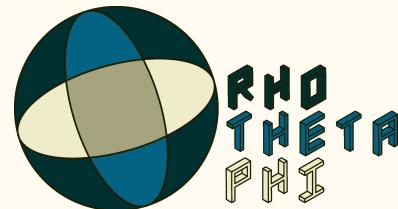
Description de la fonction:

La fonction de remorquage d'un véhicule allié permet, après analyse de l'allié, de le remorquer. Le rover déploie un crochet pour remorquer l'allié, afin de l'éloigner du front.

Comportement attendu:

- Le remorquage de l'allié ne doit pas empêcher le déplacement du rover.





Communication avec les alliés

Nom de la fonction: Communication avec les alliés.

Entrées: Information des alliés

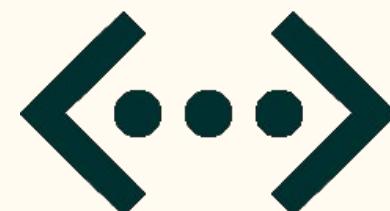
Sorties: Réponse aux alliés

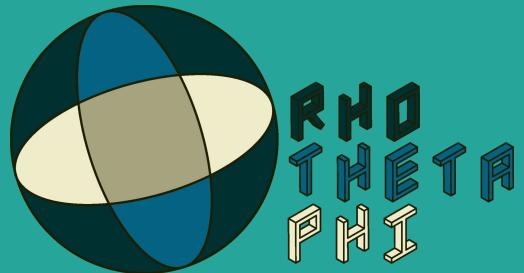
Description de la fonction:

La fonction communication avec les alliés utilise un système pour recevoir les messages et les envoyer.

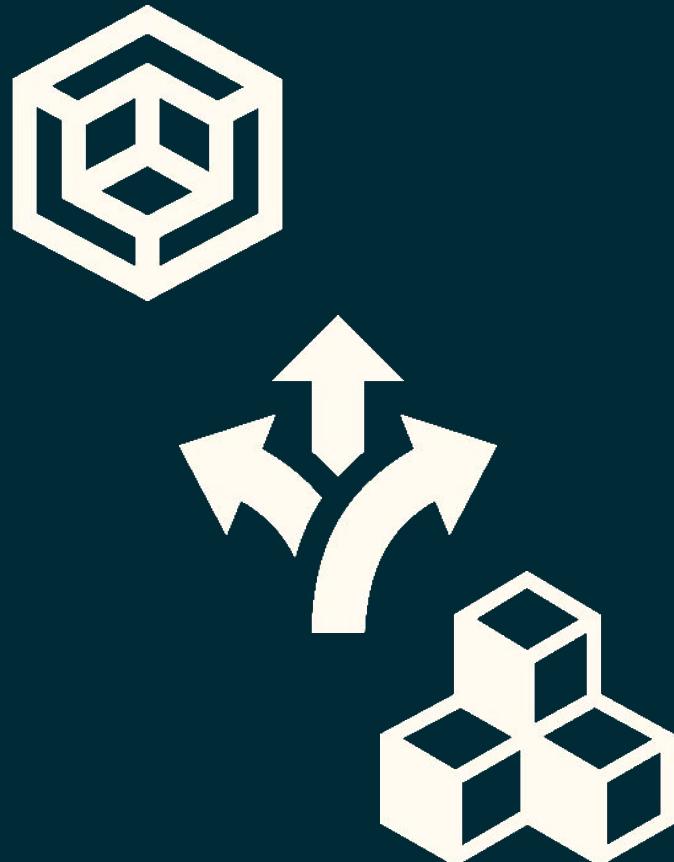
Comportement attendu:

- Le message arrive bien aux alliés.
- Le système reçoit bien les messages alliés.





Plan Global Rover



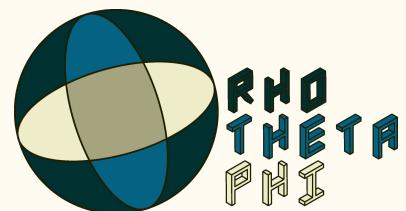
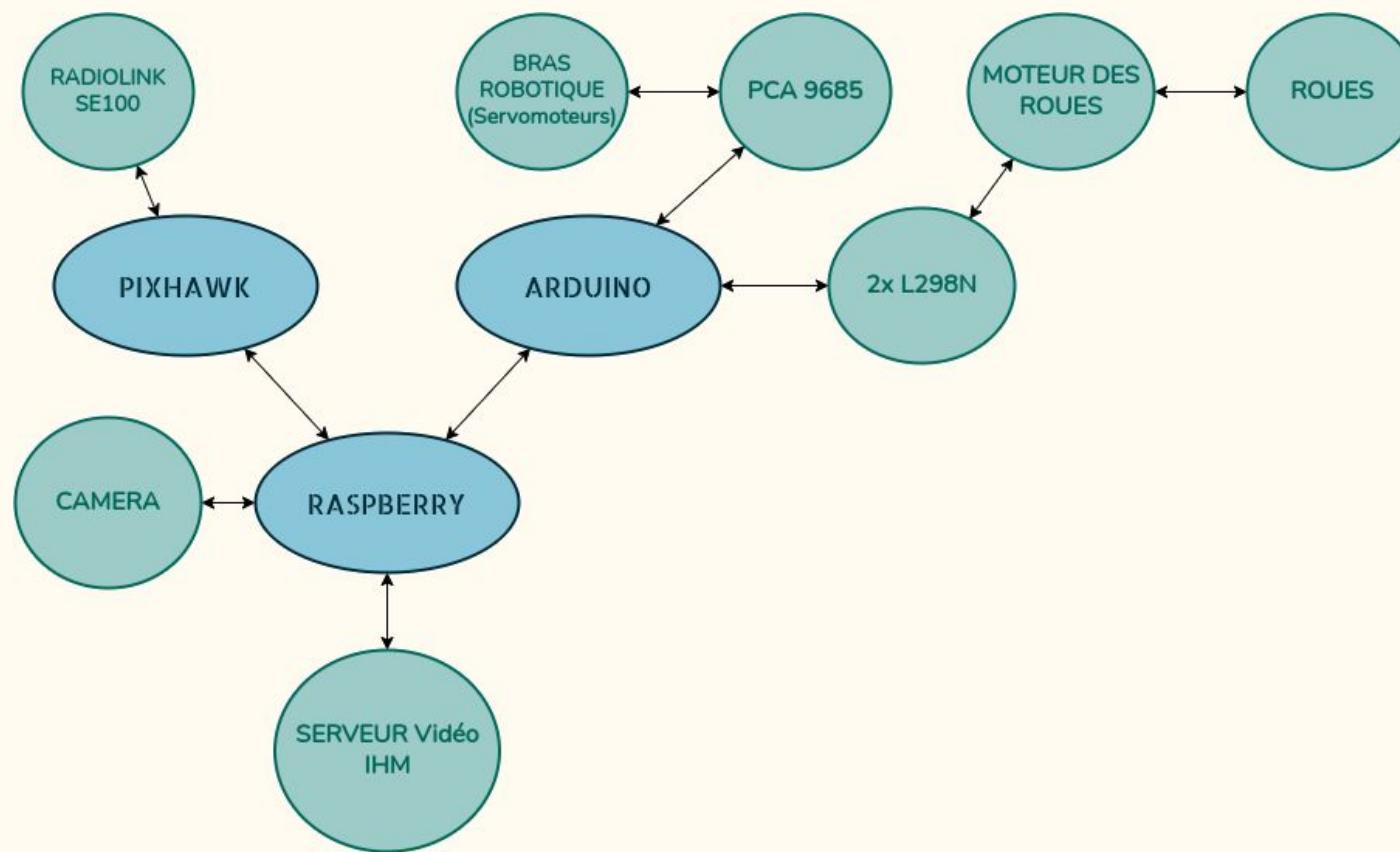
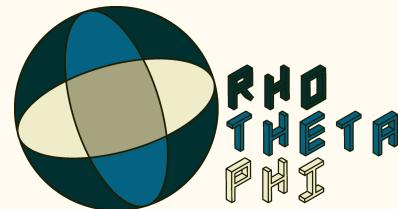


Schéma Général Rover





Codes principaux

Serveur OVH

IHM

\$_app.py
\$_app.css
\$_app.js
\$_index.html

Serveur Vidéo

Rover

Raspberry

\$_stream.sh
└── ./stream (service)
\$_serveur.c
\$_client.c
\$_token.h/token.cpp
\$_socket.c
└── ./server (service)

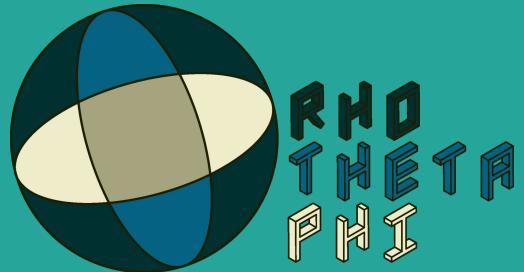
Middleware ROS (py cpp)

Arduino

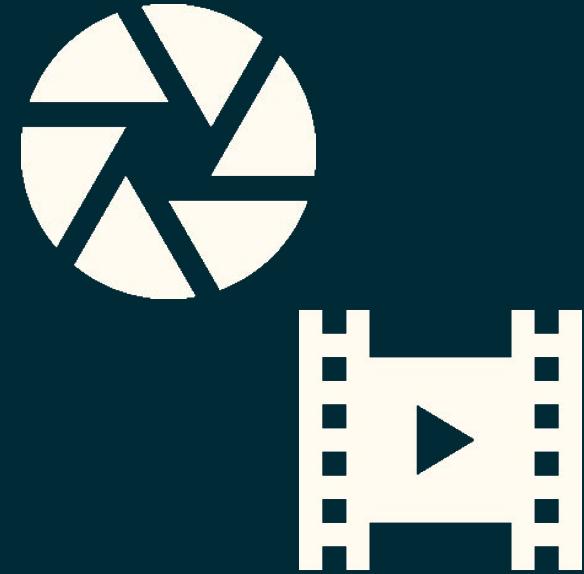
\$_Engines.h/Engines.cpp
\$_drive_arm.ino

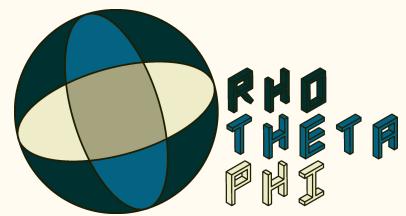
PixHawk

Firmware ArduPilot (py)



Vidéo et Streaming





Caméra : Matériel

Caméra Arducam 16MP

Capteur : Sony IMX519

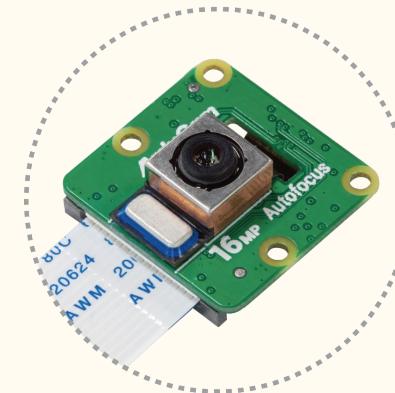
16 megapixels

1.22 µm × 1.22 µm pixel size

7.103 mm diagonal (Type 1/2.534)

Sortie : RAW10/8, COMP8

FoV : 80°



Avantages : Coût et Autofocus

Caméra : Traitement vidéo

FFmpeg : Ensemble de bibliothèques et logiciels

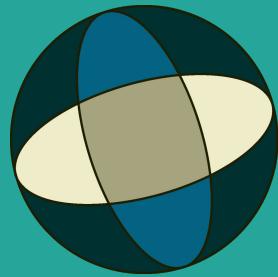
- Traitement vidéo
- Streaming
- Cross Platform
- Large documentation

Protocole RTSP=Real Time Streaming Protocol
(RFC 2326)

Protocole semblable au http pour le streaming de
vidéo et d'audio

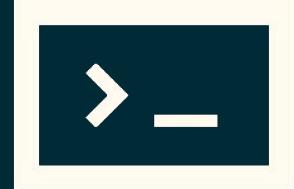
Flux streamé sur container Docker via un service



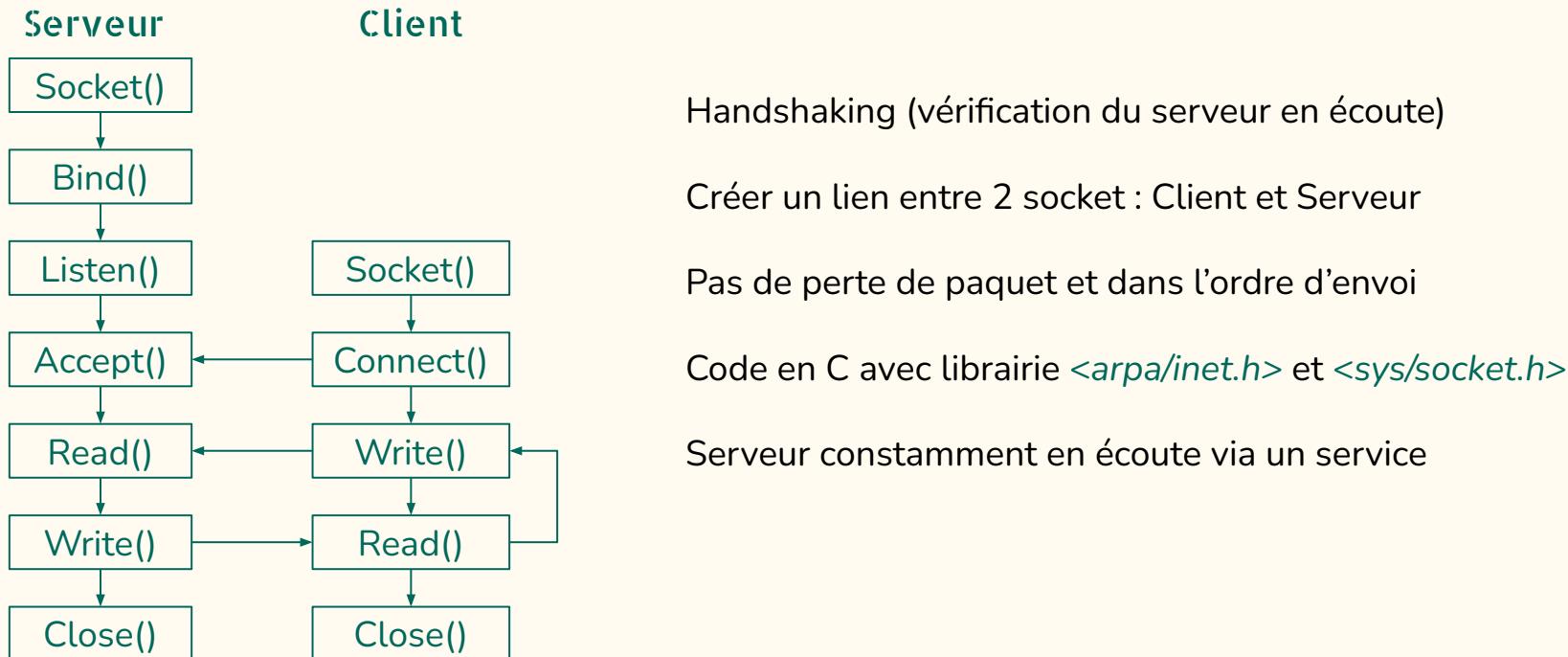


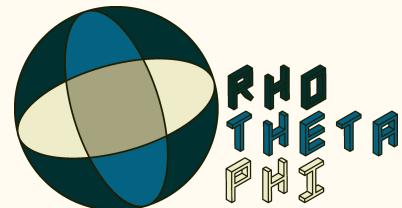
RHO
THETA
PHI

Serveur TCP



TCP





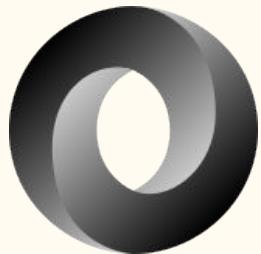
Data au format JSON

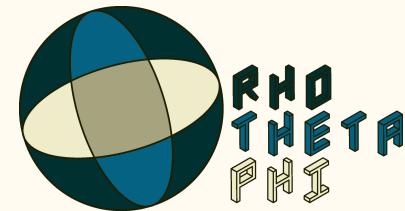
Utilisation du format JSON :

- Format Universel (bibliothèques pour C/C++ et Python)
- Facilité de transmission

Bibliothèque CJson :

- Bibliothèque légère adaptée pour l'embarquée
- Compatible pour langage C et C++





Structure JSON

JSON pour information de direction :

```
{  
  "password" : "*****",  
  "token" :  
    {"direction" : 1},  
  "value" : 1  
}
```

JSON pour information de mouvement du bras mécanique :

```
{  
  "password" : "*****",  
  "token" :  
    {"arm" : 11},  
  "value" : 2  
}
```

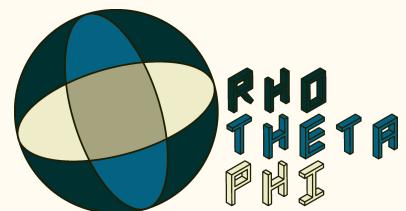
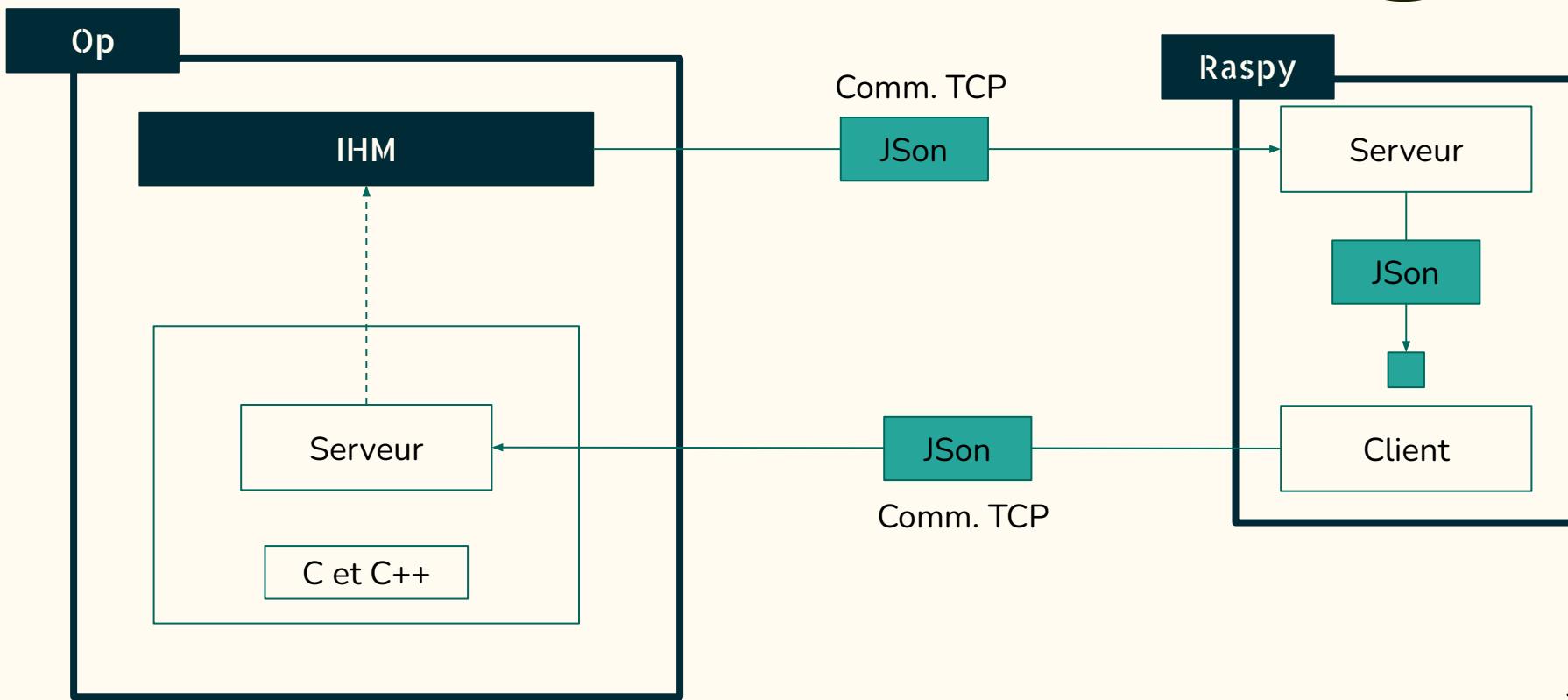
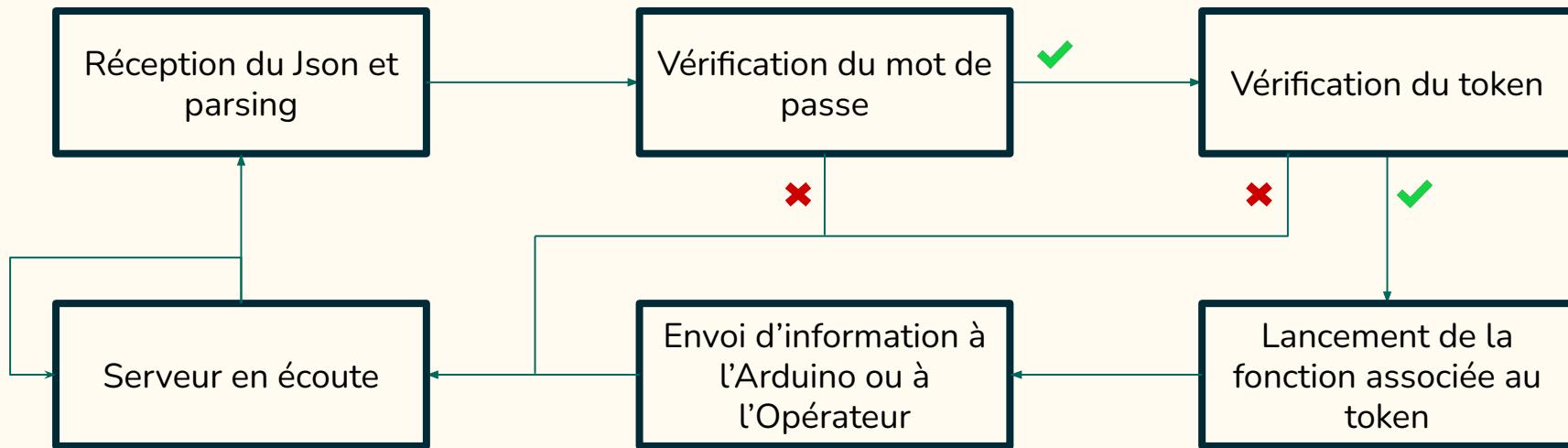


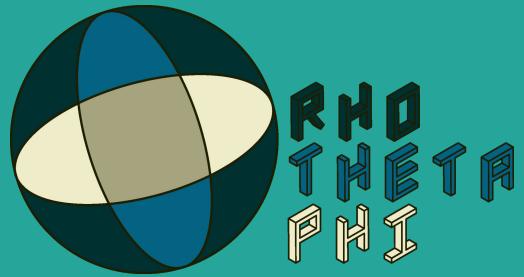
Schéma Transmission



Gestion des informations par le serveur



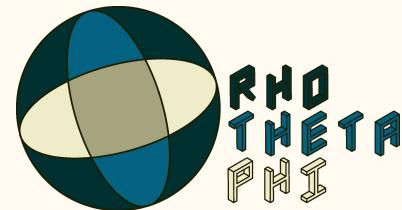
```
{
  "password" : "*****",
  "token" :
    {"direction" : 1},
  "value" : 1
}
```



Communication Raspberry et Arduino



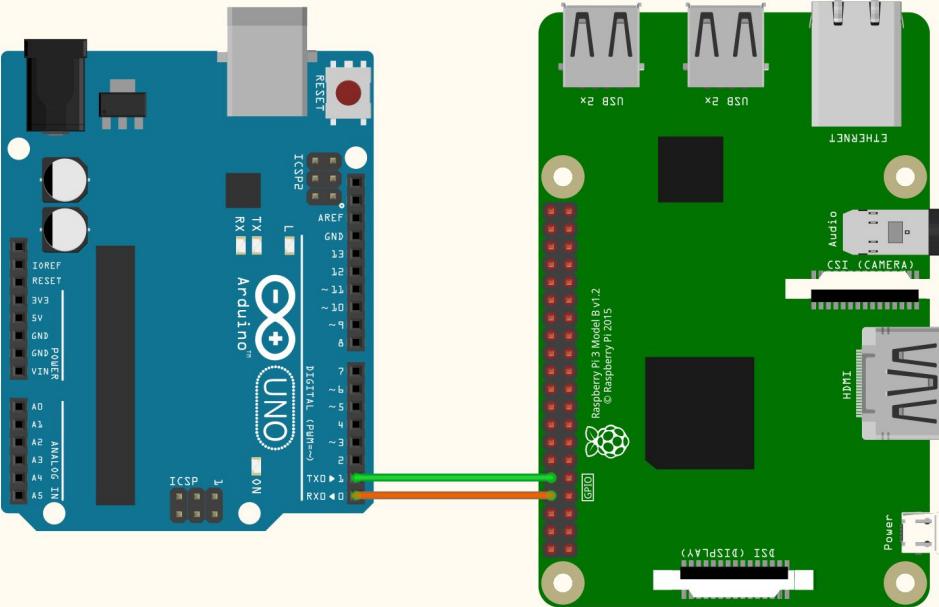
Communication Version 1



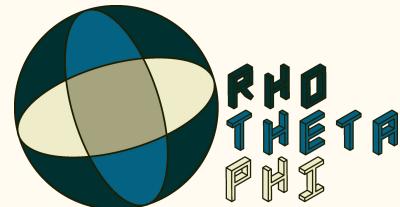
Protocole de communication UART.

Utilisation des ports Rx/Tx de l'Arduino et du Raspberry.

GPIO Raspberry	Pin Arduino
15	1
14	0



fritzing

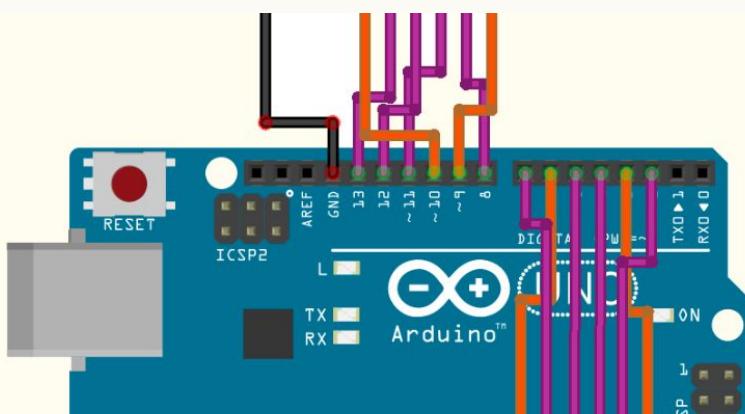


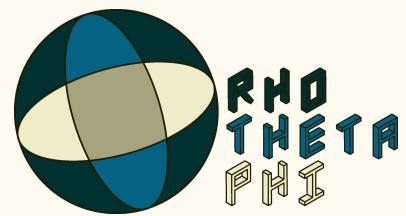
Problématiques

Communication avec le Pixhawk
uniquement possible en UART, et un
seul jeu de PIN UART sur le Raspberry
-> Communication Arduino/Raspy
en SPI

Pas assez de PIN de libre pour
protocole SPI sur l'Arduino UNO
-> Passage sur un Arduino MEGA

Raspberry Pi GPIO BCM numbering

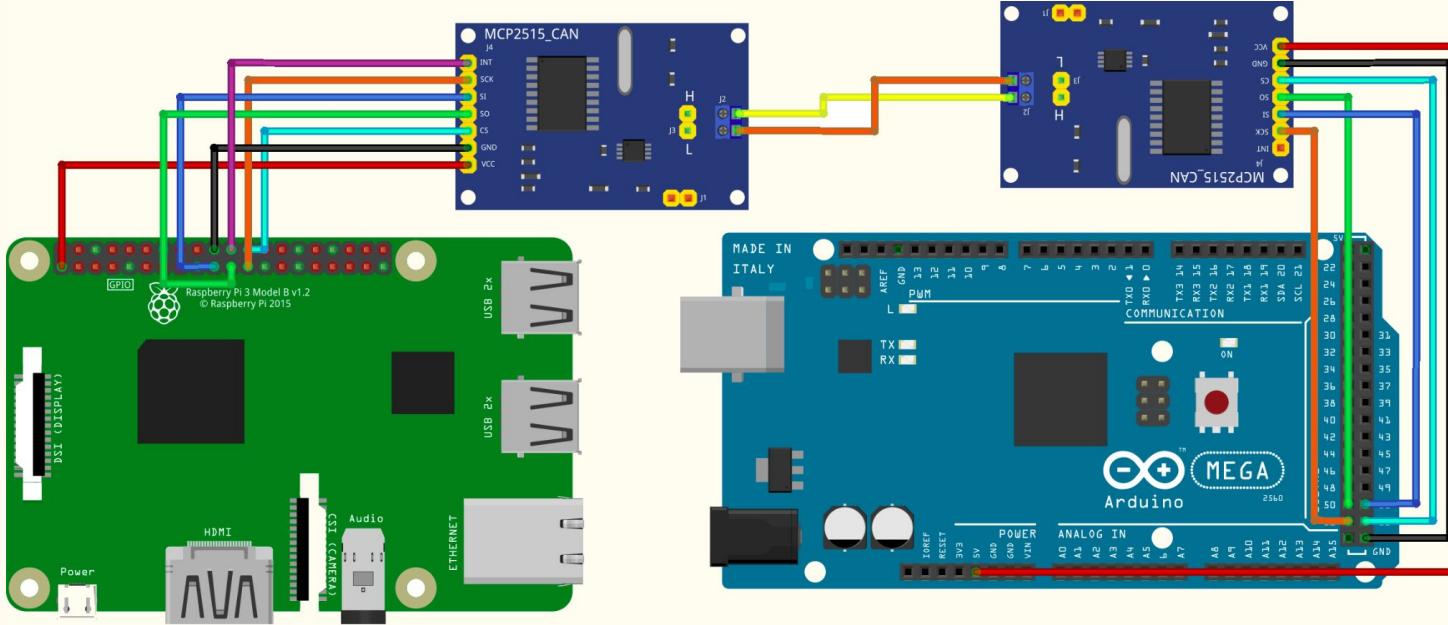


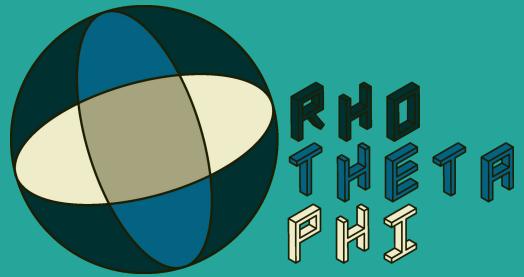


Communication Version 2

Protocole de communication SPI (4 câbles nécessaires)

Passage en bus CAN pour la différence de tension (3,3V Raspberry, 5V Arduino)

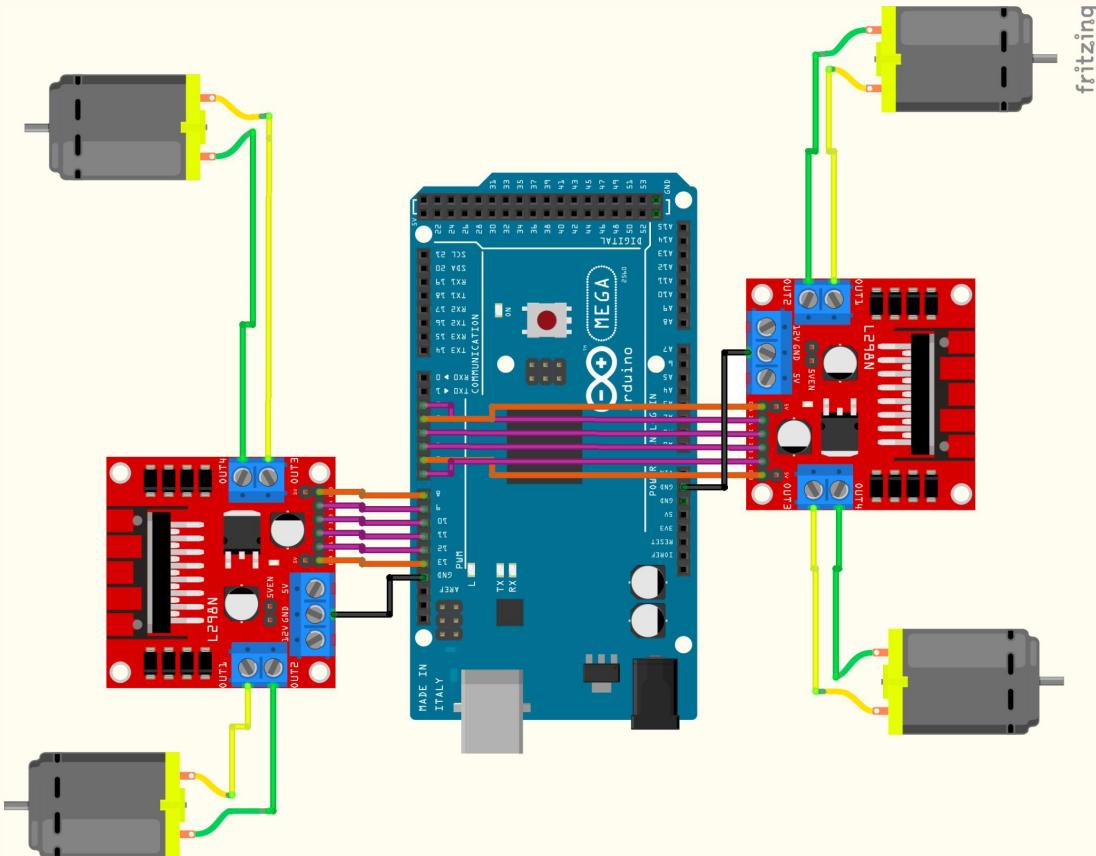




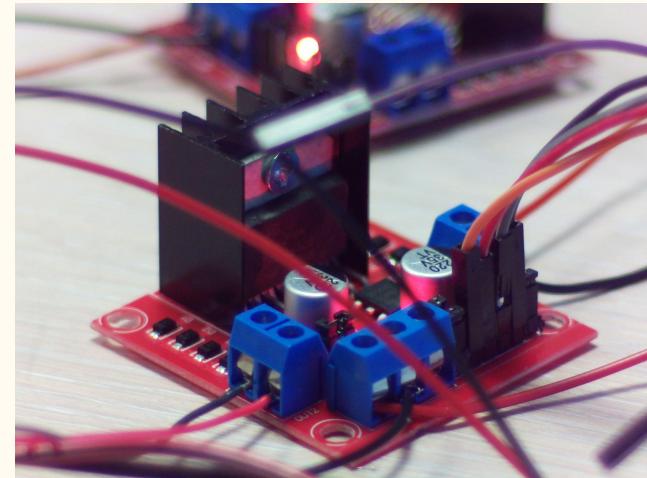
Branchement moteur et bras mécanique



Schéma Moteurs



Utilisation de 1
Arduino et 2 L298N



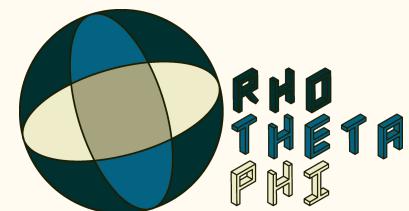
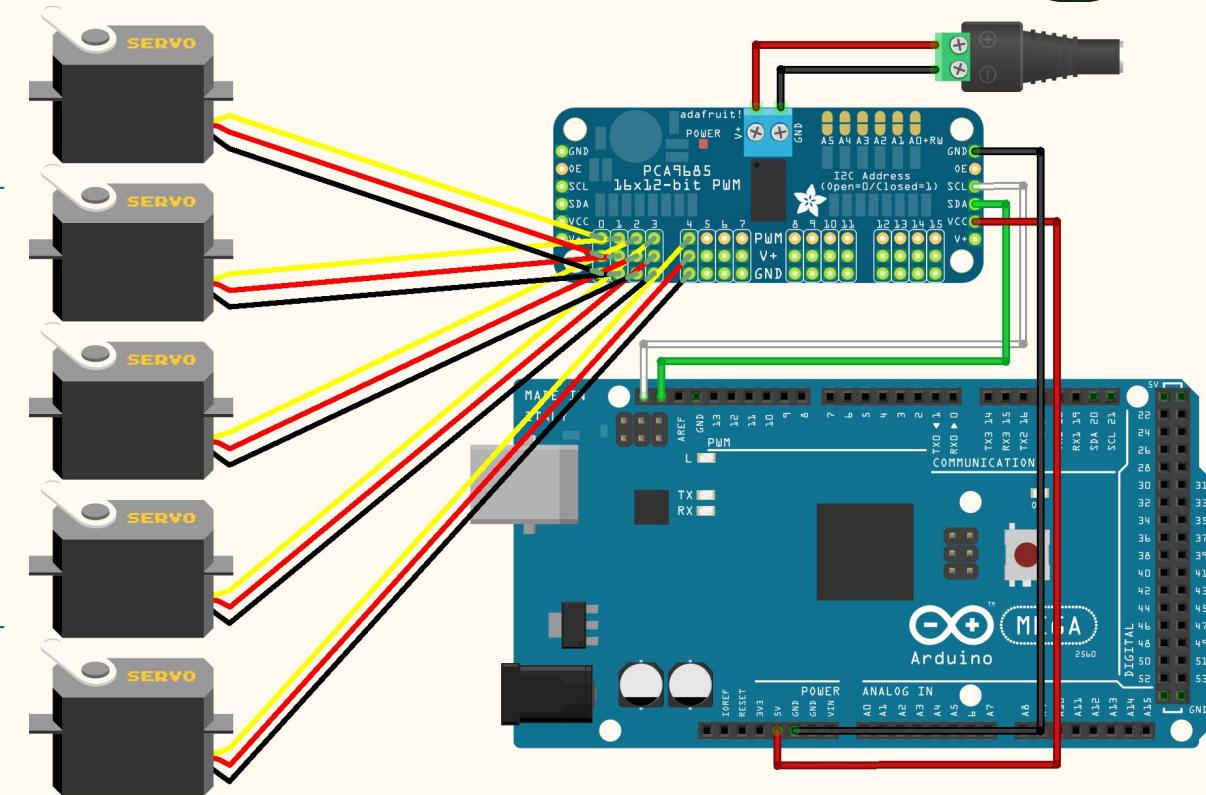


Schéma Bras Mécanique

Rotation du Bras*

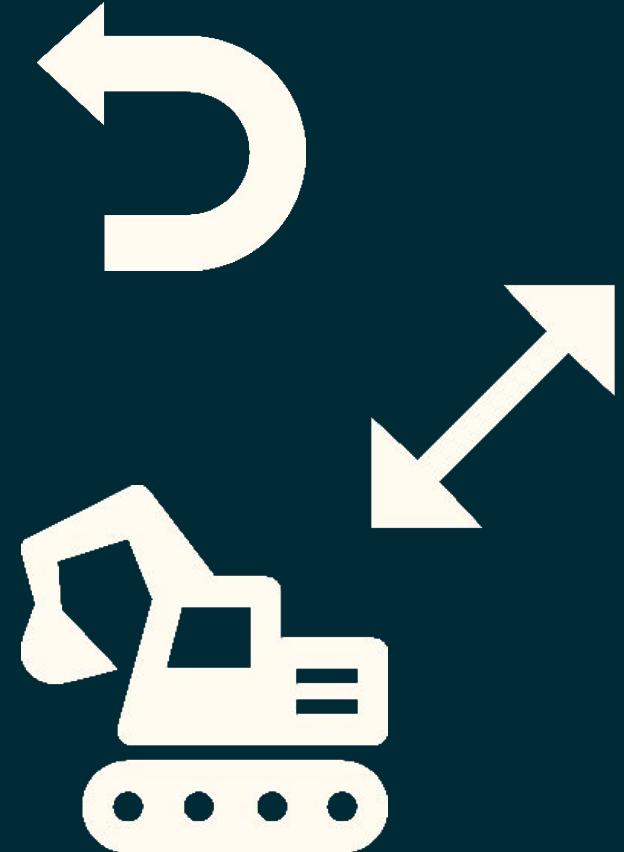
Corps du Bras

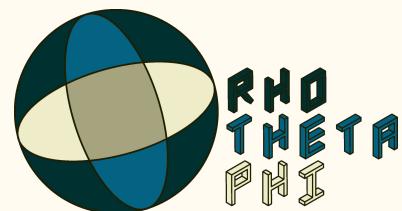
Pince du Bras





Algorithmes de déplacement et de manipulation du bras mécanique





Déplacement général

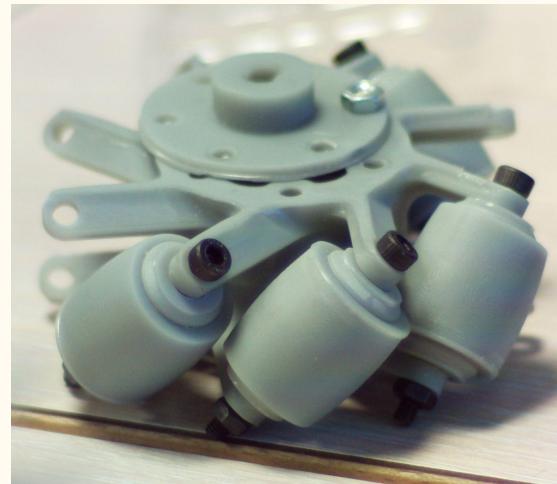
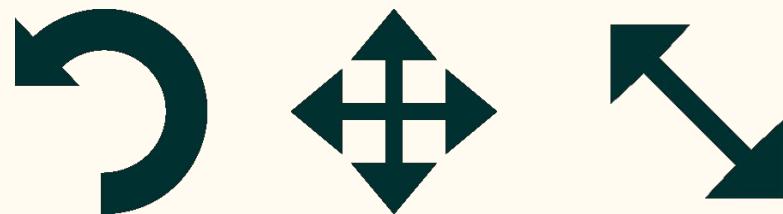
Configurations :

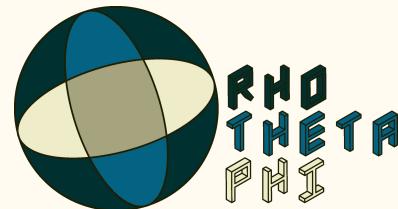
V1

- Avant / Arrière
- Pivoter
- Latérale
- Stop

V2

- Diagonale : Avant / Arrière + Latérale
- Tourner : Avant / Arrière + Pivoter
- Drift : Pivoter + Latérale





Bibliothèque Engines

Bibliothèque créée par notre équipe <*Engines.h*>

Classe composé de 4 moteurs :

```
struct Motor {  
    int in1;  
    int in2;  
    int pwn;  
};
```

Permet de gérer la vitesse et la rotation de chaque roue de manière indépendante

19 méthodes de déplacements différentes

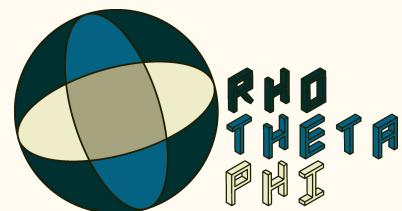
V1

```
void full_stop(int delay_time);  
  
void forward(int speed, int delay_time);  
void backward(int speed, int delay_time);  
  
void rotate_right(int speed, int delay_time);  
void rotate_left(int speed, int delay_time);
```

```
void lateral_right(int speed, int delay_time);  
void lateral_left(int speed, int delay_time);
```

V2

```
void forward_rotate_right(int speed1, int speed2, int delay_time);  
void forward_rotate_left(int speed1, int speed2, int delay_time);  
  
void backward_rotate_right(int speed1, int speed2, int delay_time);  
void backward_rotate_left(int speed1, int speed2, int delay_time);  
  
void forward_lateral_right(int speed, int delay_time);  
void forward_lateral_left(int speed, int delay_time);  
  
void backward_lateral_right(int speed, int delay_time);  
void backward_lateral_left(int speed, int delay_time);  
  
void rotate_right_lateral_right(int speed, int delay_time);  
void rotate_right_lateral_left(int speed, int delay_time);  
void rotate_left_lateral_right(int speed, int delay_time);  
void rotate_left_lateral_left(int speed, int delay_time);
```

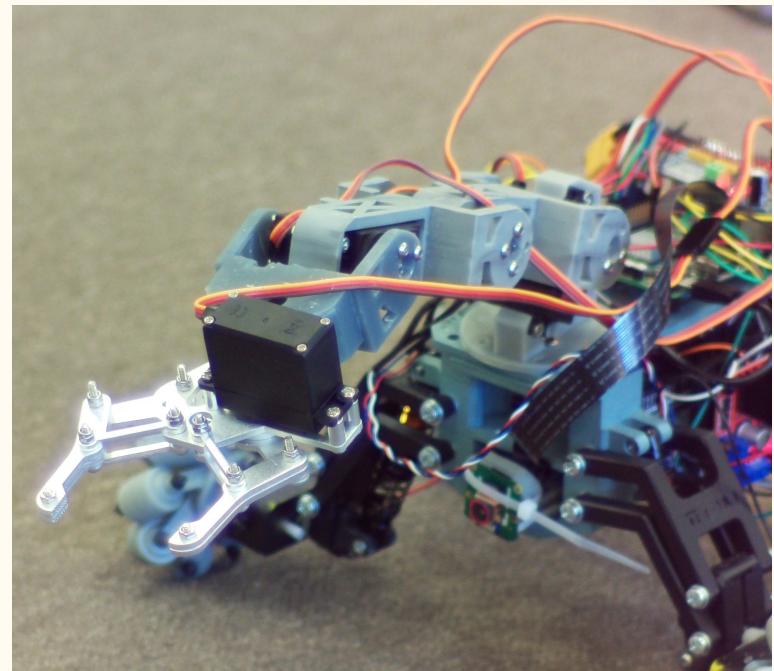


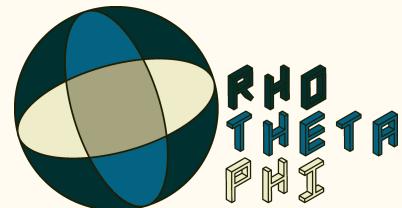
Manipulation du bras

4 moteurs :

- 3 moteurs pour la mobilité verticale du bras
→ monter / descendre
Chaque moteur est contrôlable individuellement
- 1 moteur pour la pince → ouvrir / fermer

Communication I2C vers la carte Adafruit PCA9685
avec la librairie *<PWM-Servo-Driver.h>*



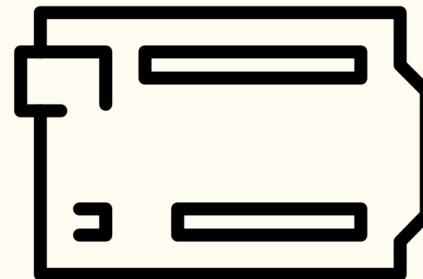


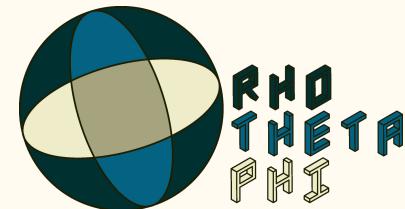
Algorithme de l'arduino

Setup :

- Réception CAN
- Communication I2C
- Position de départ du bras mécanique
- Vitesse de base pour les moteurs

Première réception d'informations du raspberry puis **loop()**





Algorithme de l'arduino

Récupération de 2 **int** via le protocole CAN (SPI) envoyé par le raspberry avec la librairie [`<mcp2515.h>`](#)

Premier int :

mode 1 (direction) ou 2 (arm)

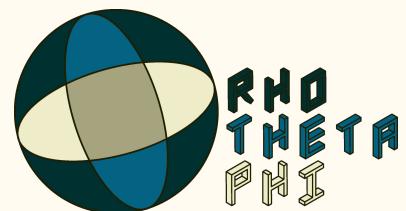
Second int :

configuration correspondant à une fonction précise selon le **mode**

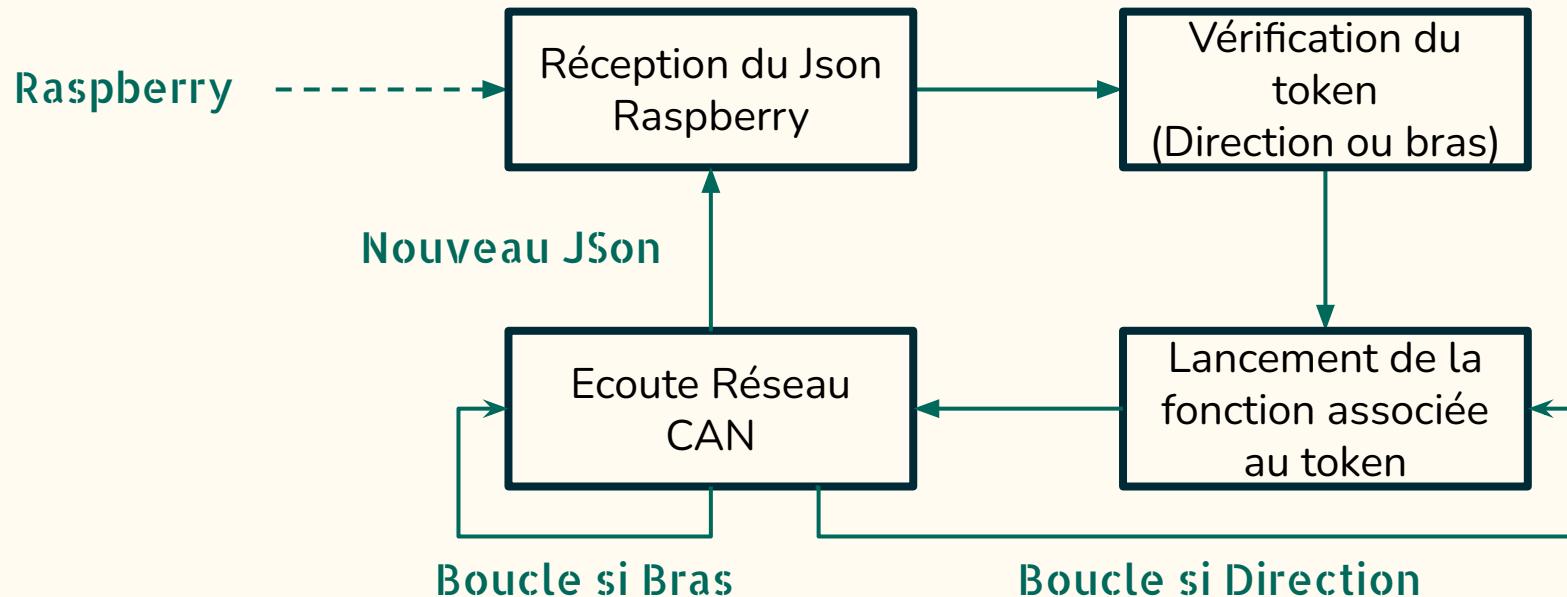
exemple : mode 1 configuration 2 → direction forward

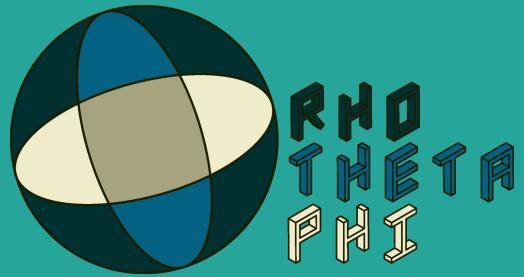
mode 2 configuration 2 → arm close claw

Informations gérées via 2 **switch** consécutifs dans le **loop()**

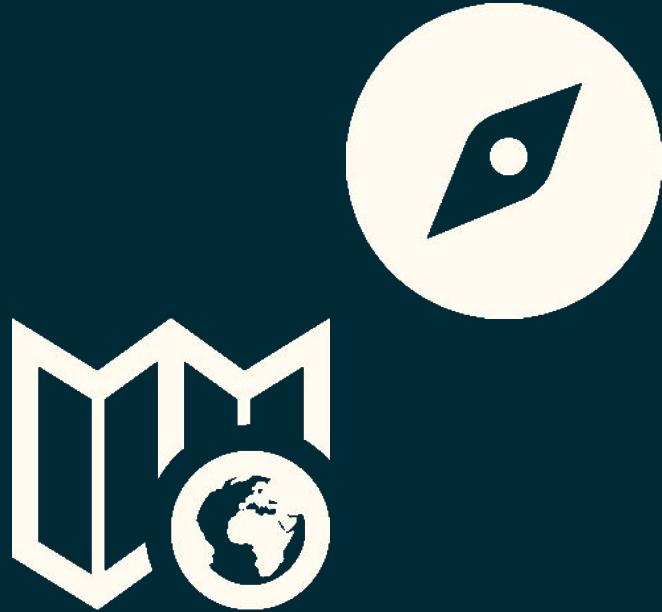


Gestion des informations par l'Arduino

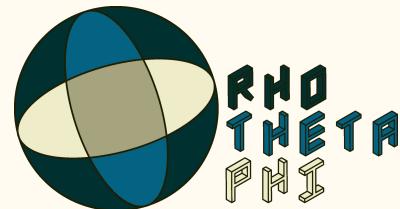




Pixhawk et
Radiolink SE100



Librairies Raspberry



Robot Operating System : Framework pour la communication avec le PixHawk



Librairie MavRos MavLink



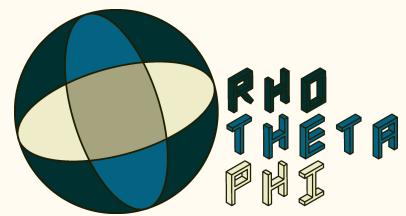
OU

Middle-ware Micro XRCE-DDS EProsim

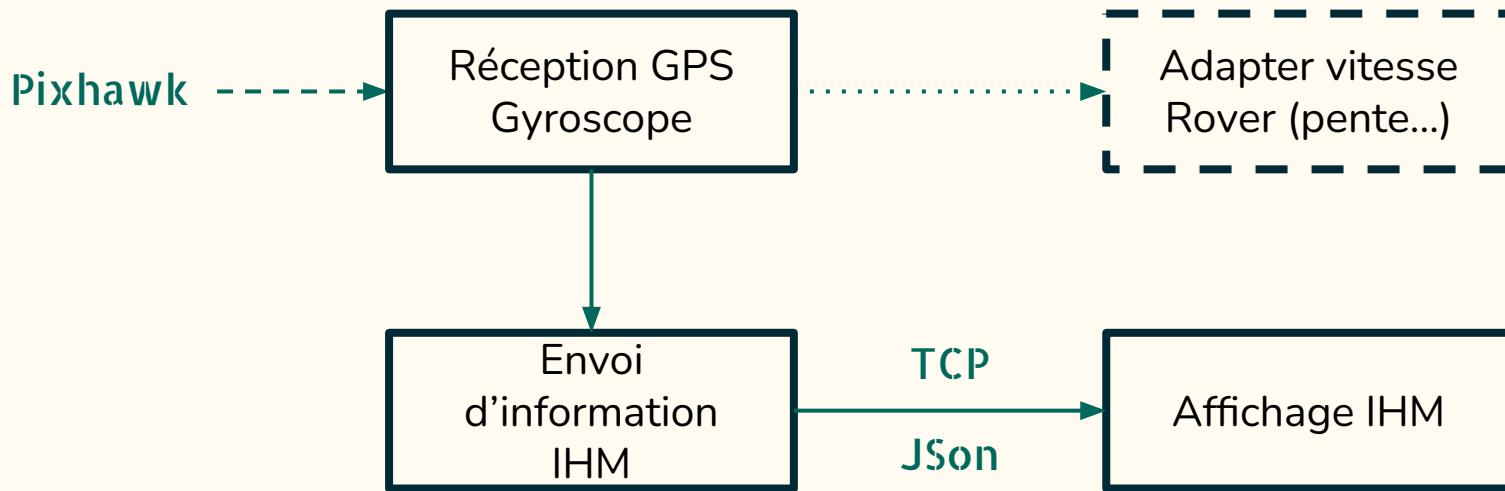


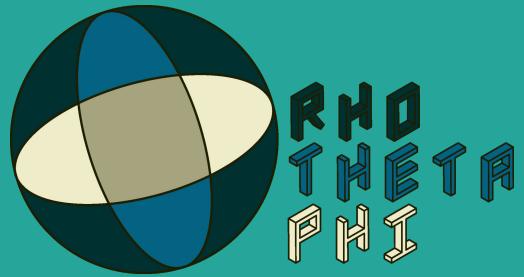
Problème : Librairies MavRos et XRCE-DDS incompatibles avec la version Raspbian utilisée (Bullseye)

Solution pour V2 : Partir sur un système d'exploitation incluant déjà les librairies nécessaires.

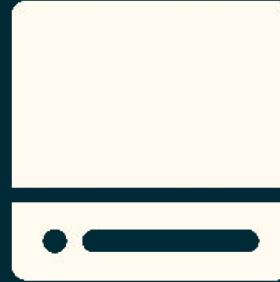


Gestion des informations du Pixhawk



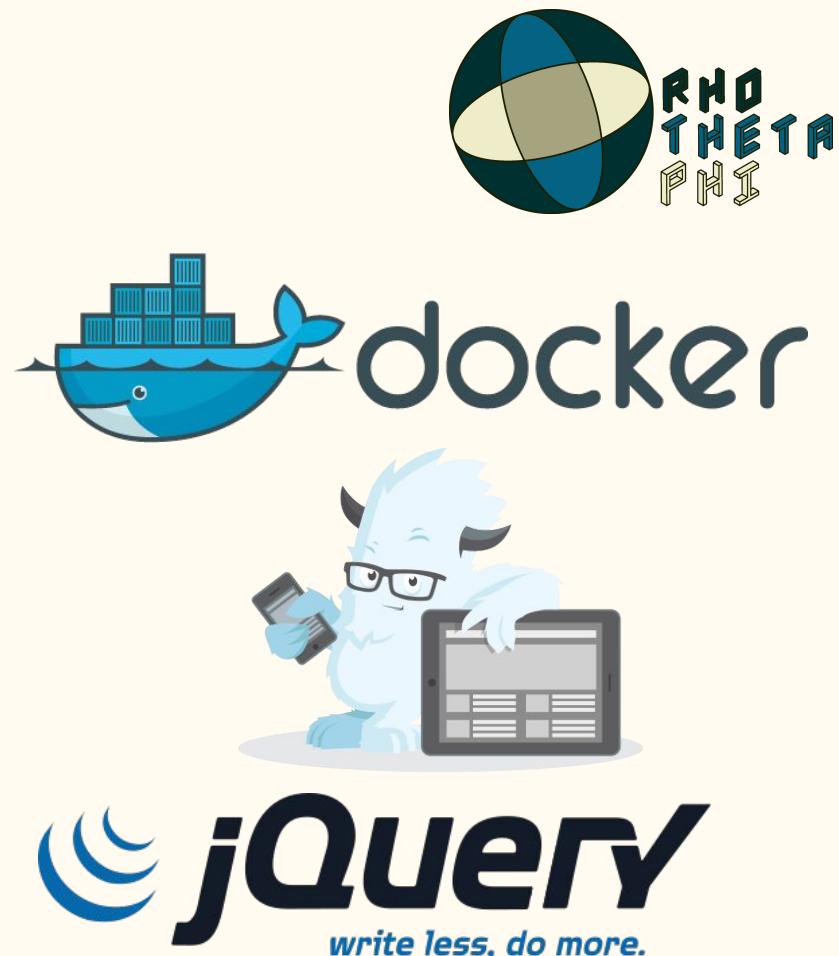


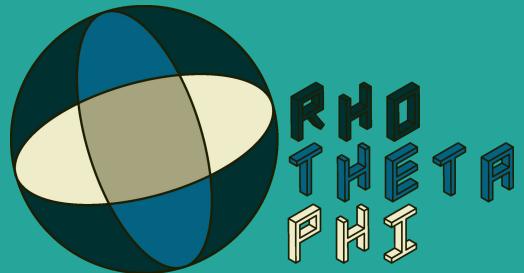
IHM Opérateur



Interface Homme Machine

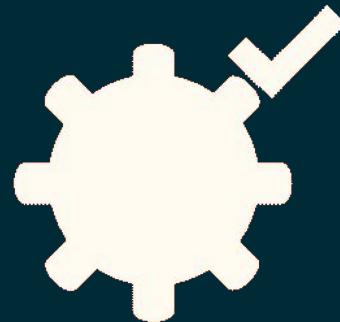
- Utilisation d'un conteneur Docker pour créer l'interface (fichier Yaml et code Python)
- Frontend créé avec Foundation Framework (HTML et CSS)
- Backend avec jQuery (Python et JavaScript)

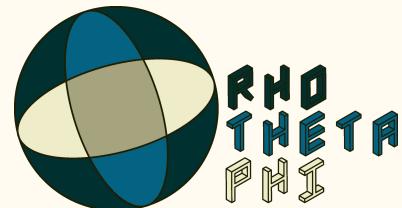




Améliorations Potentielles

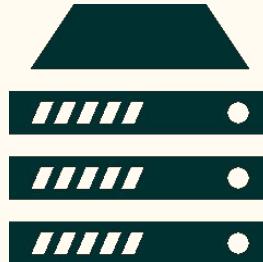
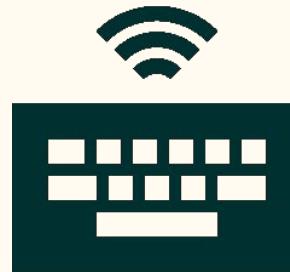
Version V2

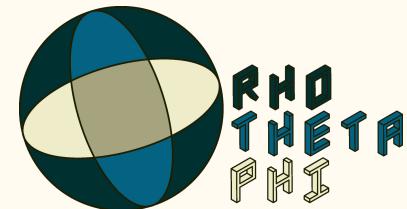




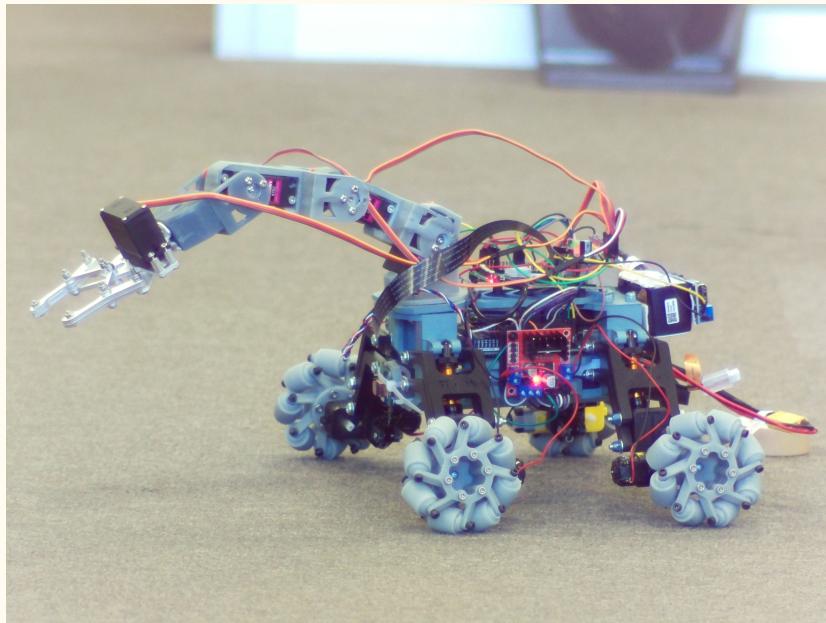
Améliorations potentielles

- Modes de déplacement
- Crochet de remorquage
- Serveur commun avec les alliés
- IHM améliorée

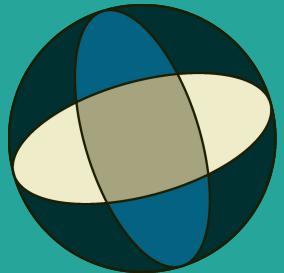




Merci de nous avoir écoutés



Questions ?



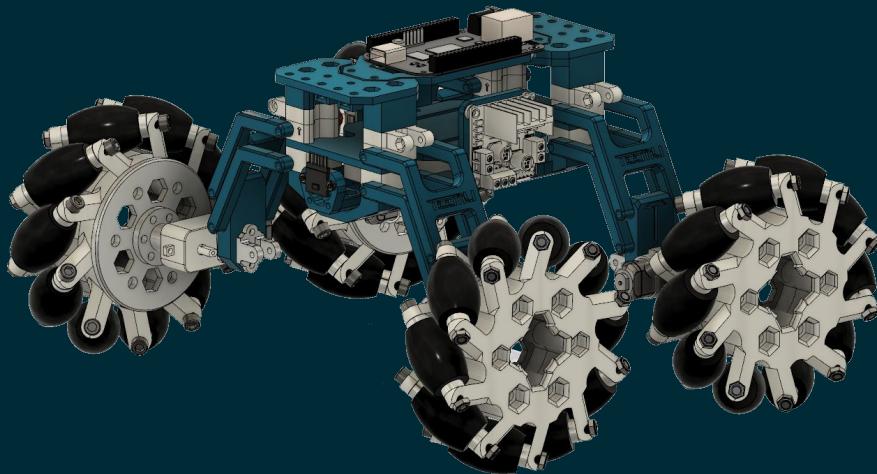
RHO
THETA
PHI

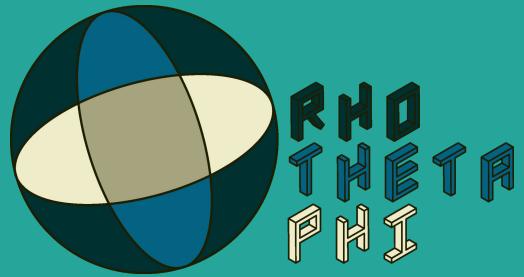


AKKODiS

6 juin 2023

Véhicule Tactique Terrestre -VTT-





Annexes



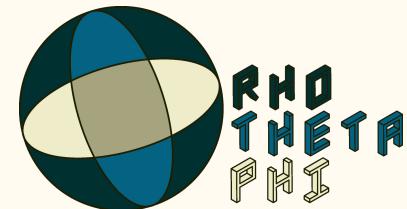


Table Correspondance JSON

Opérateur		Rover	
1	direction	5	confirmAccroche (V2)
2	accroche (V2)	6	confirmMine (V2)
3	mine (V2)	7	position
4	askNbMine (V2)	8	vitesse
11	arm	9	radar
		10	nbMine (V2)

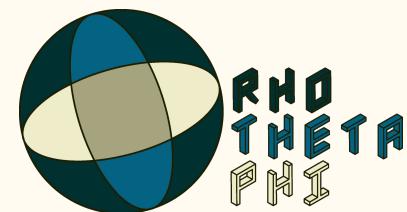


Tableau Correspondances PIN Arduino

EN : Puissance du moteur

Paire IN : Sens de rotation/blocage des roues

PIN	L298N-G		PIN	L298N-D	
13	ENA	Moteur Arrière	3	ENA	Moteur Avant
12	IN1		2	IN1	
11	IN2		4	IN2	
10	IN3		5	IN3	Moteur Arrière
9	IN4		7	IN4	
8	ENB		6	ENB	
GND	GND		GND	GND	

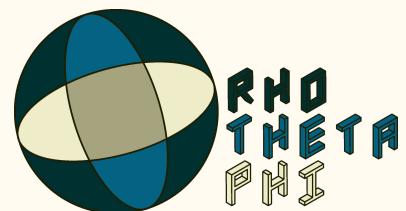
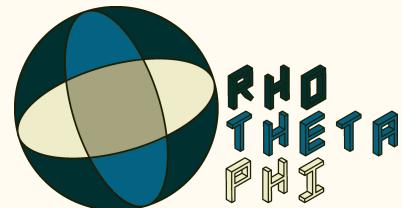


Table Communication Raspy/Arduino

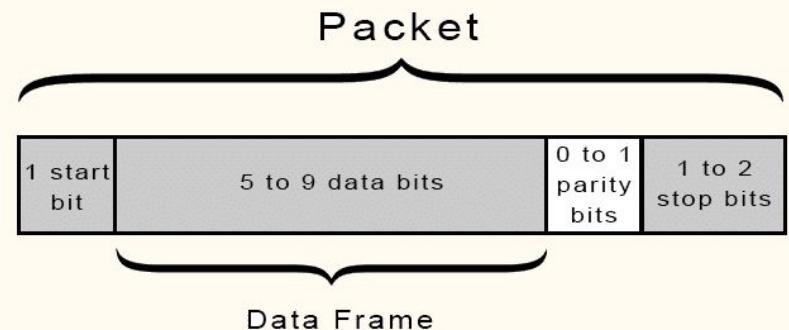
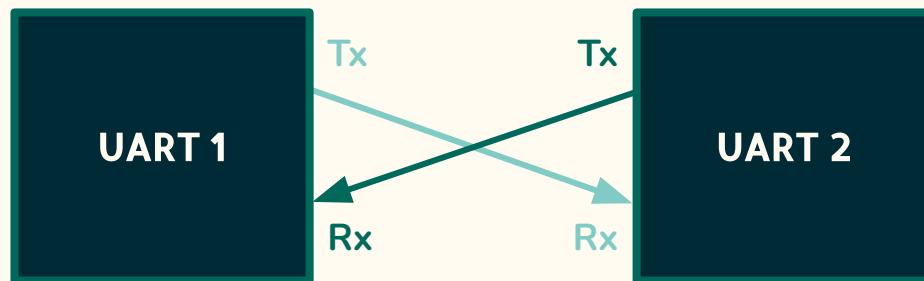
Connexion SPI pour les deux microcontrôleur

Raspberry (SPI0)				Arduino Mega	
PIN	WiringPi	GPIO	Fonction	PIN	Fonction
1	Vin	Vin	Vin (3,3V)	3V3	Vin (5V)
19	12	10	MOSI	51	MOSI
21	13	9	MISO	50	MISO
24	10	8	CS	53	SPI-CS
23	14	11	SCLK	52	SPI-SCK
22	6	25	INTERRUPT		

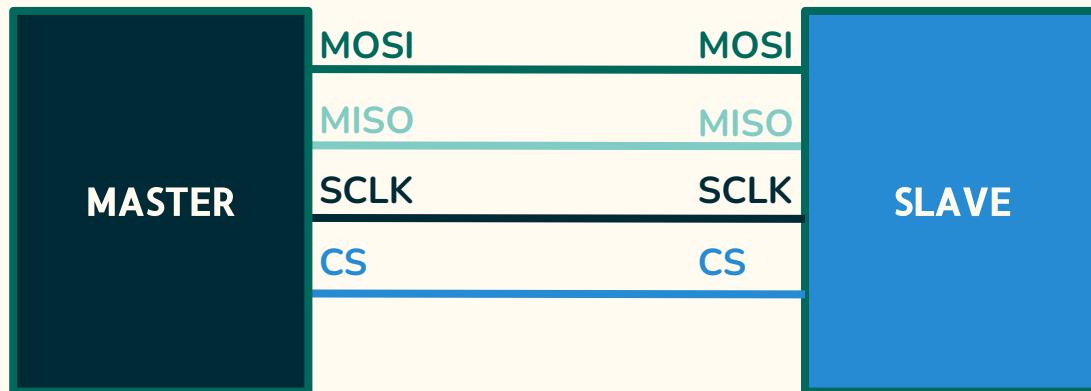


Fonctionnement UART

Protocole de communication UART.
Utilisation des ports Rx/Tx de l'Arduino et du Raspberry.



Fonctionnement Protocole SPI



MOSI : Master Output Slave Input

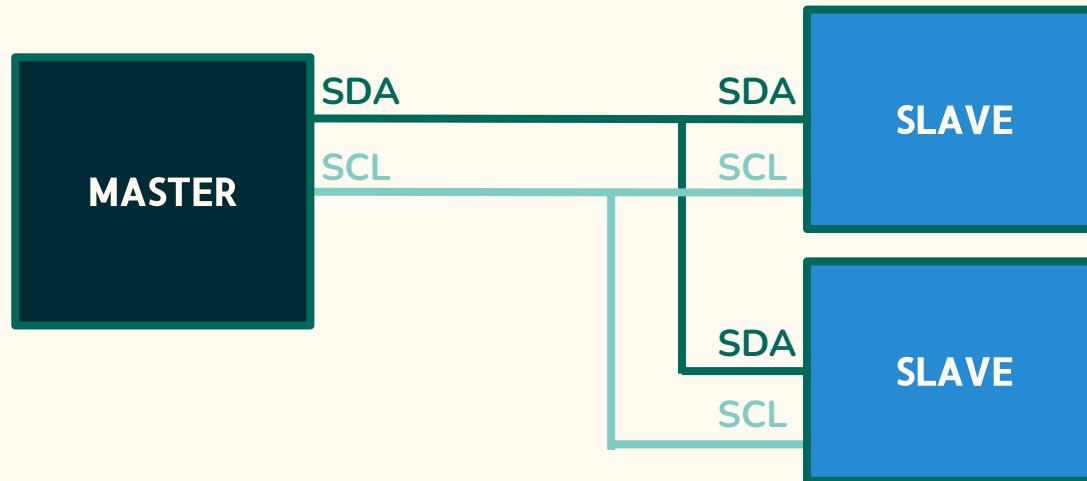
MISO : Master Input Slave Output

SCLK : Serial Clock

CS : Chip Select

Fonctionnement I2C

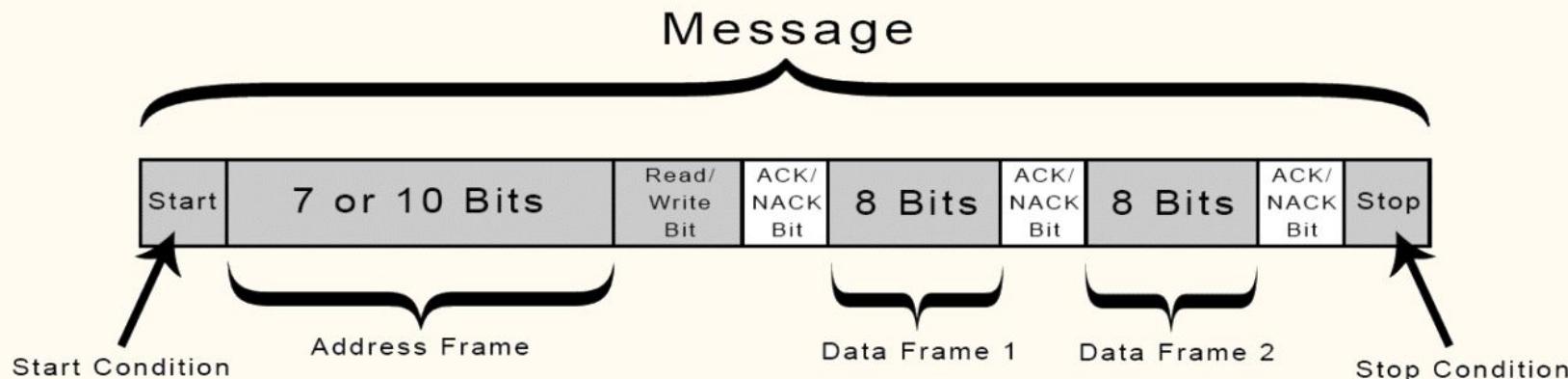
Contrôlé par l'Arduino Mega
Protocole I2C avec Adafruit PCA9685



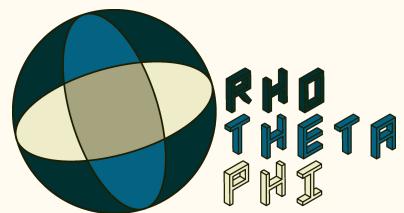
SDA : Serial Data

SCL : Serial Clock

Fonctionnement I2C

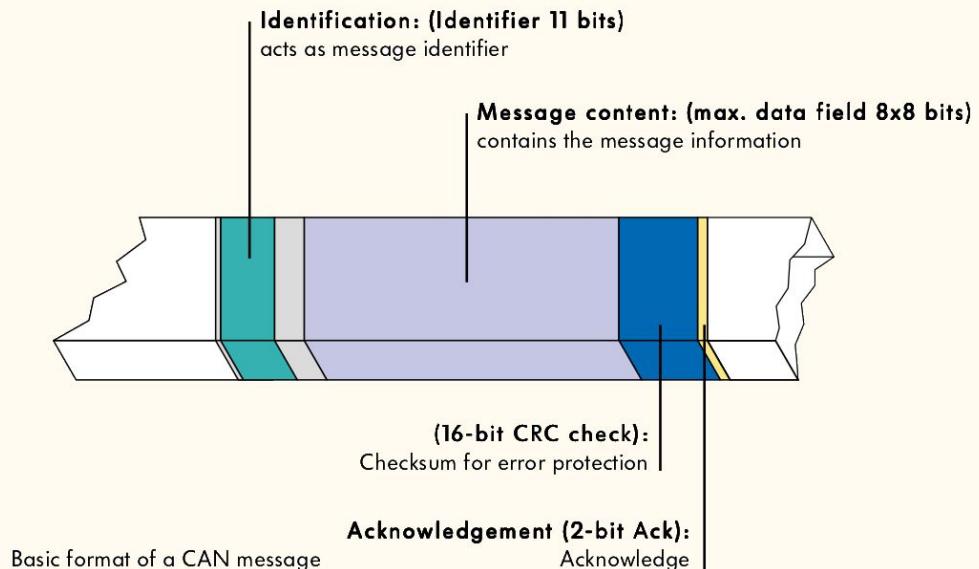
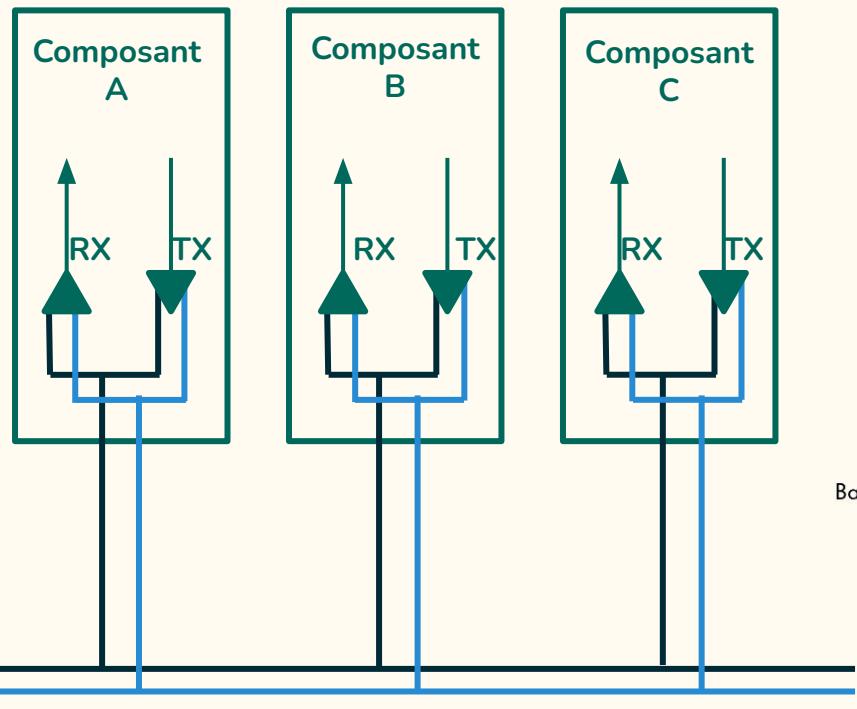


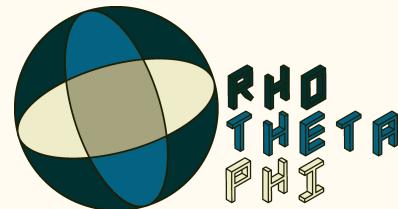
ACK/NACK Bit : acknowledge/no-acknowledge bit ; bits de contrôle



Fonctionnement Bus CAN

Mise en réseau des microcontrôleurs



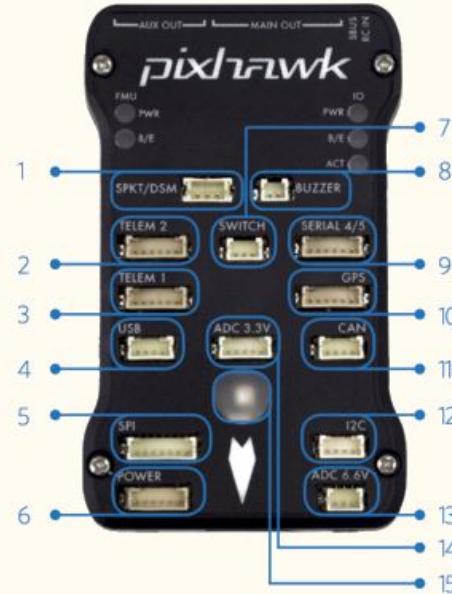


Boîtier PixHawk

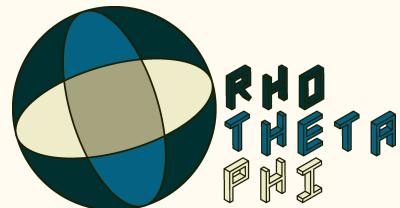
Connexion UART (Rx/Tx) avec le Raspberry (GPIO 14 et 15)

Processeur STM32F427

Utilisable pour la création de drones.

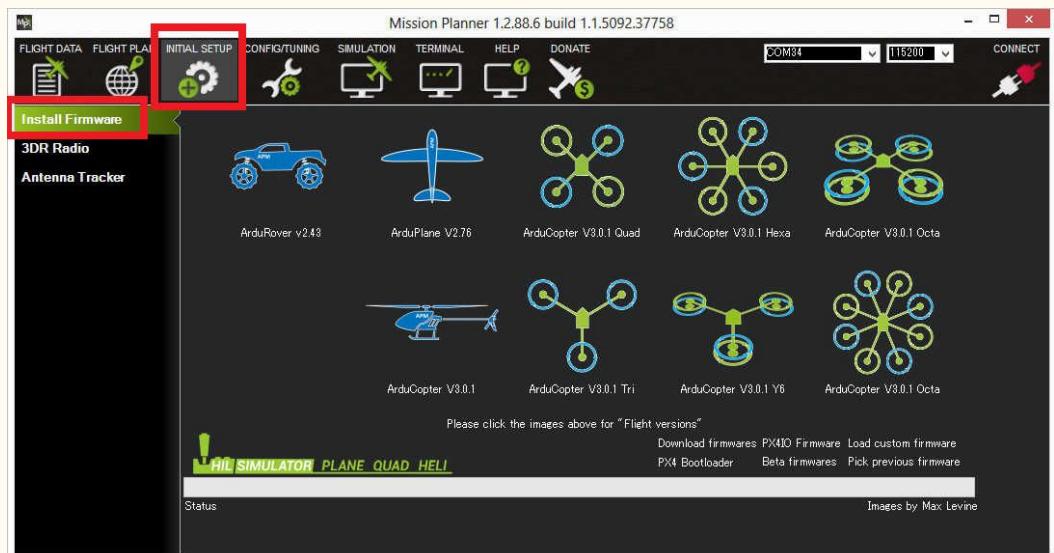


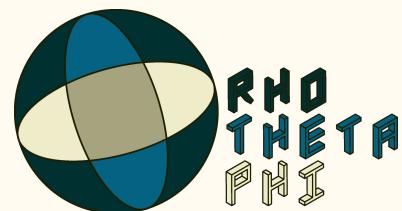
- 1 Spektrum DSM receiver
- 2 Telemetry (radio telemetry)
- 3 Telemetry (on-screen display)
- 4 USB
- 5 SPI (serial peripheral interface) bus
- 6 Power module
- 7 Safety switch button
- 8 Buzzer
- 9 Serial
- 10 GPS module
- 11 CAN (controller area network) bus
- 12 I²C splitter or compass module
- 13 Analog to digital converter 6.6 V
- 14 Analog to digital converter 3.3 V
- 15 LED indicator



Firmware Pixhawk

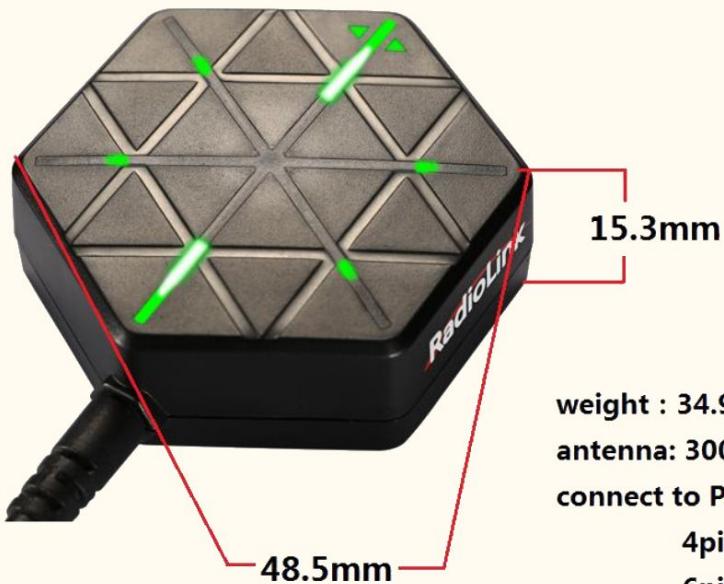
- Utilisation du logiciel Mission Planner (ArduPilot)
- Téléversement Firmware ArduRover
- Calibration des périphériques PixHawk





Radiolink SE100

Connexion I2C au PixHawk



weight : 34.9g
antenna: 300mm
connect to PIX:

4pin connect to I2C
6pin connect to GPS

