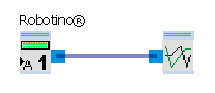
**LECTURE DES CAPTEURS ANALOGIQUES/DIGITAUX (en local)**

1. **Tester les capteurs**

Afin de savoir si les capteurs fonctionnent, il faut utiliser le logiciel RobotinoVIEW qui va nous permettre de tester toutes les entrées-sorties analogiques/numériques de la carte de Robotino.

Voici comment nous procéder :

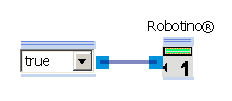
***Récupérer la valeur d’une entrée analogique :***



Pour se faire, la première box (correspondant à l’entrée analogique 1) se trouve dans la bibliothèque à droite de l’écran, rubrique : **Robotino -> Connecteur d’E/S -> Entrée analogique #1**

La deuxième box (correspondant à un oscilloscope virtuel) se trouve dans la rubrique : **Bibliothèque de blocs de fonction -> Affichages -> Oscilloscope**

***Forcer la valeur d’un des sorties TOR :***



La première box correspond à une constante, se situant dans la rubrique : **Bibliothèque de blocs de fonction -> Générateur -> Constante** (Par défaut la constante est un nombre (float) mais en la reliant à la box elle devient un booléen).

La deuxième box correspond à la sortie TOR 1, se situant dans la rubrique : **Robotino -> Connecteur d’E/S -> Sortie TOR #1**

Le fichier de démo « *Test E-S carte* » réalisant le test de la sortie digitale 1 et l’entrée analogique 1 est situé dans le répertoire APP2018/Arnaud/ ExempleLectureCapteurAnalogique sur les robotino.

1. **Interfaçage C en Java**

La solution pour pouvoir utiliser les capteurs de Robotino est d’utiliser le langage C. Mais pour éviter de refaire tout le travail effectué en Java, la solution consiste à créer une interface C que l’on mettra dans une bibliothèque et qui sera appelé dans le programme Java. C’est donc cela que je vais expliquer. Je me suis aider du tutoriel sur Développons en Java :

(<https://www.jmdoudoux.fr/java/dej/chap-jni.htm>)

Dans un premier temps, il faut déclarer la méthode native. C’est assez simple, il suffit, dans le code source Java, de déclarer la signature de la méthode avec la définition de la bibliothèque native.

Ce code est disponible sur Robotino dans le dossier : APP2018/Arnaud/ ExempleLectureCapteurAnalogique/CapteurInduction.java

Voici le programme :

**public** **class** CapteurInduction {

//Déclaration de la méthode native en JAVA

**public** **native** **float** valeurCapteurInduction(**int** numPin);

**static** {

//Chargement des bibliothèques nécessaires au programme

System.loadLibrary("rec\_robotino\_com\_c");

System.loadLibrary("CapteurInduction");

}

**public** **static** **void** main(String[] args) {

**float** valeurCptInduction;

CapteurInduction classeValeurCptInduction = **new** CapteurInduction();

//Test de la lecture du capteur du pin AIN1 et affichage

valeurCptInduction =

classeValeurCptInduction.valeurCapteurInduction(1);

System.out.println("1: " + valeurCptInduction);

}

}

Puis nous déposons ce fichier sur Robotino car il va nous falloir générer le fichier d’en-tête. Cette génération ce fait avec l’outil javah. Il est possible de faire cette génération sur le PC à condition d’utiliser la même version Java que Robotino.

La version Java sur Robotino peut s’obtenir avec la commande :

java -version

On peut voir que la version est la 1.6.0.0

On compile donc le fichier puis nous générons le code du fichier d’en-tête avec les commandes :

javac -cp .:/usr/local/OpenRobotinoAPI/1/lib/java/rec\_robotino\_com\_wrap.jar *nomClasse.java*

javah -jni *nomClasse*

Le fichier d’en-tête (ici : CapteurInduction.h) généré est le suivant :

/\* DO NOT EDIT THIS FILE - it is machine generated \*/

#include <jni.h>

/\* Header for class CapteurInduction \*/

#ifndef \_Included\_CapteurInduction

#define \_Included\_CapteurInduction

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

\* Class: CapteurInduction

\* Method: valeurCapteurInduction

\* Signature: (I)F

\*/

JNIEXPORT jfloat JNICALL Java\_CapteurInduction\_valeurCapteurInduction

(JNIEnv \*, jobject, jint);

#ifdef \_\_cplusplus

}

#endif

#endif

Maintenant, il faut réaliser l’écriture du code natif en C. La fonction est la suivante :

Il est placé dans le dossier APP2018/Arnaud/ ExempleLectureCapteurAnalogique/CapteurInduction.h

Puis il faut écrire notre fonction en C, voici le programme

// Copyright (C) 2004-2008, Robotics Equipment Corporation GmbH

#include <jni.h> //Provient du jdk

#include "CapteurInduction.h" //Fichier d'entete genere

//Librairies nécessaires au programme

#include <stdio.h>

#include <stdlib.h>

#include "rec/robotino/com/c/Com.h"

#ifdef WIN32

#include <windows.h>

// \_getch

#include <conio.h>

#else

// getchar

#include <stdio.h>

// usleep

#include <unistd.h>

#endif

//Type Booleen

typedef int bool;

#define true 1

#define false 0

/\*FONCTION EN C permettant la lecture du capteur\*/

float valeurCapteurInduction\_c(int iNumPin){

//Déclaration et initialisation des variables

ComId com;

AnalogInputId pinCapteur;

int i;

bool connectSetCom = false;

bool connectPin = false;

bool tempsInit = true;

float fvalue = -1.0;

int tailleFenetreValue = 20;

int nbValueMoyenne = 10;

float value[tailleFenetreValue];

//Connection à Robotino (en local)

com = Com\_construct();

Com\_setAddress(com,"127.0.0.1");

if( FALSE == Com\_connect(com) )

{

fprintf(stderr, "Error on connect\n");

return -1.0;

}

else

{

char addressBuffer[256];

Com\_address( com, addressBuffer, 256 );

printf( "Connected to %s\n", addressBuffer );

}

//Construction de l'objet analogique : numero du pin allant de 0 à 7 (AIN1 -> AIN8)

pinCapteur = AnalogInput\_construct(iNumPin-1);

//On associe notre objet analogique à notre interface de communication

connectSetCom = AnalogInput\_setComId(pinCapteur, com);

if(connectSetCom == false){

fprintf(stderr, "Error on connectSetComId pin\n");

return -1.0;

}

//On associe le numéro de pin à notre objet

connectPin = AnalogInput\_setInputNumber(pinCapteur, iNumPin-1);

//Initialisation capteur

while(tempsInit){

if(connectPin == false){

printf( "Erreur Pin Connexion \n" );

break;

}

//Permet d'initialiser notre capteur : on sort de la boucle une fois que l'on lit une valeur > 0

if(AnalogInput\_value(pinCapteur) > 0.001){

tempsInit = false;

}

}

//Récuperation valeur capteur

for(i=0;i<tailleFenetreValue;i++){

value[i] = AnalogInput\_value(pinCapteur);

}

//Moyenne des X dernières valeurs afin d'éliminer des erreurs de mesures

for(i=nbValueMoyenne;i<tailleFenetreValue;i++){

if(fvalue == -1.0){

fvalue = 0;

}

fvalue = fvalue + value[i];

}

//Valeur moyenne du capteur induction

fvalue = fvalue/(tailleFenetreValue-nbValueMoyenne);

fprintf(stdout, "Valeur Induction : %f \n", fvalue);

//On détruit les objets associés à notre interface de communication

AnalogInput\_destroy(pinCapteur);

Com\_destroy(com);

return fvalue;

}

//Fonction qui nous renvoie la valeur du capteur d'induction passé en paramètre d'entrée :

//Permet de "transferer"/"traduire" le code C en Java

//Définition du nom de la fonction : JNIEXPORT typeValeurRetournée JNICALL Java\_nomClasse\_nomMéthodeJava

JNIEXPORT jfloat JNICALL Java\_CapteurInduction\_valeurCapteurInduction

(JNIEnv \*env, jobject obj, jint iNumPin){

//Fonction en C qui va récupérer la valeur du capteur d'induction passé en paramètre d'entrée

return valeurCapteurInduction\_c(iNumPin);

}

Nous plaçons ce fichier sur Robotino dans le dossier APP2018/Arnaud/ ExempleLectureCapteurAnalogique/CapteurInduction.c

Cependant pour pouvoir le compiler, il va nous falloir les librairies incluses dans Robotino. Celle-ci se trouve dans le dossier (revenir à la racine):

/usr/local/OpenRobotinoAPI/1/include/rec/robotino/com

Donc la ligne de commande à exécuter dans le dossier APP2018/Arnaud/ ExempleLectureCapteurAnalogiquepour compiler notre fichier est la suivante :

gcc -c -I/usr/lib/jvm/java-6-openjdk/include/ -I/usr/local/OpenRobotinoAPI/1/include/ -o CapteurInduction.o CapteurInduction.c

Remarque : On utilise la commande gcc (si C) ou g++ (si C++)

Il faut alors créer la librairie pour pouvoir l’utiliser dans notre code source en Java. Cela s’effectue à l’aide de la commande :

gcc -o libCapteurInduction.so -shared CapteurInduction.o -Wl,-rpath, /usr/local/OpenRobotinoAPI/1/lib/linux/librec\_robotino\_com\_c.so

Attention, il faut ensuite exporter ces librairies dans le chemin PATH de robotino. Cela se fait avec la commande dans le dossier où se trouve la librairie :

export LD\_LIB RARY\_PATH=$LD\_LIBRARY\_PATH:.

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/OpenRobotinoAPI/1/lib/linux/

Enfin, il suffit d’exécuter le fichier CapteurInduction.java avec la commande :

java CapteurInduction

Remarque : Un fichier shell a été créé pour exécuter ces lignes de commande, il s’appelle buildAll.sh. Il faut penser à donner la permission au fichier avec la commande :

Chmod u+x buildAll.sh

Sauf qu’il fait penser à exporter les librairies après l’exécution de ce fichier et faire java.

Remarque : Pour compiler et exécuter un fichier C sur Robotino voici les commandes :

Il faut déposer ce fichier dans un dossier sur Filezilla et le compiler à l’aide de la commande :

gcc *nomFichier.c* -I /usr/local/OpenRobotinoAPI/1/include/ -L /usr/local/OpenRobotinoAPI/1/lib/linux/ -lm -lrec\_robotino\_com\_c -o *nomFichierSouhaite*

Dans la ligne de commande précédente : « -I /usr/local/OpenRobotinoAPI/1/include/ -L /usr/local/OpenRobotinoAPI/1/lib/linux/ -lm -lrec\_robotino\_com\_c » cela permet d’appeler la bibliothèque *rec\_robotino\_com* au chemin indiqué.

Pour exécuter le fichier, il suffit de faire la commande :

./*nomFichierSouhaite*

Remarque : Nous sommes obligés de l’exécuter depuis putty car si l’on passe par l’application nous avons une erreur d’exécution 0xc00003B. À ce jour, nous n’avons pas trouvé de solution pour résoudre ce problème.