P

MICHAUX Victor, CONDE Ibrahima, VIROUX Nicolas et LIBER Maxime

Sonde de température

Table des matières

[Introduction 2](#_Toc9245526)

[Caractéristiques techniques des éléments : 3](#_Toc9245527)

[Le descriptif de la carte ainsi que son mode d'emploi 5](#_Toc9245528)

[Les schémas électroniques (logique et PCB) 6](#_Toc9245529)

[Les codes 9](#_Toc9245530)

[C (PIC) 9](#_Toc9245531)

[JAVA 12](#_Toc9245532)

[Les tests effectués, résultats et problèmes rencontrés 13](#_Toc9245533)

[Conformité (ou non) par rapport au cahier des charges 13](#_Toc9245534)

[Planning du projet et organisation du groupe 14](#_Toc9245535)

[Bibliographie 14](#_Toc9245536)

# Introduction

Dans le cadre de notre cours d’Électronique digitale & analyse des signaux, il nous a été demandé de réaliser un thermomètre digital. Pour parvenir à cela, nous allons utiliser des connaissances acquises durant le début de notre cursus étant donné que la réalisation de ce projet nécessite des bases en électronique mais aussi en programmation.

Cette sonde de température fonctionnera suite à la récupération d’un signal analogique via une sonde de température. Ce signal sera traduit afin d’être affiché sur les deux afficheurs 7 segments que nous utiliserons.

En plus de cela, il nous a été demandé de créer une application Java qui communiquera avec notre système via une interface graphique.

# Caractéristiques techniques des éléments :

Voici les différents éléments que nous avons utilisé afin de mener à bien notre projet.

Résistances de protection :

* R1 : 10k
* R2 : 100K
* R3 : 400K
* R4 : 100K

Condensateurs :

* C1 : 1µF
* C2 : 15µF
* C3 : 15µF

Microcontrôleur :

* Modèle PIC 18F458
* Se charge de convertir et de traiter les signaux provenant de la sonde de température.

Sonde :

* Modèle LM35
* C’est un circuit qui a pour but de détecter une température. Elle renvoie une tension proportionnelle à la température détectée. Cette tension est faible donc nous avons placé un ampli opérationnel qui nous permettra de traiter plus aisément la tension envoyée.

Ampli opérationnel :

* Modèle LM741

2 LED

* Afficher le dépassement ou non du seuil de température

Afficheurs 7 segments :

* Modèle 7447
* Un afficheur pour les dizaines et un pour les unités.

Transistor :

* Modèle NPN
* Ceux-ci sont reliés aux afficheurs et ont le rôle d’interrupteur. Les deux afficheurs s’allumeront et s’éteindront simultanément afin d’afficher les unités sur l’un et les dizaines sur l’autre. Ceci se fait rapidement et est donc invisible à l’œil nu.

Décodeur :

* Modèle 4511
* Permet de traiter les valeurs de température envoyée par le PIC et de les rendre compatibles pour les afficheurs 7 segments.

Serial Port

* Modèle RS232
* Sert d’intermédiaire entre notre carte et l’application Java. Nous l’avons aussi utilisé pour « flasher » notre PIC afin d’y introduire notre code C.

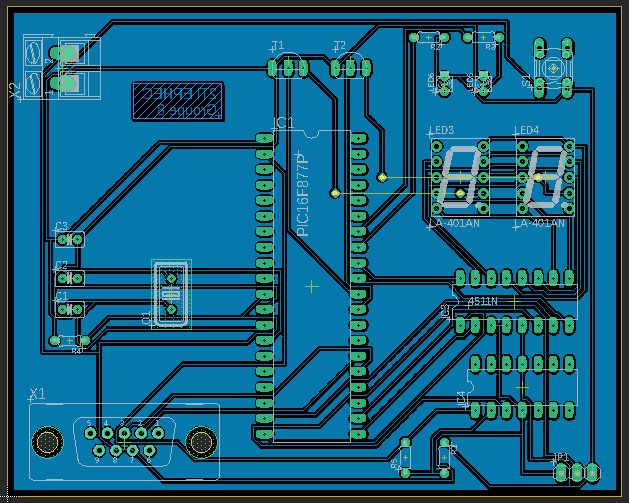
# Le descriptif de la carte ainsi que son mode d'emploi

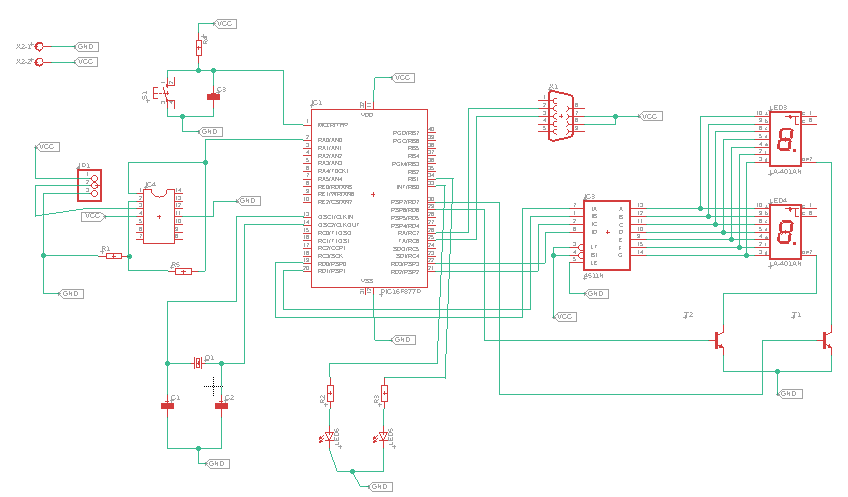
L’élément principal de notre carte est le PIC qui aura comme but de traiter les signaux reçus par la sonde de température.

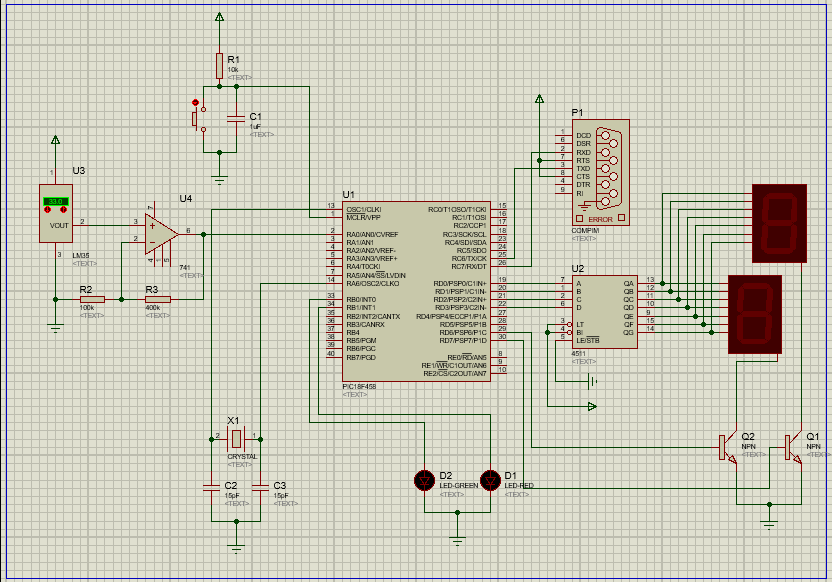
Le résultat sera affiché sur les deux afficheurs 7 segments que nous avons placé sur la carte. En plus de ces deux afficheurs, les deux LED de notre carte auront comme but de prévenir si la température affichée est dans les limites fixées dans le code C ou par l’utilisateur sur l’application Java.

Ensuite, nous avons un port série rs232 qui permet à notre plaquette d’être connectée à un ordinateur. Ce port permettra la liaison de notre carte avec l’application Java mais est aussi utile pour « flasher » le code C sur notre PIC.

# Les schémas électroniques (logique et PCB)







# Les tests effectués, résultats et problèmes rencontrés

Tout d’abord, nous avons réalisé le projet de manière virtuelle sur le logiciel Proteus. Afin d’effectuer différents tests avant d’envoyer notre circuit en production, nous avons lié notre code C que nous avions créé en parallèle de notre circuit. Ces tests se sont avérés convaincants, le résultat voulu apparaissait sur nos afficheurs 7 segments.

À la suite des résultats positifs des tests effectués, nous avons lancé la production de notre board. Une fois celle-ci finie nous avons soudé les différents composants sur le circuit imprimé. C’est alors que nous nous sommes rendus compte, que certains des composants que nous utilisions virtuellement, n’étaient pas mis à notre disposition à l’EPHEC (ex : les afficheurs 7 segments, résistances 400K…). Afin de pallier cela, nous avons décidé d’utiliser une breadboard externe à notre circuit imprimé.

Ensuite, nous avons effectué les mêmes tests que sur Proteus en conditions réelles mais malheureusement ceux-ci n’ont pas abouti positivement. Nous ne savons pas exactement à quoi sont dus ces problèmes mais nous pouvons émettre l’hypothèse d’un court-circuit ou d’un composant défaillant.

Enfin, ayant rencontré des soucis concernant notre application Java, nous n’avons donc pas pu tester le fonctionnement avec notre PIC.

# Conformité (ou non) par rapport au cahier des charges

Les éléments nécessaires au bon fonctionnement du projet sont présents sur notre carte et notre PIC a été flashée, elle contient notre code C.

Cependant, notre projet ne répond pas au cahier des charges car les afficheurs 7 segments n’affichent pas la température prise via la sonde. De plus, notre application Java n’est pas fonctionnelle car nous ne sommes pas parvenus à créer le lien entre notre application et notre plaquette. Nous avons donc été dans l’impossibilité de traiter le signal reçu par la carte.

# Planning du projet et organisation du groupe

Au lancement du projet, nous avons créé un Trello, nous l’avons mis à jour au fur et à mesure même si une discussion Messenger fut notre moyen de communication principal.

Au début du projet, nous avons fait quelques réunions au kot de Victor afin de créer notre board sur Proteus. Celles-ci nous on bien servi et cette étape s’est déroulée sans grandes difficultés.

Une fois cette partie terminée, la production de notre board a engendré une période creuse de quelques semaines car Maxime et Nicolas sont partis en semaine internationale et Victor a connu des soucis de santé. Cette période nous a malheureusement un peu perdu dans notre projet. Nous utilisions les heures de pratique afin de souder les différents composants sur notre carte mais l’application Java a été laissée de côté.

# Bibliographie

<https://www.robot-maker.com/forum/tutorials/article/20-programmation-dun-pic-sous-microc/>

<http://www.alldatasheet.com/view.jsp?Searchword=Lm35&gclid=CjwKCAjw_YPnBRBREiwAIP6TJ0Vj9keFgnB_16VTjzZczaUl6fYcLKA9QHTubSLv4lrMxa8VL_RIuxoCBU4QAvD_BwE>

<https://fr.wikipedia.org/wiki/RS-232>

<https://www.jmdoudoux.fr/java/dej/chap-rmi.htm>

<http://www.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html>

<https://www.microchip.com/wwwproducts/en/PIC18F458>

<https://www.youtube.com/watch?v=43Vdpz1YmdU>

# Les codes

## C (PIC)

#include <18F458.h>

#device ADC=10

#use delay(crystal=20000000)

#use rs232(baud=9600,parity=N,xmit=PIN\_C6,rcv=PIN\_C7,bits=8)

#define BCD\_UNITS(x) (x % 10) //unité

#define BCD\_TENS(x) (x / 10) //dizaine

void affU(int temp);

void affD(int temp);

void affT(int temp);

void allumeLED(int temp , int max, int min);

void main()

{

setup\_adc\_ports(AN0);

setup\_adc(ADC\_CLOCK\_INTERNAL); //mettre une clock interne

setup\_low\_volt\_detect(FALSE);

int max = 30;

int min = 1;

int temp = 20;

set\_adc\_channel(0);

while(TRUE)

{

temp = read\_adc() \* 100 / 1024; //convertir la tension en température

allumeLED(temp,max,min); //voir fonction plus bas

affT(temp);

}

}

void allumeLED(int temp, int max, int min){

if(temp > max || temp < min){

output\_high(PIN\_B1); //allume la LED rouge

output\_low(PIN\_B0); //éteint la LED verte

}

else{

output\_low(PIN\_B1); //éteint la LED rouge

output\_high(PIN\_B0); //allume la LED verte

}

}

void affT(int temp){

while(TRUE){

affU(temp);

delay\_ms(10);

affD(temp);

delay\_ms(10);

}

}

void affU(int temp){

output\_d(BCD\_UNITS(temp) | (1 << 4));

output\_low(PIN\_D7);

output\_high(PIN\_D6);

}

void affD(int temp){

output\_d(BCD\_TENS(temp) | (1 << 5));

output\_low(PIN\_D6);

output\_high(PIN\_D7);

}

## JAVA

### Class VueThermo

package vue;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JTextField;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import javax.swing.JButton;

import javax.swing.JTextArea;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JSlider;

import java.awt.Color;

public class VueThermo {

public JFrame frame;

private JTextField Grp8;

private JTextField txtTemperatureActuelle;

private JTextField txtTemperatureMin;

private JTextField txtTemperatureMax;

private JTextField minField;

public int min=-30;

public int max=70;

//public int tempMax = 100;

private JTextField maxField;

public VueThermo() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

frame = new JFrame();

frame.getContentPane().setBackground(Color.YELLOW);

frame.setBounds(100, 100, 450, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(null);

Grp8 = new JTextField("Bienvenue sur la sonde de temperature du groupe 8 ");

Grp8.setBounds(58, 11, 299, 20);

frame.getContentPane().add(Grp8);

Grp8.setColumns(10);

Grp8.setEditable(false);

txtTemperatureActuelle = new JTextField();

txtTemperatureActuelle.setText("Temperature Actuelle : ");

txtTemperatureActuelle.setBounds(10, 56, 136, 20);

frame.getContentPane().add(txtTemperatureActuelle);

txtTemperatureActuelle.setColumns(10);

txtTemperatureActuelle.setEditable(false);

txtTemperatureMin = new JTextField();

txtTemperatureMin.setText("Temperature Min : ");

txtTemperatureMin.setBounds(10, 87, 136, 20);

frame.getContentPane().add(txtTemperatureMin);

txtTemperatureMin.setColumns(10);

txtTemperatureMin.setEditable(false);

txtTemperatureMax = new JTextField();

txtTemperatureMax.setText("Temperature Max : ");

txtTemperatureMax.setBounds(10, 118, 136, 20);

frame.getContentPane().add(txtTemperatureMax);

txtTemperatureMax.setColumns(10);

txtTemperatureMax.setEditable(false);

JButton btnExit = new JButton("EXIT");

btnExit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

System.exit(0);

}

});

btnExit.setBounds(335, 227, 89, 23);

frame.getContentPane().add(btnExit);

JTextArea tmpAct = new JTextArea();

tmpAct.setBounds(156, 54, 47, 20);

frame.getContentPane().add(tmpAct);

tmpAct.setEditable(false);

tmpAct.setText("20 °");

JSlider sliderMin = new JSlider();

sliderMin.setBounds(213, 81, 200, 26);

frame.getContentPane().add(sliderMin);

JSlider sliderMax = new JSlider();

sliderMax.setBounds(213, 112, 200, 26);

frame.getContentPane().add(sliderMax);

minField = new JTextField();

minField.setBounds(156, 87, 47, 20);

frame.getContentPane().add(minField);

minField.setColumns(10);

minField.setEditable(false);

maxField = new JTextField();

maxField.setBounds(156, 118, 47, 20);

frame.getContentPane().add(maxField);

maxField.setColumns(10);

maxField.setEditable(false);

sliderMin.addChangeListener(new ChangeListener() {

public void stateChanged(ChangeEvent arg0) {

min=sliderMin.getValue();

minField.setText(min+"");

sliderMin.setMaximum(70);

sliderMin.setMinimum(-30);

}

});

sliderMax.addChangeListener(new ChangeListener() {

public void stateChanged(ChangeEvent arg0) {

min=sliderMax.getValue();

maxField.setText(min+"");

sliderMax.setMaximum(70);

sliderMax.setMinimum(-30);

}

});

}

}

### Class main

package main;

import java.awt.EventQueue;

import vue.VueThermo;

public class main {

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

VueThermo window = new VueThermo();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

}

### Class ThermController

package controller;

import vue.VueThermo;

public class ThermController {

protected VueThermo vue;

public void addView(VueThermo vue) {

this.vue = vue;

}

}

### Class COMListener

package model;

import gnu.io.CommPort;

import gnu.io.CommPortIdentifier;

import gnu.io.SerialPort;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.PrintWriter;

import java.io.UnsupportedEncodingException;

public class COMListener {

public InputStream in;

public OutputStream out;

private int rate=9600;

public COMListener(String portName) throws Exception {

super();

CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(portName);

if (portIdentifier.isCurrentlyOwned()) {

System.out.println("Error: Port is currently in use");

}

else {

CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);

if (commPort instanceof SerialPort) {

SerialPort serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(rate, SerialPort.DATABITS\_8,SerialPort.STOPBITS\_1, SerialPort.PARITY\_NONE);

in = serialPort.getInputStream();

out = serialPort.getOutputStream();

}

else {System.out.println("Error: Only serial ports are handled by this example.");

}

}

}

}