

ARDUINO+LABVIEW

CONTROLE DE LA TEMPERATURE

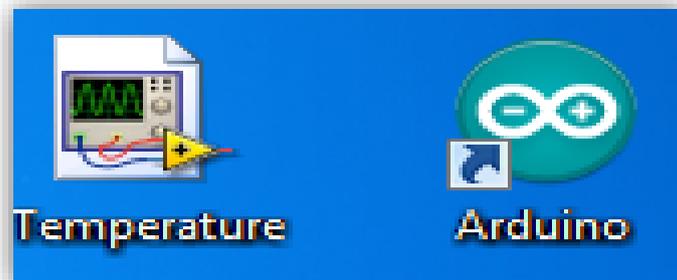


Table des matières

Introduction	3
I. Préparation de l'environnement du Labview+Arduino	3
1. Arduino IDE	4
2. VISA NI.....	5
3. JKI VI	6
II. Montage sous ISIS	8
III. Code du projet sous Arduino	10
IV. Création du projet sous LABVIEW	17
V. Communication entre LABVIEW et ISIS	22
Conclusion	28

Liste des Figures

Figure 1 : Logiciel Arduino(IDE)	5
Figure 2 : Emplacement du VISA sous LabVIEW	6
Figure 3 : Arduino sous JKI VI.....	7
Figure 4 :L'installation de l'interface ARDUINO sous LabVIEW	7
Figure 5 : Arduino sous LabVIEW	8
Figure 6 : Composants utilisés	8
Figure 7 : Schéma du montage sous ISIS.....	10
Figure 8:Dossier LIFA_Base	11
Figure 9 : Fichiers LIFA_Base nécessaires.....	11
Figure 10 : Fichiers du LIFA_Base sous Arduino	12
Figure 11:DS18S20	12
Figure 12 : Code du DS18S20	13
Figure 13 : Suite du code DS18S20.....	14
Figure 14 : Les bibliothèques	15
Figure 15 : Ajout des variables	15
Figure 16 : Case 0x34	16
Figure 17 : Fonction « ObtenirTemperature »	16
Figure 18 : Compilation terminée	17
Figure 19 : ObtenirTemperature (Block Diagram+Front Panel).....	18
Figure 20 : Arduino Resource	19
Figure 21 : Arduino Resource 2.....	19
Figure 22 : Numéro.....	20
Figure 23 : Block Diagram « Température »	21
Figure 24 : Front Panel « Température »	22
Figure 25 : Fichier (.hex)	22
Figure 26 : Chargement du code	23
Figure 27 : VSPE	24
Figure 28: Ports pour ISIS et Arduino(IDE).....	24
Figure 29 : Ports pour ISIS et Arduino(IDE).....	24
Figure 30 : Ports pour ISIS et Arduino(IDE).....	25
Figure 31 : Port pour LabVIEW	26
Figure 32 : Port pour LabVIEW	26
Figure 33 : VSPE(les virtuels ports)	27
Figure 34 : Configuration du COMPIM	27
Figure 35 : VISA (COM3).....	28

Liste des Tableaux

Tableau 1 : Les étapes du projet	3
Tableau 2 : Tableau des composants	10

Introduction

Ce projet consiste à contrôler la température par l'intervention de plusieurs logiciels :

- Arduino.
- LabVIEW.
- ISIS.

Ce rapport comporte 5 étapes :

N°	Étape	Rôle
1	<ul style="list-style-type: none"> ▪ Préparation de l'environnement du Labview+Arduino 	C'est une étape d'initialisation qui nous permet d'accéder à la carte Arduino sous LabVIEW.
2	<ul style="list-style-type: none"> ▪ Montage sous ISIS 	Cette étape définit les composants utilisés et le schéma sous ISIS.
3	<ul style="list-style-type: none"> ▪ Code du projet sous Arduino 	Cette étape permet d'avoir comment coder ARDUINO sous LabVIEW.
4	<ul style="list-style-type: none"> ▪ Création du projet sous LabVIEW 	Block Diagram+Front Panel.
5	<ul style="list-style-type: none"> ▪ Communication entre LabVIEW et ISIS 	Comment réaliser une communication entre ISIS et LabVIEW.

Tableau 1 : Les étapes du projet

I. Préparation de l'environnement du Labview+Arduino

Pour installer Arduino sous LabVIEW, il faut d'abord avoir installé :

- VISA NI.
- Arduino IDE.
- JKI VI.

- Qu'elle est l'utilité de chaque logiciel et comment les utilisés pour avoir ARDUINO sous LabVIEW ?

1 Arduino IDE

Arduino est une plateforme de prototypage open-source basé sur le matériel et un logiciel facile à utiliser. Cartes Arduino sont capables de lire les entrées - la lumière sur un capteur, un doigt sur un bouton, ou un message Twitter - et la transformer en une sortie - l'activation d'un moteur, d'allumer une LED, publier quelque chose en ligne. Vous pouvez dire à votre conseil quoi faire en envoyant un ensemble d'instructions au microcontrôleur sur la carte. Pour ce faire, vous utilisez le langage de programmation Arduino (basé sur le câblage), et le logiciel Arduino (IDE), basée sur le traitement. Ci-dessous vous trouvez la fenêtre du logiciel Arduino (IDE) sous lequel on va taper notre code.

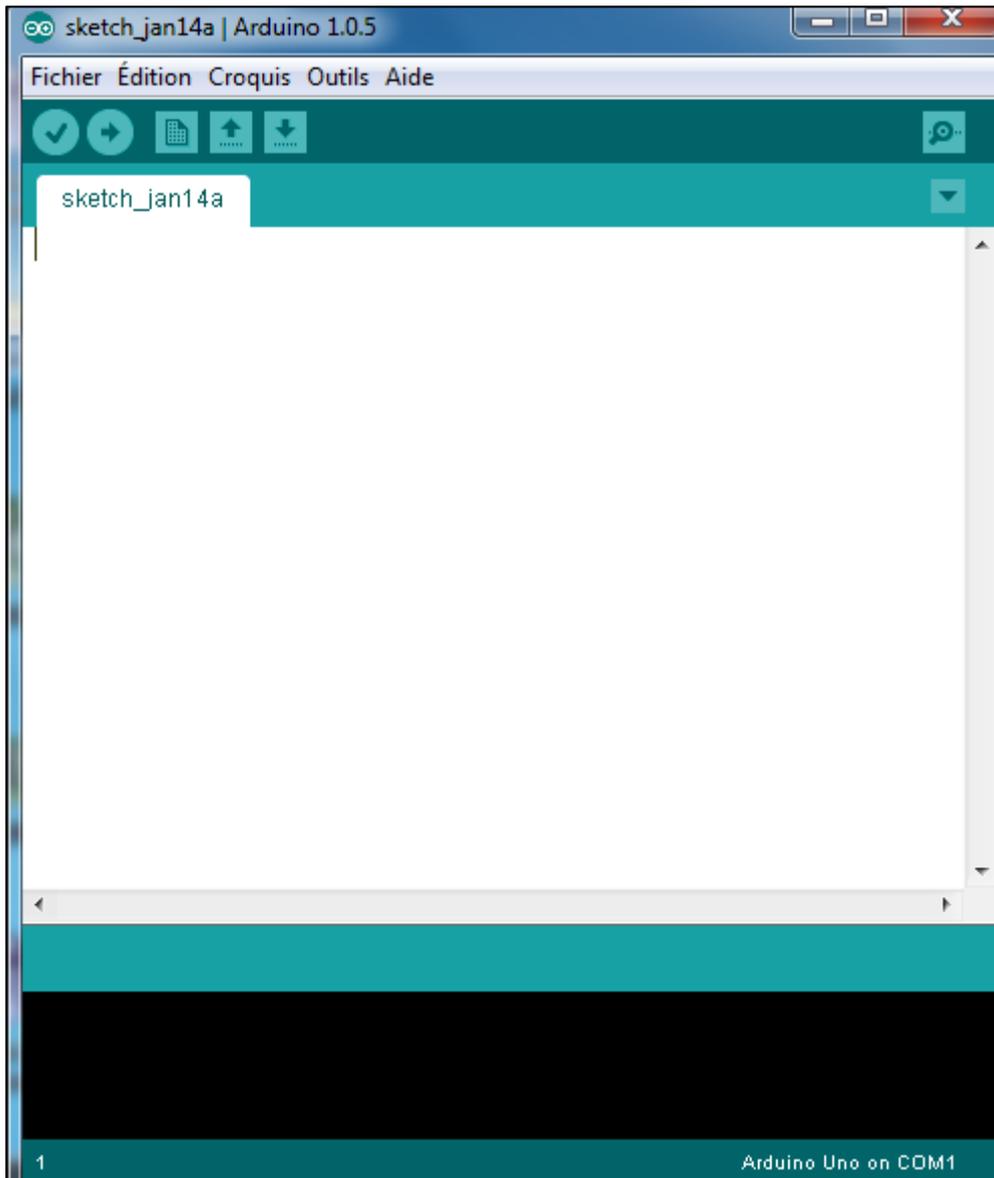


Figure 1 : Logiciel Arduino(IDE)

2. VISA NI

Le Salon Virtuel Instrument Software Architecture (VISA) est un standard pour la configuration, la programmation et le dépannage des systèmes d'instrumentation comprenant des interfaces GPIB, VXI, PXI, série, Ethernet et / ou USB. VISA fournit l'interface de programmation entre les environnements matériels et de développement tels que LabVIEW, LabWindows / CVI et Measurement Studio pour Microsoft Visual Studio. NI-VISA est la mise en œuvre de National Instruments de la norme / S VISA je. NI-VISA comprend des bibliothèques de logiciels, les services publics interactifs tels que NI I / O Trace et le VISA Interactive Control, et des programmes de configuration via Measurement & Automation

Explorer pour tous vos besoins de développement. NI-VISA est standard sur toute la gamme de produits de National Instruments.

IL faut mettre VISA sous le dossier National Instruments du LabVIEW.

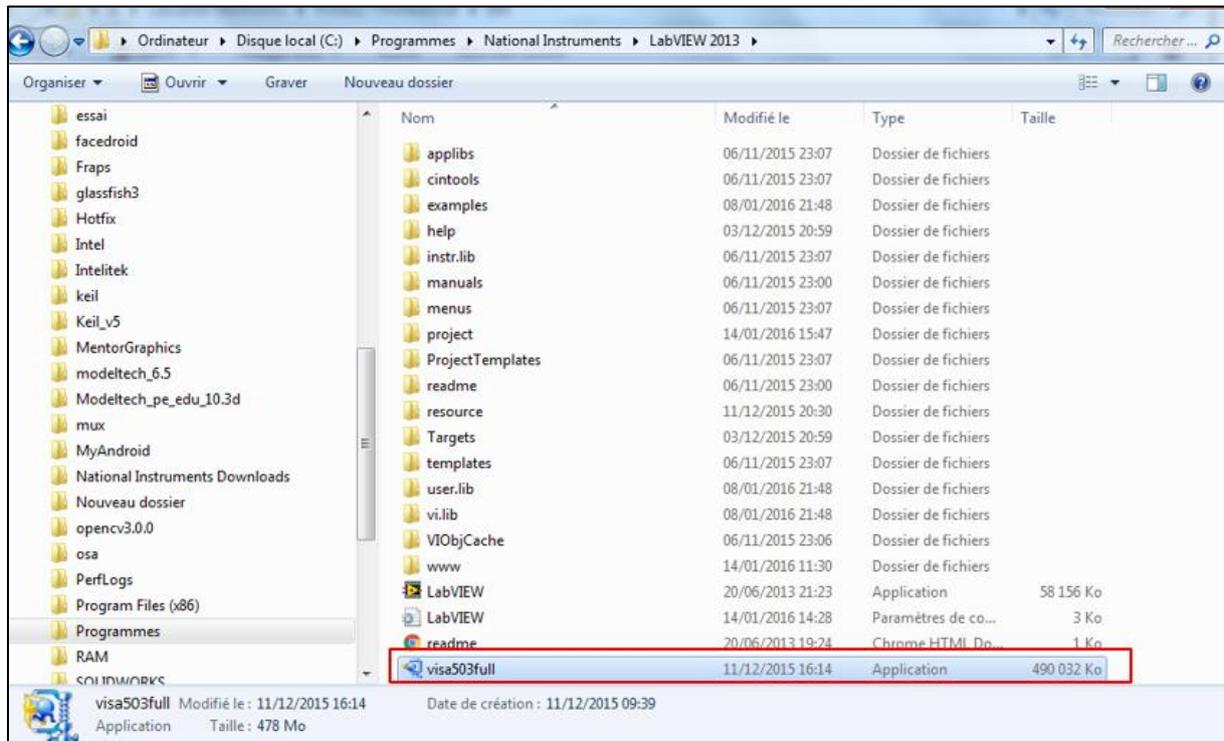


Figure 2 : Emplacement du VISA sous LabVIEW

3. JKI VI

VI Package Manager (de VIPM) est un outil de gestion des paquets qui organise et gère les paquets au sein de votre environnement LabVIEW. Il est l'outil pour l'obtention et de la configuration des bibliothèques et des outils de développement.

Après l'installation du VI, on tape Arduino dans l'icône de la recherche et on obtient la fenêtre suivante.

Voir la figure ci-dessous.

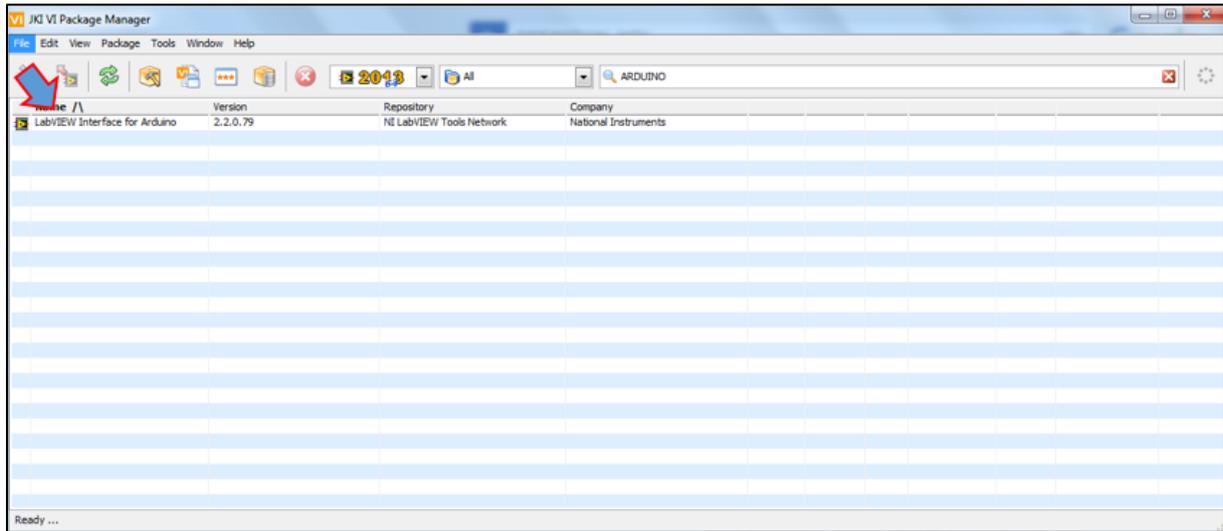


Figure 3 : *Arduino sous JKI VI*

Double cliques sur « LabVIEW Interface for Arduino », puis Install.

A la fin de l'installation, on a pour résultat la fenêtre suivante.

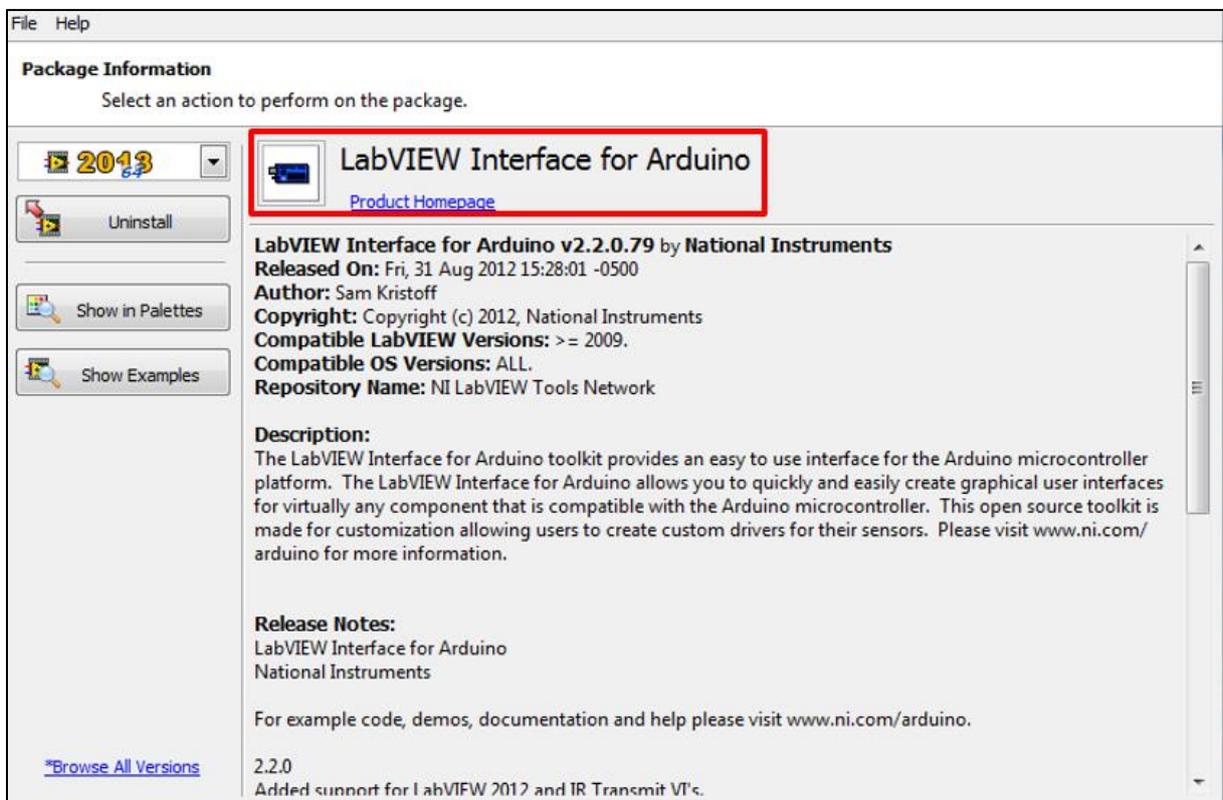


Figure 4 : *L'installation de l'interface ARDUINO sous LabVIEW*

Maintenant quand on lance le logiciel LabVIEW, on vérifie quand la carte Arduino est ajoutée on non.

Clique droit → on trouve les fonctions → on cherche le mot Arduino → Et voilà on trouve cette fenêtre qui contient les composants qui définissent Arduino sous LabVIEW.

Et par suite nous pouvons commencer notre projet qui permet de contrôler la température.

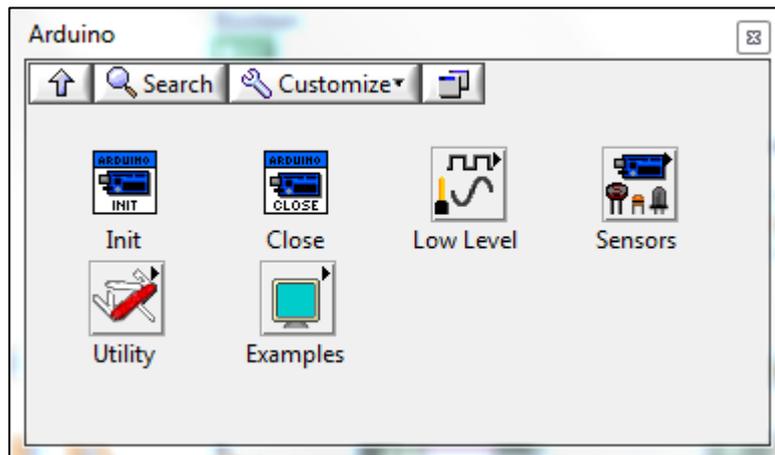


Figure 5 : Arduino sous LabVIEW

II. Montage sous ISIS

Pour contrôler la température on va utiliser les composants suivantes :



Figure 6 : Composants utilisés

Non du Rôle Composant		Schéma sous ISIS
<p>DS18B20</p>	<p>Le thermomètre numérique DS18B20 9 bits fournit à Celsius mesures de température de 12 bits et a une fonction d'alarme avec des points non volatiles programmables par l'utilisateur déclenchement supérieur et inférieur. Le DS18B20 communique sur un bus 1-Wire® que, par définition, ne nécessite qu'une seule ligne de données (et la terre) pour la communication avec un microprocesseur central. Il a une plage de température de fonctionnement de -55 ° C à + 125 ° C et une précision de ± 0,5 ° C sur la plage de -10 ° C à + 85 ° C. En outre, le DS18B20 peut tirer un pouvoir directement depuis la ligne de données («puissance parasite»), éliminant le besoin d'une alimentation externe.</p>	
<p>RELAY</p>	<p>Un relais est un commutateur à commande électrique. Les relais sont utilisés là où il est nécessaire de commander un circuit par un signal de faible puissance (par isolation électrique complète entre les circuits de commande et contrôle), ou où plusieurs circuits doivent être contrôlés par un seul signal.</p>	
<p>BC547</p>	<p>Le BC547 est un transistor à usage général de jonction bipolaire trouvé couramment dans les équipements électroniques.</p>	

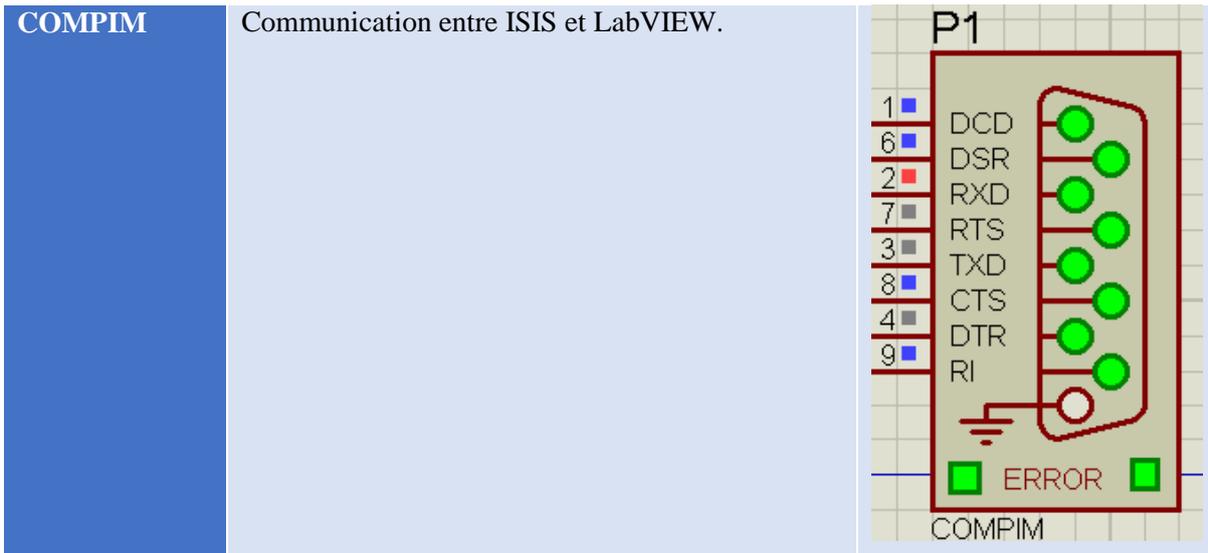


Tableau 2 : Tableau des composants

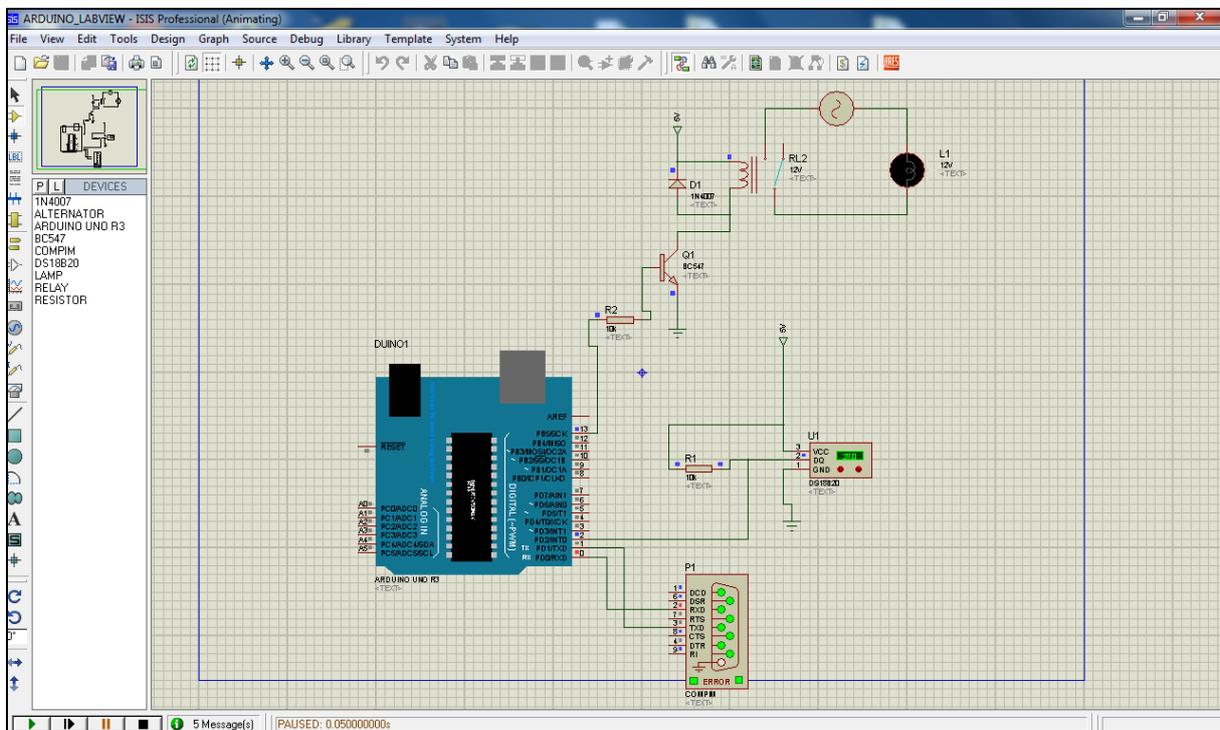


Figure 7 : Schéma du montage sous ISIS

III. Code du projet sous Arduino

Pour coder Arduino sous LabVIEW, il faut accéder au chemin suivant « [C : \Program Files\National Instruments\LabVIEW 2013\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA Base](C:\Program Files\National Instruments\LabVIEW 2013\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA Base) »

On obtient la fenêtre ci-dessous.

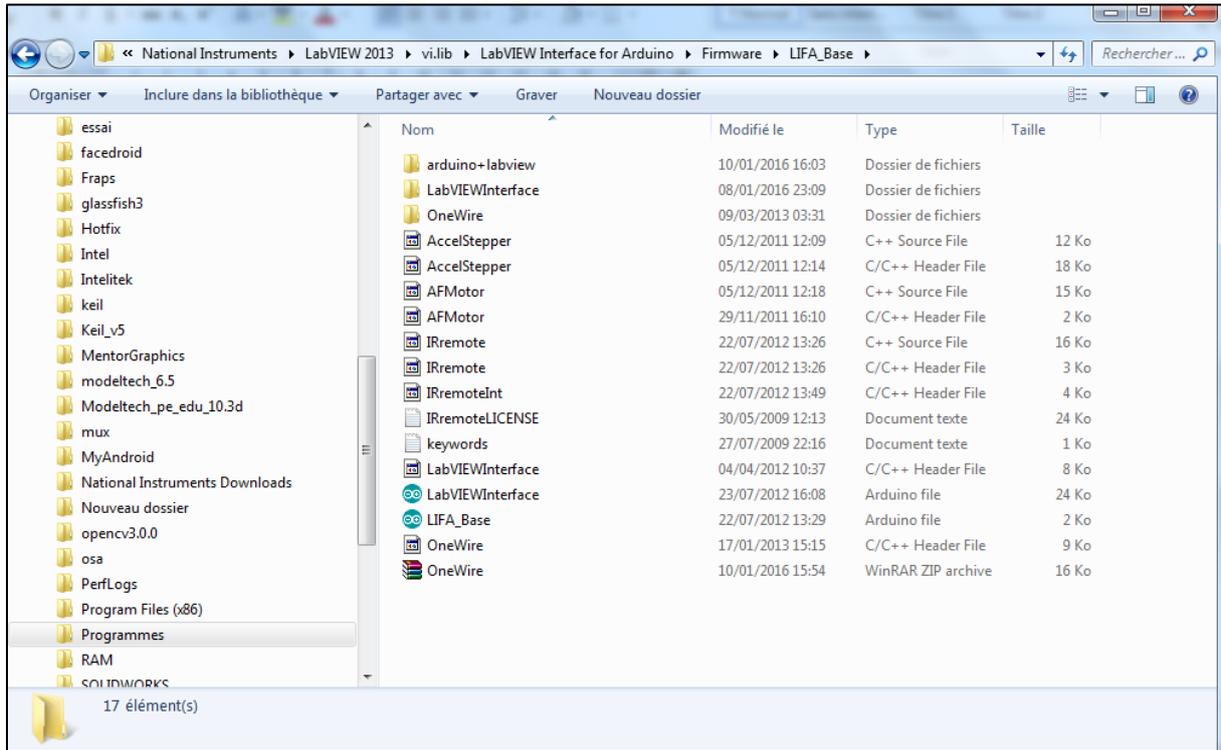


Figure 8: Dossier LIFA Base

On copie les fichiers qui sont indiqués dans la figure qui suit et on les met dans un dossier, puis on lance le logiciel Arduino et on copie ces fichiers dans le sketch ouvert.

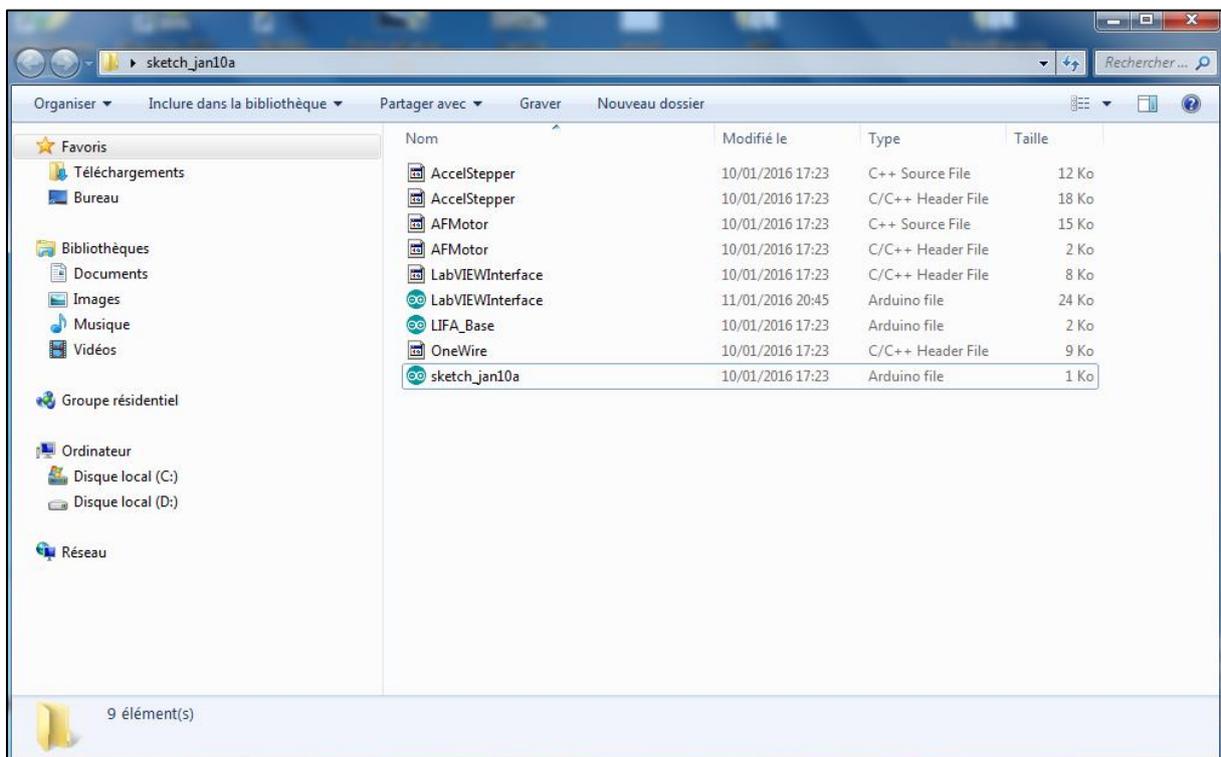


Figure 9 : Fichiers LIFA Base nécessaires

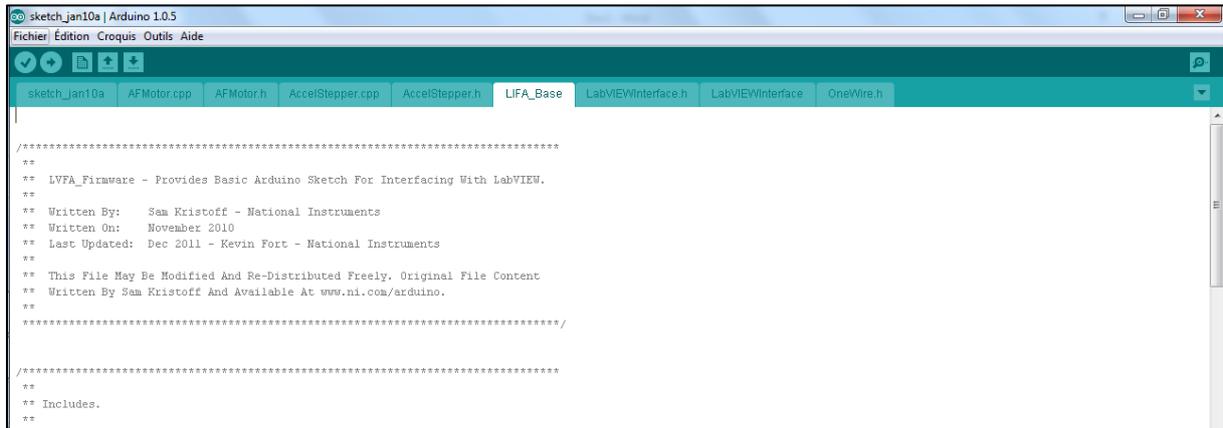


Figure 10 : Fichiers du LIFA Base sous Arduino

Maintenant, on va ajouter le code qui nous permet de contrôler la température.

On va télécharger le code DS18S20.

Voici le code.

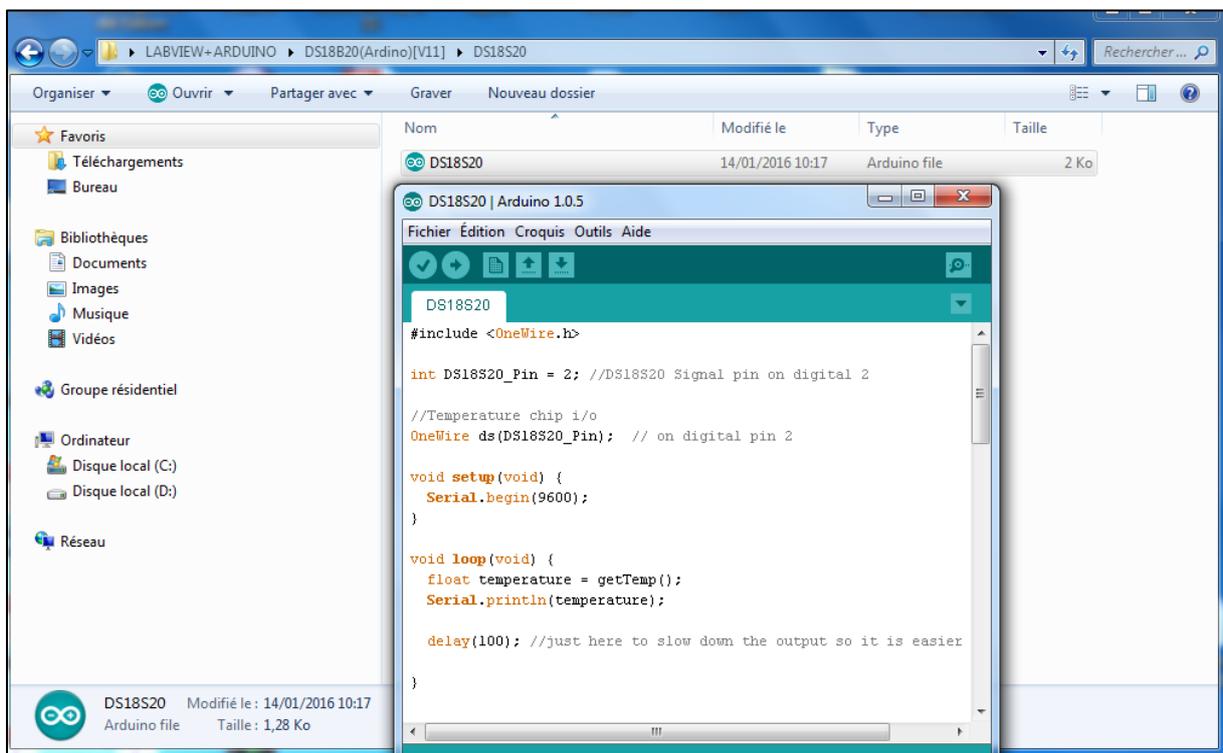


Figure 11:DS18S20

```
DS18S20
#include <OneWire.h>

int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2

//Temperature chip i/o
OneWire ds(DS18S20_Pin); // on digital pin 2

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  float temperature = getTemp();
  Serial.println(temperature);

  delay(100); //just here to slow down the output so it is easier to read
}

float getTemp(){
  //returns the temperature from one DS18S20 in DEG Celsius

  byte data[12];
  byte addr[8];

  if ( !ds.search(addr) ) {
    //no more sensors on chain, reset search
    ds.reset_search();
    return -1000;
  }
}
```

Figure 12 : Code du DS18S20

```
if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.println("CRC is not valid!");
    return -1000;
}

if ( addr[0] != 0x10 && addr[0] != 0x28) {
    Serial.print("Device is not recognized");
    return -1000;
}

ds.reset();
ds.select(addr);
ds.write(0x44,1); // start conversion, with parasite power on at the end

byte present = ds.reset();
ds.select(addr);
ds.write(0xBE); // Read Scratchpad
for (int i = 0; i < 9; i++) { // we need 9 bytes
    data[i] = ds.read();
}

ds.reset_search();

byte MSB = data[1];
byte LSB = data[0];
float tempRead = ((MSB << 8) | LSB); //using two's compliment
float TemperatureSum = tempRead / 16;
return TemperatureSum;
}
```

Figure 13 : Suite du code DS18S20

On revient maintenant sketch qui contient les fichiers LIFA_Base, on s'intéresse au fichier « LabVIEW Interface »

On va suivre les étapes présentées par les figures suivantes.

- On ajoute #include<OneWire.h>

```

Fichier Edition Croquis Outils Aide
sketch_jan10a AFMotor.cpp AFMotor.h AccelStepper.cpp AccelStepper.h LIFA_Base LabVIEWInterface.h LabVIEWInterface OneWire.h
/*****
**
** LVIFA_Firmware - Provides Functions For Interfacing With The Arduino Uno
**
** Written By: Sam Kristoff - National Instruments
** Written On: November 2010
** Last Updated: Dec 2011 - Kevin Fort - National Instruments
**
** This File May Be Modified And Re-Distributed Freely. Original File Content
** Written By Sam Kristoff And Available At www.ni.com/arduino.
**
*****/

#include <Wire.h>
#include <SPI.h>
#include <LiquidCrystal.h>
#include <OneWire.h>

/*****
** Optionally Include And Configure Stepper Support
*****/
#ifdef STEPPER_SUPPORT

// Stepper Modifications
#include "AFMotor.h"
#include "AccelStepper.h"
    
```

Figure 14 : Les librairies

- On ajoute les variables utilisées dans le code DS18S20.

```

Fichier Edition Croquis Outils Aide
sketch_jan10a AFMotor.cpp AFMotor.h AccelStepper.cpp AccelStepper.h LIFA_Base LabVIEWInterface.h LabVIEWInterface OneWire.h
}

AccelStepper steppers[8]; //Create array of 8 stepper objects

#endif

// Variables
int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2

//Temperature chip i/o
OneWire ds(DS18S20_Pin); // on digital pin 2

unsigned int retVal;
int sevenSegmentPins[8];
int currentMode;
unsigned int freq;
unsigned long duration;
int i2cReadTimeouts = 0;
char spiBytesToSend = 0;
char spiBytesSent = 0;
char spiCSPin = 0;
char spiWordSize = 0;
Servo *servos;
byte customChar[8];
LiquidCrystal lcd(0,0,0,0,0,0,0);

// Sets the mode of the Arduino (Reserved For Future Use)
void setMode(int mode)
{
    currentMode = mode;
    
```

Figure 15 : Ajout des variables

Puis on ajoute une nouvelle case qui fait appel à la fonction qui donne la température.

```

    }
    Serial.write('0');
    break;
case 0x31: // Stepper Write
  AccelStepper_Write(command);
  Serial.write('0');
  break;
case 0x32: // Stepper Detach
  steppers[command[2]].disableOutputs();
  Serial.write('0');
  break;
case 0x33: // Stepper steps to go
  retVal = 0;
  for(int i=0; i<8; i++){
    retVal += steppers[i].distanceToGo();
  }
  Serial.write( (retVal & 0xFF) );
  Serial.write( (retVal >> 8) );

  break;
case 0x34: //Temperature
  Serial.print(ObtenirTemperature());
  break;
#endif
/*****
** Unknown Packet
*****/
default: // Default Case
  Serial.flush();
  break;

```

Figure 16 : Case 0x34

- Maintenant, on ajoute la fonction « ObtenirTemperature » qui constitue le reste du code DS18S20.

```

// Checksum Failed, Flush Serial Buffer
Serial.flush();
}
}

/*****
** Functions
*****/
/*fonction Temperature*/
float ObtenirTemperature() {
  //returns the temperature from one DS18S20 in DEG Celsius

  byte data[12];
  byte addr[8];

  if ( !ds.search(addr) ) {
    //no more sensors on chain, reset search
    ds.reset_search();
    return -1000;
  }

  if ( OneWire::crc8( addr, 7) != addr[7] ) {
    Serial.println("CRC is not valid!");
    return -1000;
  }

  if ( addr[0] != 0x10 && addr[0] != 0x28 ) {
    Serial.print("Device is not recognized");
    return -1000;
  }
}

```

Figure 17 : Fonction « ObtenirTemperature »

- Maintenant on peut compiler le programme et corriger les erreurs.

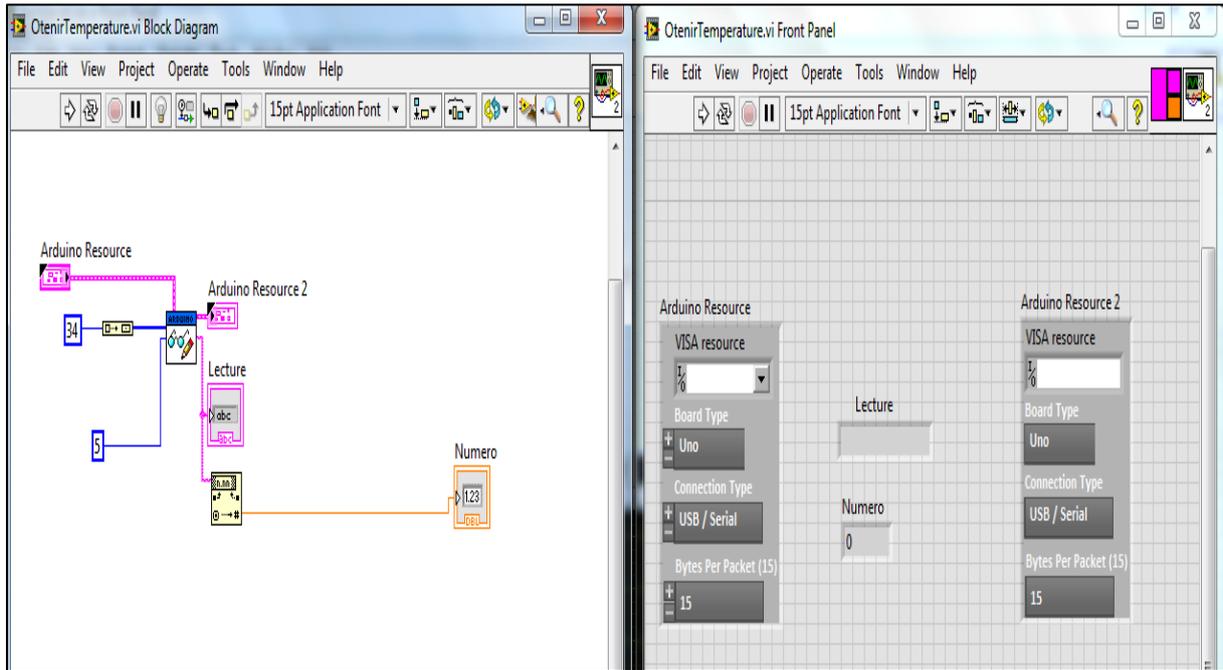


Figure 19 : ObtenirTemperature (Block Diagram+Front Panel)

Pour ce bloc on va créer 3 entrées pour qu'on puisse l'utiliser par suite comme de fonction sous le projet global.

Voici comment se manipuler ça par figure.

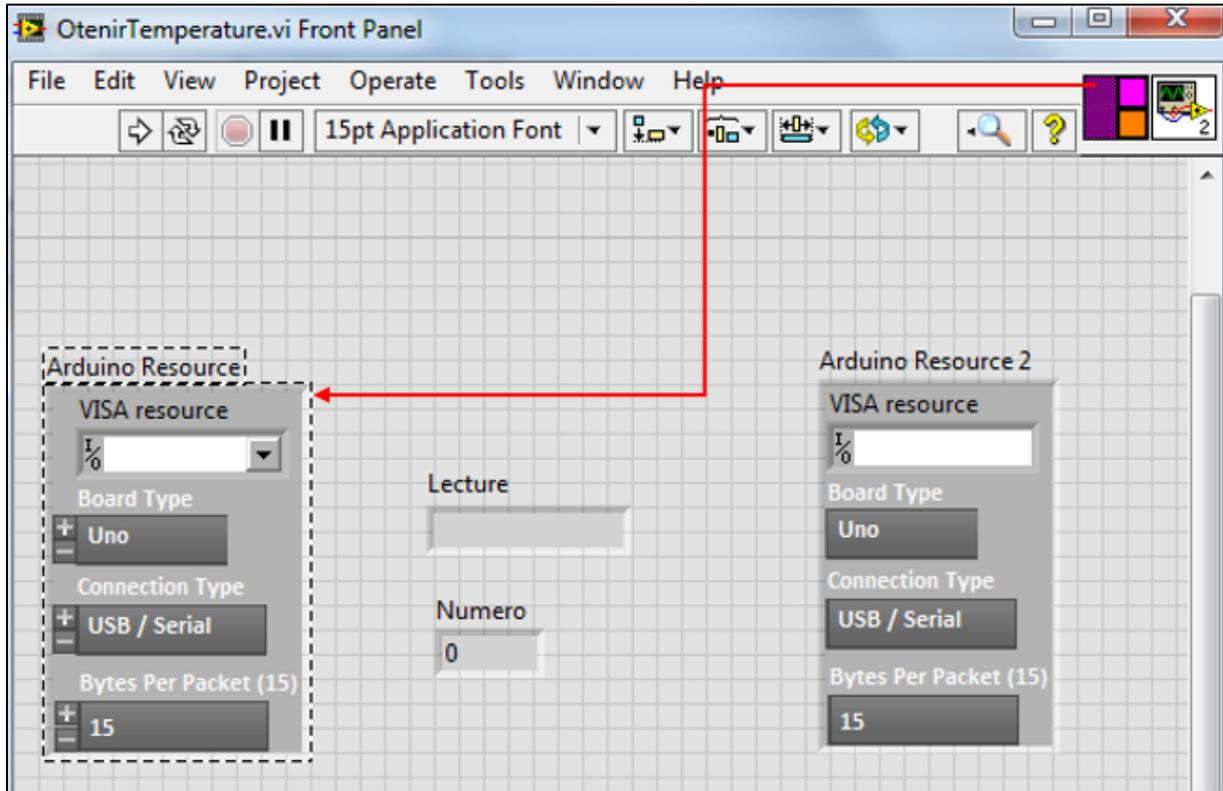


Figure 20 : Arduino Resource

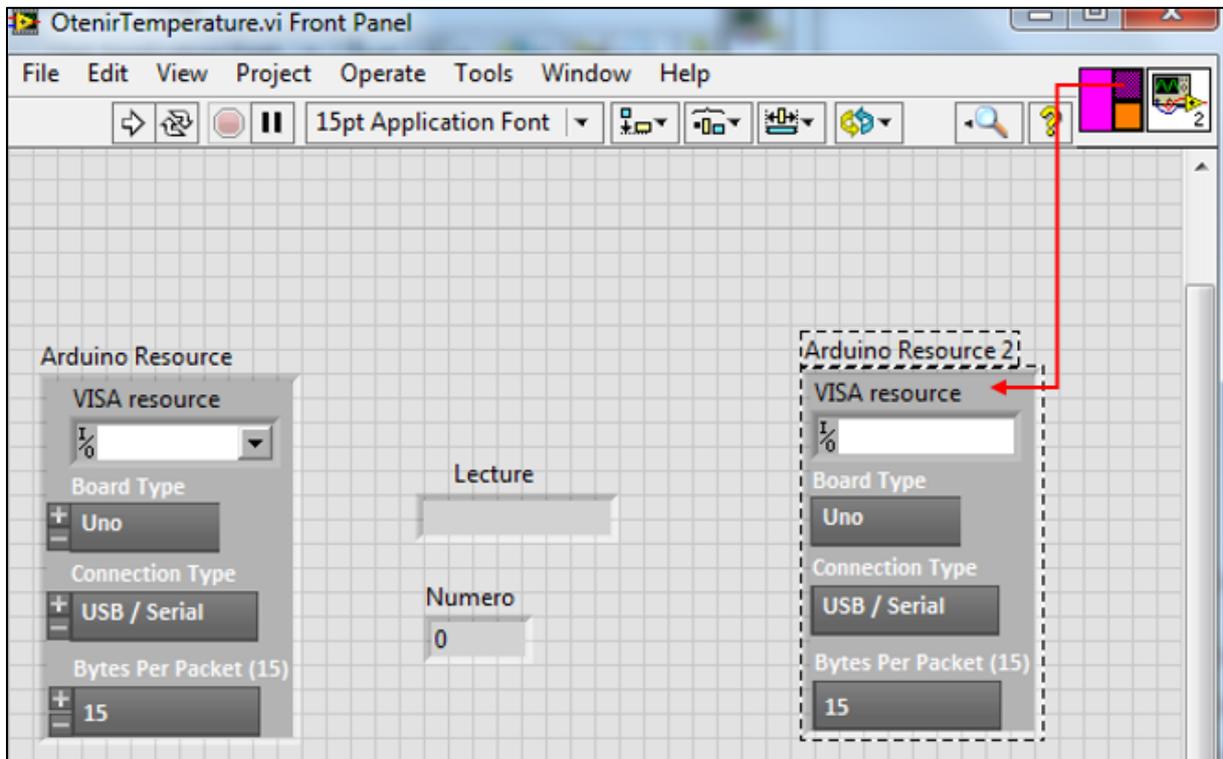


Figure 21 : Arduino Resource 2

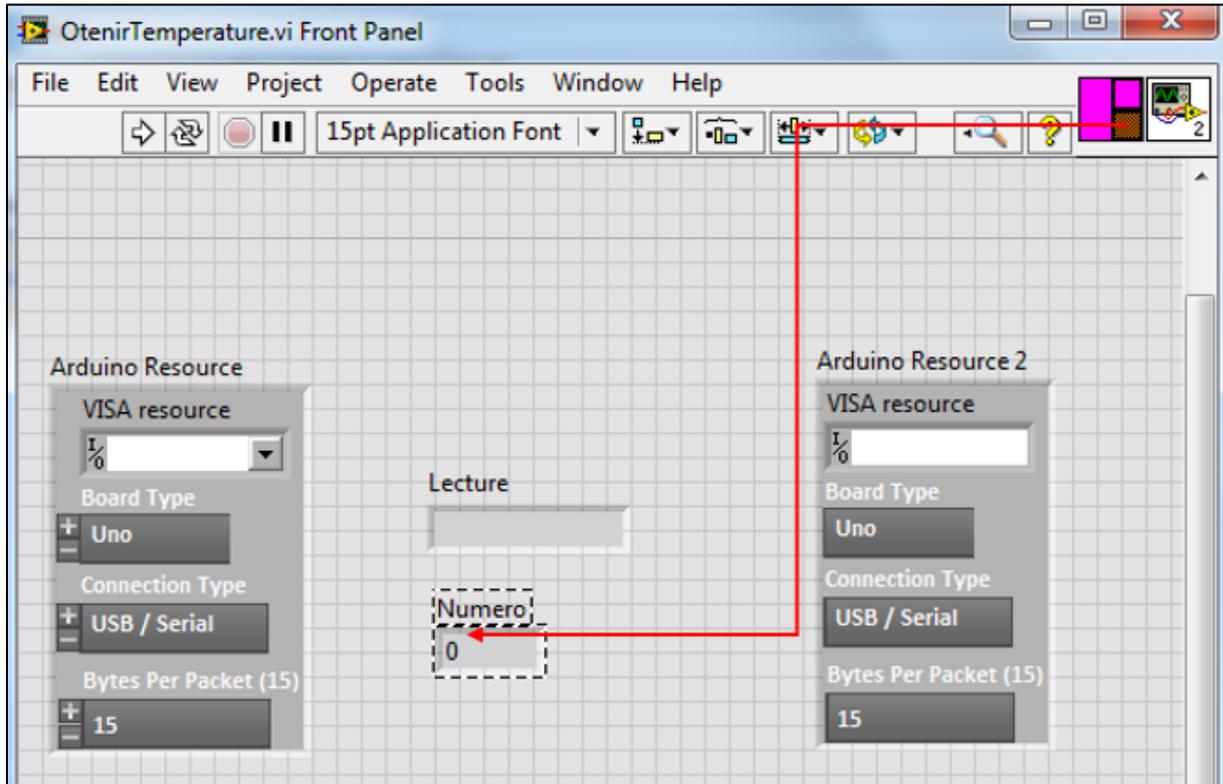


Figure 22 : Numéro

Voici par la suite le projet global sous LabVIEW.



Pour éliminer les erreurs il est utile de vérifier que les fonctions de l'arduino sont bien liées en entrée et en sortie.

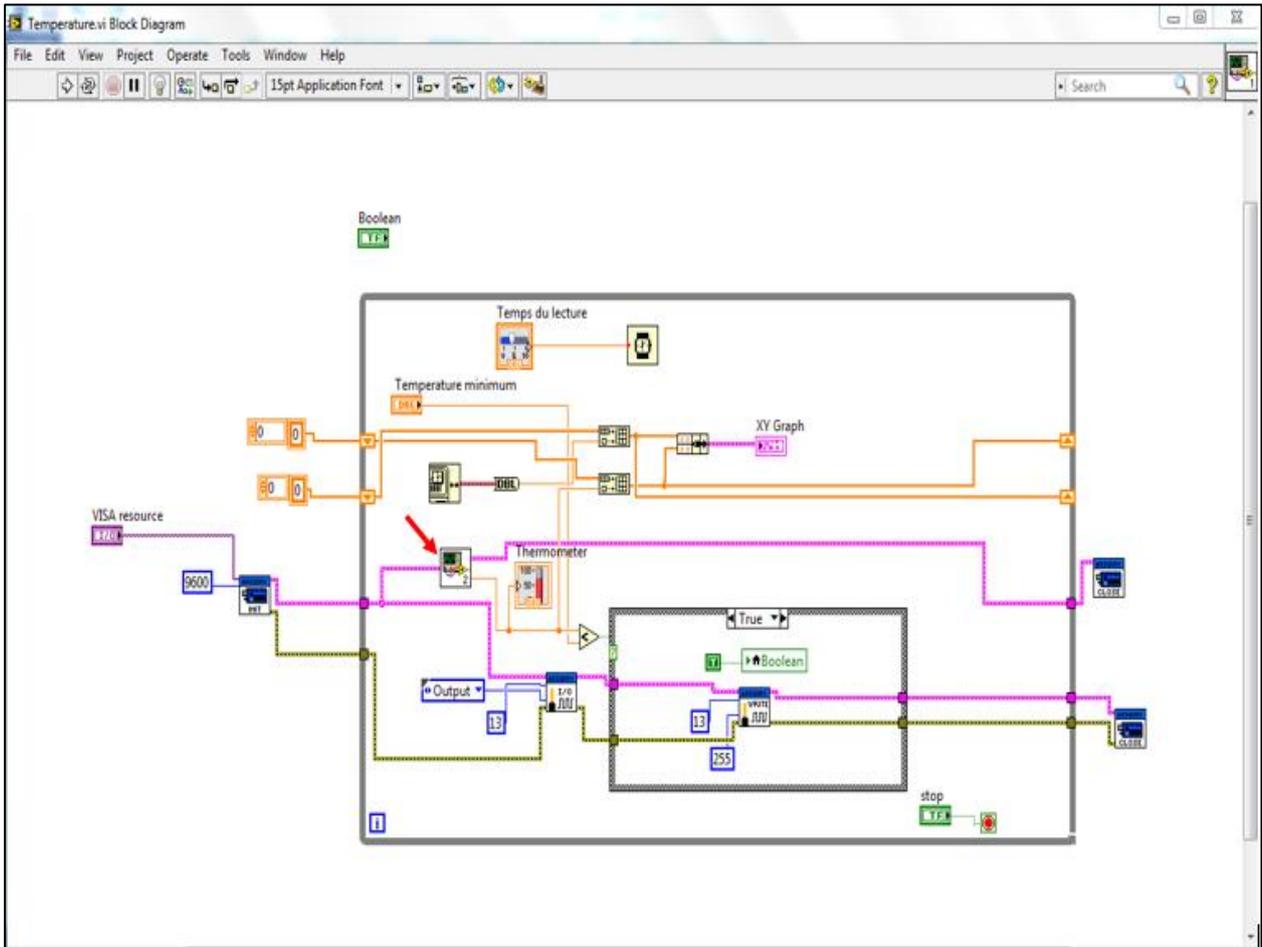


Figure 23 : Block Diagram « Température »

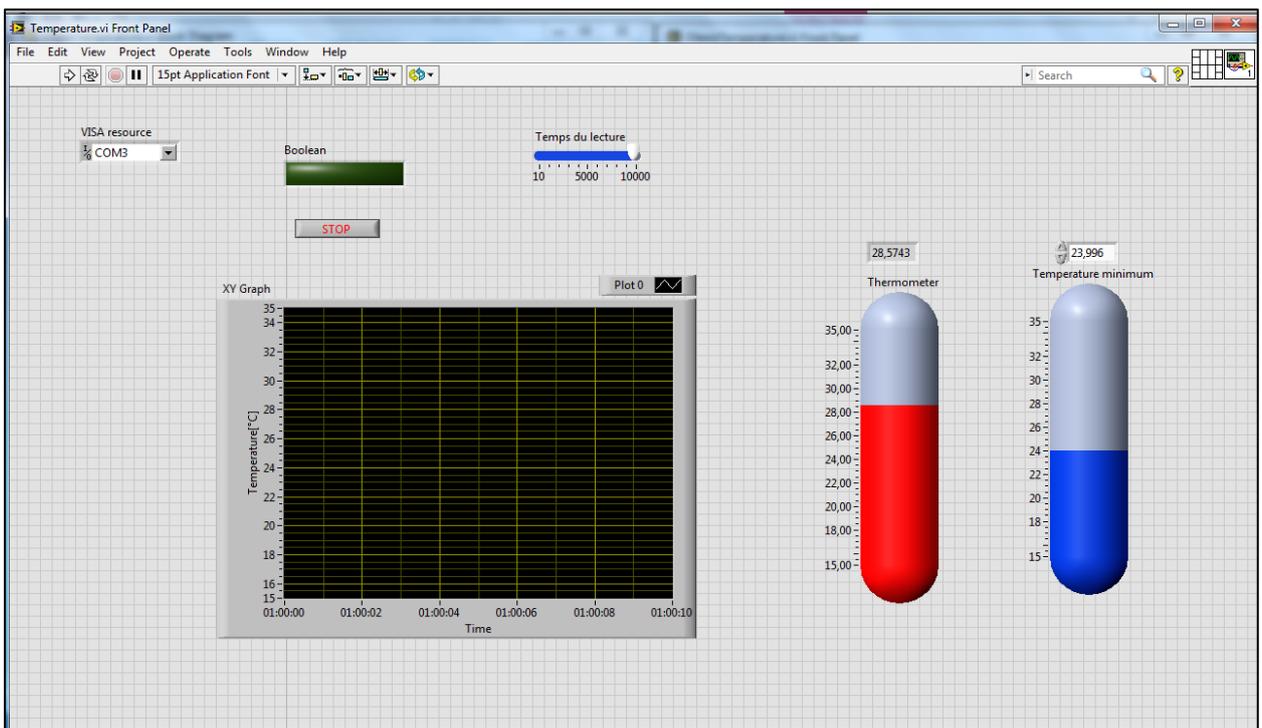


Figure 24 : Front Panel « Température »

V. Communication entre LABVIEW et ISIS

Enfin c'est l'étape de simulation, on va d'abord charger le code sous la carte Arduino.

Après la compilation du code, un fichier (.hex) se produit, on copie l'emplacement du fichier (comme c'est montré par figure) et on le met dans le programme file de la carte Arduino sous ISIS.

```

sketch_jan10a | Arduino 1.0.5
Fichier Edition Croquis Outils Aide

sketch_jan10a AFMotor.cpp AFMotor.h AccelStepper.cpp AccelStepper.h LFA_Base LabVIEWInterface.h LabVIEWInterface OneWire.h

//*****
// LVIFA_Firmware - Provides Functions For Interfacing With The Arduino Uno
//
// Written By: Sam Kristoff - National Instruments
// Written On: November 2010
// Last Updated: Dec 2011 - Kevin Fort - National Instruments
//
// This File May Be Modified And Re-Distributed Freely. Original File Content
// Written By Sam Kristoff And Available At www.ni.com/arduino.
//
//*****

#include <Wire.h>
#include <SPI.h>
#include <LiquidCrystal.h>
#include <OneWire.h>

//*****
// Optionally Include And Configure Stepper Support
//*****
#ifdef STEPPER_SUPPORT

// Stepper Modifications
#include "AFMotor.h"
#include "AccelStepper.h"

Compilation terminée.
C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avr-objcopy -O ihex -R .eeprom C:\Users\hp\AppData\Local\Temp\build5201294613662814551.tmp\sketch_jan10a.cpp.elf
C:\Users\hp\AppData\Local\Temp\build5201294613662814551.tmp\sketch_jan10a.cpp.hex
taille binaire du croquis : 22 490 octets (d'un max de 32 256 octets)

70 Arduino Uno on COM1
    
```

Figure 25 : Fichier (.hex)

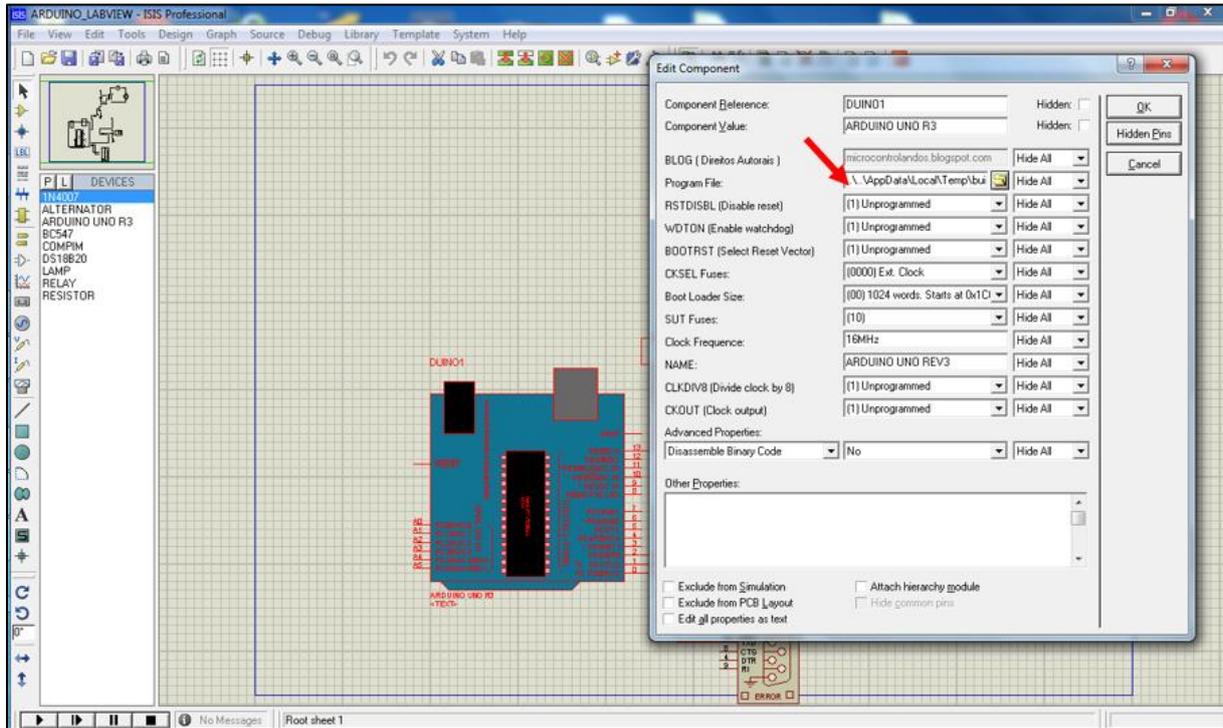


Figure 26 : Chargement du code

Pour faire la communication entre LabVIEW et ISIS, on a utilisé le logiciel « VSPE » (Virtual Serial Ports Emulator). Comme le montre son nom il permet d'avoir une communication virtuelle.

- Cliquez sous l'icône suivante (dans la figure).

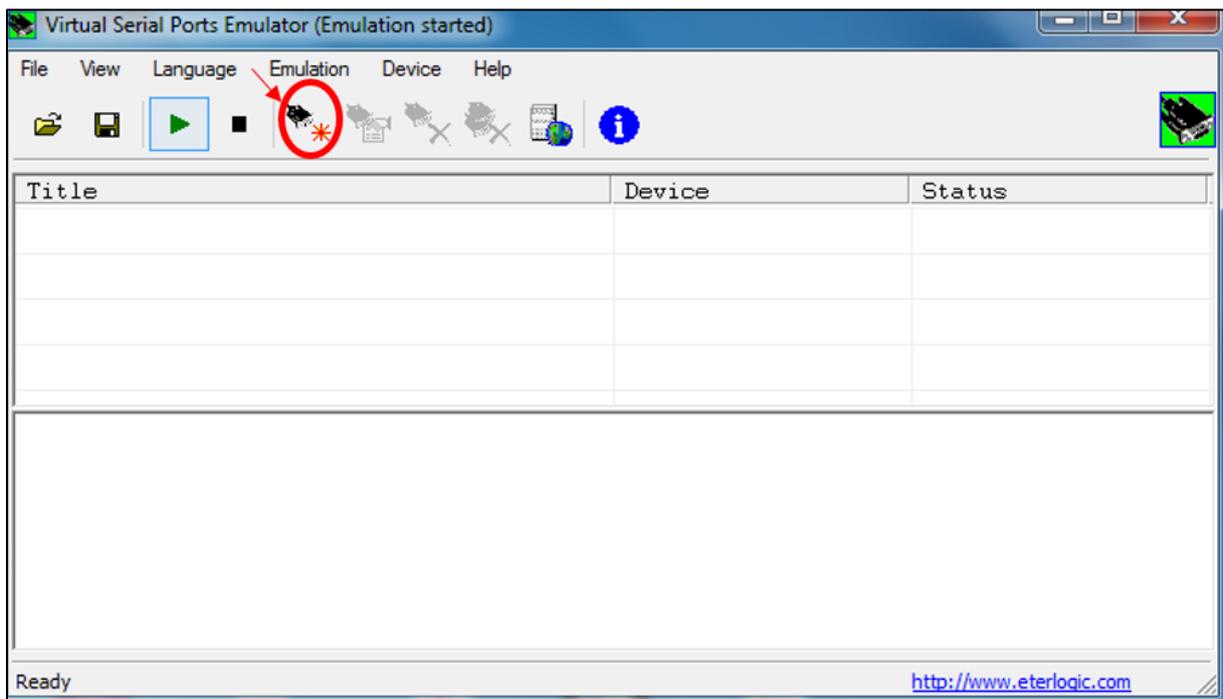


Figure 27 : VSPE

- Choisir Pair.
- Suivant.

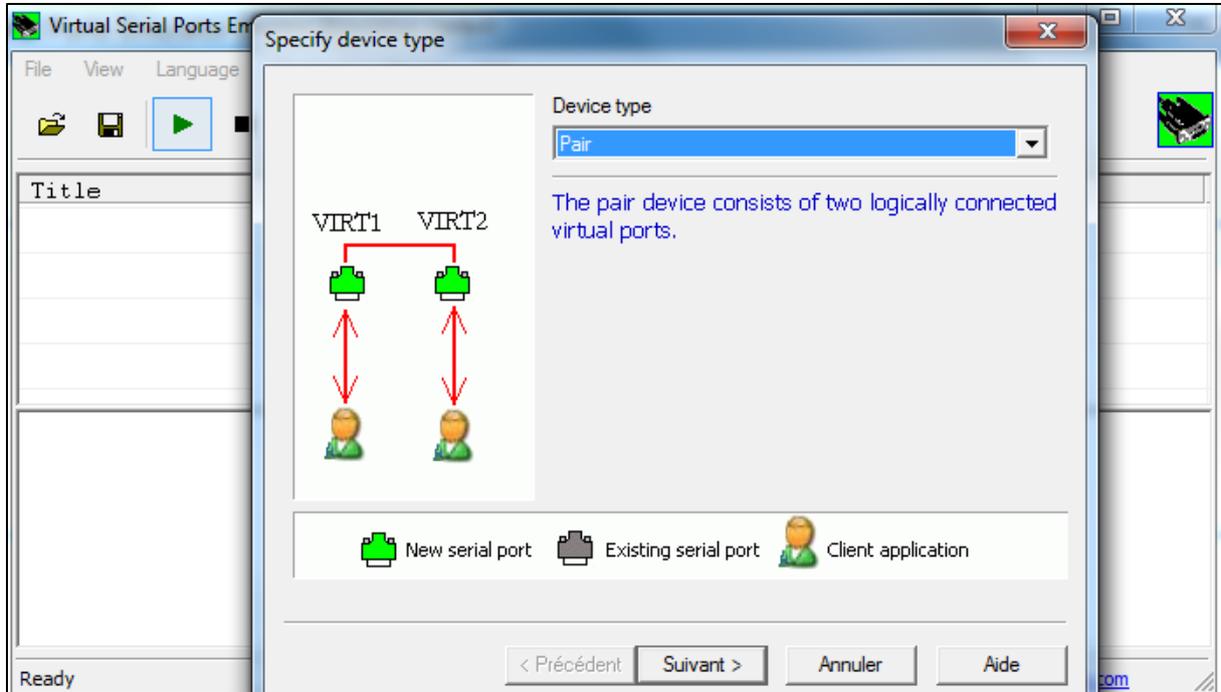


Figure 28: Ports pour ISIS et Arduino(IDE)

- Terminer.

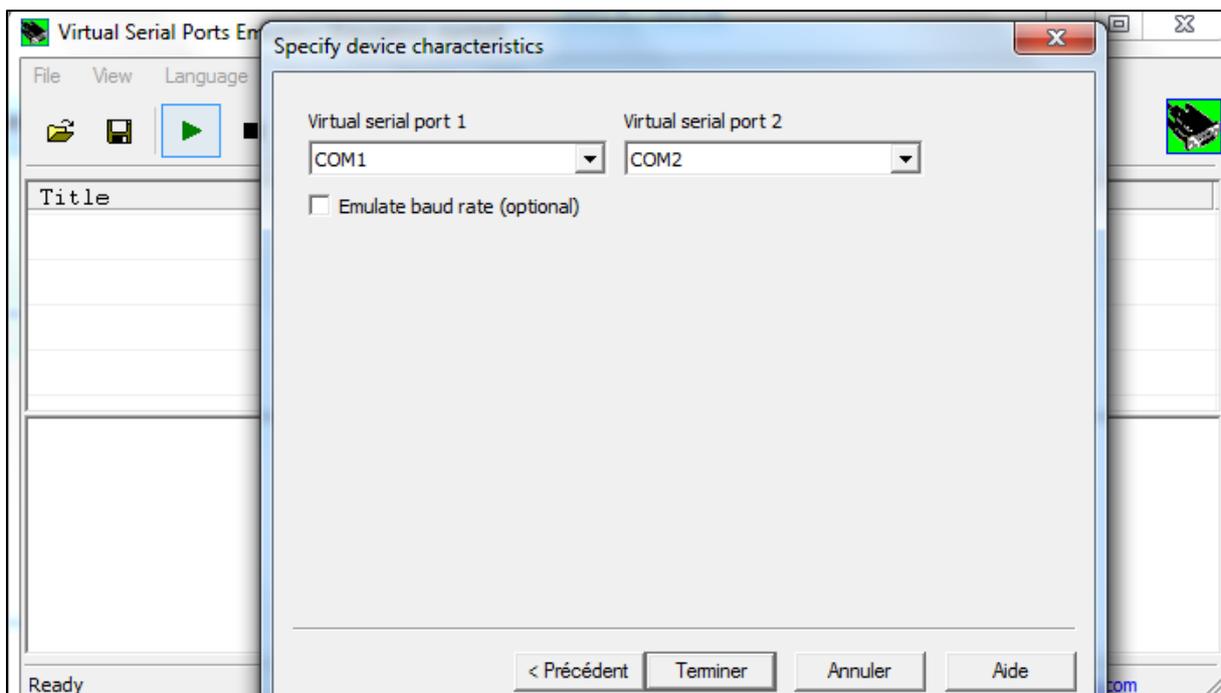


Figure 29 : Ports pour ISIS et Arduino(IDE)

COM1→Arduino(IDE).

COM2→ISIS.

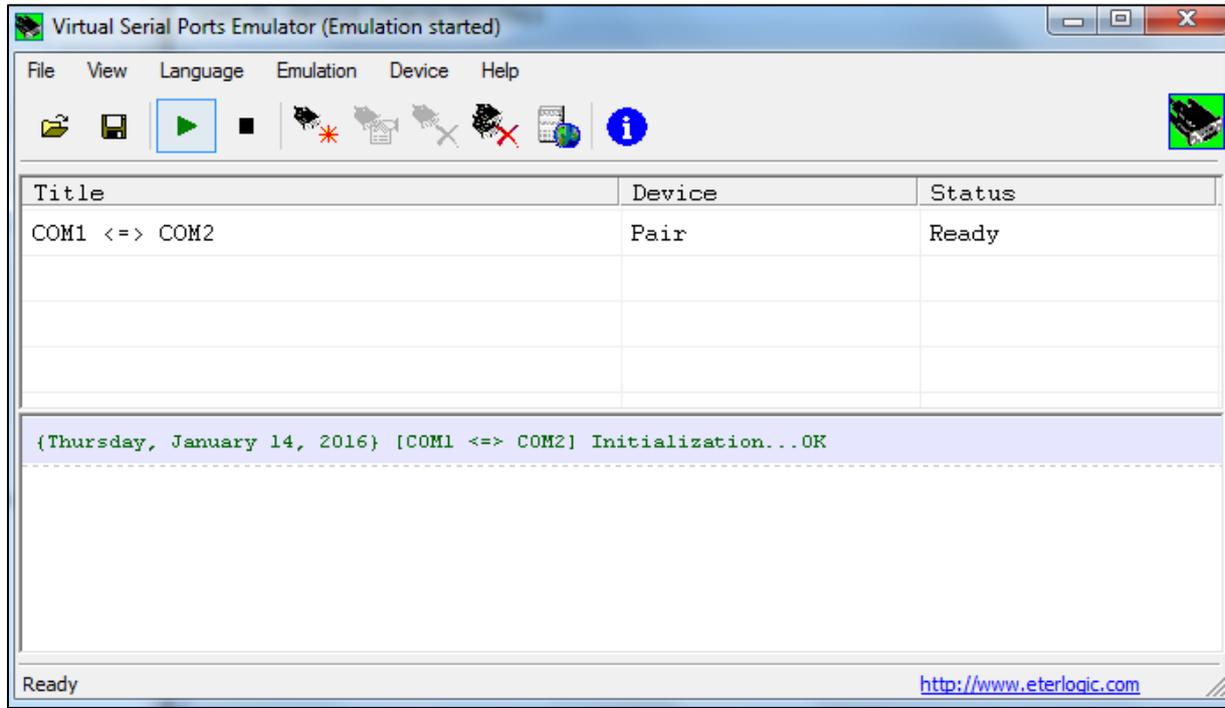


Figure 30 : Ports pour ISIS et Arduino(IDE)

- Connecter.
- Suivant.

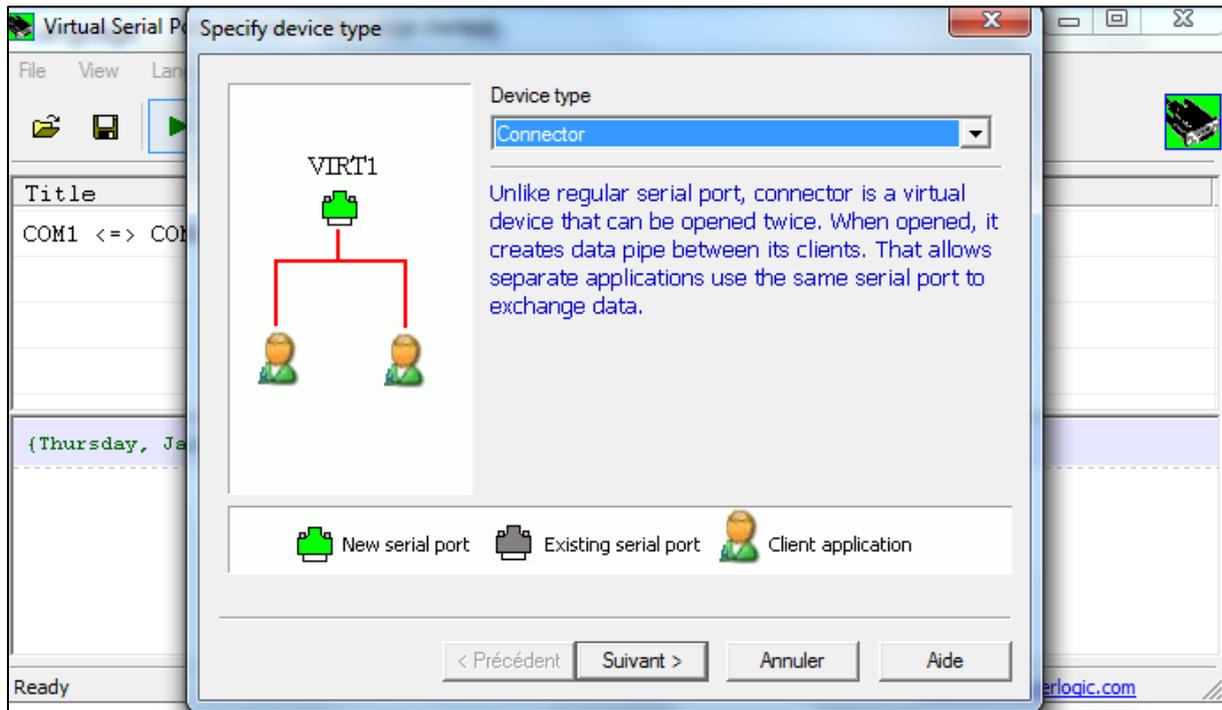


Figure 31 : Port pour LabVIEW

- COM3.
- Terminer.

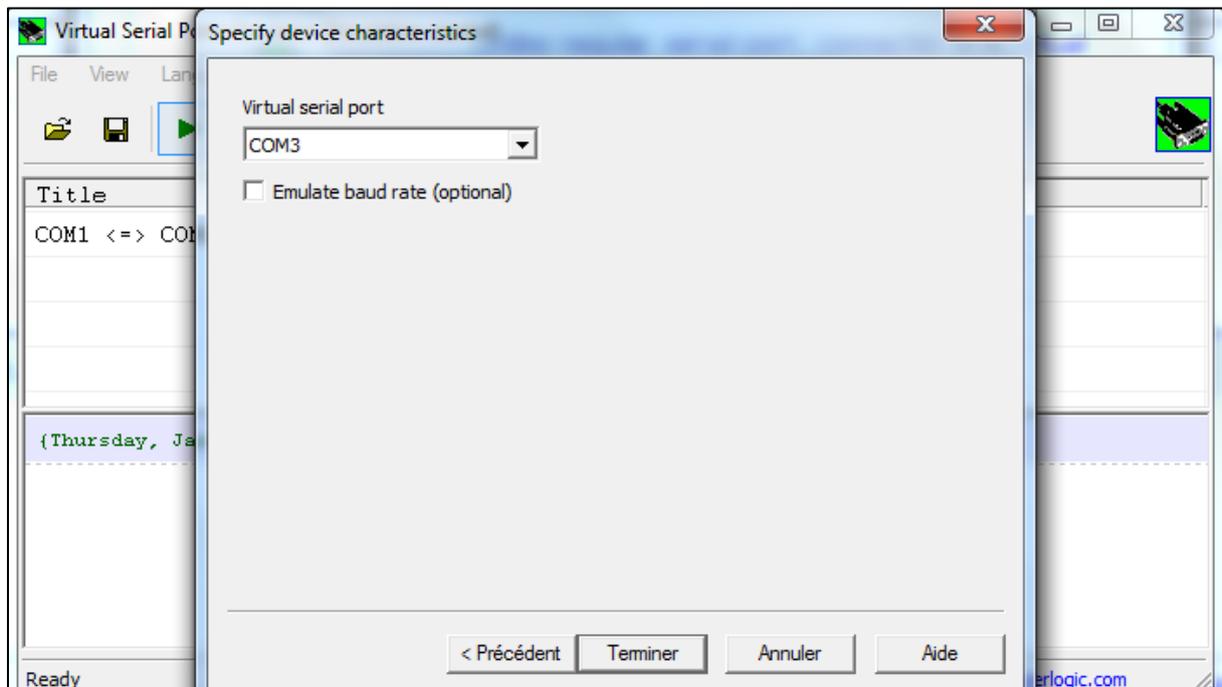


Figure 32 : Port pour LabVIEW

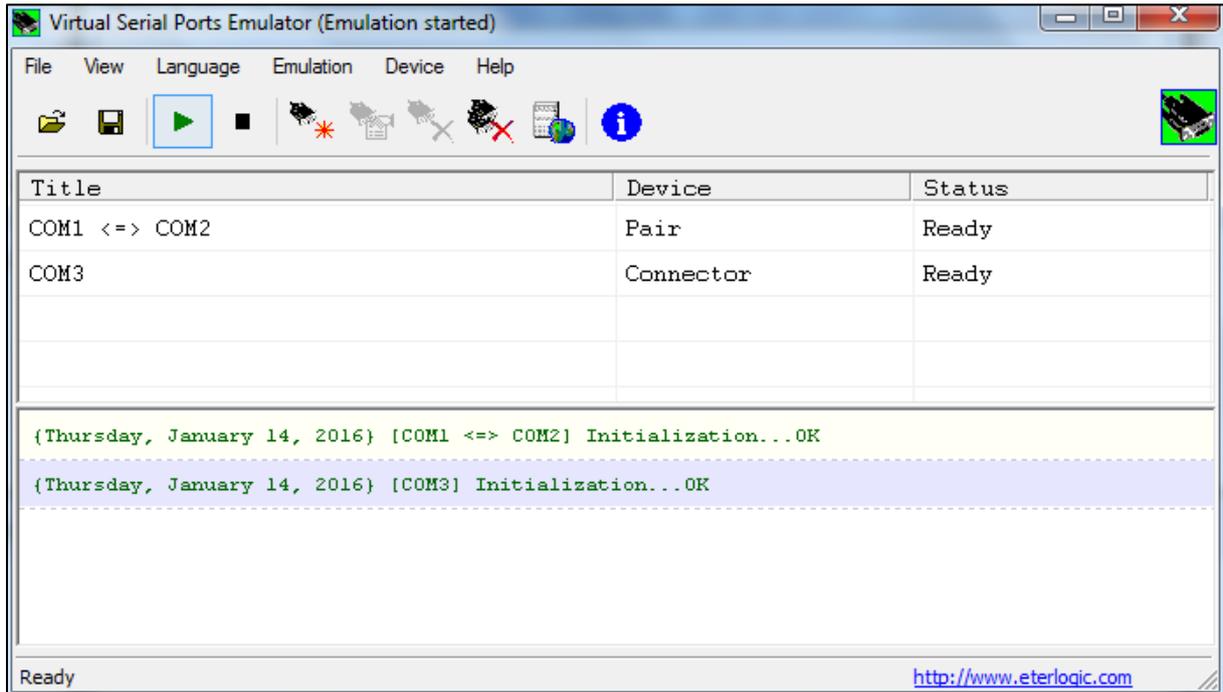


Figure 33 : VSPE(les virtuels ports)

- Double clique COMPIM (schéma ISIS).
- Faire la configuration comme indiqué dans la figure.

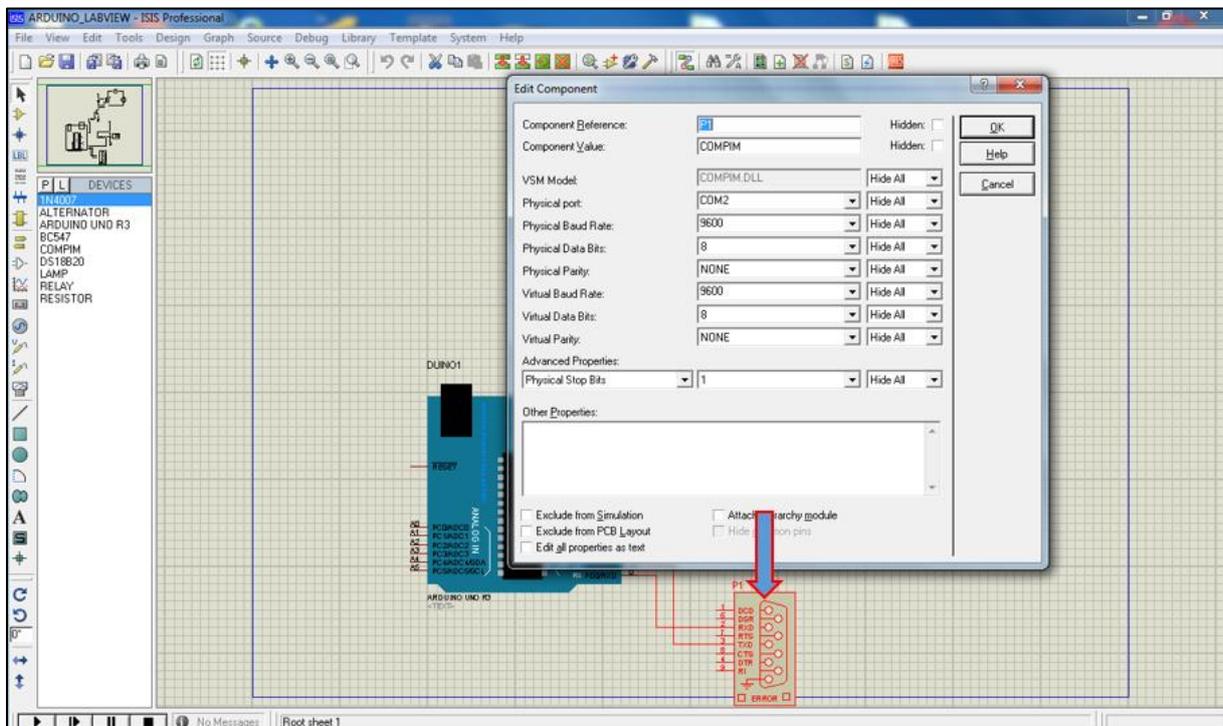


Figure 34 : Configuration du COMPIM

- Sous LabVIEW, choisir COM3.

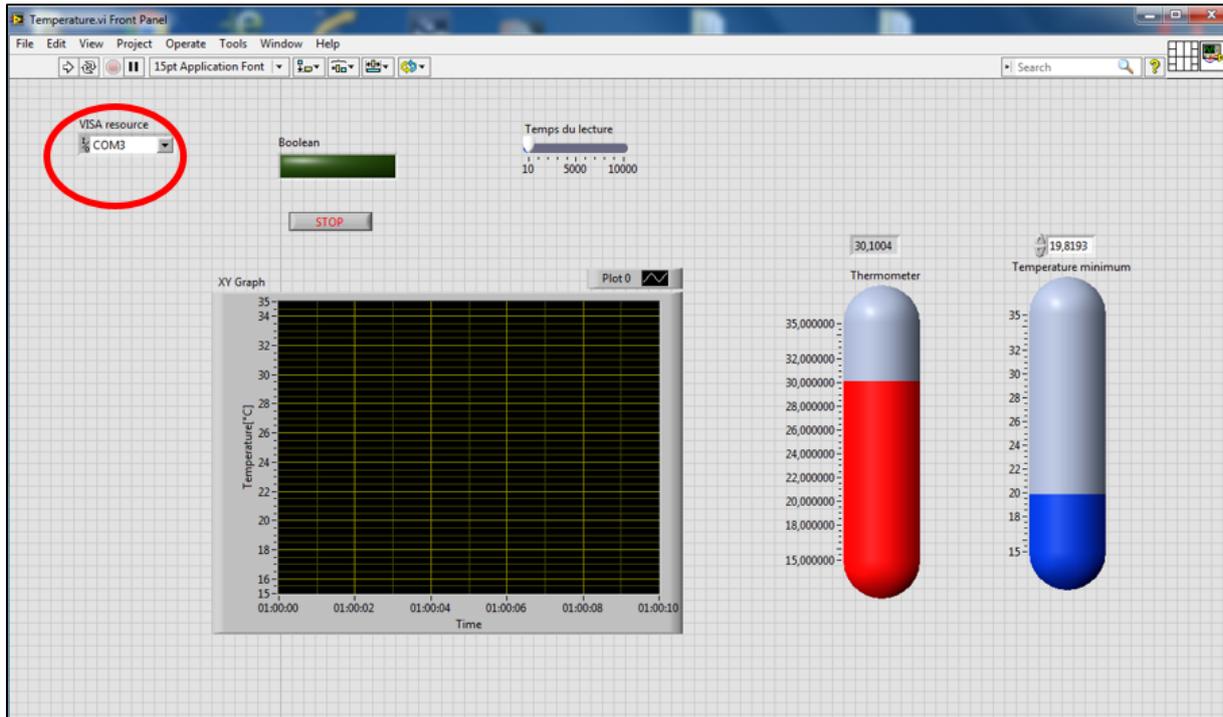


Figure 35 : VISA (COM3)

Conclusion

Ce projet m'a permis de découvrir un nouvel outil de programmation « LabVIEW » qui me semble différent en sa méthode qui se base sur des fonctions en forme des blocs. De plus, l'intégration de la carte Arduino sous LabVIEW m'a permis de manipuler autres logiciels savoir faire la communication entre eux.

