Ajout d’une nouvelle fonction dans l’interface ArdPilot

Table des matières

[Création du bouton 3](#_Toc509261996)

[Fonctions gateway dans le serveur node.js 4](#_Toc509261997)

[Implémentation dans le serveur (sur le raspberry) 5](#_Toc509261998)

[Appel d’une fonction sur le robot depuis le serveur 6](#_Toc509261999)

[Implémentation de la fonction dans le robot 7](#_Toc509262000)

[Réception et gestion du code retour du robot dans le serveur 8](#_Toc509262001)

# Création du bouton

XCode [ Raspberry/projets/ardpilot/node/index.html ]

Placer à l’endroit souhaité le bouton :

<input type="button" value="Nom affiché" id="Identifiant unique" />

Dans la méthode « $(document).ready » ajouter la fonction d’action :

$('#Identifiant').click(function () {

this.style.background = 'SandyBrown';

// si on souhaite un changement de couleur

//temporaire en attente du résultat

socket.emit('Identifiant', 'parametre');

// ‘parametre’ est une chaine de caractères

})

Et enfin si on souhaite colorer le bouton en fonction d’un résultat (true / false), toujours dans la méthode « ready » :

socket.on('valeur True Unique', function(message) {

document.getElementById("locate").style.background='LightGreen';

})

socket.on('valeur False Unique', function(message) {

document.getElementById("locate").style.background='LightCoral';

})

# Fonctions gateway dans le serveur node.js

Créer les fonctions passe-plat dans le serveur node.js.

Xcode [ Raspberry/projets/ardpilot/node/index.js ]

1. Depuis interface web vers le serveur :

dans la fonction io.sockets.on('connection'…

socket.on(‘Identifiant’, function (message) {

client.write('IDT');

});

Identifiant est la chaine de caractère passée en premier paramètre à « socket.emit » dans index.html

Si il y a un parametre (reçu dans ‘message’) on peut le concaténer pour le passer au serveur:

socket.on(‘Identifiant’, function (message) {

client.write('IDT' + message);

});

1. Retour du serveur vers l’interface Web

Dans la fonction client.on(‘data’… ajouter un test

else if (chaine.indexOf('valeur retour F') == 0) {

socket.emit('valeur False Unique', ‘message F\n');

} else if (chaine.indexOf('valeur retour T’) == 0) {

socket.emit('valeur True Unique', ‘message T\n');

}

‘message T ou F’ sera affiché dans la console de message de l’interface Web, en plus de la mise à jour de la couleur du bouton.

# Implémentation dans le serveur (sur le raspberry)

Xcode [ Raspberry/projets/ardpilot/src/arduino.c ]

Dans la fonction read\_cmd :

else if (!strncmp(buffer, "IDT", 3))

{

fonction();

}

// IDT est l’identifiant reçu du serveur node.js

// 3 est en fait la longueur de cet identifiant (ici 3 en l’occurrence)

S’il y a un paramètre (chaine de caractère) à prendre en compte :

else if (!strncmp(buffer, "IDT", 3))

{

sprintf (message, "%s", buffer+3);

fonction(message);

}

S’il y a un paramètre (un caractère) à prendre en compte :

else if (!strncmp(buffer, "IDT", 3))

{

fonction(buffer[3]);

}

Si on souhaite renvoyer le code retour à l’interface pour changer la couleur du bouton :

else if (!strncmp(buffer, "IDT", 3))

{

if (fonction() == 1)

control\_message(MSG\_INFO, "valeur retour T");

else

control\_message(MSG\_INFO, "valeur retour F");

}

Déclarer la fonction dans le fichier adequat

Compiler :

Sur le raspberry

cd **~/**projets/ardpilot/src

make

# Appel d’une fonction sur le robot depuis le serveur

Définir le code de la fonction dans le fichier de grammaire

Xcode [ Arduino/z\_ardpilot/grammar.h ]

Ajouter un code contrôle unique dans la section // controls

#define C\_MA\_COMMANDE 0xHH

Fonction bloquante (on attend de recevoir le résultat avant d’accepter autre chose :

int fonction ()

{

unsigned char sequence;

int i;

sequence = Sequence[C\_MA\_COMMANDE] + 1;

if ((sequence == '\*') || (sequence == 0x00))

sequence++;

corps\_fonction (sequence);

boucle\_attente (C\_MA\_COMMANDE, sequence, 0);

return 1;

}

La fonction boucle\_attente va attendre que le compteur «  Sequence[C\_MA\_COMMANDE] » soit incrémenté (lors d’une interruption asynchrone en réception d’un message du robot).

Sinon on peut appeler directement la fonction corps :

void corps\_fonction (char sequence)

{

extern int ardfd;

char message[4];

message[0] = C\_MA\_COMMMANDE;

message[1] = sequence;

message[2] = '\0'; // ou bien des valeurs de paramètre

// mais toujours finir par \0

write\_ard (ardfd, message);

}

Dans ce cas, le serveur n’est pas bloqué et peut faire autre chose en attendant d’être interrompu par le retour du robot (le cas échéant). Il peut même envoyer d’autres commandes au robot.

# Implémentation de la fonction dans le robot

Lancer l’IDE Arduino, aller dans l’onglet d\_ardpilot\_commandes

Dans la fonction « manage\_command » ajouter un cas :

case C\_MA\_COMMANDE:

ma\_commande (sequence);

break;

Corps de la fonction et retour du résultat vers le serveur :

void ma\_commande (unsigned char sequence)

{

// execution du code

// …

// envoi d’un parameter en retour

obuffer[0] = C\_MA\_COMMANDE;

obuffer[1] = sequence;

itoa (UneValeurEntièreParExemple, &(obuffer[2]), 10);

wifi\_write ();

}

Compiler le croquis :

Brancher le cable USB sur le robot

Maintenir enfoncé le bouton « reset » de la carte WIFI (bouton vertical sur le côté droit du robot) afin de libérer le port série

Cliquer sur le bouton « téléverser »

Relacher le bouton « reset »

# Réception et gestion du code retour du robot dans le serveur

Xcode [ Raspberry/projets/ardpilot/src/arduino.c ]

Dans la fonction read\_ard ajouter un cas :

case C\_MA\_COMMANDE :

sscanf(&(buffer[2]), "%d", &x);

// si j’ai un paramètre entier à récupérer, par exemple

// eventuellement du code à réaliser au retour

control\_message(MSG\_INFO, "Un message à afficher dans la console");

break;

Ne pas mettre de code ici si on est sur un appel bloquant.

La page HTML peut intercepter le message reçu par « control\_message » et faire autre chose que l’afficher dans la console (comme le cas de la boussole par exemple)