

RoBoNav

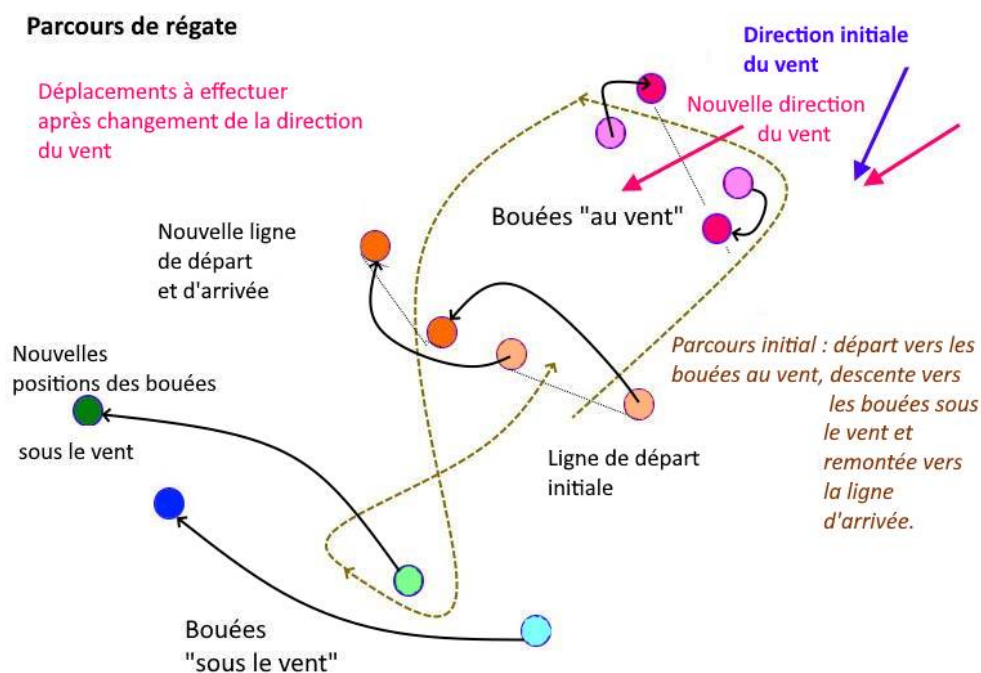
Bouées de régates autonomes

Application pour smartphone

Nicolas FERRY <nicolas.ferry@icam.fr> (ICAM),
Enora FREMY <enora.fremy@2024.icam.fr> ICAM),
Marie LOUVET <marie.louvet@2024.icam.fr> (ICAM),
Jean FRUITET <jean.fruitet@free.fr> (ARBL)

Lors de régates de voiliers radiocommandés l'une des difficultés dans le placement des bouées provient de la variabilité du vent qui peut changer plusieurs fois de direction en cours de régate en fonction des conditions météo. Or les régates doivent s'effectuer sur un parcours type constitué de 6 bouées disposées par paires en travers de l'axe du vent.

Le repositionnement est consommateur de temps et n'est pas sans risque pour ceux qui s'en chargent. Il nécessite de mettre une annexe à l'eau ; d'autre part il y a des plans d'eau sur lesquels il n'est pas possible de mouiller des bouées ce qui est un argument pour des bouées autonomes positionnées par GPS.



Problématique : repositionner les bouées en travers du vent quand la direction de celui-ci change.

Introduction

RoBoNav est un projet destiné à mettre en œuvre une constellation de bouées de régates autonomes pour la voile radio commandée.

Les bouées sont pilotées à distance soit par des radiocommandes soit avec une interface homme-machine tournant comme une application sur Smartphone ou Tablette.

Ce document détaille l'implantation de cette interface.

Architecture de l'application RoBoNav

Nous avons utilisé MIT App Inventor pour développer cette application. MIT App Inventor permet de créer une application graphique portable multi plateforme en programmation de blocs.

Lien permettant de créer une application Android gratuitement et en ligne:

<https://appinventor.mit.edu/>



L'application générée est de type Ground Control Station – Station de Contrôle à Terre destinée à piloter à distance une constellation de bouées autonomes dans la cadre du Projet RoBoNav.

L'interface mime une radio commande avec ses différents manches de contrôles (GAZ, ROTATION, INTERRUPTEUR), cartographie du plan d'eau et pour chaque bouée de la constellation, affichage de la position des bouées, état et niveau des capteurs.

Quand une action manuelle est effectuée sur l'un des actionneurs un ordre est envoyé à la bouée sélectionnée, soit pour la déplacer, soit pour lui attribuer une mission.

La communication à distance utilise un point d'accès WiFi et des trames (diagrammes) UDP. Les ordres sont communiqués sous la forme de "Blockchains" (train de chaînes de caractères). Le traitement des commandes reçues par la bouée est effectué par une carte électronique embarquée de type Arduino qui convertit les ordres reçus en signaux PWM activant différents effecteurs de bouée ciblée : interrupteurs et moteurs.

La bouée est munie de capteurs de position (GPS, Boussole) et de tension électrique dont les valeurs sont retournés à l'application permettant de positionner les bouées sur une carte du plan d'eau.

Les trains de commandes sont préfixés par des "mots clés" qui permettent de repérer et d'interpréter ces commandes .

Création d'une application MIT App Inventor

Le générateur d'application MIT App Inventor génère un fichier de type .apk pour Android qui peut ensuite être publié sur un Google PlayStore ou un AppleStore.

Procédure de mise en ligne de l'application Android : <https://fr.yeeply.com/blog/publier-application-google-play-store/>

Pour résumer les étapes, il faut:

- Créer un compte développeur sur GooglePlay (coût unique de 25 USD) ;
- Cliquer sur "Toutes les applications" puis "Créer une application" ;
- Créer et remplir une liste Play Stores (avec les informations accessibles aux utilisateurs)
- Téléverser le fichier .apk de l'application et choisir le type de publication (Test interne, fermé, ouvert)
- Définir la classification du contenu de l'application (primordial car si non classifié, l'application pourrait être retirée)
- Définir le prix et la distribution de l'application
- Soumettre l'application à un examen (il faut attendre un contrôle vert sur chaque section)
- Attendre que la version de déploiement soit disponible (cela peut prendre quelques jours) .

Page d'accueil de l'application RoBoNav

La page d'accueil a plus une fonction graphique que de commande de la bouée. Elle permet de présenter l'application et les personnes partenaires de l'application. Elle permet également de faire le lien avec la page où se trouve la télécommande et la page où se trouve la carte.



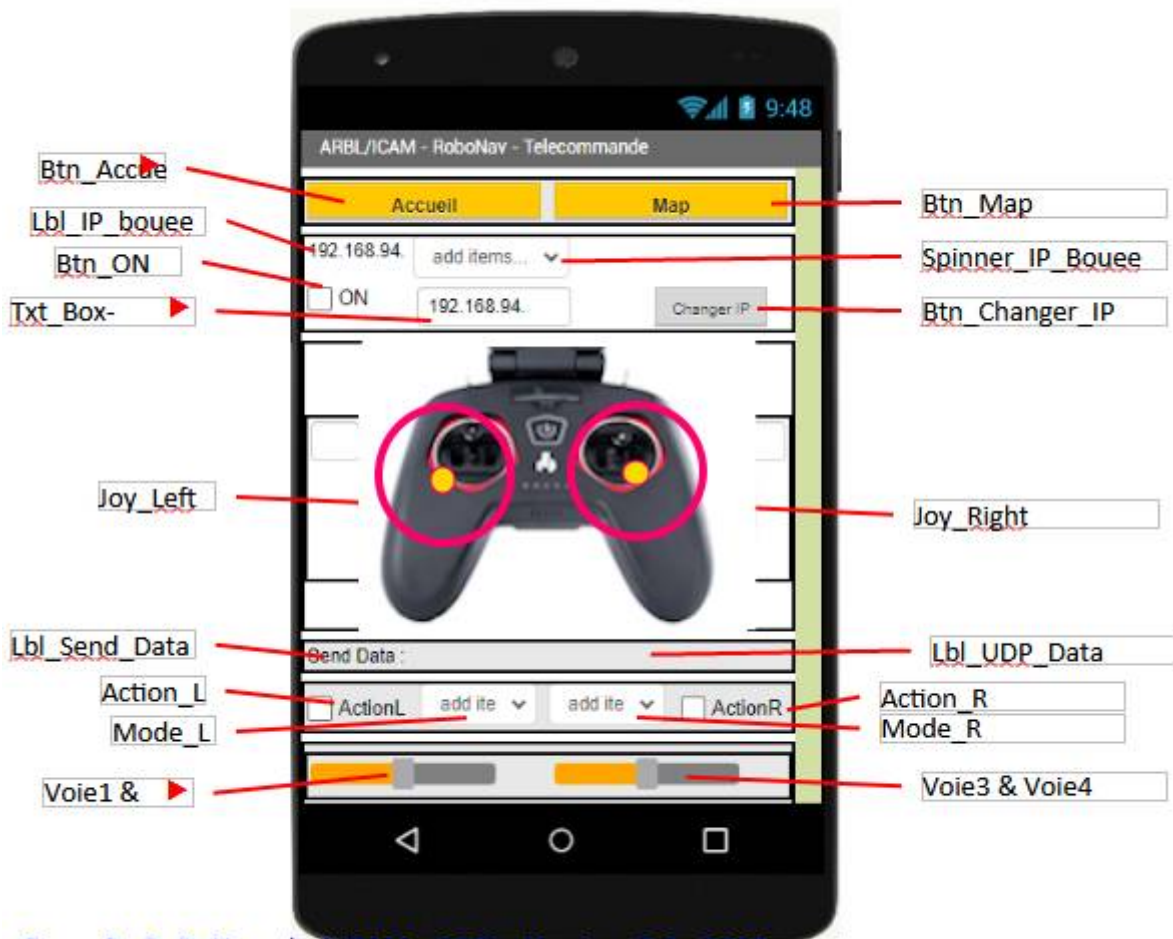
Ecran 1 - RoBoNav - (cc) ICAM - ARBL - 2023

Cette fenêtre n'ayant pas beaucoup de fonctionnalités, peu de blocs y sont associés. Seuls les blocs de changement d'écran (screen) existent. L'écran 2 (screen 2) correspond à la page de la cartographie (map).



Télécommande

Voici à quoi ressemble l'interface de l'application sur l'écran télécommande. En légende sont inscrites toutes les appellations utilisées dans la partie des blocs.

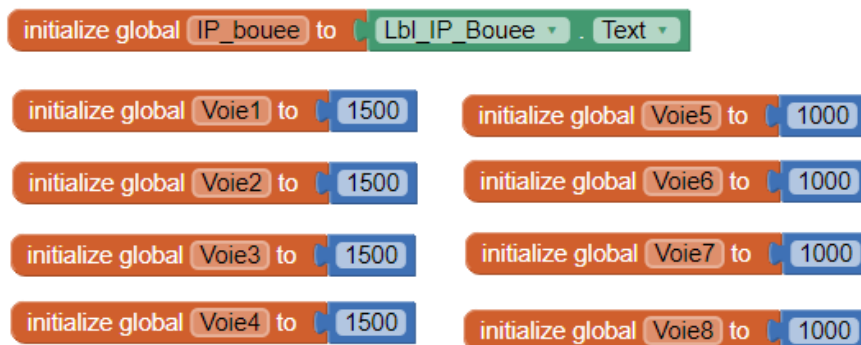


Ecran 2 - RoBoNav - (cc) ICAM - ARBL - Version 0.1 - 2023

Bloc d'initialisation

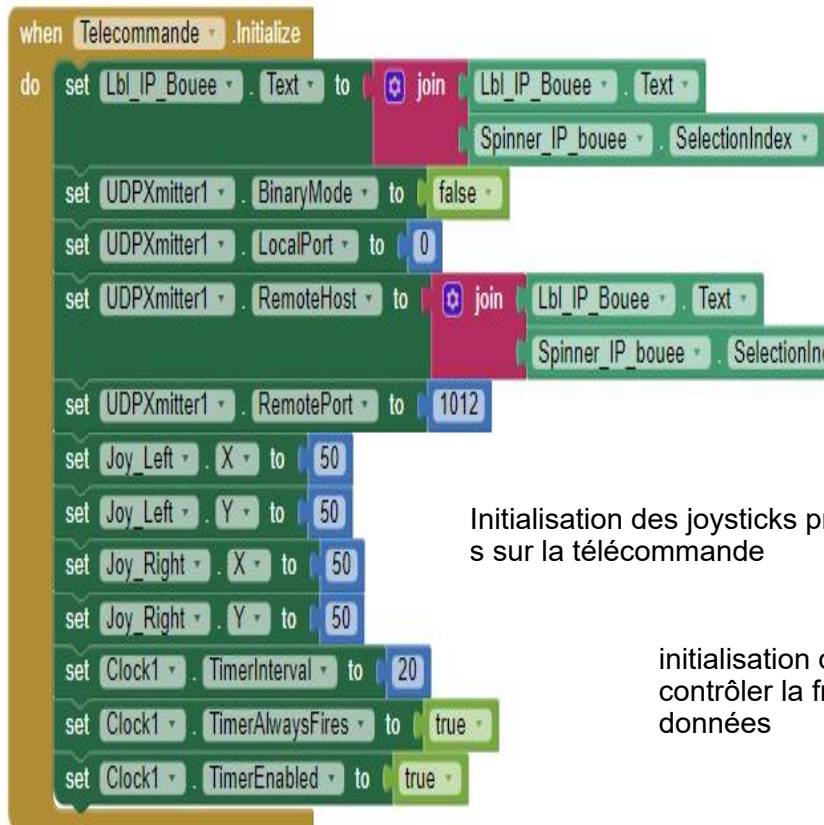
Ce bloc a pour objectif d'initialiser toutes les variables lors du lancement de l'application, ou lors de l'ouverture de nouvelle fenêtre.

Tout d'abord nous initialisons toutes les variables globales:



Chaque voie correspond à une voie physique d'une télécommande classique.

Puis nous initialisons les variables liées à la télécommande:



when **Telecommande** .Initialize

do

- set **Lbl_IP_Bouee** .Text to join **Lbl_IP_Bouee** .Text and **Spinner_IP_bouee** .SelectionIndex
- set **UDPXmitter1** .BinaryMode to false
- set **UDPXmitter1** .LocalPort to 0
- set **UDPXmitter1** .RemoteHost to join **Lbl_IP_Bouee** .Text and **Spinner_IP_bouee** .SelectionIndex
- set **UDPXmitter1** .RemotePort to 1012
- set **Joy_Left** .X to 50
- set **Joy_Left** .Y to 50
- set **Joy_Right** .X to 50
- set **Joy_Right** .Y to 50
- set **Clock1** .TimerInterval to 20
- set **Clock1** .TimerAlwaysFires to true
- set **Clock1** .TimerEnabled to true

Initialisation du label de la bouée

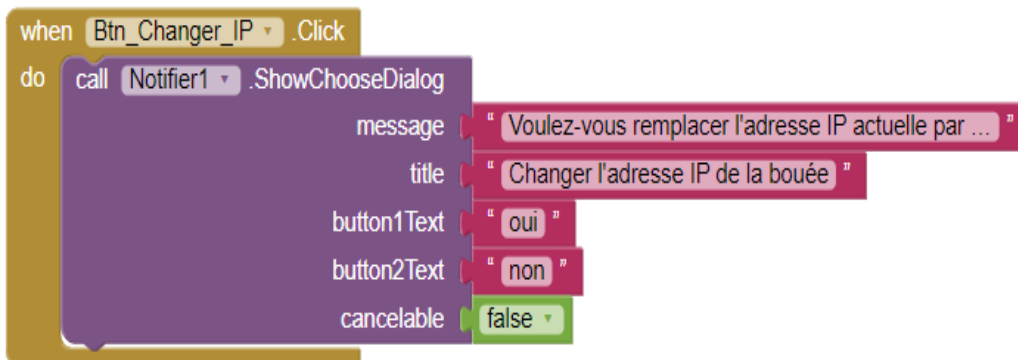
Initialisation de L'UDPXmitter1 qui permet la transmission d'information de l'app vers la bouée

Initialisation des joysticks présents sur la télécommande

initialisation du timer permettant de contrôler la fréquence d'envoi des données

Changement de l'adresse IP de la bouée

Lorsque l'on clique sur le bouton *changer IP*, alors un pop-up apparaît avec le texte du *title* et du *message* ainsi que 2 boutons: "oui", "non"

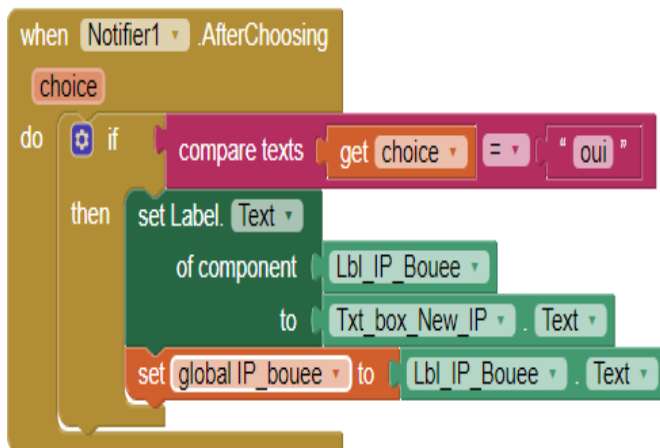


when **Btn_Changer_IP** .Click

do

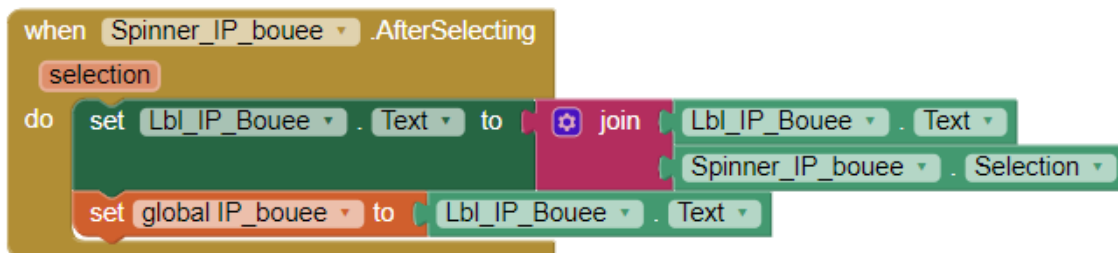
- call **Notifier1** .ShowChooseDialog
 - message "Voulez-vous remplacer l'adresse IP actuelle par ..."
 - title "Changer l'adresse IP de la bouée"
 - button1Text "oui"
 - button2Text "non"
 - cancelable false

Si l'utilisateur choisit de cliquer sur « oui », alors le bloc suivant permet de rentrer et dans le label et dans la variable *IP_bouee* la nouvelle adresse IP (texte qui était noté dans la Texte Box "Txt_box_Nex_IP").



Ajout du numéro de bouée à la fin de l'adresse IP, en fonction du choix de bouée

Ce bloc permet que lorsque l'on change le numéro de bouée choisit dans le *spinner*, le numéro choisi vient s'ajouter à l'adresse IP préexistante.

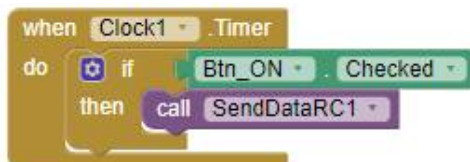


N.B. : Pour le moment les chiffres ne viennent pas se remplacer mais s'ajouter. Ainsi pour changer de numéro de bouée, il faut d'abord changer l'IP grâce au bouton, puis choisir le numéro de bouée !

Contrôler l'envoi des données

Afin de contrôler l'envoi des données et éviter la saturation des informations envoyées à la carte de programmation lors de l'utilisation de la télécommande présente sur l'application, nous avons utilisé une horloge (ou clock)

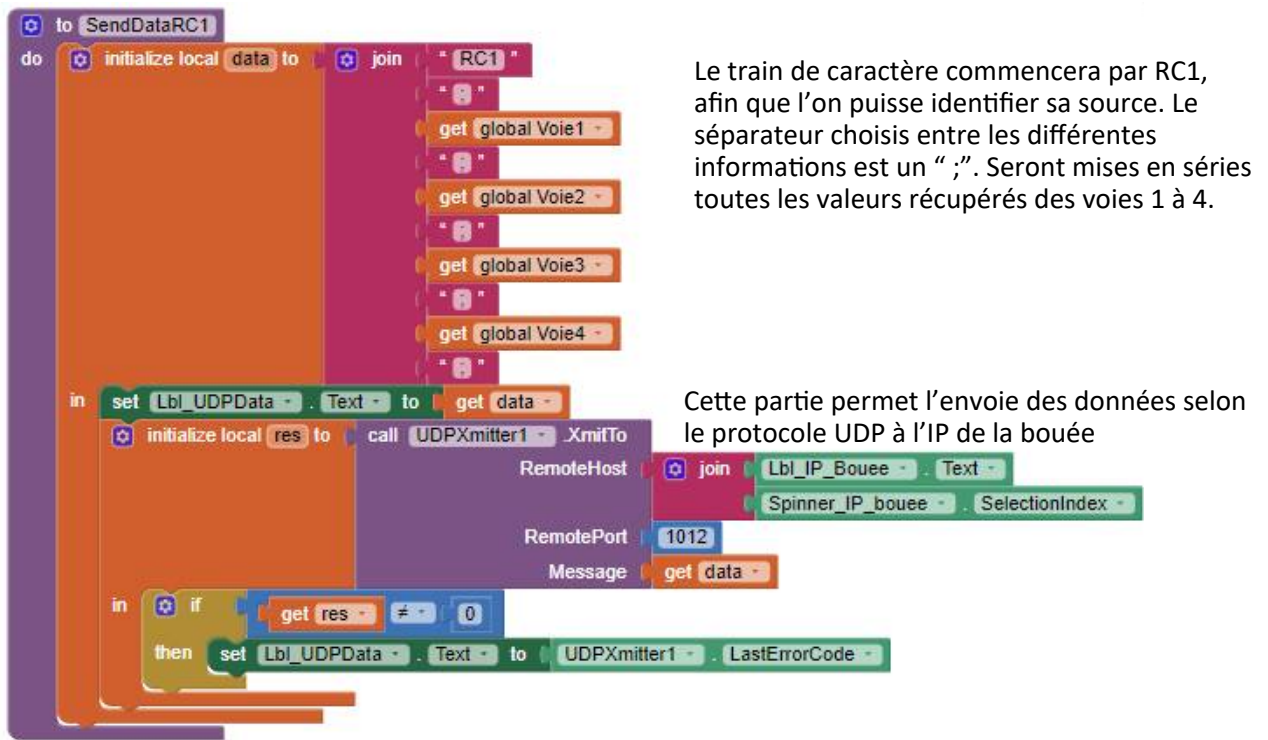
Voici le bloc associé à cette fonction:



Envoi des blocs de caractères correspondant aux voies de la bouée:

Pour pouvoir faire le lien entre l'application et la bouée, nous envoyons des trains de données grâce au protocole UDP.

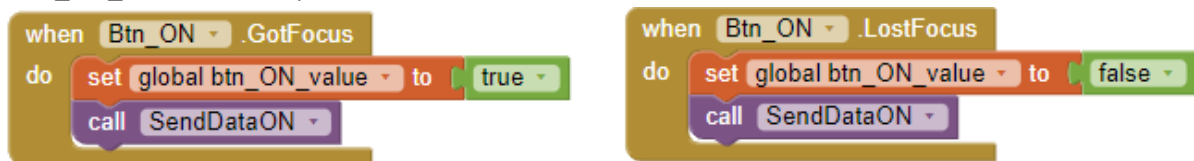
Pour récupérer les données concernant RC1 (le mode 4 voies), voici le bloc utilisé:



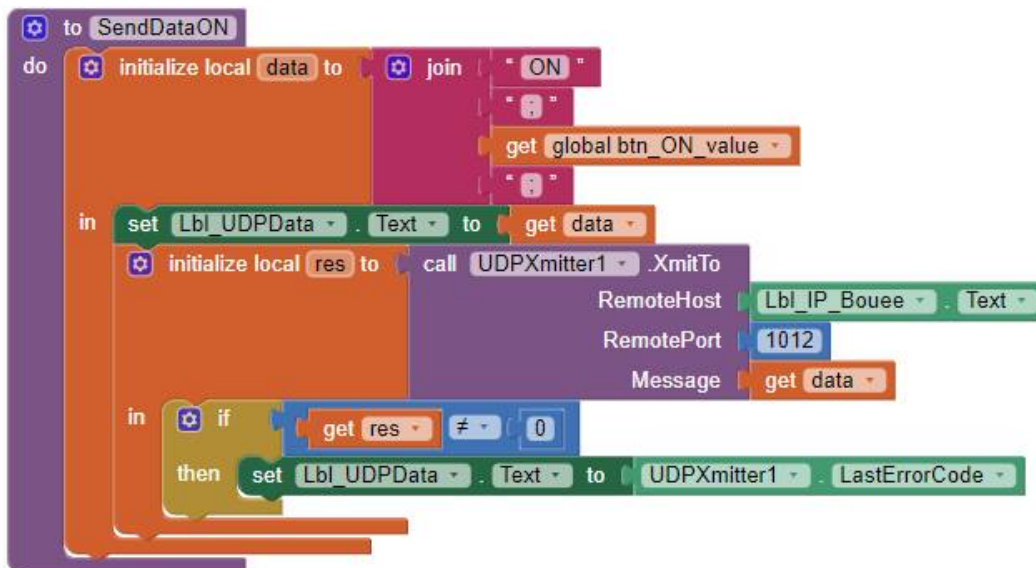
De même, nous avons construit le même type de bloc pour RC2, mais cette fois-ci avec non pas 4 voies dans le haut du bloc, mais 8 voies.

Envoie des blocs de caractères correspondant au “Bouton ON” de la bouée:

Ces blocs correspondent à l’appelle de la fonction “SendDataON” lorsque le bouton est coché et décoché. Lorsque celui-ci est coché, on attribue la valeur “true” à la variable globale “btn_ON_value”, et lorsque celui-ci est décoché, la valeur false.

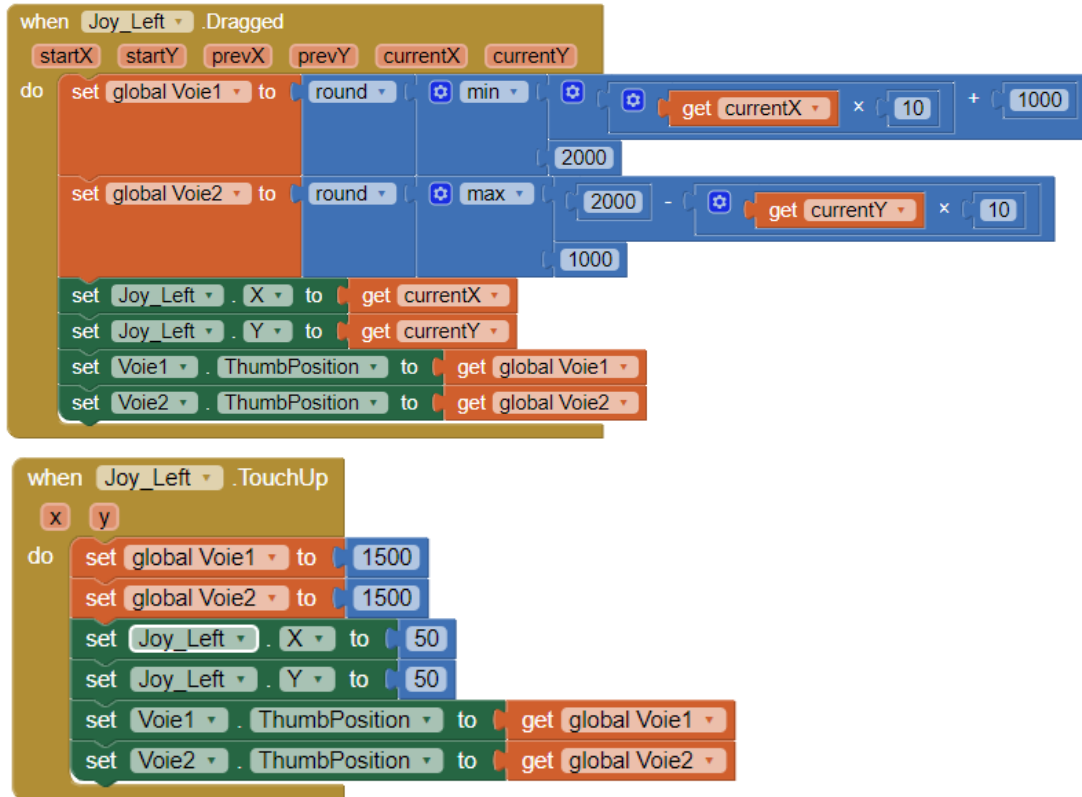


La fonction “SendDataON” correspond à l’envoi du train de caractère à la carte programmable:



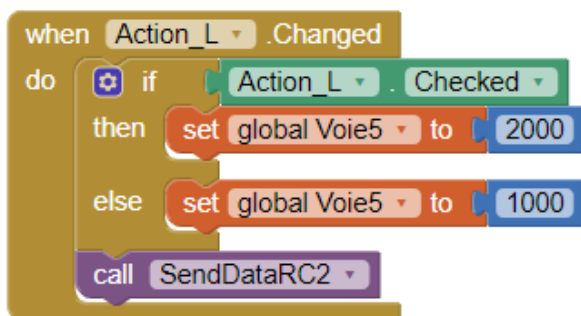
Récupération des données du joystick et mise à l'échelle des valeurs

Les blocs ci-dessous correspondent aux blocs permettant de modéliser les joysticks de la télécommande. Vous ne trouverez que ceux pour le joystick de gauche. Cependant il existe exactement le même bloc pour le joystick droit.



Modélisation des interrupteurs de la télécommande

Afin de modéliser les huit voies de la télécommande, nous avons rajouté des boutons "Action L" et "Action R" pour modéliser les commutateurs 2 positions, ainsi que des spinners "Mode L" et "Mode R" pour modéliser les commutateurs trois positions.



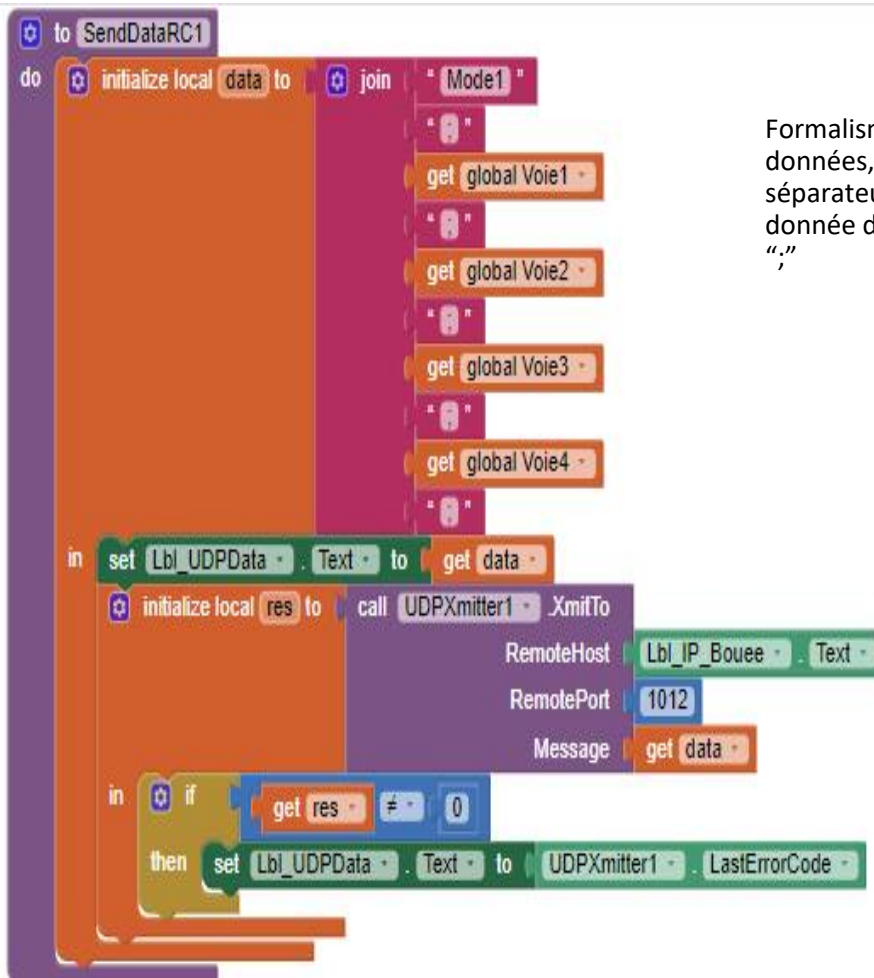
Bloc permettant que lorsque le Action_L est sélectionné, une valeur de 2000 est renvoyée sur la voie 5



Ce bloc permet qu'en fonction de la sélection (Mode 1, 2, 3), les valeurs renvoyées sur la voie 6 soient différentes

Récupération des données de la voies 1 à 4

Le rôle de l'application est de capter les ordres réalisés sur l'écran et de les transmettre à la bouée. Pour cela nous utilisons un protocole UDP, et envoyons les valeurs récupérées sur chaque voies sous forme d'un train de caractères. Voici le bloc permettant cet envoi.

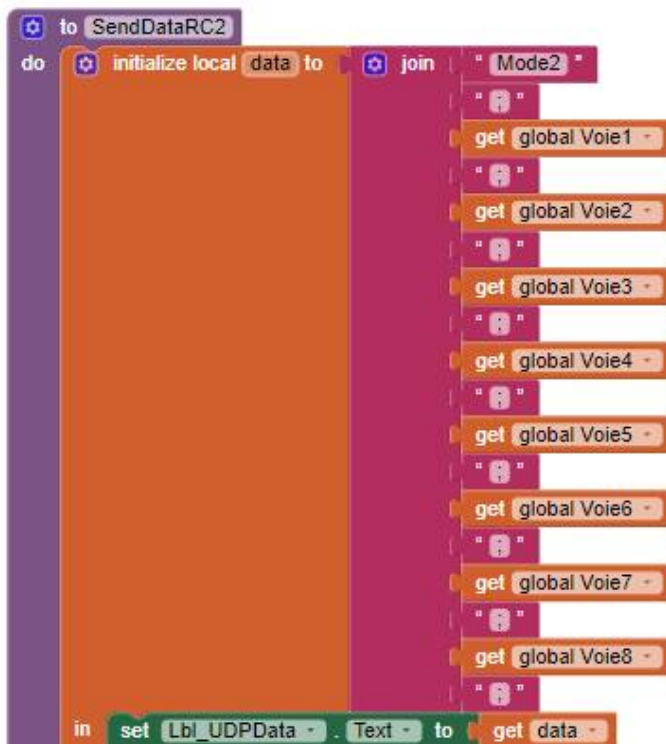


Formalisme d'envoi des données, avec pour séparateur de chaque donnée de chaque voie: un ","

Protocole UDP d'envoi des données à la carte Arduino par Wifi

Récupération des données de la voie 1 à 8:

Pour la récupération des 8 voies d'un seul coup, nous avons réalisé le même type de bloc que dans la partie ci-dessus, en ajoutant simplement les voies de 5 à 8 dans la première partie du bloc. Voici la partie concernée par le changement dans le bloc.

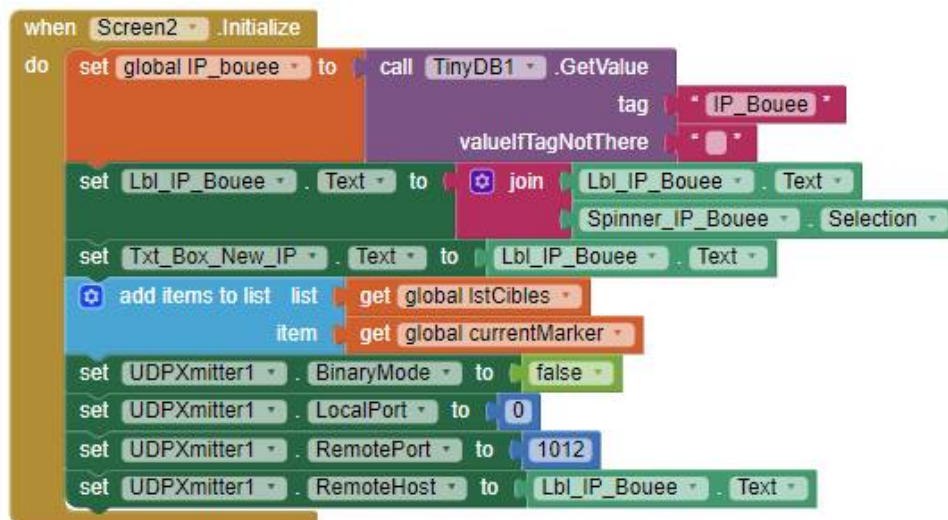


Map



Map - RoBoNav - (cc) IRCAM - ARBL - 2023

Initialisation des paramètres



Création d'un point GPS cible sur la carte:

The script is triggered by a 'when Map_Zone - LongPressAtPoint' event. It contains the following blocks:

- do** block:
 - initialize local** block: 'newMarker' to 'call Map_Zone - CreateMarker'.
 - call** block: 'Map_Zone - CreateMarker' with inputs 'latitude' (from 'get latitude -') and 'longitude' (from 'get longitude -').
 - in** block:
 - set Marker. Title -** block: 'of component' 'get newMarker -' to 'join' 'Cible N°' and 'length of list - list' (from 'get global IstCibles -').
 - set Marker. Description -** block: 'of component' 'get newMarker -' to 'length of list - list' (from 'get global IstCibles -').
 - set Marker. Draggable -** block: 'of component' 'get newMarker -' to 'true -'.
 - set Marker. Visible -** block: 'of component' 'get newMarker -' to 'true -'.
 - insert list item** block: 'list' (from 'get global IstCibles -') at 'index' 'length of list - list' (from 'get global IstCibles -') with 'item' 'get newMarker -'.
 - set Map_Zone - Features -** block: 'to' 'get global IstCibles -'.
 - set global currentMarker -** block: 'to' 'get newMarker -'.
 - call** block: 'UpdateCibleDisplay -'.

Création du marker

Association des paramètres (GPS et n° de bouée) au nouveau marker créé

Incrémentement du numéro de bouée en fonction du nombre de cible déjà créés

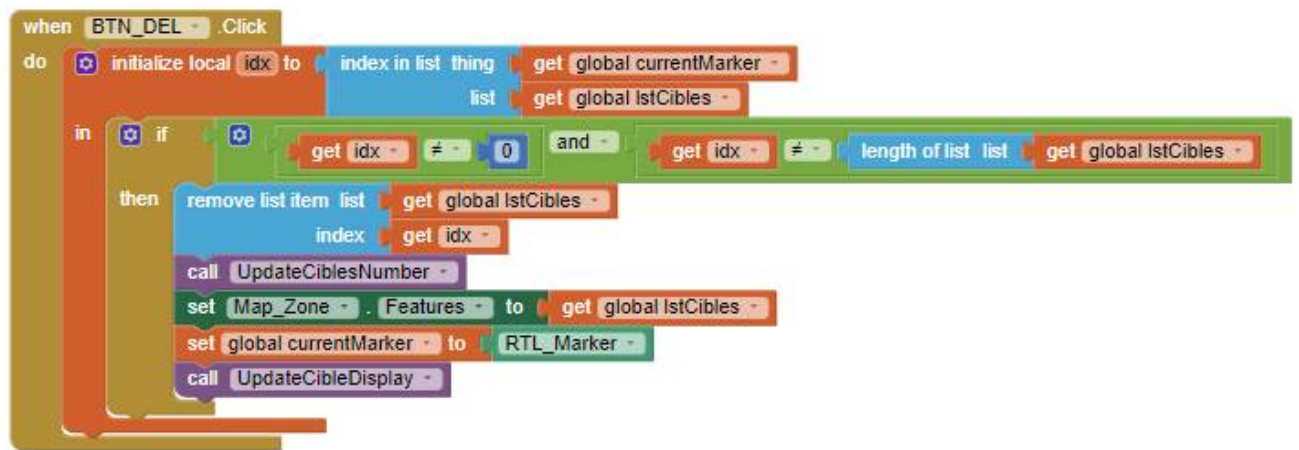
Ce bloc fait appel, à la fin, à la fonction "UpdateCibleDisplay". La fonction est détaillée ci-dessous. Cette fonction permet de mettre à jour l'affichage des données concernant le numéro de cible, et les coordonnées GPS du marqueur créé.

The function is defined as 'to UpdateCibleDisplay' and contains the following blocks:

- do** block:
 - initialize local** block: 'Bouee' to 'get global currentMarker -'.
 - in** block:
 - set Label1 - Text -** block: 'to' 'Marker. Title -' (from 'of component' 'get global currentMarker -').
 - set Lbl_Lat - Text -** block: 'to' 'Marker. Latitude -' (from 'of component' 'get global currentMarker -').
 - set Lbl_long - Text -** block: 'to' 'Marker. Longitude -' (from 'of component' 'get global currentMarker -').

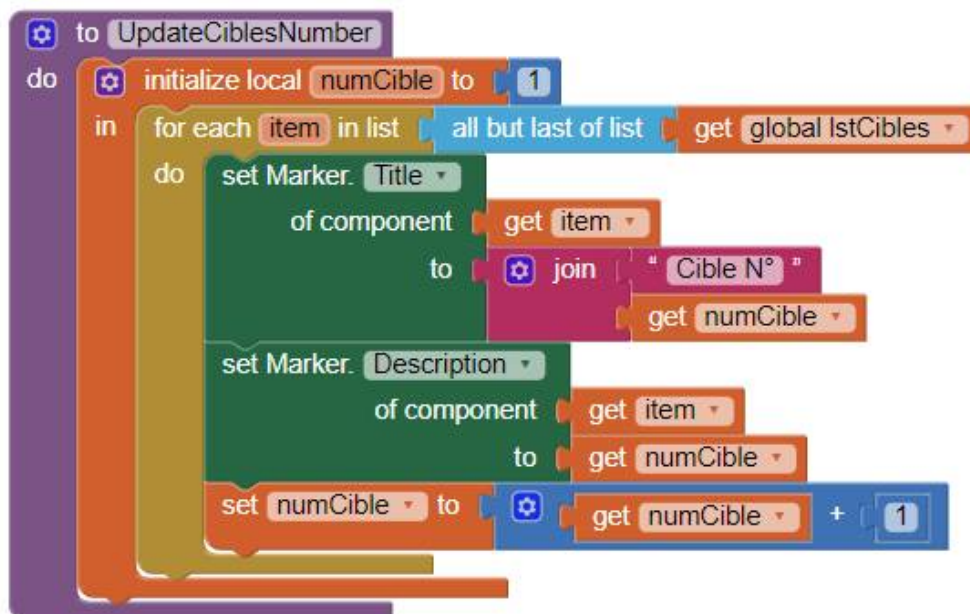
Suppression d'un marqueur cible:

Ce bloc permet que lorsque l'on appuis sur le bouton "DEL", le marqueur sélectionné à ce moment-là est supprimé, et les données associées également.



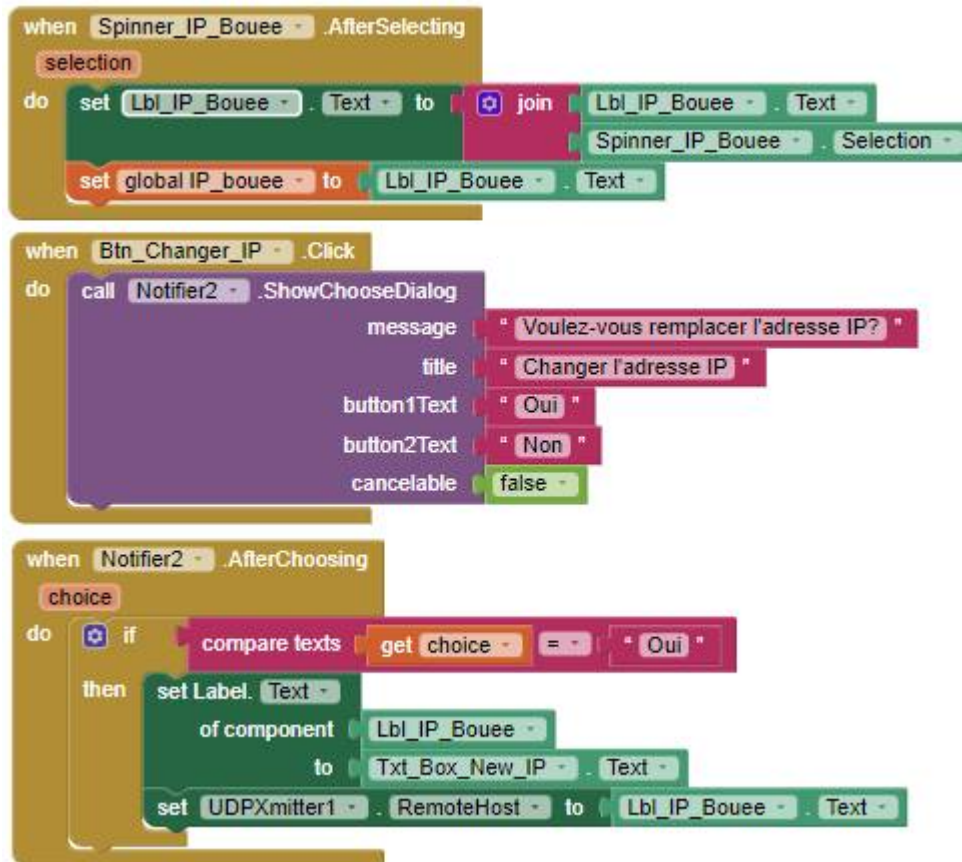
Ce bloc fait appelle à deux fonctions:
 UpdateCibleDisplay (déjà présenté ci-dessus)
 UdateCiblesNumber

Cette deuxième fonction permet d'ajouter un numéro de bouée, et donc une nouvelle cible dans la liste des bouées:



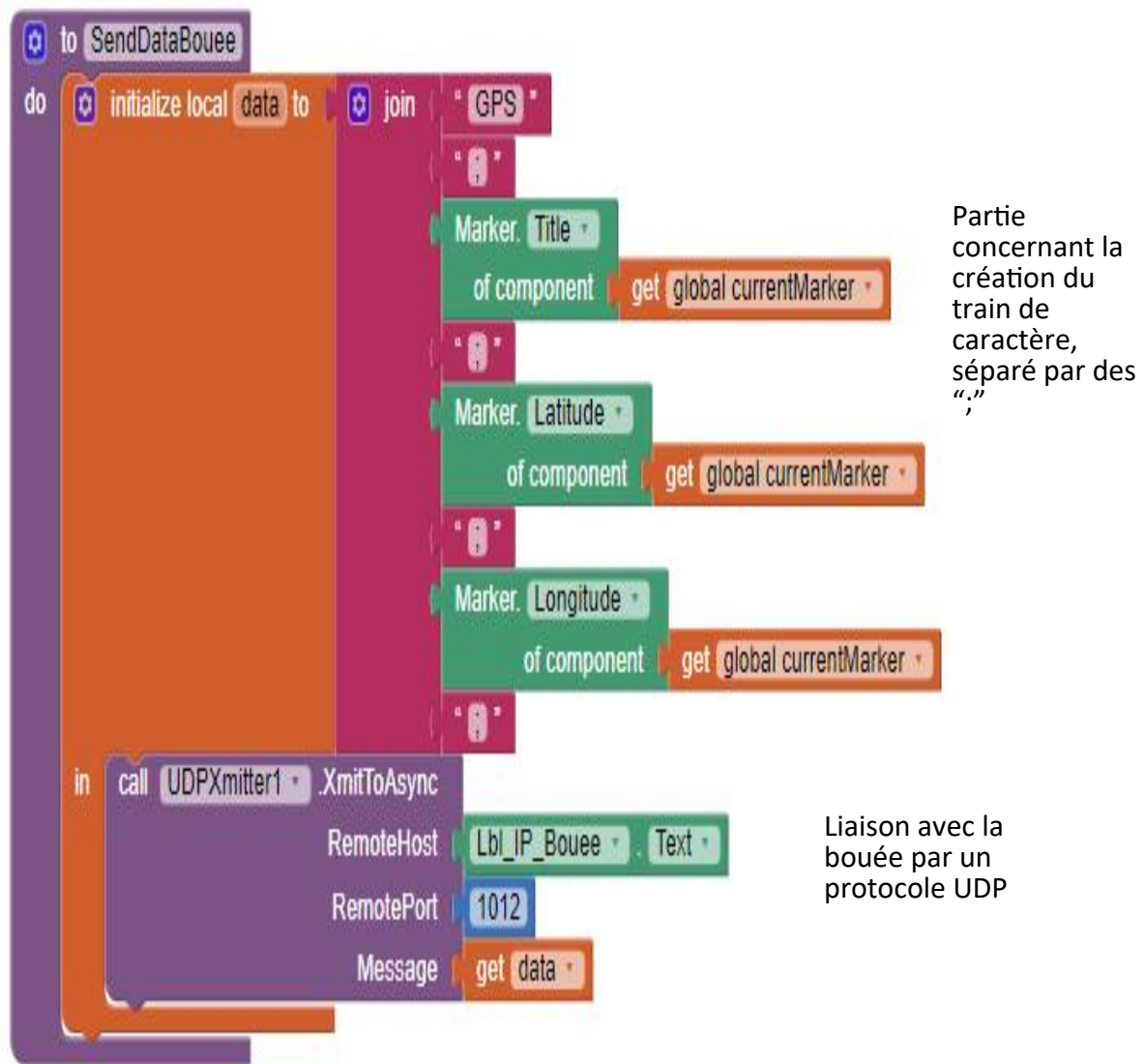
Gestion de l'adressage IP de la bouée:

Pour le changement d'adresse IP, nous avons procédé de la même manière que pour la télécommande, voici donc les blocs concernés:

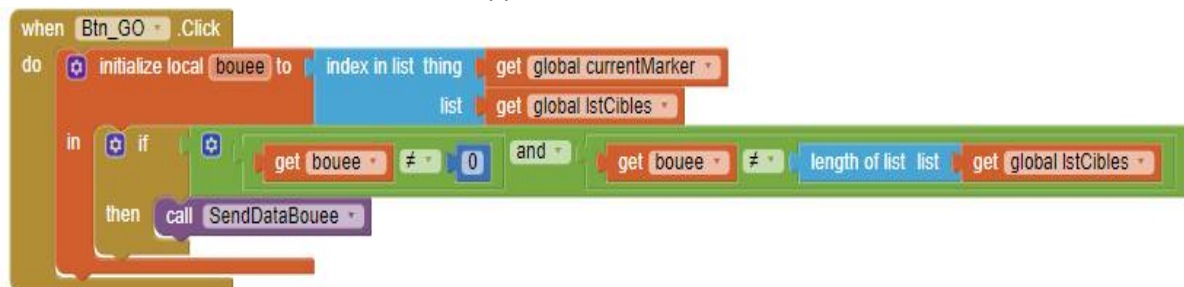


Envoi du train de caractère pour récupérer le numéro de bouée et les coordonnées GPS de la bouée:

Pour la réalisation de l'envoi de trains de caractère permettant de récupérer pour une bouée donnée les coordonnées (longitude et latitude) visées, nous avons utilisé deux blocs différents. Tout d'abord nous avons fabriqué la fonction permettant de créer le train de caractère que nous souhaitons envoyer. Nous avons choisi le mot "GPS" comme mot repère du bloc, qui nous permettra ainsi d'identifier la réception d'un bloc possédant les coordonnées visées de la bouée lors du traitement des blocs dans la partie Arduino.



Puis nous avons réalisé les conditions d’appel de la fonction :



Lorsque le Bouton Go est cliqué, alors les coordonnées de la cible sélectionnée sont envoyées au récepteur WiFi de la carte embarquée.

Conclusion

Ce projet a été réalisé dans le cadre de la formation de 4ème année d'Enora FREMY et Marie LOUVET à l'ICAM de Nantes sous la direction de Nicolas FREMY.

<https://www.icam.fr/les-campus/icam-nantes/>

Jean FRUITET à assuré le suivi du projet pour l'Association Radiomodéliste des Bords de Loire

<https://www.arbl.fr>

Il est soumis à une licence de CopyLeft Creative Commons « Attribution - Partage dans les Mêmes Conditions » [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/)



Références

« RoBoNav - Bouées robot pour la VRC », projet de mineure en quatrième année de formation à l'Ecole d'ingénieur ICAM Nantes.

MIT App Inventor : Générateur d'application multi plateforme par assemblage de blocs.

<https://appinventor.mit.edu/>

Ce projet utilise une extension MIT App Inventor pour la communication UDP entre les bouées.

<https://ullisroboterseite.de/android-AI2-UDP-en.html>

Ce projet est inspiré des BuoyBots de Andrew R. Wilson <andrewRwilson62@gmail.com>

<https://www.youtube.com/watch?v=dJOPf-hZqxM&t=12s>

<https://www.facebook.com/buoybot/>

<https://www.buoybot.com.au/>

<https://www.youtube.com/channel/UC4bFbzEi-i1VW942aT6dpFA>