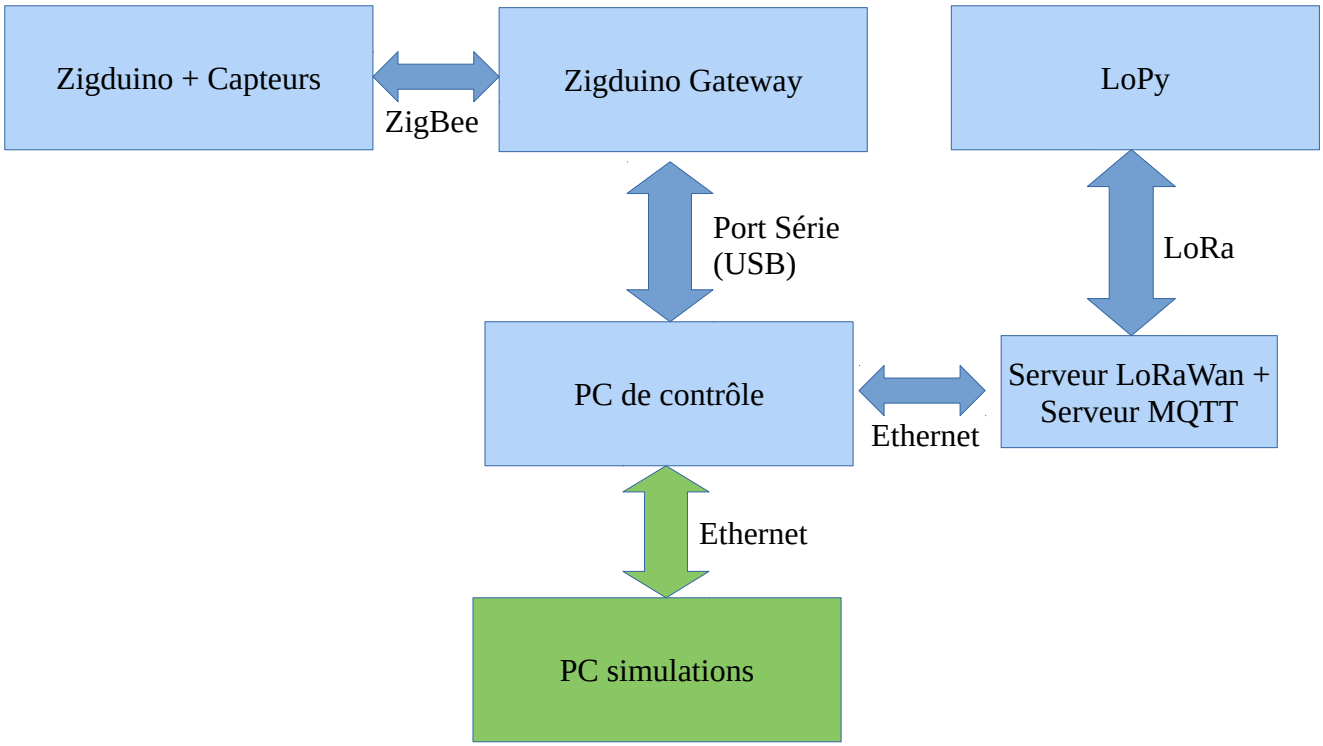
Rapport Projet
Robotique et domotique communicantes

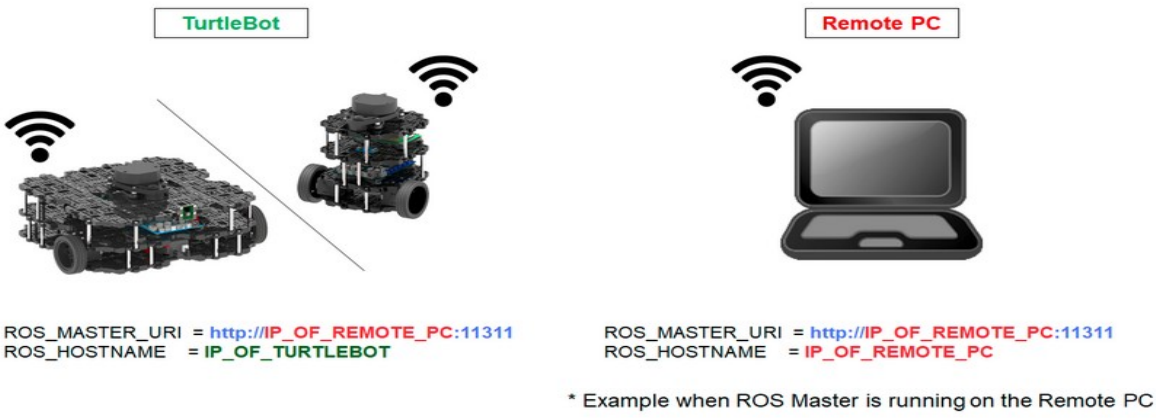
Bouvry Antonin
Seigneuret Constan
Pacheu Maxime

1/ Introduction

Architecture matérielle :

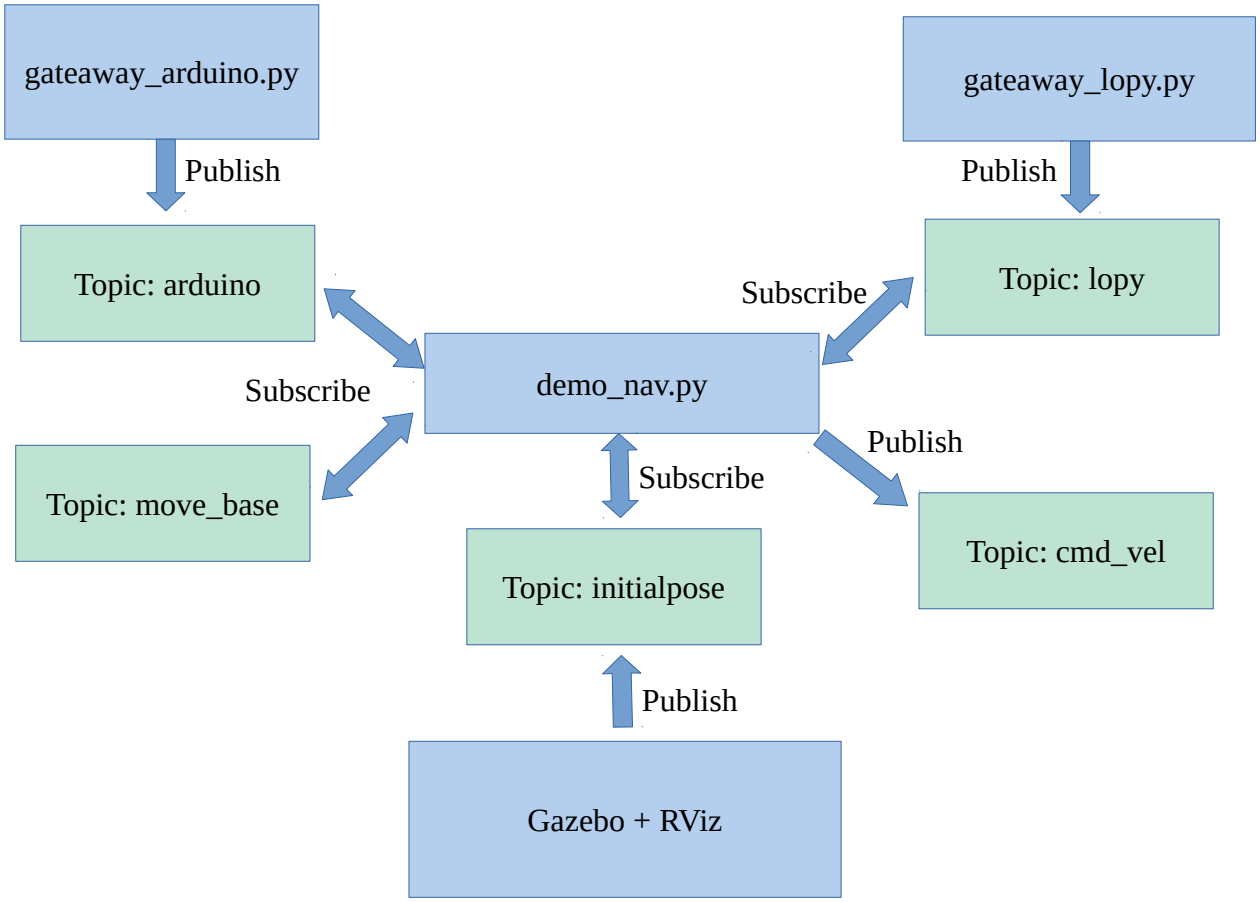


Nous n'avons pas implémenté la partie verte, c'est à dire la séparation de la simulation (logiciels Gazebo et RViz) et du programme de contrôle principal. Nous avons essayé de la faire en modifiant les fichiers bashrc des deux pc comme si-dessous :



Malheureusement nous avons rencontré un problème de droits qui nous empêchait de nous connecter à distance. Nous ne pouvions pas lancer le programme en root car l'installation de ROS n'a pas été faite pour l'utilisateur root.
De ce fait, tout s'effectue sur un seul PC dans notre projet.

Architecture logicielle :



Nous utilisons l'ensemble d'outils informatiques open source ROS pour simuler un robot turtlebot3.
Les logiciels de simulations utilisés sont :

- Gazebo : permet de voir le robot évoluer dans son environnement.
- RViz : permet de générer des cartes virtuelles, de simuler le déplacement du robot (en tenant compte de l'environnement). Ce logiciel nous permet de donner la position initial du robot.

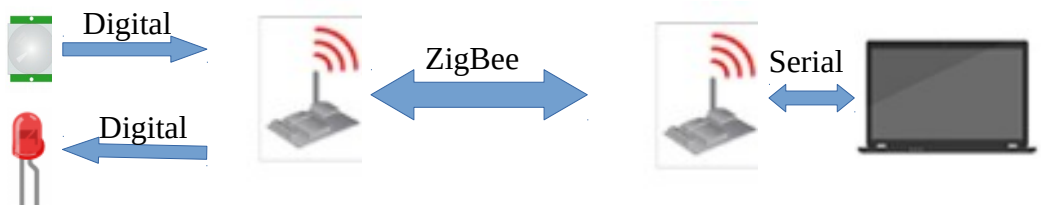
2/ Capteurs

On distingue deux catégories de capteurs : les cartes Zigduino et le LoPy.

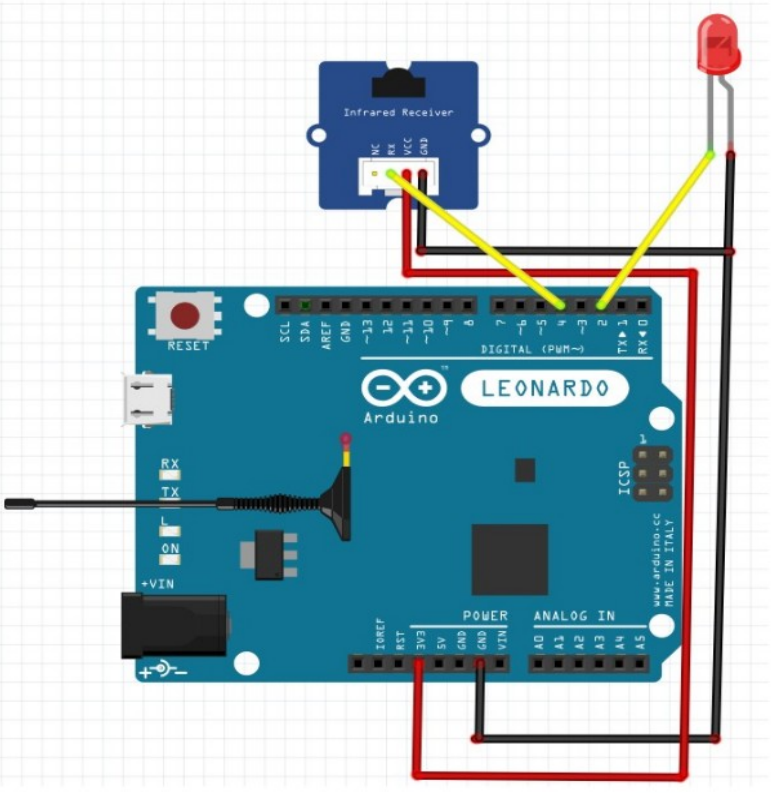
A) Zigduino :

Fonctionnement :

Capteur + LED ⟸⟹ Zigduino 1 ⟸⟹ 802.15.4 ⟸⟹ Zigduino 2 ⟸⟹ Port Série PC



Voici un schéma simple représentant les branchements réels sur les deux cartes. Ce schéma représente la carte sur laquelle les capteurs /actionneurs sont branchés. Le premier capteur est un capteur de présence permettant de détecter le mouvement, branché sur le port D4.
Le second est une LED simulant une ampoule, branchée sur D2.
Ce schéma représente les branchements sur une carte Leonardo, mais dans la réalité, nous avons utilisé un shield et une carte Zigduino.



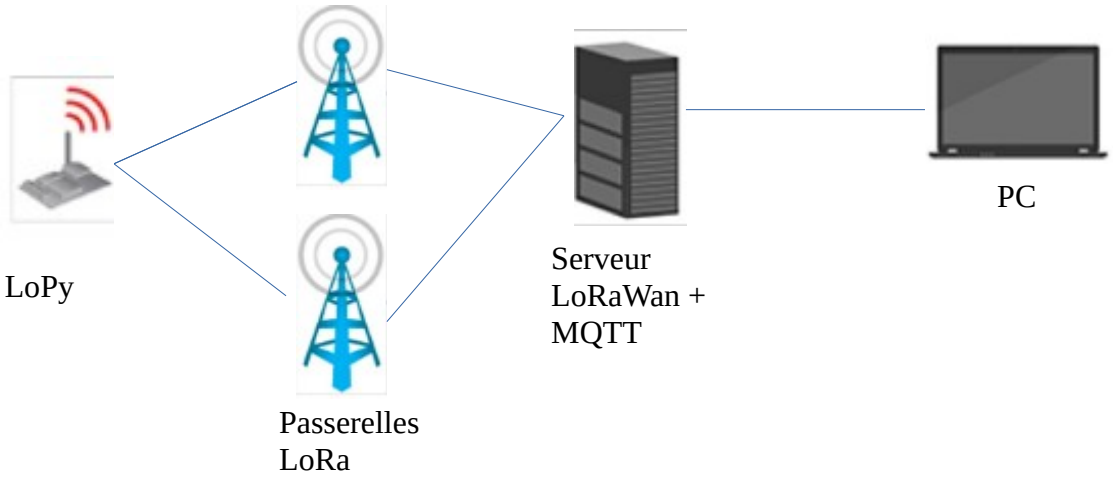
Nous avons conservé les modes passif et actif du TP4, mais nous n'utilisons que le mode passif ici.

Le mode passif consiste à capter la présence de mouvement et d'allumer la LED en conséquence de façon continu, sauf si on décide de passer en mode actif avec l'envoie d'une commande. Le mode passif de la Zigduino 1 envoie seulement les changements d'états de la LED à la Zigduino 2, qui les transmet sur le port série. Nous avons de plus ajouté une optimisation pour éviter des perturbations du capteur de mouvement en ajoutant un système d'anti rebond.

Nous avons modifié la librairie ZigduinoRadio pour pouvoir récupérer les 7 premiers octets, car celle-ci les ignore de base. Nous y mettons ainsi un identifiant pour représenter un destinataire.
Un programme python qui fait office de passerelle (gateway_arduino.py) récupère ainsi la commande sur le port série et l'envoie vers le programme principal. Pour cela, la passerelle crée un topic ROS et y publie la commande (l0 ou l1).

B) LoPy :

Notre équipement LoPy se connecte à une passerelle LoRa et y envoie un message lorsque son accéléromètre détecte un mouvement. Les passerelles envoient ensuite ce message au serveur LoRaWan qui dispose d'un broker MQTT. On se subscribe ensuite au serveur MQTT sur un programme annexe (gateway_lopy.py) pour récupérer le message et crée un topic ROS pour publier le message, qui sera récupéré par le programme principal. La librairie python pour MQTT est « paho.mqtt ».



Nous récupérons et envoyons le message en entier. Nous regardons seulement si nous avons reçu un message pour faire une décision. Pour récupérer une valeur précise, nous aurions pu importer la librairie JSON dans le code python afin de faire du parsing et décider en fonction de la valeur récupérée. Par exemple, dans le message ci-dessous, dans le champ « object », nous avons mis arbitrairement la valeur 12 pour simuler un ordre. On pourrait récupérer que cette valeur, le reste n'étant pas utile.


```
[INFO] [1547135339.063388, 178.326000]: application/14/device/70b3d5499462415d/rx {"applicationID":"14","applicat
ionName":"pytrack","deviceName":"pytrack-01","devEUI":"70b3d5499462415d","rxInfo":[{"gatewayID":"b827ebfffe0cbdc2
","name":"lorabox04","time":"2019-01-10T15:48:58.627147Z","rssi":-107,"loRaSNR":8.8,"location":{"latitude":48.525
74,"longitude":7.73782,"altitude":150}}],"txInfo":{"frequency":868100000,"dr":5},"adr":true,"fCnt":92,"fPort":2,"
data":"AAAMBXQPiw==", object":{"digitalInput":{"0":12}, voltage":{"5":3.979}}}
```


3/ Programme principal :

Le programme principal « demo_nav.py » permet de déplacer le robot dans la simulation selon deux modes. Au début du programme, il est demandé à l'utilisateur de pointer sur RViz la position initiale du robot et d'entrer le mode de fonctionnement souhaité.

```
INFO] [1547135140.646741, 10.675000]: Waiting for move_base action server...
INFO] [1547135140.794208, 10.795000]: Connected to move base server
INFO] [1547135140.795684, 10.798000]: *** Click the 2D Pose Estimate button in RViz to set the robot's initial pose...
```

```
[INFO] [1547135262.585685, 113.424000]: Starting navigation test
Mode programme ou mode evenementiel (prog/event) ?event
```

Le mode programmé permet de déplacer le robot vers trois positions en boucle (TP2).

Le mode événementiel attend qu'un message provenant du LoPy ou que la commande « l1 » provenant de la Zigduino arrivent. Si un message ou une commande arrivent, le robot se déplace vers la position du capteur concerné. Si cela arrive pendant le déplacement, il est ignoré et le robot continue sa course. Ce n'est qu'une fois arrivé à destination que le robot se remet en attente d'un événement. Si l'événement provient du capteur de la position actuelle, le robot n'en tient pas compte.

Pour récupérer un message du LoPy ou de la Zigduino, le programme se subscribe au topic correspondant.


```
[INFO] [1547135329.402674, 170.080000]: Destination suite a un event: lopy
[INFO] [1547135329.406394, 170.081000]: Updating current pose.
[INFO] [1547135329.407909, 170.083000]: Going to: lopy
[INFO] [1547135329.410612, 170.086000]: Goal failed with error code: RECALLED
[INFO] [1547135329.411784, 170.087000]: Success so far: 0/1 = 0%
[INFO] [1547135329.412719, 170.088000]: Running time: 2.6 min Distance: 0.0 m
[INFO] [1547135339.066497, 178.328000]: Changemnet de str_lopy: {"applicationID":"14","applicationName":"pytrack","devic
eName":"pytrack-01","devEUI":"70b3d5499462415d","rxInfo":[{"gatewayID":"b827ebfffe0cbdc2","name":"lorabox04","time":"201
9-01-10T15:48:58.627147Z","rssi":-107,"loRaSNR":8.8,"location":{"latitude":48.52574,"longitude":7.73782,"altitude":150}}
],"txInfo":{"frequency":868100000,"dr":5},"adr":true,"fCnt":92,"fPort":2,"data":"AAAMBXQPiw==","object":{"digitalInput":
{"0":12},"voltage":{"5":3.979}}}
```


Par exemple, on a récupéré via le topic « lopy » exactement le même message récupéré du broker MQTT qui a été publié dans le topic par la gateway lopy (message en haut de page).

Voici un exemple de déplacement du robot vers la position du LoPy suite à un évènement.