



6, bd maréchal Juin
F-14050 Caen cedex 4

Filière Informatique & Réseaux
2^e année

ENSICAEN

Rapport de projet

Etude des microcontrôleurs TINI

M'barki Med Amine
Chamakh Akram

Suivi Ensicaen :
Sir Philippe Lefebvre

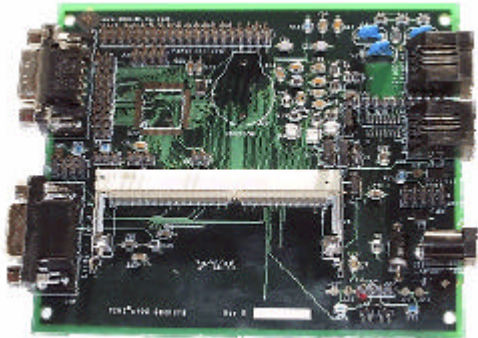

Année 2003-2004

Table des matières

1	Introduction.....	3
1.1	Qu'est ce que le TINI?.....	3
1.2	Equipement requis.....	3
2	Démarrage du kit	5
2.1	La procédure de mise en marche.....	5
2.2	Bibliothèques de tini, utilités, bibliothèques 1 Wire du tini.....	6
2.3	Exécuter un " <i>simple</i> " programme java par le tini.....	10
3	Mesure de température.....	10
3.1	Caractéristiques du Ibutton DS1920.....	10
3.2	Schéma électrique	11
3.3	Le programme « TemperatureServer.java ».....	11
3.4	Chargement du programme de température dans le Tini-m400.....	12
4	Conclusion	13
	Références bibliographiques et contact.....	14
	Annexe : (Le programme « TemperatureServer.java » commenté).....	15

1 Introduction

Les panneaux de douilles de TINI sont des cartes mères conçues pour accueillir le module correspondant d'évaluation de TINI. La combinaison des deux cartes permet la pleine évaluation des dispositifs du microprocesseur de cible. Pour évaluer le DS80C400, commandez un DSTINIm400 et un DSTINIs400. Bien que les divers conseils soutiennent différents modules d'évaluation, le plus populaire est le DSTINIs400. Il inclut une 144-goupille connecteur de SODIMM et fournit 1-Wire®, CAN2.0B, et un Ethernet PHY de 10/100 pour relier le TINIm400 au monde physique. Les DSTINIm400 et les DSTINIs400 sont entièrement assemblés et les cartes examinées. Une fois utilisés ensemble ils forment un système complet d'évaluation pour le microcontrôleur du réseau DS80C400. Le module DSTINIm400 inclut le microcontrôleur du réseau DS80C400, une horloge en temps réel, un flash 1MB, la RAM 1MB statique, et le soutien d'un Ethernet externe PHY pour se relier à une grande variété de réseaux. La carte est conçue comme module à brancher au 144-goupille connecteur de SODIMM sur le DSTINIS400.

TINIs400	TINIm400
	

1.1 Qu'est ce que le TINI ?

Notre projet consiste à faire une étude sur les microcontrôleurs Tini-m400 et Tini-s400 de la famille TINI. Un TINI est un microcontrôleur qui exécute des programmes java et il a la gestion de réseau et les interfaces intégrées d'Ethernet pour relier différents types de matériel. Le nom TINI représente l'interface minuscule d'Internet et se rapporte au jeu de puces de TINI et à la carte de TINI.

L'unité centrale de traitement de TINI est le DS80C390 de Dallas. Semiconductor. TINI a des ports en série, en parallèle, un 1-Wire, un I²C et des ports du bus CAN, avec les goupilles supplémentaires pour commander les dispositifs facultatifs. Il peut adresser jusqu'à 1Mo de la RAM et 1Mo du Flash ROM. La carte TINI contient une interface RS232, une horloge en temps réel, une unique adresse MAC d'Ethernet, et une protection de batterie pour la RAM. Le contrôleur d'Ethernet soutient la gestion de réseau 10BaseT et vous permet beaucoup d'applications sur internet. La ROM Flash contient les bibliothèques, la machine virtuelle de Java et les bibliothèques des classes Java de TINI, Elle a également l'espace pour stocker un programme de circuit fermé.

1.2 L'équipement requis :

Les hardware et software minimum requis sont:

- Les Besoins en matériel de TINI
Module de vérification du TINIm400.

Panneau ou équivalent des douilles TINIs400.

- Conditions De Logiciel de TINI

Version 1.1 ou plus du kit de développement du logiciel de TINI à l'adresse:

<http://www.ibutton.com/TINI/software/index.html>.

JDK (kit de développement de Java) à l'adresse:

<http://java.sun.com/>

Communications api de Java à l'adresse: <http://www.rxtx.org>.

- Conditions De Système De Développement

Conditions de réunion de logiciel d'exploitation du JDK de « Sun microsystems ».

1 port RS232 (COM) 115200 bauds recommandé.

Câble serie DB9 male de RS232C à la femelle DB9- catalogue numéro 26-117 de Radio Shack ou équivalent.

Câble d'Ethernet de croisement - le catalogue numéro 950-0368 de Radio Shack ou l'équivalent pour le raccordement direct à l'ordinateur principal, un câble traversant droit devrait être employé pour se relier à un routeur ou un hub voir l'adresse:

<http://www.radioshack.com>.

Approvisionnement d'alimentation CC 5V capable de fournir 150mA voir le catalogue numéro 900-2740 de Radio Shack.

2 Démarrage du kit.

2-1 La procédure de mise en marche.

- 1- Avant de commencer, vérifiez que la puissance n'est pas reliée au panneau des douilles TINIs400.
- 2- Insérez le module de la vérification TINIm400 dans le connecteur 144 sur le panneau des douilles TINIs400.
- 3- Vérifiez que le DTR (J14) est placé sur le panneau de douilles.
- 4- Attachez un câble RS232 au connecteur marqué «loader serial 0» comme représenté sur le schéma 1.
- 5- Reliez le câble série à une entrée série sur votre PC.
- 6- Après, attachez le câble d'Ethernet de croisement entre le PC et les douilles du TINIs400.
- 7- Reliez l'adaptateur de puissance à vos douilles TINIs400 comme représenté sur le schéma 1.
Une alimentation par 5V DC doit être employée avec le panneau des douilles TINIs400.
D'autres panneaux de douilles peuvent avoir une différente alimentation électrique.
Référez-vous à la documentation de panneau de douilles pour plus d'information.
- 8- Branchez l'adaptateur de puissance à une prise murale.

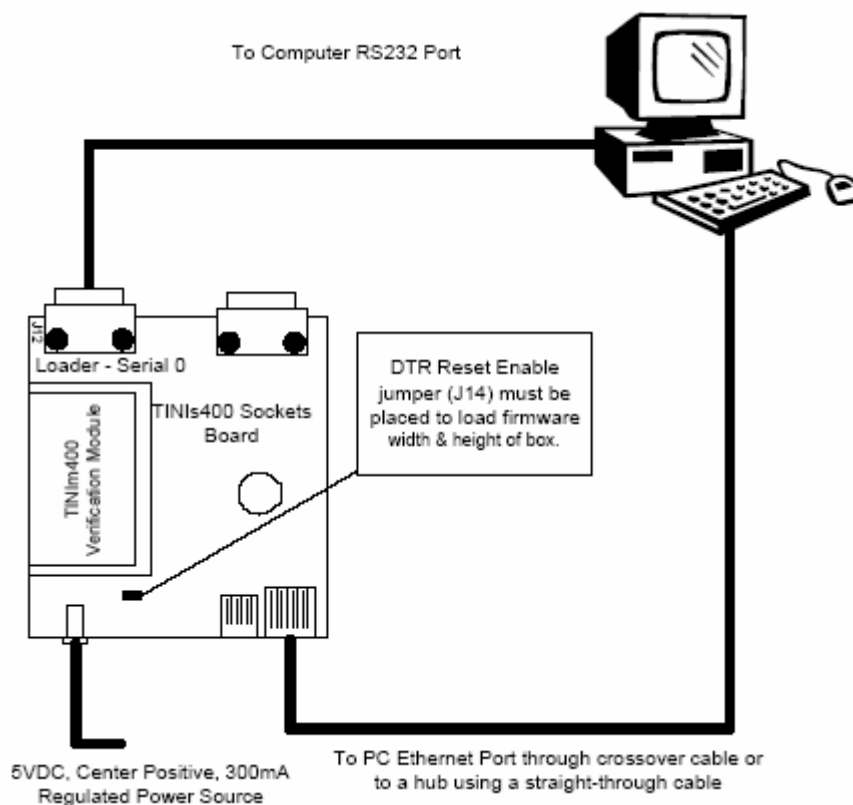


Figure1

2-2 bibliothèques de tini, utilités, bibliothèques 1 Wire du Tini

Vu qu'on a travaillé sous l'environnement Linux on va développer les différentes étapes à ce sujet en prenant en compte cette environnement. la bibliothèque et les utilités principales des logiciels du Tini sont assurées par Dallas Semiconductor.

D'abord on télécharge le TINI SDK (tini1_02.tgz ou la version courante) dans un répertoire temporaire. Ensuite on décompresse le fichier dans le répertoire /opt comme suit:

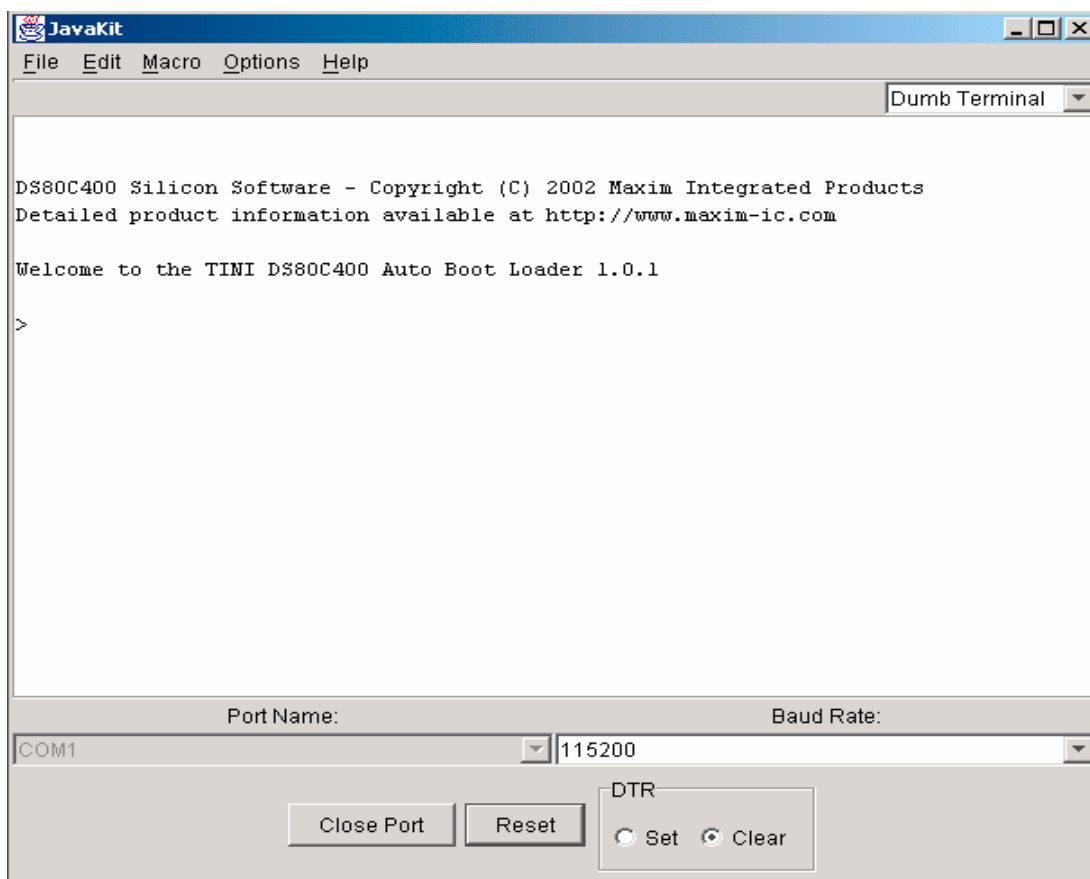
```
% cd downloads
% mv tini1_02.tgz /opt
% cd /opt
% uncompress tini1_02.tgz
% tar -xf tini1_02.tar
```

Après on doit ajouter **tini.jar** à la variable d'environnement **CLASSPATH** ainsi qu'une variable d'environnement qui indique le répertoire **TINI_HOME**.

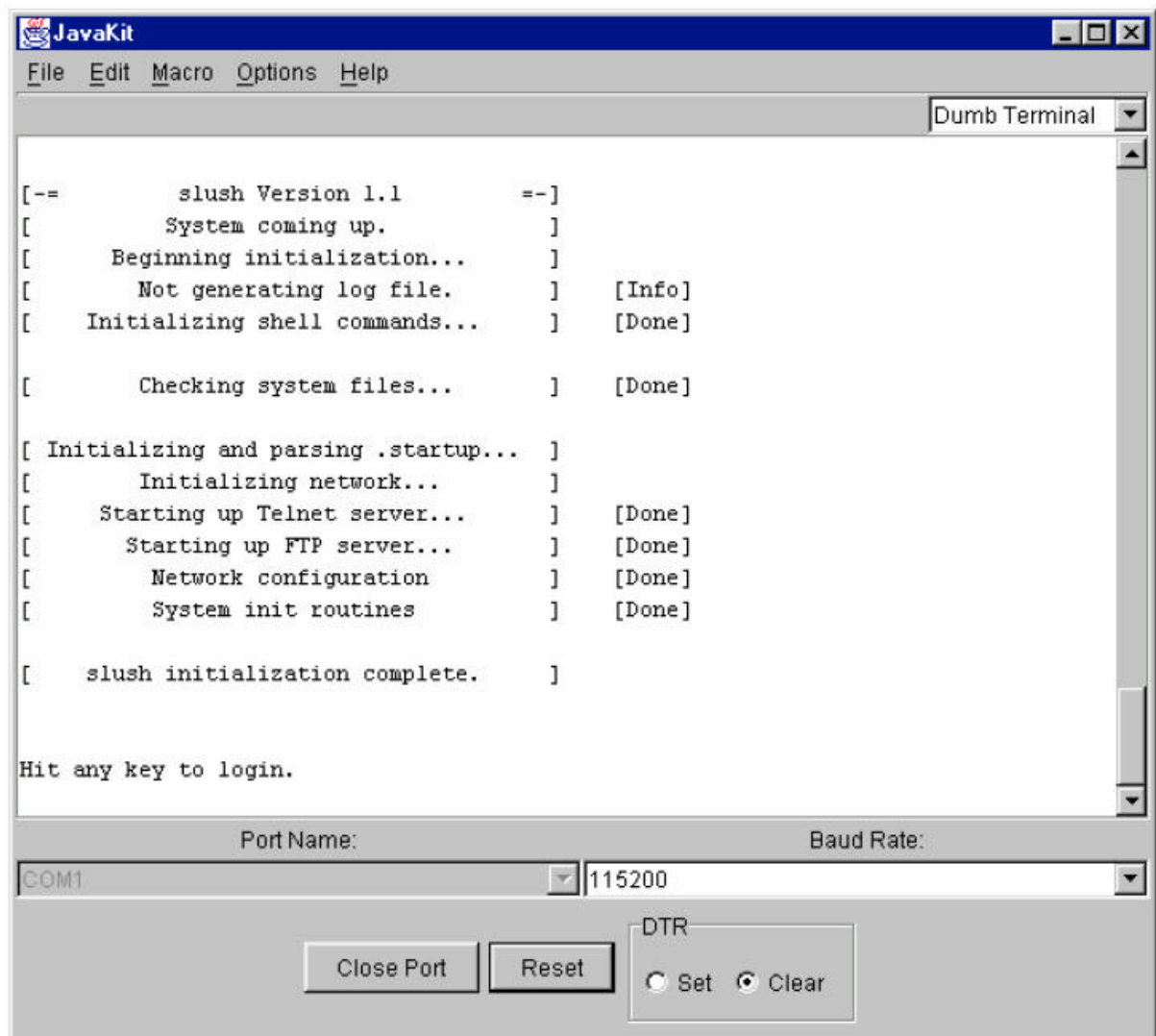
On installe le kit de développement Java depuis le site <http://java.sun.com/products/> on télécharge ensuite le Java 2 SDK (j2sdk1.4.2_02-linux-rpm.bin).

Pour démarrer le **JavaKit** il faut relier le port série de l'ordinateur à celui de la carte TINI, celui ci est utilisé pour configurer les progiciels du TINI. Les étapes sont:

- 1- Ouvrir une console Shell puis se positionner sur le répertoire <TINI SDK Install Dir>\bin.
- 2- Ensuite, taper la commande **java -classpath tini.jar JavaKit -400 -flash 40**, si le kit de développement Java et les API de communication Java sont proprement installés, le **JavaKit** devra normalement apparaître à l'écran.
- 3- Sélectionner le nom du port auquel le **TINI** est attaché. Appuyer sur le bouton «**Open Port**». La vitesse par défaut en Baud est 115200.
- 4- Après avoir appuyer sur le bouton pour ouvrir le port on appuie sur le bouton «**Reset**», ainsi on voit apparaître une invite de commande.



- 5- Si le message d'invite n'est pas affiché, vérifier si le bon port a été sélectionné, sinon si tout se passe comme prévu, lire le fichier **Running_JavaKit** contenu dans le **TINI SDK** pour plus d'informations.
- 6- taper **B0**, entrer puis taper sur **F0** et entrer. Ceci devra effacer la **RAM** du **TINI**. A chaque fois qu'un nouveau temps d'exécution d'un environnement est chargé, la mémoire devra être effacé.
- 7- Maintenant cliquer sur l'onglet **File** de la barre d'outil et sélectionner l'option **Load File**. Un fichier de dialogue s'affiche alors à l'écran. Passer en revue le répertoire <**TINI SDK Install Dir**>\bin, sélectionner le fichier **tini_400.tbin** puis cliquer sur le bouton **Open**. Le téléchargement prendra 30 secondes.
- 8- Pour charger la commande Shell, cliquer sur le menu **File** puis sélectionner **Load File** encore. Sélectionner le fichier **slush_400.tbin** puis cliquer sur le bouton **Open**.
- 9- Appuyer sur le bouton **Reset** et le prompt de JavaKit apparaîtra.
- 10- À l'invite de commande, taper E puis appuyer sur la touche entrée, un texte de «Booting» apparaît alors sur l'écran.

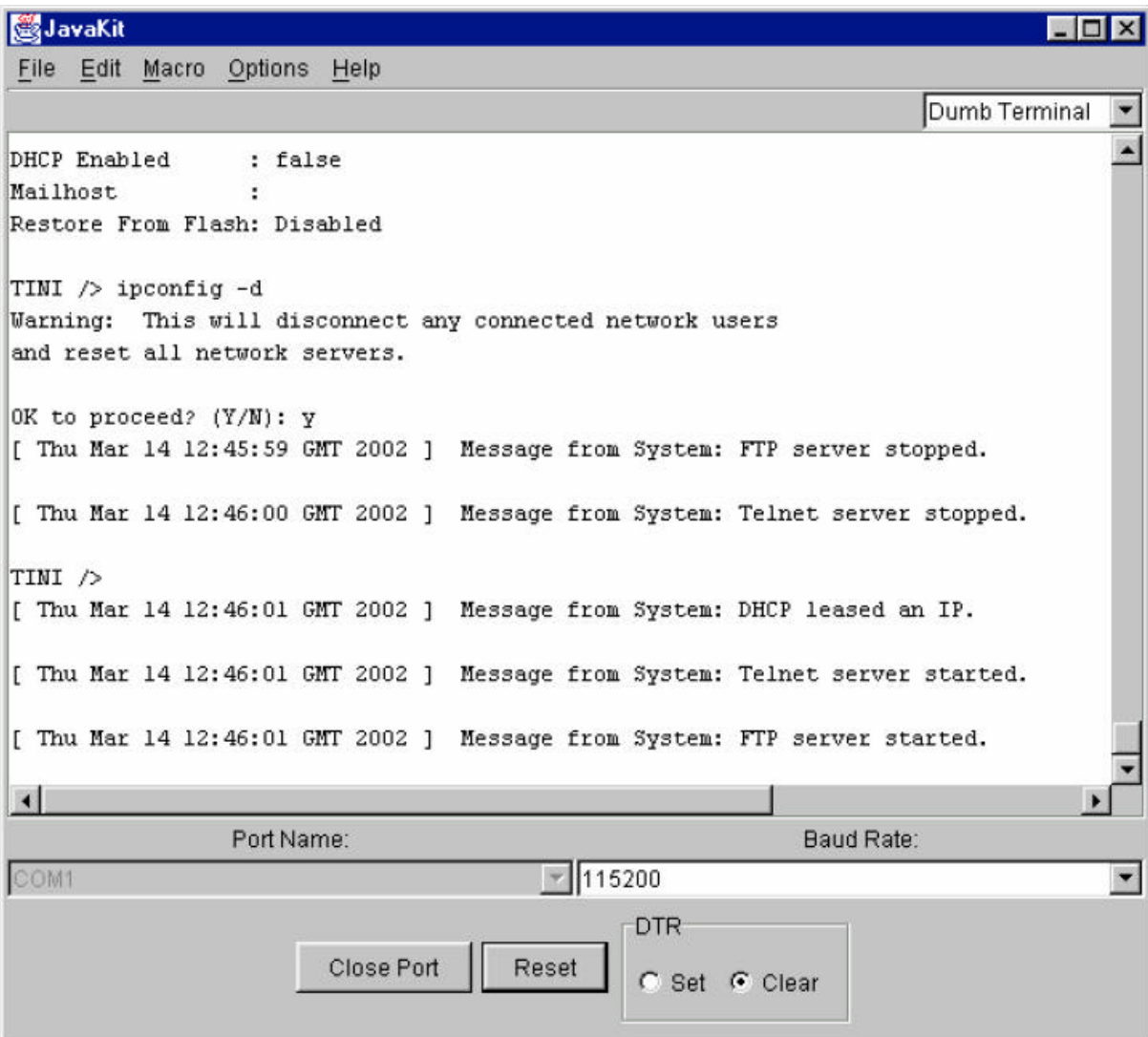


- 11- Après appuyer sur entrée, un login vous sera demandé. Taper le nom d'utilisateur **root** et appuyer sur le bouton entrée. Le premier mot de passe est par contre **tini**.

12- Pour visionner l'ensemble des commandes taper **help**.

Pour accéder au réseau, le TINI aura accès à une configuration réseau grâce à la commande **ipconfig**. En tapant la commande **ipconfig -h** vous aurez accès aux options détaillées de la configuration. **-h** est utilisé avec n'importe quel commande pour voir les options.

La méthode la plus simple de se connecter au réseau est le **DHCP** si il y'a un serveur DHCP. Pour permettre DHCP taper **ipconfig -d** est entrer, une fois que le module de vérification de TINI a loué avec succès une adresse de réseau, un message est affiché comme sur le schéma suivant:



```
JavaKit
File Edit Macro Options Help
Dumb Terminal

DHCP Enabled      : false
Mailhost          :
Restore From Flash: Disabled

TINI /> ipconfig -d
Warning: This will disconnect any connected network users
and reset all network servers.

OK to proceed? (Y/N): y
[ Thu Mar 14 12:45:59 GMT 2002 ] Message from System: FTP server stopped.

[ Thu Mar 14 12:46:00 GMT 2002 ] Message from System: Telnet server stopped.

TINI />
[ Thu Mar 14 12:46:01 GMT 2002 ] Message from System: DHCP leased an IP.

[ Thu Mar 14 12:46:01 GMT 2002 ] Message from System: Telnet server started.

[ Thu Mar 14 12:46:01 GMT 2002 ] Message from System: FTP server started.

Port Name: COM1 Baud Rate: 115200
Close Port Reset DTR
Set Clear
```

Si votre réseau ne supporte pas DHCP, il est alors nécessaire d'utiliser une adresse IP statique. D'abord, vous devez obtenir une adresse de réseau et un masque de sous réseau valide de votre administrateur réseau. Afin de configurer une adresse IP statique, utiliser **ipconfig -a x.x.x.x -m y.y.y.y** ou l'adresse IP est noté par **x.x.x.x** et le masque par **y.y.y.y**. Par exemple, pour configurer une adresse IP de 192.168.1.50 avec un masque de 255.255.255.0, taper **ipconfig -a 192.168.1.50 -m 255.255.255.0** à l'invite de commande puis entrer.

Une bonne façon de tester les configurations du réseau est d'utilisé la commande **ping**.

Le **TINI SDK** contient plusieurs exemples d'applications. Chaque exemple d'application à son fichier batch pour le compiler et le convertir en fichier binaire pour une exécution en

temps réel dans l'environnement TINI. Une simple application inclus avec le TINI SDK est **Blinky**. Cette application fait clignoter une LED (DS1) du TINIm400. Pour transférer cette application au TINIm400, ouvrez une invite de commande et se positionner sur le répertoire **<TINI SDK Install Dir>\examples\Blinky\bin** ce répertoire contient un fichier nommé Blinky.tini. Depuis l'invite de commande de l'ordinateur principale taper ftp. Le mot de passe par défaut est « tini ».

```
ftp> open 192.168.1.50
Connected to 192.168.1.50.
220 Welcome to slush. (Version 1.1) Ready for user login.
User (192.168.1.50:(none)): root
331 Password Required for root
Password:
230 User root logged in.
ftp> bin
200 Type set to Binary
ftp> put Blinky.tini
200 Command successful.
150 BINARY connection open, putting Blinky.tini
226 Closing data connection.
ftp: 514 bytes sent in 0.00Seconds 514000.00Kbytes/sec.
ftp> bye
221 Goodbye
```

Le fichier existe maintenant dans le système de fichiers de TINI. Après, se connecter à votre TINIm400 en utilisant telnet sur votre PC. Vérifiez l'existence du fichier Blinky.tini en utilisant la commande **LS**. Pour exécuter le programme de Blinky.tini, taper **Java Blinky.tini** suivi d'entrée. Le programme de Blinky s'exécute et la LED devrait commencer un clignotement régulier.

Les changement de profile se traduiront comme suit:

```
# .bashrc
# User specific aliases and functions
# Source global definitions

TINI_HOME=/opt/tini1.12
export TINI_HOME

CLASSPATH=/usr/java/j2sdk1.4.2_02/lib:/usr/java/j2sdk1.4.2_02/commapi/comm.
jar:$TINI_HOME/bin/tini.jar:.
PATH=/usr/java/j2sdk1.4.2_02/bin:$PATH

OW_HOME=/opt/onewire
CLASSPATH=$CLASSPATH:$OW_HOME/lib/OneWireAPI.jar

export OW_HOME CLASSPATH

alias ll="ls -l"

alias javakit='java -cp
/usr/java/j2sdk1.4.2_02/lib:/usr/java/j2sdk1.4.2_02/commapi/comm.jar:
/opt/tini1.12/bin/tini.jar JavaKit -400 -flash 40&'
```

2.3 Exécuter un « *simple* » programme java par le tini :

Considérant un *simple* fichier java « HelloWorld.java » il faut passer par les étapes suivantes :

- 1- **Javac HelloWorld.java** ceci nous donne un fichier **HelloWorld.class**
- 2- on converti le fichier .class en fichier .tini en faisant : **java TINIconvertor -f HelloWorld.class -o HelloWorld.tini -d \$TINI_HOME/bin/tini.db**
- 3- ensuite on envoie le fichier **tini** à la maquette en utilisant **ftp**.
- 4- enfin on se connecte au **tini** soit par **telnet** soit le **JavaKit** et on tape la commande :
java HelloWorld.

3 Mesure de température

3.1 Caractéristiques du Ibutton DS1920.

Le DS1920 fournit les lectures numériques de la température 9-bit sur une gamme de -55°C à +100°C dans les incréments 0.5°. L'iButton® communique avec un processeur en utilisant le protocole 1-Wire® par une interface de port de matériel. L'interface gauche fournit le lien physique et manipule les protocoles de transmission qui permettent au processeur d'accéder à des ressources d'iButton avec des commandes simples. Deux bytes d'EEPROM peuvent être employés ou pour placer des déclenchements d'alarme ou pour stocker des données d'utilisateur.

Blindé dans un iButton, le thermomètre peut mesurer les températures dans les environnements exigeants. Comme chaque autre iButton, le DS1920 a un numéro de série unique. L'interface 1-Wire ramène le rendement de données, l'adresse, et la puissance à une ligne. Un numéro de série unique permet aux multiples 1920s de partager le même bus, avec beaucoup de sondes placées à différents emplacements faisant rapport à un port simple sur un processeur. Les DS1920s sentent la température de bas puissance dans les bâtiments, l'équipement, ou les machines intérieur disponible et dans de processus le contrôle. Avec les accessoires appropriés le DS1920 s'attache facilement à un objet ou à une carte.

Les caractéristiques du DS1920 sont :

- Le thermomètre mesure les températures de -55°C à +100°C, typiquement en 0.2 seconde.
- L'exactitude à $\pm 0.5^{\circ}\text{C}$ entre 0°C et +70°C.
- Alimentation générale nulle.
- Alimentation d'énergie par le contact de données.
- Accès aux compteurs internes permet la résolution accrue par l'interpolation.
- Deux octets d'EEPROM pour l'usage comme déclenchements d'alarme ou mémoire d'utilisateur.
- Contrôleur intégré de réseau ; la recherche d'alarme identifie directement des dispositifs sentant les températures alarmantes.
- CRC de 8 bits pour le contrôle en marche de l'intégrité des données.

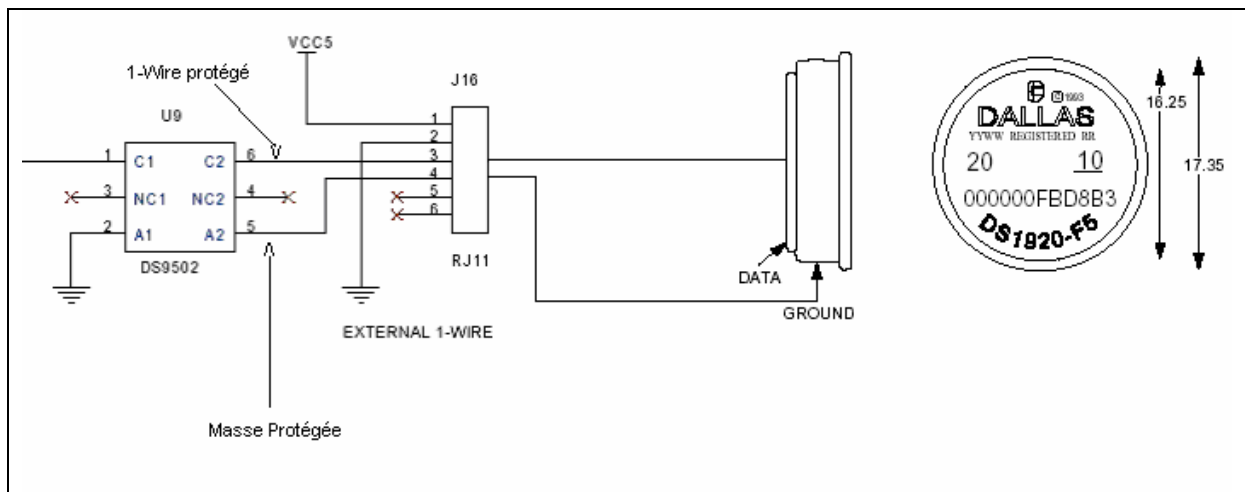
Le **DS1920** à une mémoire de type **EEPROM** et de taille 16 bits, son prix est d'environ 3.34\$, la photo du dispositif est la suivante :



DS1920

3.2 Schéma électrique :

Le Schéma électrique du branchement est comme suit :



3.3 Le programme « TemperatureServer.java » :

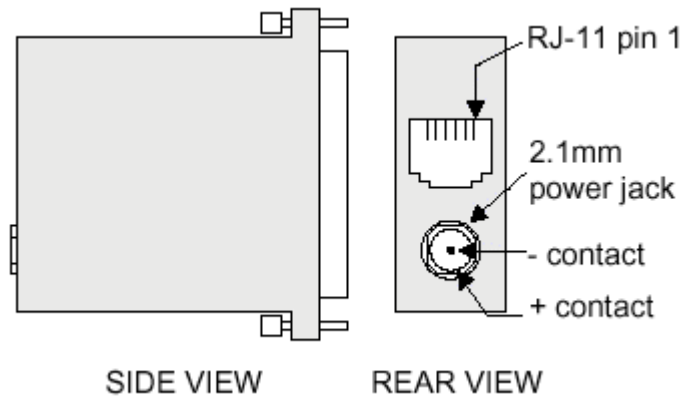
Le programme se présente comme suit :

TemperatureServer

```
+ adapter :: DSPortAdapter
+ container :: OneWireContainer10
+ state :: * byte
+ ROM_ID :: String
+ degC :: double
+ degF :: double

+ TemperatureServer() {}
+ TemperatureServer( String PAadapt, String Port) {}
+ TemperatureServer( String ROM, String PAadapt, String Port) {}
+ Init(String ROM) { }
+ measureT() {}
+ main(String[] args) { }
+ writePage(BufferedWriter wr, String ct,double degc,double degf, String rom) {}
```

- Le champ **adapter** correspondra à un adaptateur qui peut aussi servir à relier le capteur à notre maquette son schéma est comme suit (exemple du DS9097U) :



- Le champ **container** permettra d'accéder aux différentes fonctions liés à la famille des capteurs de température comme les méthodes : **adapter.getFirstDeviceContainer()** et **getAddressAsString()** et **container.readDevice()**, etc ...
- Le champs **state** va contenir l'information numérique fourni après avoir converti la mesure analogique effectué par le DS1920.
- Le champ **ROM_ID** correspond à l'ID de la ROM du dispositif (dans notre cas du DS1920).
- Les champs **degC** et **degF** correspondent aux mesures en degrés Celcius et fahrenheit de la température.

Concernant les méthodes il y'a trois **constructeurs**, le choix d'utilisation se fera en fonction des adaptateurs (DS1410E, DS9097U, TINIEExternalAdapter ...) et des ports (COM1, LPT1, /dev/ttyS0, or serial1) utilisés. Une méthode **Init** qui initialise le thread suivant qu'on utilise un dispositif ou un adaptateur particulier. La méthode **measureT** s'occupe de renvoyé la température mesurée par le capteur puis fait la conversion en fahrenheit. Le **main** quand à lui contient un programme serveur qui écoute les requêtes clientes depuis le port 80. La méthode **writePage** sert à mettre en page le flot de sortie, on trouve dans cette méthode essentiellement du code HTML.

Pour plus de détails voir l'annexe.

3.4 Chargement du programme de température dans le Tini-m400.

Le programme de température utilise des classes qui appartiennent à un package permettant à la maquette tini de communiquer avec tous les dispositifs iButtons. Les étapes sont comme suit :

- On convertit le fichier « *TemperatureServer.java* » en .class on faisant :
**Javac -classpath \$TINI_HOME/bin/owapi_dependencies_TINI.jar :
TemperatureServer.java**
- Pou transformer le fichier .class que l'on met dans le répertoire **bin** en fichier .tini on fait comme suit:
**Java -cp \$TINI_HOME/bin/tini.jar BuildDependency -f /bin -o
TemperatureServer.tini -d \$TINI_HOME/bin/tini.db -p
\$TINI_HOME/bin/owapi_dependencies_TINI.jar -x
\$TINI_HOME/bin/owapi_dep.txt -add Thermometers**

4 Conclusion

Les avancées récentes renvoyées dans la technologie de gestion de réseau ont stimulé l'intérêt énorme pour le développement des dispositifs commandés par ordinateur, des appareils futés, où prétendus dispositifs électroniques communiquent directement avec l'Internet.

Il y a plusieurs raisons de relier certains dispositifs à un LAN ou à l'Internet. Une des raisons plus contraignantes est qu'un contrôleur incorporé n'a plus besoin d'une interface utilisateur d'affichage ou de matériel. Un navigateur Web peut devenir l'interface utilisateur standard pour votre dispositif et peut vous réduire le coût de votre conception en éliminant le besoin de matériel d'interface utilisateur. Un autre avantage est que l'Internet fournit une manière pour vous de rassembler des données opérationnelles au sujet de votre dispositif et d'apprendre comment et quand vos clients l'emploient et quels dispositifs ils emploient ou utilisent.

En employant un réseau standard et un TINI, vous pourriez surveiller la météo en utilisant des capteurs, distribués par exemple partout dans le monde tandis que TINI. Un TINI géré en réseau pourrait alors rechercher l'information reliée par temps additionnel des serveurs accessibles. En plus, des données qui peuvent être trop informatique intensives, comme la période du lever de soleil ou la phase de la lune, pourraient être calculées et recherchées à partir d'un ordinateur sur le réseau qui est davantage convenu à la tâche.

Références bibliographiques et contact

- Conditions De Logiciel de TINi
Version 1.1 ou plus du kit de développement du logiciel de TINi à l'adresse:
<http://www.ibutton.com/TINI/software/index.html>.
- JDK (kit de développement de Java) à l'adresse:
<http://java.sun.com/>
Communications api de Java à l'adresse:
<http://www.rxtx.org>.
- Le catalogue Radioshack est à l'adresse :
<http://www.radioshack.com>
- On installe le kit de développement Java depuis le site :
<http://java.sun.com/products/>
- Le livre « Designing Embedded Internet Devices » avec son CD ou l'on trouve des programmes destinés aux microcontrôleurs TINi.

Pour toute demande de renseignement contacter :

- mbarki@ensicaen.ismra.fr
- chamakh@ensicaen.ismra.fr
- lefebvre@ensicaen.ismra.fr

Annexe (Le programme java commenté).

```
import com.dalsemi.onewire.*;
import com.dalsemi.onewire.adapter.*;
import com.dalsemi.onewire.container.*;
import java.util.*;
import java.io.*;
import java.net.*;

public class TemperatureServer {
    DSPortAdapter adapter; //interface de communication
    OneWireContainer10 container; // dispositif individuel

    byte[] state; //information courante du capteur avant d'etre analysé
    //dans une lecture réelle de la temperature
    String ROM_ID;
    double degC;
    double degF;
/*
- chaque constructeur appelle la fonction Init qui:
essaye d'utiliser l'adapteur de port par défaut et essaye de trouver
le premier dispositif 1-wire avec un code de famille de 10h à identifier.
*/
    public TemperatureServer() {
        try {
            adapter =
(DSPortAdapter)OneWireAccessProvider.getDefaultAdapter();
            Init("");
        } catch (Exception e) {
            System.out.println("problem in constructor");
            System.out.println(e);
        }
    }

    //PAadapt: DS1410E, DS9097U, TINIEExternalAdapter
    //Port: COM1, LPT1, /dev/ttyS0, or serial1
    public TemperatureServer( String PAadapt, String Port) {
        try {
            adapter =
(DSPortAdapter)OneWireAccessProvider.getAdapter(PAadapt,Port);
            Init("");
        } catch (Exception e) {
            System.out.println("problem in constructor");
            System.out.println(e);
        }
    }

    public TemperatureServer( String ROM, String PAadapt, String Port) {
        try {
            adapter =
```

```

(DSPortAdapter)OneWireAccessProvider.getAdapter(PAadapt, Port);
        Init(ROM);
    } catch (Exception e) {
        System.out.println("problem in constructor");
        System.out.println(e);
    }
}

public void Init(String ROM) {
    try {
        //beginExclusive() et reset() preparent le bus 1-Wire et empechent
        //d'autres threads d'interrompre les communications courantes
        adapter.beginExclusive(true);
        adapter.reset();

        //la valeur 0x10 demande au bus 1-Wire de chercher les dispositifs
        //appartenant à la famille de code 10h
        adapter.targetFamily(0x10);

        // si on spécifie un dispositif, on le prend sinon on prend le premier
        if (ROM.length() < 1) {
            container = (OneWireContainer10)adapter.getFirstDeviceContainer();
        }
        else {
            container = (OneWireContainer10)adapter.getDeviceContainer(ROM);
        }

        // cast the generic container into a TemperatureContainer
        ROM_ID = container.getAddressAsString();

        //readDevice() permet de lire l'etat du iButton
        //c'est une sorte de CAN
        state = container.readDevice();

        //on transmet alors le champs "state" à la méthode
        //doTemperatureConvert() qui transforme la donnée
        //analogique en donnée numérique
        container.doTemperatureConvert(state);

        degC = container.getTemperature(state);
        degF = container.convertToFahrenheit(degC);

        //pour finir, nous libérons le bus 1-Wire
        //de la commande exclusive avec endExclusive()
        adapter.endExclusive();

    } catch (Exception e) {
        System.out.println("problem in constructor");
        System.out.println(e);
    }
}

```



```

    }
}

public void measureT() {
    try {
        adapter.beginExclusive(true);
        adapter.reset();
        state = container.readDevice();
        container.doTemperatureConvert(state);
        degC = container.getTemperature(state);
        degF = container.convertToFahrenheit(degC);
        container.writeDevice(state);
        adapter.endExclusive();
    } catch(Exception e) {
        System.out.println("problem in measure");
        System.out.println(e);
    }
}

//l'execution peut se faire comme:
//1- java TemperatureServer
//2- java TemperatureServer DS9097U COM1
//3- java TemperatureServer DS9097U /dev/ttyS0
public static void main(String[] args) {

    //port 80: port des serveurs web, un seul serveur web par machine
    int port = 80;

    TemperatureServer myTherm;
    try {
        if (args.length==2) {
            myTherm = new TemperatureServer( args[0], args[1] );
        }
        else {
            myTherm = new TemperatureServer();
        }

        //la classe ServerSocket sert à recueillir les requêtes
        //de connexion de clients
        ServerSocket srv = new ServerSocket(port);

        Date currentDate;
        String currentTime;
        while (true) {
            currentDate = new Date();
            currentTime = currentDate.toString();

```

```

//la méthode accept de la classe ServerSocket réceptionne
//la requête de connexion et retourne une socket,
//instance de la classe Socket pour gérer la communication
//avec le nouveau client.
        Socket mySocket = srv.accept();

        myTherm.measureT();
        System.out.println("Connection Accepted @" + currentTime);

        //on ouvre un flot de sortie sur la socket
        BufferedWriter serverResponse = new BufferedWriter(
            new
OutputStreamWriter(mySocket.getOutputStream()));
        writePage(serverResponse,
currentTime,myTherm.degC,myTherm.degF,myTherm.ROM_ID);
        mySocket.close();
    }
} catch (IOException e) {
    e.printStackTrace();
    System.out.println("There is an IOException in HttpSocketServer");
}
}

public static void writePage(BufferedWriter wr, String ct,double degc,double degf,
String rom) {
    try {
        wr.write("HTTP/1.0 200 OK\n");
        wr.write("Content-type: text/html\n\n");
        wr.write("<HTML><HEAD>\n");
        wr.write("<TITLE>Hello Web Client!</TITLE>\n");
        wr.write("<H3><CENTER>CAPTEUR DE TEMPERATURE
DS1920 de Dallas-semiconductor!</CENTER></H3>\n");
        wr.write("</HEAD>\n");
        wr.write("<BODY><CENTER>\n");
        wr.write("la date est" + ct);
        wr.write("The device ROM ID: " +rom);
        wr.write("la Température en deg celcius" + degc);
        wr.write("la Température en fahrenheit\n" + degf);
        wr.write("</CENTER></BODY></HTML>\n");
        wr.flush();
        wr.close();
    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("There is an IOException in writePage");
    }
}
}

```