
DISTRIBUTED One-WIRE TEMPERATURE MEASUREMENTS USING EMBEDDED SYSTEMS

Graduate Program on Instrumentation Physics

Master Thesis

SURYA DARMA

630202022X

Supervisor: Dr. rer. nat. Martarizal



**DEPARTMENT OF PHYSICS
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
UNIVERSITY OF INDONESIA
2005**

ENDORSEMENT SHEET

Research proposal with the title "Distributed One-Wire Temperature Measurements Using Embedded Systems" has been checked and approved as Master Thesis at the Master Program in Physics Department, Faculty of Mathematics and Natural Sciences, Universitas Indonesia.

Depok, October 11th, 2010
Master Thesis Supervisor,

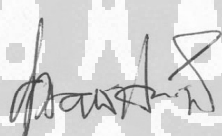


Dr.rer.nat Martarizal

Reviewers,



(Dr. Sastra Kusuma W.)



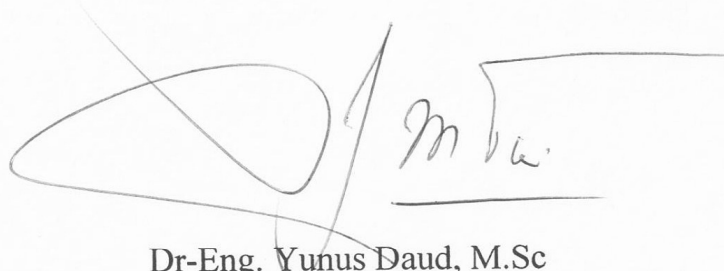
(Dr. Prawito)



(Dr. Muhammad Hikam)

Approved by,

Head of Physics Graduate Program



Dr-Eng. Yunus Daud, M.Sc

A distributed system is a collection of independent computers that appears to its users as a single coherent system.

A. S. Tanenbaum, M. Van Steen ^[01]

Preface

This work investigates the use of embedded systems implementing several interconnected data acquisition points which simultaneously collect temperature data and then display them to the end user. Monitoring of the overall system can be done through Internet access from any computer, making it a distributed system conforming to Tanenbaum's definition.

Acknowledgements

I would like to thank several people who contributed to this work in special ways. First of all, I thank Dr. Ulrich Raich, who gave me the idea to take this topic when he was explaining graphical user interfaces and connectivity with legacy equipment at the Second Workshop on Distributed Laboratory Instrumentation Systems in Trieste, Italy. I also want to thank him for his willingness to become my advisor.

The analysis of my work by Dr. rer. nat Martarizal and his valuable comments are greatly acknowledged. I warmly thank him for being a supervisor of my thesis. I am grateful to Rachmat Widodo Adi, Ph.D for sharing some of his insights with me, being a great friend who dedicated time to revise my thesis, and for many unforgettable discussions on some general concepts of instrumentation.

My great thanks also acknowledge to Prof. Dr. Djarwani S Soejoko for her support to give me an extended time to finish this thesis, and Dr. Dedi Suyanto as chairperson of the graduate program who participate in supporting the extended time. I also sincerely thank Prof. Dr. Carlos Kavka and Dr. Paul Bartholdi for their lectures during the Second Workshop on Distributed Laboratory Instrumentation Systems, from which I used some of their example programs and included them in this project. I thank my colleagues at the University of Indonesia for fruitful discussions and a lot of fun: Iwan Sugihartono, R Bagus Suryasa, Lutfi Rohman, Eko Ketut, Arief Fitrianto as well as my former colleagues Evi Ulina Margaretha S, Handoko Hutapea. I especially thank my colleagues Asido Patar Nainggolan and Oscar R Tillmans, for numerous good advices, proofreading drafts of this work, and a great cooperation during many years.

Special thanks to the entire of my family. This work is dedicated to *Riani* and *Muthia*, who gave me love and many encouragements for finishing this thesis. Without their support, I believe this work could not be finished.

Surya Dharma

Depok, June 30, 2005

Abstract

In this paper, I describe a recently developed experimental setup for monitoring temperature, which can be accessed via Ethernet network. The experiment consists of three primary elements communicating with each other: *i)* an embedded system acting as a *server*. It consists of an embedded Ethernet controller and a 1-Wire sensor for temperature measurement which is called TINI and *ii)* a *second* embedded 1-Wire temperature measurement system which is build on the ATMEL AVR micro-controller and *iii)* a PC based client computer, for monitoring the overall process. The AVR embedded system whose programs were written in the C programming language, sends data to the TINI using serial communications, while the TINI, which executes Java programs communicates to PC using the TCP/IP protocol.

The client computer plots the data sent to it through TINI and provides a visual display of a sequence of temperature measurement data, which were detected by sensors with 1-Wire technology on either of the two embedded systems.

Key Words: AVR micro-controller, TINI, embedded system, C and Java programming language and 1-Wire technology.

vii + 75 halaman, gambar, tabel, lampiran

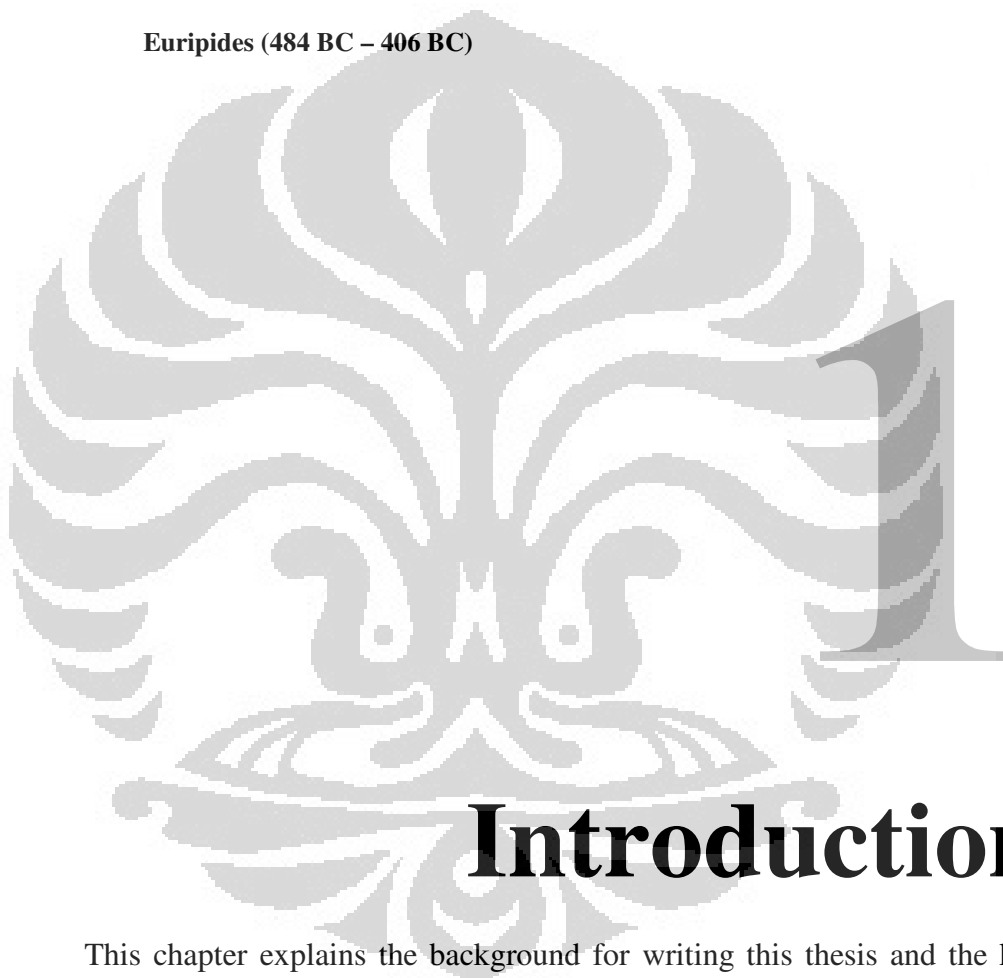
Contents

Endorsement Sheet	ii
Preface	iii
Abstract	v
Contents	vi
1 Introduction	1
1.1 Background	1
1.2 Objectives	4
1.3 Problem Statement	4
1.4 Research Methodology	4
1.5 Thesis Overview	5
2 Basic Concepts: System Design & Assembly	7
2.1 Definition of A Distributed Systems	8
2.2 Distributed Concepts	8
2.2.1 1-Wire® Bus Technology	9
2.2.2 Embedded AVR AT90S8535 unit	10
2.2.3 The Tiny Network Interface (TINI™)	12
2.3 Software Concepts	14
2.3.1 AVR core software	14
2.3.2 TINI® kernel	16
2.3.3 Java implementation in TINI®	20
2.4 Debugging	22

3 Distributed System Architecture	24
3.1 System Overview	25
3.2 Database	26
3.3 System Integration	28
3.3.1 Embedded unit integration	30
4 Results and Analysis	31
4.1 Output of the System	32
4.1.1 The indoor temperature measurement.	33
4.1.2 The outdoor temperature measurement.	35
4.2 System Test	37
4.3 Error Analysis	38
5 Conclusions and Future Works	40
Reference	42
Appendix A: Supporting Source Codes	44
Appendix B: Main Source Codes	64

“A bad beginning makes a bad ending.”

Euripides (484 BC – 406 BC)



Introduction

This chapter explains the background for writing this thesis and the key reason for choosing the topic. It also outlines the structure of the document.

1.1 Background

Micro-controllers are used in low-cost embedded systems that control and monitor consumer appliances, robots, machinery, etc. Micro-controllers are

widely used in applications that necessitate computing power delivered within a small form factor, in e.g., cellular phones, calculators, digital wristwatches, etc. Because of dimension and cost limitations, micro-controllers have limited connectivity capabilities. For example, the type of connectivity provided for data communication to a personal computer (PC) user for data visualization and parameter adjustment “on-the-fly” is usually limited to a serial port interface. This serial interface is restricted in distance and depends much on the operating systems, which allow only one user to control the micro-controller in a time. On the other hand, some modern embedded systems include more elaborate communication interfaces such as Ethernet access, instrumentation buses and multitasking OS. Typical examples are the Rabbit Core and the Tiny Network Interface so called TINI. The TINI allows us to run programs in the Java environment which, until now was rarely used in the micro-controller based applications. Since it was launched in the late 2000, it has made tremendous breakthrough in embedded system applications. Tini’s on board Ethernet controller gives remote Computers easy access to micro-controller resources through the ubiquitous Ethernet data communication network, and allows developers and end-users to monitor and control micro-controller operated devices with great flexibility. Such a system makes any data logger easier to receive, by connecting the network with other media to transmit the data we can get a distributed measurement, especially in a very difficult area to reach simultaneously.

In recent years, the Ethernet network protocol has been widely adopted as the method of choice for data communication on personal computers and other digital devices. Due to the immense use of the Internet which is connected to the Ethernet network makes its popularity incredibly high, an information exchange that communicates data via the Ethernet network, by the general public infrastructure. Furthermore, Ethernet communication is readily available on most of the currently deployed PCs for a very cheap price. As a data communication protocol, the Ethernet is efficient. In all Ethernet networks, devices can easily communicate at speeds of 10 or 100 megabits-per-second, with some of the most recent Ethernet networks communicating even at data speeds of 1 gigabit-per-second.

In this thesis, I address issues of the use of communication between embedded systems and the PC: RS-232 and network, in getting data acquisition with the distributed 1-Wire temperature measurements. The temperature sensors use the 1-wire multi-drop serial bus. 1-Wire communication provides the capabilities to recognize devices by use of unique device identifiers data transmission using the “one-wire protocol” on a single data-line. For a certain application, power supply for sensor can also be delivered in the same wire.

TINI provided an operating system kernel, which allows users to execute multi-threaded applications making it ideally suited as a server. The embedded system is used to acquire data from the physical quantity and provide the data to the external world. These data can then be accessed through Ethernet and, providing a Web server and the necessary protocols, they can be accesses through

a Web browser on the Internet. All of these concepts will be used in this thesis to realize distributed temperature measurements and display the result on the end user's terminal.

1.2 Objectives

This thesis has the following objectives:

1. Exploring distributed system aspects with a simple demonstration system.
2. Implementation of network nodes using TINI and how to use them in a temperature data acquisition system and also as a server collecting data from other embedded system applications.
3. The design of embedded systems which involved 1-Wire technology on the AVR micro-controller.
4. Management of a distributed system.

1.3 Problem Statements

The thesis focuses on TINI system installation, network node programming and interfacing of TINIs to external embedded systems as well as access to TINI servers through PC based computers via the Ethernet. All these computers are then integrated into a single distributed data acquisition system. On the embedded system side temperature data acquisition programs are described while on the PC side application programs with comfortable graphical user

interfaces are provided. The PC is used as an end user terminal controlling the whole distributed system.

1.4 Research Methodology

Before writing this thesis, I had to fully understand the concepts of the entire systems. In first place I had to define the entire problem, subdivide it into manageable sub-tasks and map these tasks onto the distributed hardware. To understand the concepts, implementations and to see the development in this field I consulted system documentation and journals as references. Based on the previous work I then designed hardware and software for the embedded systems, then implemented it. In parallel, I developed the program to be downloaded into the TINI. Some design errors were detected and fixed. Some problems on the AVR embedded system could be diagnosed by connecting it directly on the PC (instead of TINI). This allowed me to use known and proven software tools for hardware debugging.

Once the hardware was corrected I wrote the software for the individual embedded systems and the end user PC and finally integrate the bits and pieces into the final distributed system. Some laboratory measurements were done in order to prove the correct functioning of the system. The experience collected during the whole process as well as the final results have been compiled into this report.

1.5 Thesis Overview

The thesis is organized as follows. In Chapter 1 some introductory background material is given and the problem is explained. Chapter 2 contains some fundamental concepts used in the implementation. In this chapter the concepts will be divided into several sections regarding to the topic covered.

Chapter 3, I describe the embedded design hardware development and environment then the software environment, respectively, used in this thesis. In this chapter I also write the result of modified concept according to the experiences in the laboratory. In Chapter 4, I provide the experimental data measured with the partial final version of my hardware and software. Finally, some concluding remarks and suggestions are given in Chapter 5. Support materials and the source code of my programs are given in the appendix.

*“Prediction is very difficult, especially
about the future.”*

Niels Bohr (1922)



Basic Concepts Systems Design & Assembly

This chapter introduces some terminology used in this document, provides definitions of the distributed systems and how does it works, highlights several important concepts, discusses how distributed system can be developed, and

mentions applications that benefit from the distributed systems. And also the fundamental of the embedded system involved.

2.1 Definition of A Distributed System.

A distributed system is a collection of independent computing devices that appears to its users as a single coherent system. Computing devices can consist of any type technology, any type of CPU, any computing power etc., as long as they can communicate. The only requirement is that the individual computing nodes are linked together in such a way, that the user sees a single coherent system. In this experiment I use two types of embedded systems:

- A micro-controller board with an AVR ATmega90S8535 which provides a 1-wire interface through a single bi-directional I/O line and which is connected to a TINI through a RS-232 serial interface.
- A TINI used as a bridging node, giving access to the AVR system through Ethernet and having a 1-wire interface itself, to which a 1-wire temperature sensor can be connected.

2.2 Distributed System Concepts

Even though all distributed systems consist of multiple CPUs, there are several different ways the hardware can be organized, especially in terms of how they are interconnected and how they communicate. In this section we will look briefly at distributed system hardware in used, in particular, how the machines are

connected together. Various classification schemes for multiple CPU computer systems have been proposed over the years, but none of them have really caught on and been widely adopted. For our purposes, we consider only systems built from a collection of independent computing systems.

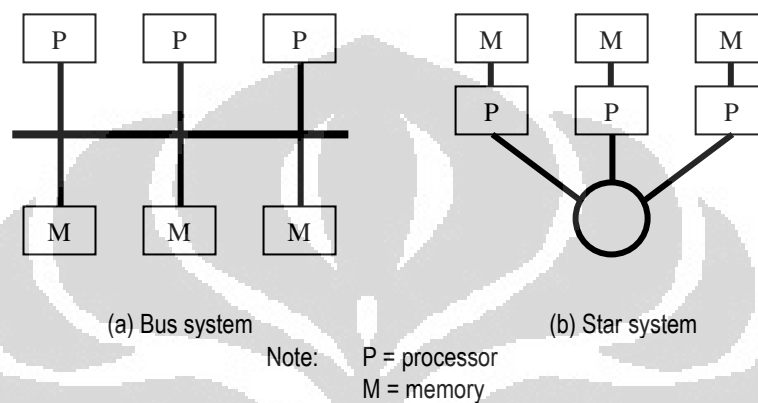


Figure 2.1 Different basic organizations of processors and memories in distributed computers systems.

In Fig. 2-1, we divide all computing devices into two groups: those that have shared memory, usually called multiprocessors, and those that do not, sometimes called multicomputers. The essential difference is this: in a multiprocessor, there is a single physical address space that is shared by all CPUs. If any CPU writes, for example, the value 44 to address 1000, any other CPU subsequently reading from *its* address 1000 will get the value 44. All the machines share the same memory. In contrast, in a multicomputer, every machine has its own private memory. After one CPU writes the value 44 to address 1000, if another CPU reads address 1000 it will get whatever value was there before. The write of 44 does not affect its memory at all. A common example of a multicomputer is a collection of personal computers connected by a network.

2.2.1 1-Wire® Bus Technology

The 1-Wire® protocol, many years ago was originally designed for communication with nearby devices on a short connection. One simple application at the time was the way to add auxiliary memory on a single microprocessor port pin. 1-Wire bus means of using only one line connection to communicate with other device in addition to ground using the 1-Wire protocol. Power supply also can be delivered in the same line (Figure 2.2).



Figure 2.2 1-Wire bus.

This type of bus allows only one master and max 8 slaves attach on the one line. Communication between master and slaves must follow the 1-Wire protocol with a very strict timeslot^[02].

2.2.2 Embedded AVR AT90S8535 unit

In building the embedded systems for temperature measurements, we used 1-Wire technology developed by Dallas Semiconductors then processing it with the low-power and low-cost AT90S8535 CMOS micro-controllers produced by ATMEL. The AT90S8535 builds on the basis of the AVR RISC architecture technology, run the instructions in the 1 MIPS per MHz which makes it easy to optimize power consumption versus the processing speed^[03].

Figure 2.3 shows the simple board connectivity of AT90S8535 with supporting components. In this experiment, we deliver power on the 1-Wire line instead of using the external power attaches directly to the sensor.

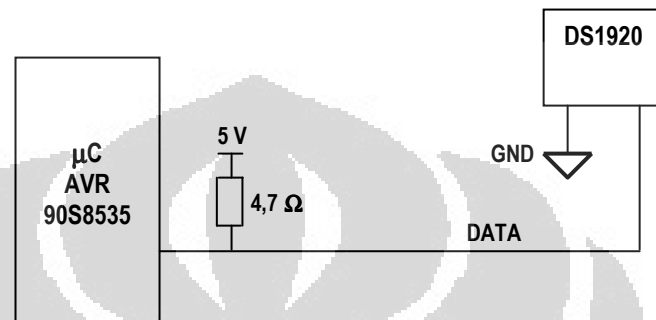


Figure 2.3 AVR system connectivity; DS1920 to micro-controller

On the system, the 1-Wire bus placed on pin PC.0 of the micro-controller. To give enough current feeding slaves, we provide the pull-up resistor attach on bus.

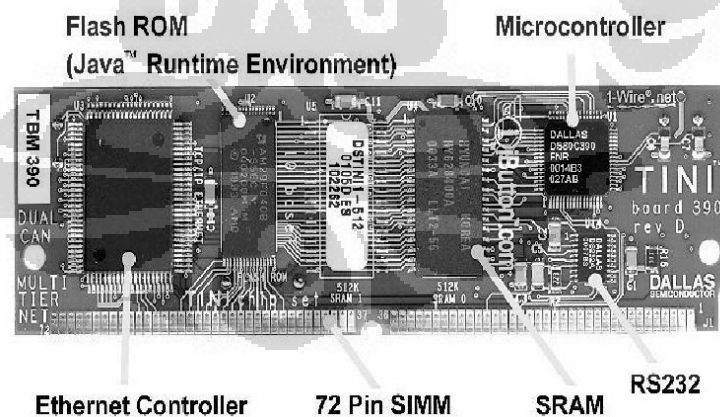
The 1-Wire master then have to initiate signals to start communicates with the slaves ^[02]. First master sends a reset pulse (480 µs low level signal) and then slaves will respond with a presence signal (60 µs low level signal within 60 µs after the bus release by master). After completed the presence time slot, master sends a ROM command which effectively addresses one or several slave devices. The last on the sequence, master send the memory command to access the information provide from the sensor.

To program the AVR, we use the CodeVisionAVR IDE which we developed the software in C. The IDE provide all the library, header and methods for AT90S8535 to communicate with the 1-Wire protocol. The

methods provide in the form of function which we can include it in our program. The program will be provided in section 2.3.

2.2.3 The Tiny Networks Interface (TINI™)

The TINI board model 390, commonly known as the DSTINI1 reference board, is a complete TINI hardware reference design. The DSTINI1 is currently available with either 512kB or 1MB of NV SRAM, in addition to 512kB of flash. It is available as a 72-pin SIMM module and is shown in Figures 2.4^[04]. The DSTINI1 module based on the DS80C390 processor, which could run Java™ programs using the TINI 1.12 firmware. This module can then be attach on the Eurocard (an extension to all peripheral its support). By connecting them we get features consisted of the TCP/IP connectivity, serial and parallel communications, SPI, CAN, I2C and iButton data logger.



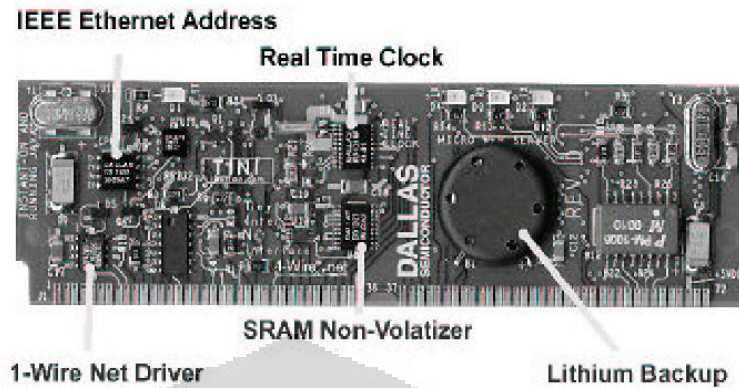


Figure 2.4 TBM 390 72-SIMM TINI (double side) [Courtesy of Dallas Semiconductor corp.].

Because of these feature availability we can use it as a server in collecting data from several embedded systems.

The TINI diagram is shown in Figure 2.5 shows the overall architecture block connectivity and it's interfacing to outside world. The memory is loaded with the kernel which includes the RTOS, TCP/IP Stack, JVM™, and Java program applications. On the other side, TINI is completed with the availability of RTC which produce the exact time generated. The Ethernet controller will give the conversion of the digital data into the right format of internet protocol.

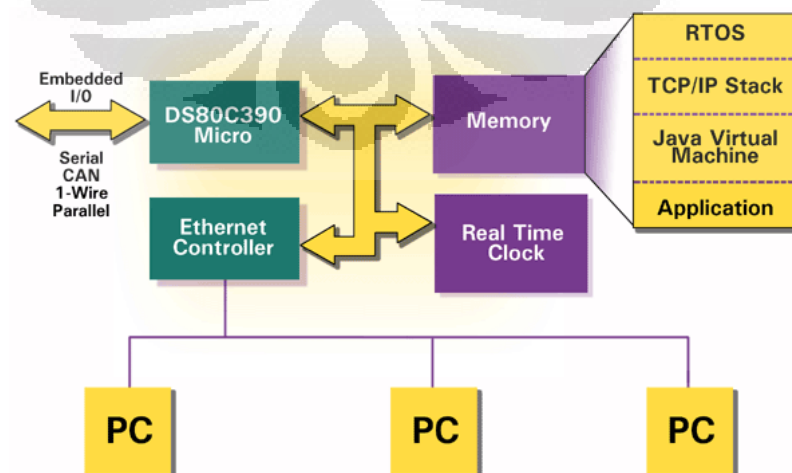


Figure 2.5 The TINI™ diagram; internal block connectivity and outside interfacing. [Courtesy of Dallas Semiconductor corp.]

We implement serial communication as the connection between the embedded systems with the TINI. The serial have the protocol 8 data bits, 1 stop bit with no parity at 9600 bps. AVR unit is listening to serial line until it receive the letter S, then it detects the temperature and send the data with the iButton string address to the TINI.

2.3 Software Concepts

Hardware for distributed systems is important, but it is software that largely determines what a distributed system actually looks like. In this section we provide the explanation of software concept implemented.

2.3.1 AVR core software

In recognizing the 1-Wire sensor the CodeVisionAVR provide the header file `1wire.h`, `Ds1820.h`, `90s8535.h`, `Stdio.h` and `Delay.h`. Meanwhile the entire search program is provided by sub unit library; `Ds1820.lib` and `Stdio.lib`. In the other hand to arrange the sequence we put program `surya.c`. The C source code will be discussed in this section, meanwhile header and library codes provided in appendix A.

As a general idea of the software implemented by the CodeVisionAVR to read the information on sensor, we provide the algorithm in the figure 2.6 below.

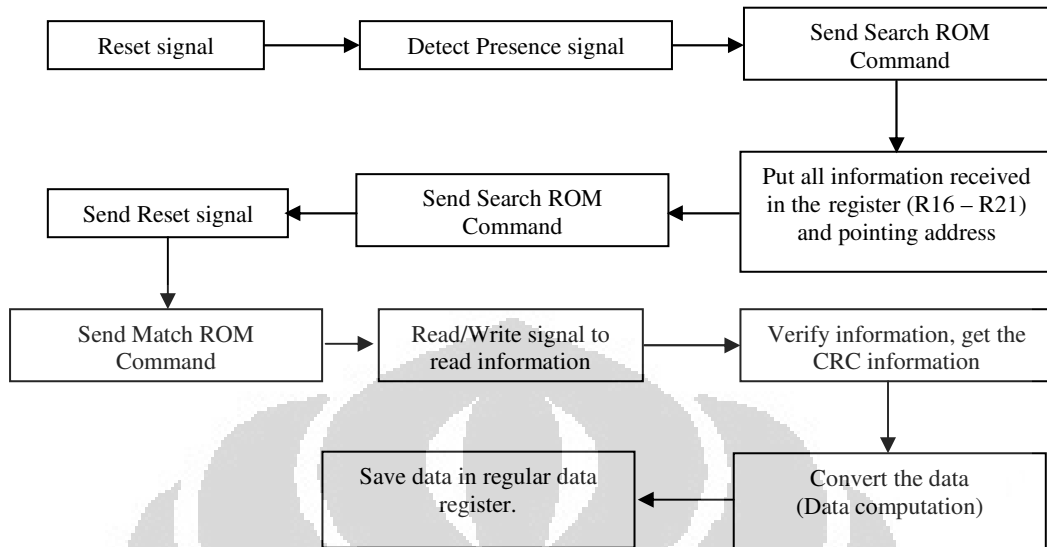


Figure 2.6 Flowchart to read the information on the 1-wire sensor.

In order to get the sequence of reading information within 1-Wire bus and since the bus is connected to pin PC.0 on the board then the initialization of port in used is arranged as follow:

```

.equ w1_port=0x15 //Port C
.equ w1_bit=0 //PC.0
  
```

The 0x15 determine the port C address at AT90S8535 which declare in the header file 90s8535.h and bit 0 will refer to the PC.0 pin connection. In adapting the properties of DS1920 which is the 1-Wire systems, we are using the same configuration of DS18B20 definition file (Appendix A). These properties then hook up into the program by putting “#include <ds1820.h>”.

Inside the header ds1820 declared the Dallas Semiconductor DS1820 1-Wire bus functions; w1_init(), w1_write(), w1_read(), w1_search(), ds_1820_select(), w1_dow_crc8, w1_search(), etc. In the header, author classified the function in the specific family code. Since this header is only provide information for family of 0x10 then the use ROM command will be

more efficient. In regard to access the data then we have to search for the 9 ROM code sequences then write it in the buffer and recognize the sensor from the first 8 bytes. The search process shown as a partial program follows:

```

devices = wl_search(SEARCH_ROM,&rom_code[0][0]);
printf("%u device(s) found\n\r",devices);
for (i=0;i<devices;i++){
    printf("Dallas #%u serial number:",n++);
    for (j=1;j<=8;j++)
        printf("%02X",rom_code[i][j]);
    printf("\n\r"); };

```

This program intends to recognize the family number or the serial number that lasered into the sensor semiconductor. In the first line we put the search method (which is included in the header file) for detecting the code for the ROM initialization to recognize the family or the device number and we make the looping procedure to locate if there are a number of other 1-Wire sensors. The *printf* command will print the result at the standard output every 1-Wire sensor detected in the bus connection.

In detecting the temperature measurements from sensor we used the program as shown below:

```

while (1)
{
    for (i=0;i<devices;i++)
    {
        temp=ds1820_temperature_10(&rom_code[i][0]);
        j='+';
        if (temp<0)
        {
            j='-';
            temp=-temp;
        };
        printf("t%u=%c%i.%u\xdfC",++i,j,temp/10,temp%10);
        delay_ms(800);
    };
};

```

We make loop for every devices to print out the measurements readings. We have also put the minus sign if the reading of temperature is less than zero which is done by checking the temp variable where the temperature data is stored. We put delay of 0.8 second in order to get good reading for each iteration. If we put the delay less than 0.5 second, we sometimes gain an error measurement reading.

2.3.2 TINI® kernel

In order to get the TINI® running, we need some requirements before using it. First of all we need to download the kernel as a bootloader of this device. To be able downloading this lootloader we will need the java environment which in this case we are using Java development environment 1.5.0_05 with netbeans Beta5.0, the Java Communications API, and the TINI SDK on the personal computer. The steps required for installing the TINI JavaRuntime Environment depend on the hardware and firmware versions chosen.

After configuring the PATH and CLASSPATH, then install all the equipments as it shown in the figure 2.7 and run the JavaKit from the tini1.1x.

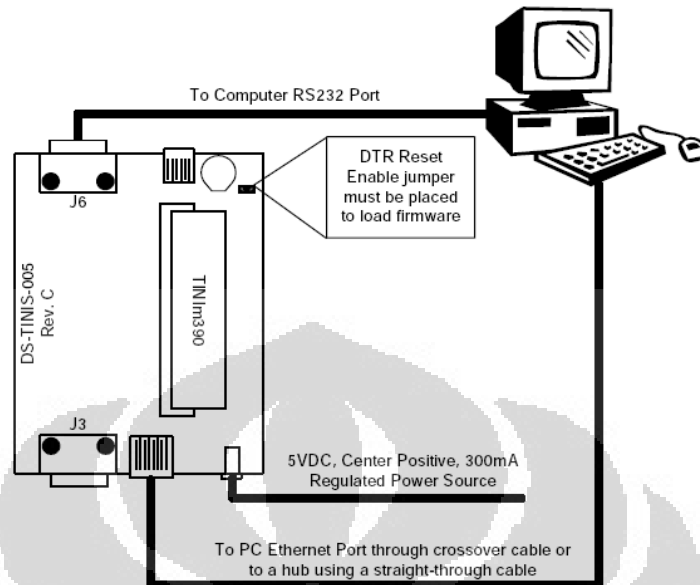


Figure 2.7 TINI connection to PC in downloading the slush

In JavaKit, we select the COM port which will be used for the serial I/O. Then we have to select the speed as 115200 as shown in figure 2.8. Press the OPEN PORT button as it mention in figure 2.8.

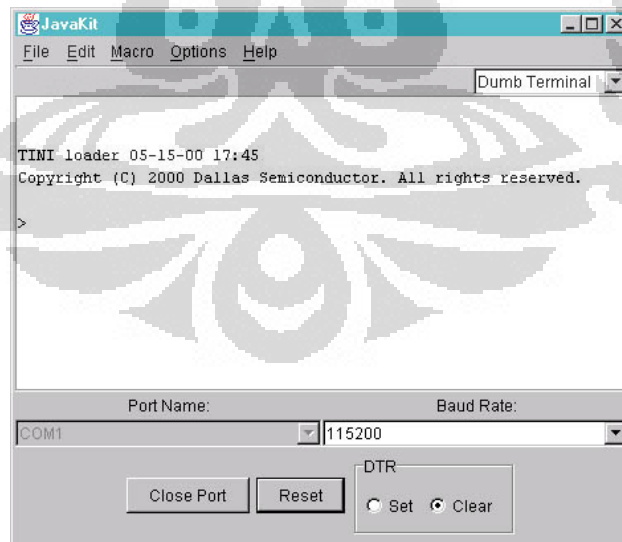


Figure 2.8 The JavaKit environment display.

The next step we should press the RESET button when it becomes enabled. This sends a reset signal to the TINI Board. The reset circuitry of the TINI Board is connected to the DTR signal of the RS232 cable. By pressing reset, JavaKit generates a short pulse on the DTR line so that the board is going into the power on reset internal procedure. This procedure starts the "loader" that allows us to communicate with the board and download the firmware. In the text area above a prompt should appear along with the words:

```
TINI loader 05-15-00 17:45
Copyright (C) 2000 Dallas Semiconductor. All rights reserved.
```

>

Go up to the FILE menu above and select LOAD. Load the following file: %TINI_HOME%\bin\tini.tbin. There is no need to change banks first. The file has the bank information embedded in it. It should report the bank(s) it was loaded in. This will take several seconds.

Next, clear the heap by doing the following:

```
b18 //changes to bank 18
f0 //fills bank 18 with 0's, effectively erasing it
```

Select FILE/LOAD one more time and load the following file: %TINI_HOME%\bin\slush.tbin. If everything is loaded correctly, type 'e' to execute. Then we will see output similar to:

```
----> TINI Boot <----
TINI OS 1.01
API Version 8006
Copyright (C) 1999, 2000 Dallas Semiconductor Corporation
01000000
Doing First Birthday
Memory Size: 07E600
Addresses: 181A00,200000
Skip List MM
L01
```

```

Running POR Code
Memory POR Routines
000020
Transient block freed: 0000, size: 000000
Persistant block freed: 0000, size: 000000
KM_Init Passed

```

```

Ethernet MAC Address Part Found
TTS Revision: 154 , Date: 7/19/00 3:13p
Thread_Init Passed
External Serial Port Init
External serial ports not enabled
Memory Available: 075F00

```

```

Creating Task:
0100
01
Loading application at 0x070100
Creating Task:
0200
02
Application load complete

```

```

[==      slush Version 1.01      ==]
[      System coming up.      ]
[      Beginning initialization...  ]
[      Not generating log file.    ]      [Info]
[      Initializing shell commands... ]      [Done]
[      Checking system files...    ]      [Done]
[      Initializing and parsing .startup... ]
[      Initializing network...    ]
[      Network configurations not set. ]      [Skip]
[      Network configuration      ]      [Done]
[      System init routines      ]      [Done]

[      slush initialization complete.  ]

```

Hit any key to login.

Welcome to slush. (Version 1.01)

```

TINI login: root
TINI password:

```

```
TINI />
```

There are two default accounts on this revision of slush, 'guest' with the password 'guest' and 'root' with the password 'tini'.

On DS80C390-based TINI systems, the boot loader program occupies all of bank 0 (address range 0 to 10000h). Loading the 1.14 firmware then consists of loading the files `tini.tbin` and `slush.tbin`. The file `tini.tbin` loads the TINI firmware and core API from address 10000h to address 70000h. The file `slush.tbin` loads the slush application from address 70100h to address 80000h. The 100h bytes in between the end of the `tini.tbin` file and the start of the `slush.tbin` file is reserved for flash preservation of network configuration information, such as in conjunction with Slush's "`ipconfig -C`" command.

2.3.3 Java implementation in TINI®

In performing specific task in TINI as a bridging and temperature measurement system, we need to develop our own program. Since the system has to communicate between serial ports to the AVR and then send the information through the Ethernet socket, so we have to make the data read from serial line and put it in buffer and then send it from socket connection. We provide the algorithm used in the main java program and follow by some explanation of it. The full source code is available at the Appendix B.

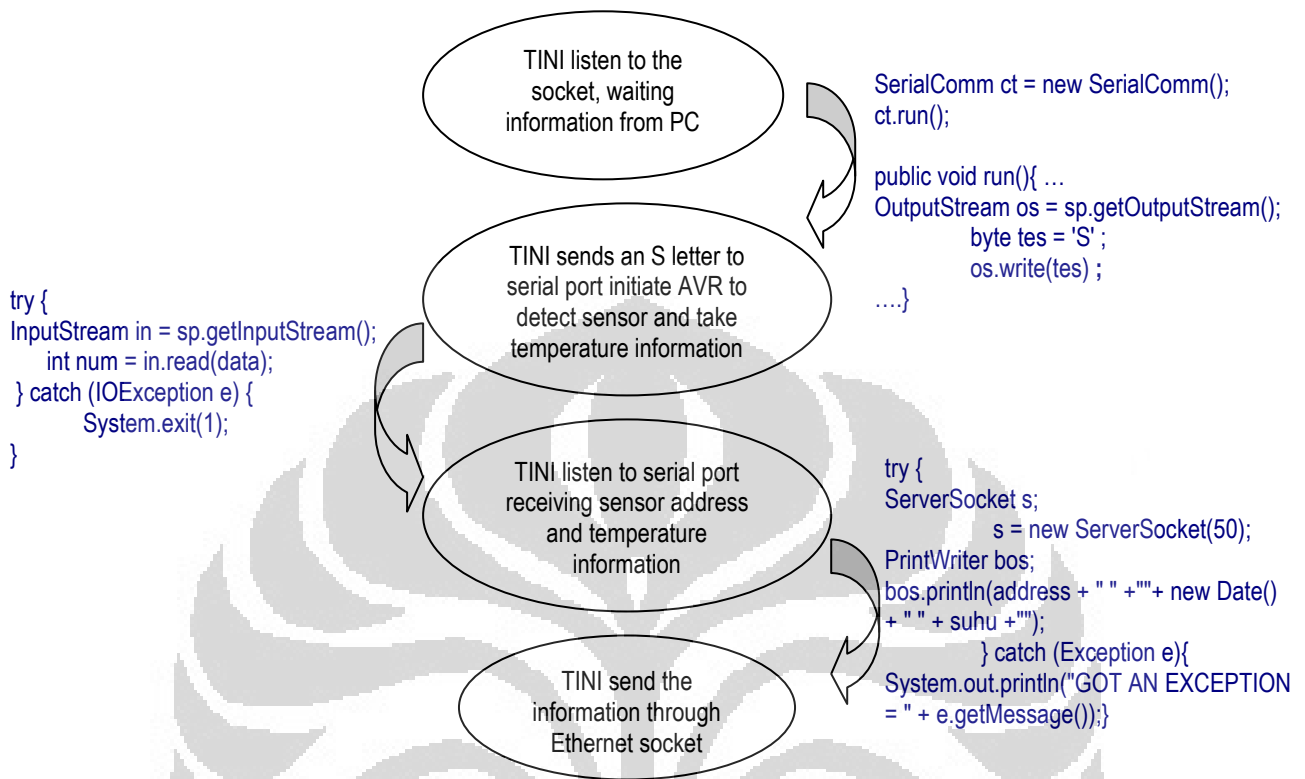


Figure 2.9 Flowchart of java program implemented in TINI.

In figure 2.9 first, the program should be run as a background from the slush with the arguments of port that used to communicate <java Temperature.tini 50 &>. Then we activate the program at the PC side which at the background telnet the TINI passing the arguments <telnet tini 50>. By this signal, then TINI sends the activation letter S to AVR. The AVR communicates to the sensor and takes the temperature information to be sent to TINI through serial ports. TINI which listen to the serial will receive the data and try to send the information through socket. The data which reach the PC will be directed to database. Java program at the PC can execute the query data into a graphical display.

2.4 Debugging

In order to get the system working, we have to be sure that every part of the system is working, before they placed into integration. For the embedded AVR system, we connect the board with the serial port of the PC. On the PC, we run the minicom open source software written by Miquel van Smoorenburg 1991-1995 for serial communication. This minicom has the same functions as the hyperterminal at windows operating system. After the connection established we send the letter S from the minicom and then waiting for the respons. Once we have the response we got the string address of the iButton and the temperature reading.

On TINI side we also connect the serial port of the TINI (serial port when used in loading the slush) to the PC with a null modem cable without connection at pin 4 (DB9). This pin 4 absence will not disturb the DTR signal at the TINI side which can cause TINI to restart. Before we get any information on the serial line we have to activate java program in TINI by typing “java Temperature.tini 50 &”. At the PC, we should telnet the TINI by typing “telnet tini 50”. Then the PC serial should receive the letter S send by TINI. After the letter S has appears we can type 22 characters which are 16 characters of simulated string address, 1 character space bar, 4 digits of simulated temperature measurements and enter (\r). Once the TINI have this information, it will directly send the information with the timestamp given from TINI. If we lose the data at each step, then we have to see

the signal at the serial line using the oscilloscope whether there is something happen or not when we sending or receiving the data.



“A system must be programmed to perform tasks. Different tasks require different programs.”

Cay Horstmann (Comp.Concepts)



Distributed System Architecture

In the previous chapter, we have discussed all aspects of the fundamental concepts of distributed system used in this experiment: the information of hardware and the appropriate software concepts used. In this chapter, we focus on the architecture systems and its connectivity to the database. 1-Wire temperature

measurement concept implemented into embedded systems design which the data will be sent through serial communication to the server. The sequence in getting the data will be described. On the embedded server side (TINI®), the Java program give access to the incoming data through serial port, then process and display it onto the monitor whenever a call program run by the PC. During the way to display it to user, the data then converted to TCP/IP format by the Ethernet controller which makes it accessible remotely.

3.1 System Overview

Several ways have been proposed to improve the use of electronics equipments in a group. The most prominent way to group electronics devices, i.e. computer or TCP/IP basis is to use the wide connection of the internet. But in fact, to connect electronic devices to the internet line is not an easy task to do. In this thesis we report the use of TINI as a bridge the legacy embedded systems to the internet access. In Figure 3.1 we designed the remote monitoring temperature measurements.

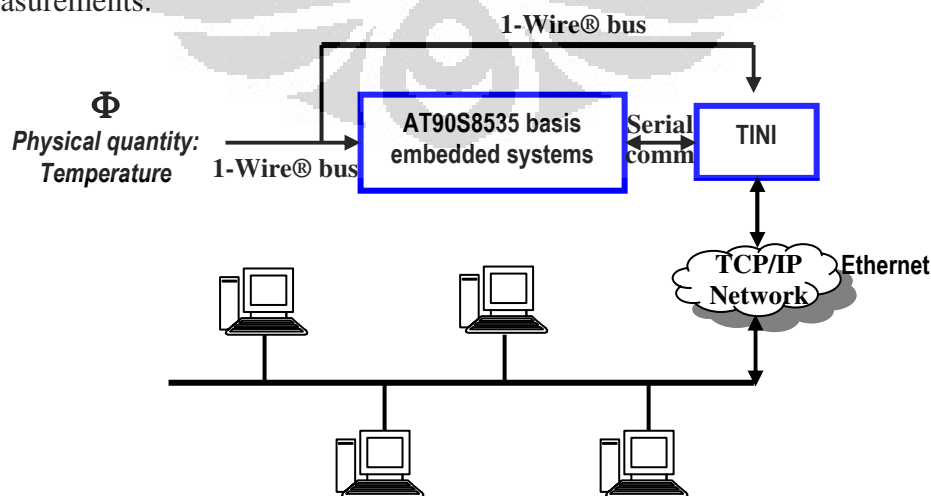


Figure 3.1 System overview; system design and assembly (with box)

The physical quantity measured with the sensor DS1920 (which recognize as iButton™) using the 1-wire protocol. The processor has to be able in searching each ROM value of the device to get the specific identification number of each sensor which will be attach on the data send, to distinguish where the information come from. On this system there are two iButtons located, first at the AVR board which attach to the port C (PC.0) and the second attach on the TINI board. The AVR board then sends the temperature reading with the identification number of iButton will be forward to the PC plus timestamp. TINI itself collect the temperature reading from its board and send the same information into PC. In PC both information will be recorded in the same database with distinguish from iButton's identification numbers. Then java program at PC with a specific query can plot the measurement result in a graphical mode.

3.2 Database

Temperature measurement system intends to make a long record duration of temperature change. Usually for getting a precise interpretation from the data, it needs to use a small time interval in between the data which usually difficult to observe. To take over this problem we use database. The database records every measurement done by TINI and AVR board. To distinguish the data between both data, we differentiate them using the iButton string address identification. The database consists of fields which are date, time, identification number (ID) and temperature reading. In order to get the data of temperature reading and

identification, a sequence of action should be performed which take data into database from specific system. In the following we describe the communication system and it's relational with database.

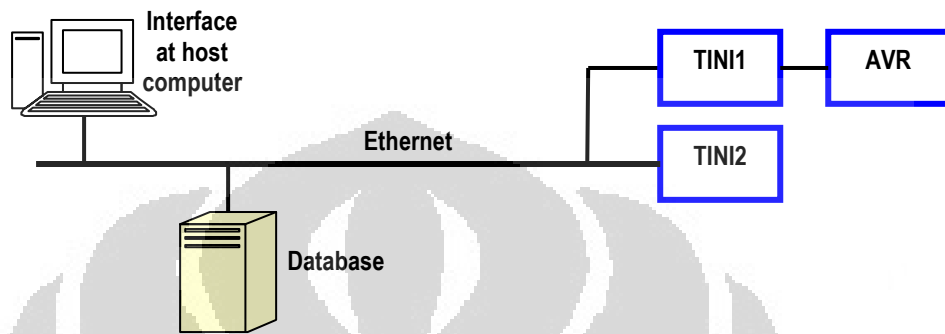


Figure 3.2 Database connectivity

On figure 3.2 we can observe the PC stands as a user interface. The user then run the java program we developed [Appendix B]. To run the program we have to type <java Tini_Client> at the console terminal. Then at screen the output user interface will appear as shown in figure 3.3.

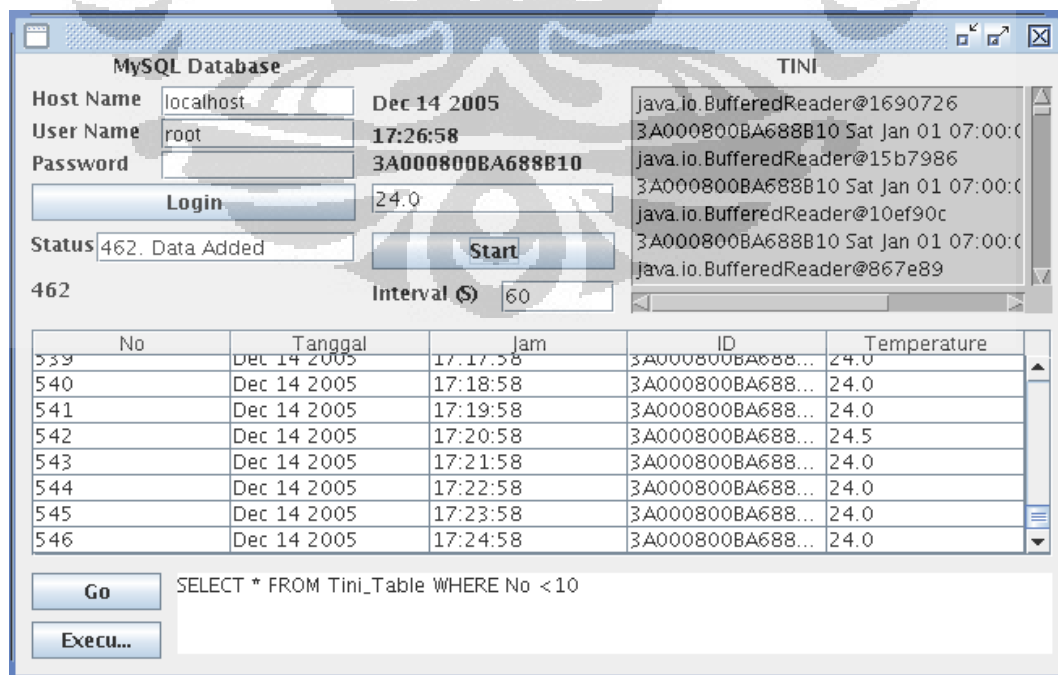


Figure 3.3 User interface for remote measurements

To start access the data, we have to make sure the program in TINI® already activated. Then we have to initiate the interval needed that the data will be taken, in this regards we limit the interval cannot be lower than 60 seconds. This is only our limitation, not an aspect of hardware limitation. Once we have finished, by pressing start we connect into database and ready to fill every field that have made before. On the next step, java runs the telnet program to open a session with TINI and passing the port information onto it, which trigger the java program on TINI to collect the information. This information then forward by TINI to PC during the telnet session and close it after last information transfer is finished. Meanwhile in the java program timer has been activated. It will count for the duration of interval that we inputted in the user interface. After timer reach the value given then the telnet session being activated once more. It will collect the data as it is happen before. This program will last forever if there is no disturbance or interrupt given into PC or systems. To close the connection, simply press the button at the top right corner of the user interface.

3.3 System Integration

So far we have described all parts as a partial stand alone system which we debug it solely using the PC. In this section we describe the integration process of combining the partial elements into the whole system. As discussed before, the entire system will look like shown in the figure 3.1. To get the entire idea of this

system we provide a flowchart of total process in figure 3.4. In the flowchart shown that the initialization of partial system happened in each device itself. It means the process has to be done separately and independent program from the program running on PC.

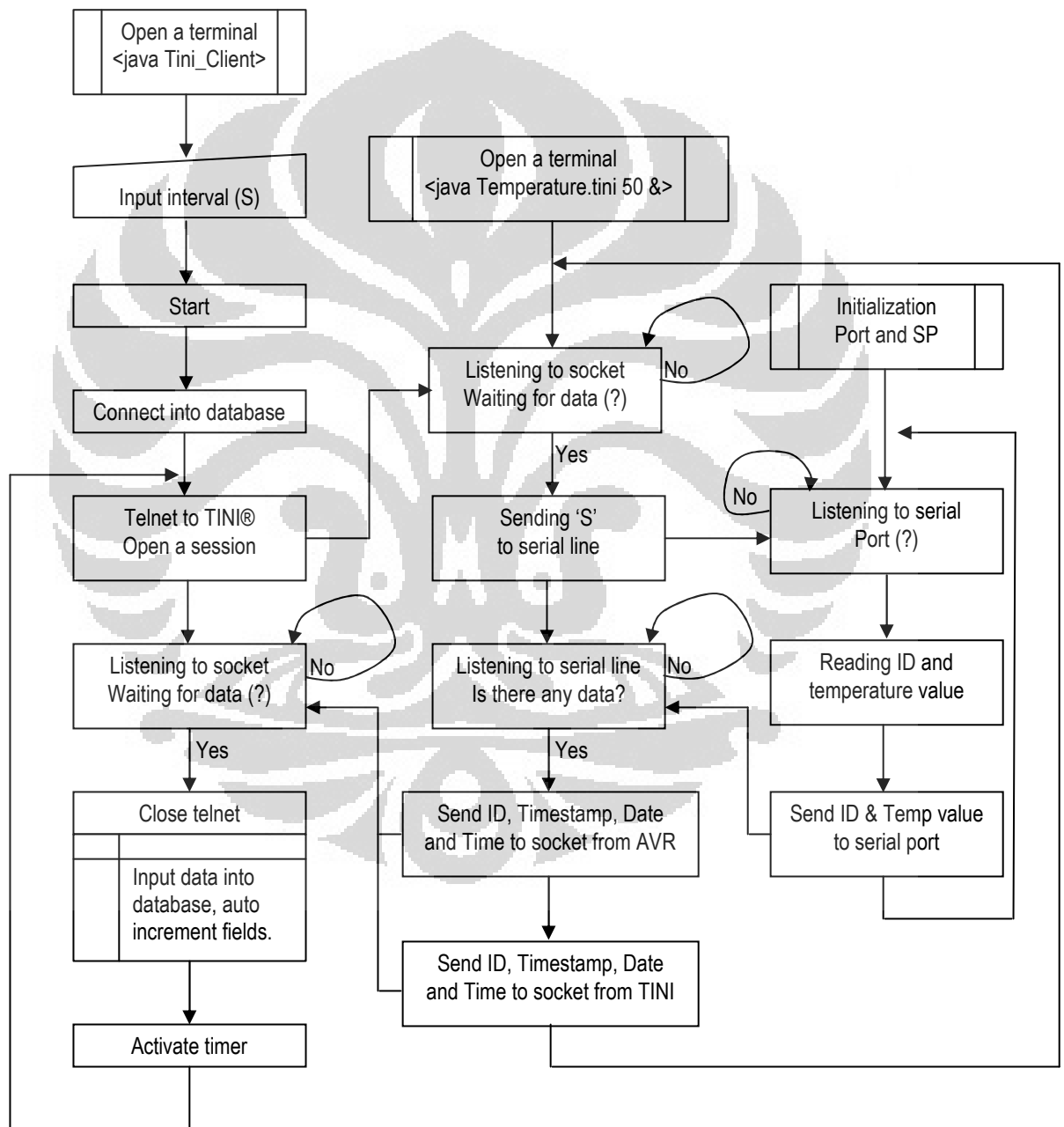


Figure 3.4 Flowchart of the overall process

On the connection, the only database use is TINI_DB, since it is already declared in the program. If we want to put the new data into different database that one should change it in the source code. These processes are in series, therefore the time to run process will be slower than it should be. This is because of the different cycling time among processors. This phenomenon will also occur in the result data.

3.3.1 Embedded unit integration

In the figure 3.5 below, we can see that both serial connections directly connected. On the section 2.4, we debug the TINI with the null modem configuration cable to PC, meanwhile in the design we only connect TxD, RxD and ground pin.

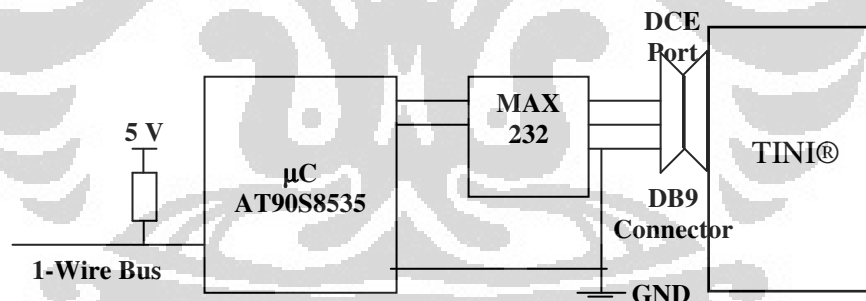
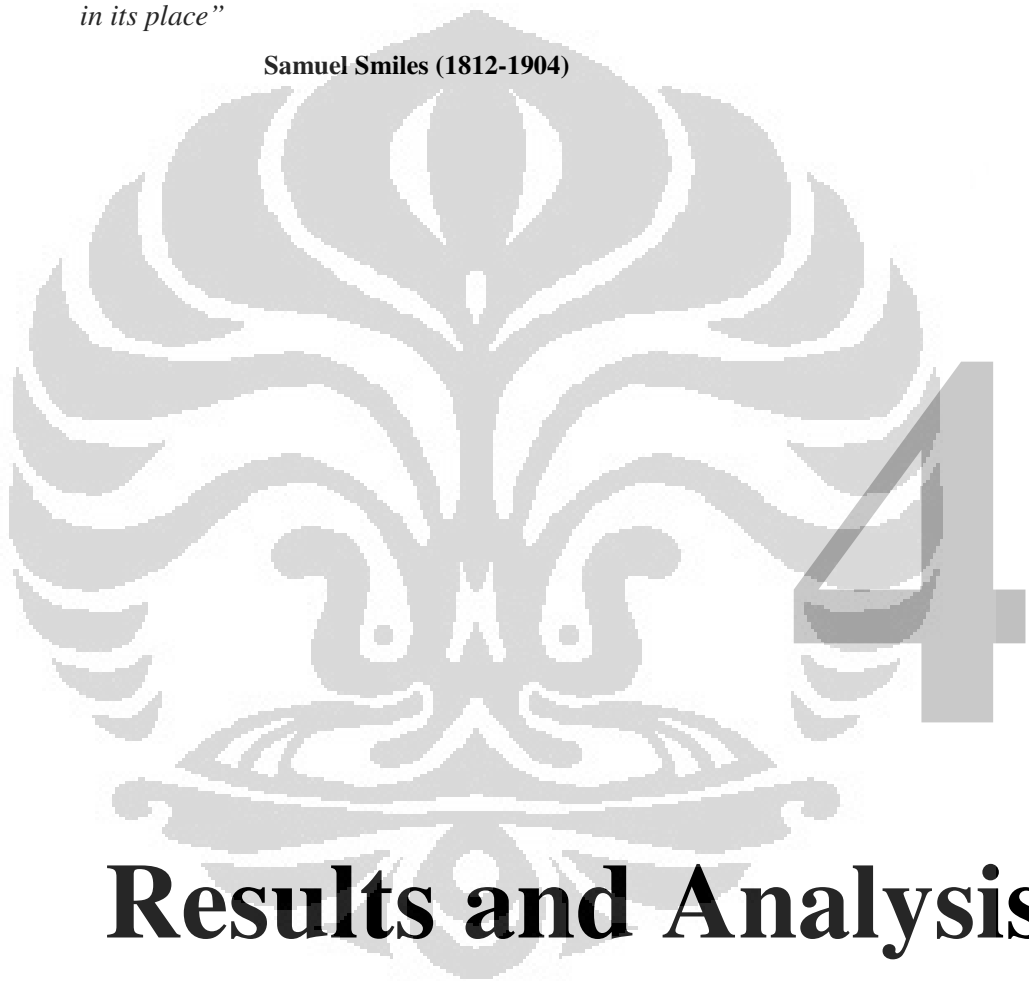


Figure 3.5 AVR – TINI connection

This design is compatible with TINI, by the absence of others pin just give us no problem. We don't have to worry about the reset of TINI although we are using the serial0 line to it.

*“A place for everything and everything
in its place”*

Samuel Smiles (1812-1904)



Results and Analysis

So far we have seen the hardware and software construction of the distributed systems adapted in this experiment. To be consistent with the goal of this research, which is building a system consisted of subsystems that working as a single coherent system, - then we have to evaluate the result produced by the systems. Therefore in this chapter, we provide the data/information obtained from

entire system. In this chapter we also should see the reliability of the system when the application used for long time, the consistence of the result and the analysis regarding to the information. Also we will see the probability of error that unpredictable in organizing the program used.

4.1 Output of the System

In the user interface we monitor the input output process through the buffer reader that displayed on the right boxes shown in figure 4.1.

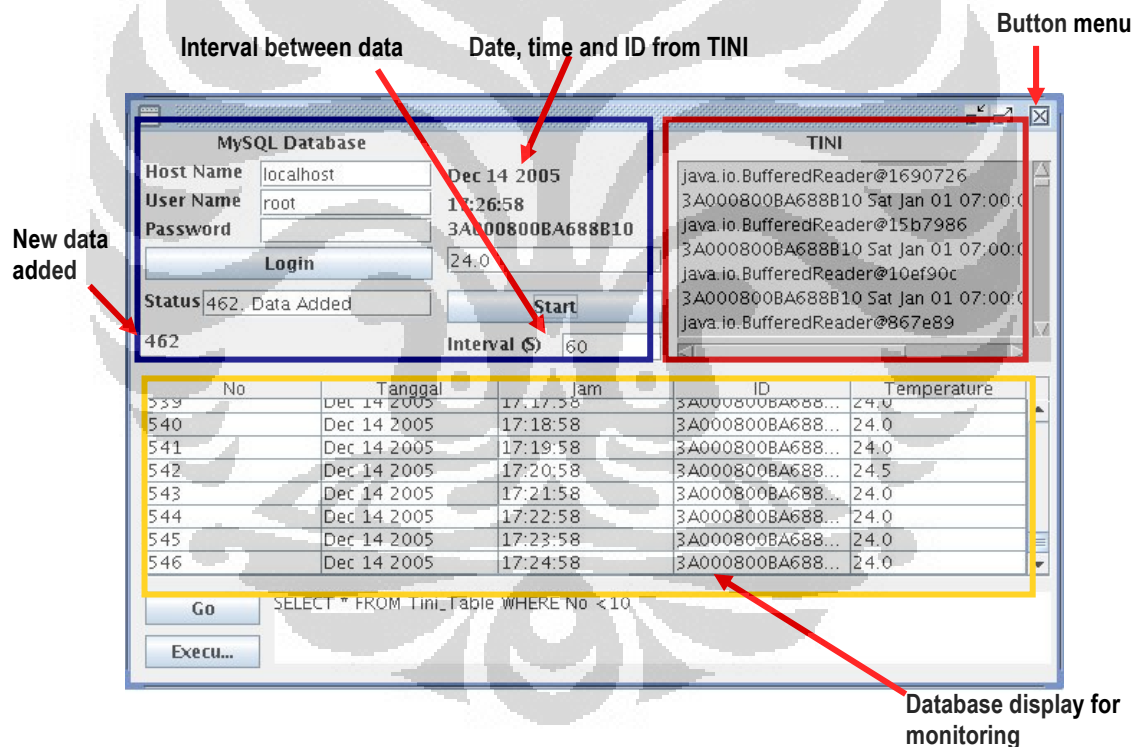


Figure 4.1 Java I/O monitor dialog (with sign) on PC

From this dialog box, we can locate the problem which might occur during the communication between PC and embedded system. If the buffer reader (information in the red blocks) cannot get access to information then we can

assume that the problem should be on the embedded systems. In this regard, we can trace the output of the AVR or TINI machine independently. On the left dialog box (with the blue box), we can find all control panels for accessing the temperature measurements and databases. Also in this dialog box we will get the next time plot for taking new data from the TINI where the time is read also from the timestamp sent by TINI. The next time plot is incremented according to the time interval that we inputted. On the left of that box we will find the number of new data added into the database. This feature is intended to show the cumulative of new data to be compared to whatever number of data we need. On the bottom of the user interface we can see the database display of entry data. We can scroll the data to see the newest and oldest data available. In this experiment, we did temperature measurements at two point of interest. The first measurement was indoor measurement which its temperature influences by the old version air conditioner (without temperature control) and the second measurement was outdoor temperature measurement. Also to be consistent with the temperature measurements, we provide the comparison data with the old style standard temperature measurement system (thermometer Hg).

4.1.1 The indoor temperature measurement

The indoor temperature measurement is done at the Instrumentation and Control Laboratory (LINK), 4th floor of the Department of Physics University of Indonesia. This system is then activated for the duration of 72 hours and the timeslot inputted into system is 60 seconds. So the total of the temperature measurement will approximately 4000 data. To give a simple interpretation on

the data, we then presented the data in a graphical format as shown in figure 4.2. Since it is too many data presented on the graphics then the axis indicator are overlapped. In order to have a nice graphics, we can easily change the iteration number to be displayed in the source code. But if we interest in a specific area, we can point to it and see the number of the data entry which we can see other properties of it in the database.

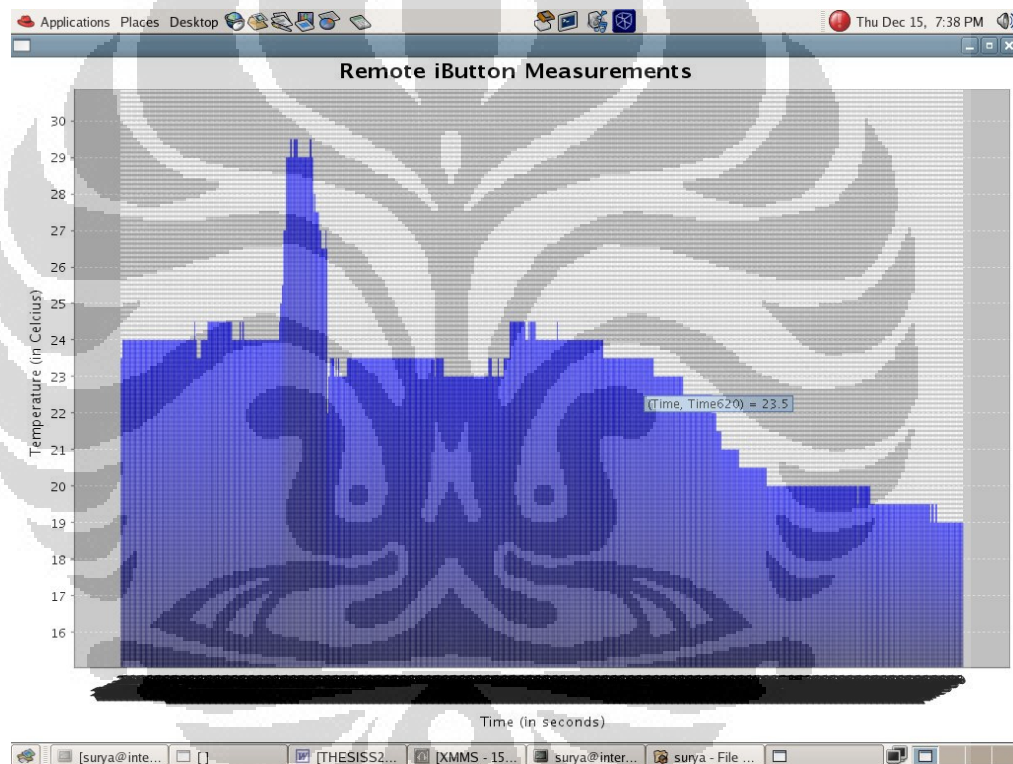


Figure 4.2 Indoor temperature measurement data

From the result at the graphics, we can easily understood that the temperature drops indicate that the day is changing become night which usually colder that daylight. On the peak range around 29°C, indicating that the room temperature is somehow influence by a factor which is in our record this happened because of the air conditioner is off and the system in the beginning of it used. The

system was on for the duration of four hours at that time and then crash in the system because of IP conflict, and then in the morning we trace the problem which we can manage to make the system running again. This is why we get the value which is the differences is too high. By this measurement, we can see that during the daylight we will have the temperature around 23 – 24 °C, meanwhile for the night we will have the temperature around 17 – 19 °C depending on the weather during the day.

4.1.2 The outdoor temperature measurement

Since the indoor temperature measurement is to do, it is a different concept with the outdoor temperature measurement. This is because of the security problem. To maintain indoor measurement is not difficult. We just installed them and let them run accordingly. But for outdoor temperature measurement we need to be on the site for the duration of the measurement. Other factor also we have to count that, the possibility of other physical quantity which influence the system, the most frequently disturb the system is the wind and direct heating from the sun.

As a result of this measurement, we plot the measurement taken on Dec 15th 2005 23 pm until Dec 16th 2005 5.45 am, as shown in figure 4.3. This measurement is to accomplish the previous measurements during the daylight. From this measurement we can see that the temperature around the UI Depok campus is roughly around 29 – 31 °C. Meanwhile in the daylight the temperature is roughly around 33 - 34 °C. But it is also depend on the weather is it rain, windy or not.

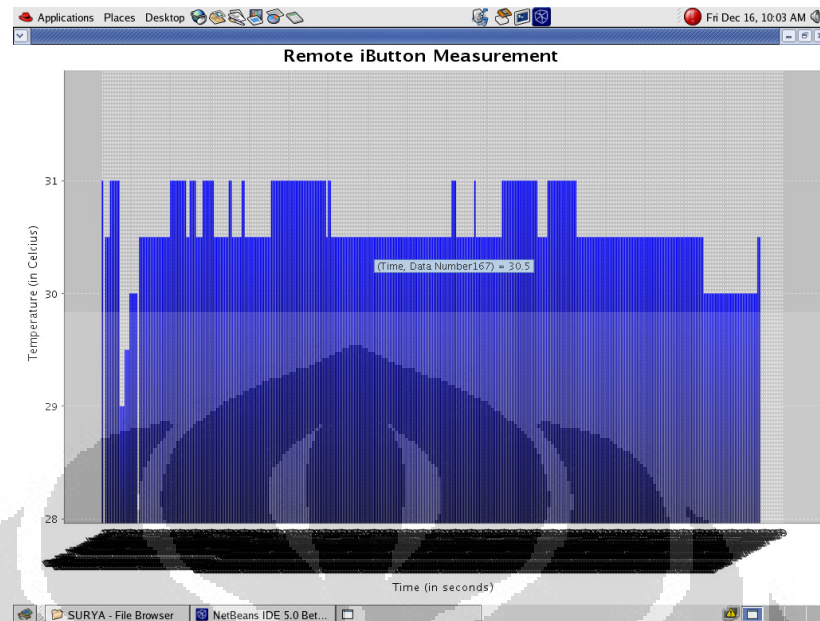


Figure 4.3 Outdoor temperature measurement data

Since the data of outside temperature measurement influence by other factors, we cannot see the pattern of the temperature change in only one day data.

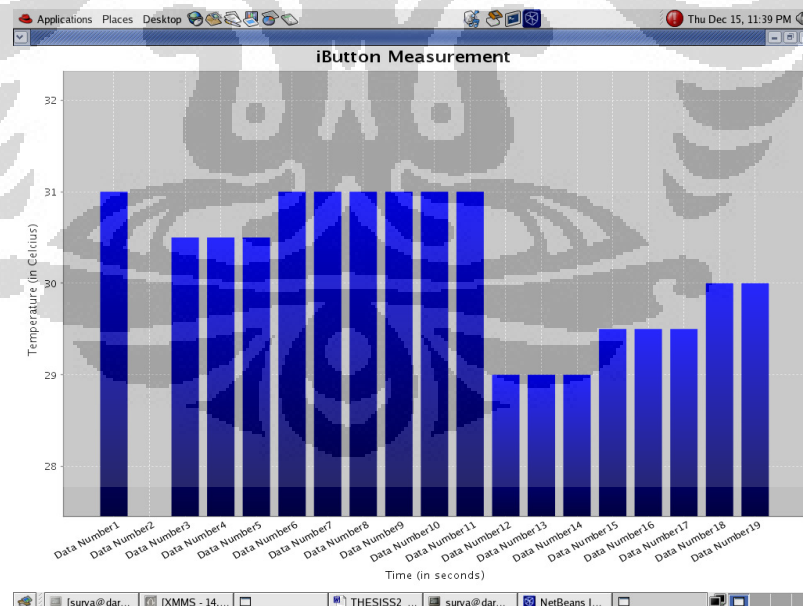


Figure 4.4 A few numbers of outdoor temperature measurement data

To be able to get a right interpretation on this, we should try a longer duration of measurements once we have the safe place to put all the equipments to install.

4.2 System Test

During the measurement, we faced that the system is relatively good. The data taken by the system are stable. The connection method by the telnet connection is very good, since the connection will last until the data is reached. It is shown by the periodically data taken by the system during the same period. When the timer counts, there is no need to access the network. Also when a telnet session is broken then the system will create a new telnet session after the interval of time counted. As a whole datalogger, the system is used for a long period of data measurement; means that some missing data still can be neglected. The available data still can be interpreted nicely.

The data integrity is also in a considerable range. It is because the TINI is a good modular system and the RTC inside the TINI module is quite precise. We can see these phenomena from the error in timestamp which occurs during the measurements; we believe that for the temperature measurements system this thesis experiment can still be adapted. Even though we can see a second change happened in the measurements, but it did not much differentiate the time plot we planned. For example, in the figure 4.5 we can locate a one second change timestamp for the duration of fifteen hours measurements. But sometimes it just can get back to the previous value before it come into a new permanent timestamp.

The screenshot shows a terminal window titled 'surya@interface1:~'. The terminal displays a table with the following columns: Line Number, Date, Time, ID, and Value. A blue box highlights the 'Time' column, and a blue arrow points to it with the text 'Consistent timestamp'.

Line	Date	Time	ID	Value
90	Dec 16 2005	00:28:13	B9000800BA49AB10	30.5
91	Dec 16 2005	00:29:13	B9000800BA49AB10	30.5
92	Dec 16 2005	00:30:13	B9000800BA49AB10	30.5
93	Dec 16 2005	00:31:13	B9000800BA49AB10	30.5
94	Dec 16 2005	00:32:13	B9000800BA49AB10	30.5
95	Dec 16 2005	00:33:13	B9000800BA49AB10	30.5
96	Dec 16 2005	00:34:13	B9000800BA49AB10	30.5
97	Dec 16 2005	00:35:12	B9000800BA49AB10	30.5
98	Dec 16 2005	00:36:12	B9000800BA49AB10	30.5
99	Dec 16 2005	00:37:12	B9000800BA49AB10	30.5
100	Dec 16 2005	00:38:12	B9000800BA49AB10	30.5
101	Dec 16 2005	00:39:12	B9000800BA49AB10	30.5
102	Dec 16 2005	00:40:12	B9000800BA49AB10	30.5
103	Dec 16 2005	00:41:12	B9000800BA49AB10	30.5
104	Dec 16 2005	00:42:12	B9000800BA49AB10	30.5
105	Dec 16 2005	00:43:13	B9000800BA49AB10	31.0
106	Dec 16 2005	00:44:12	B9000800BA49AB10	31.0
107	Dec 16 2005	00:45:12	B9000800BA49AB10	31.0
108	Dec 16 2005	00:46:12	B9000800BA49AB10	31.0
109	Dec 16 2005	00:47:12	B9000800BA49AB10	31.0
110	Dec 16 2005	00:48:12	B9000800BA49AB10	31.0
111	Dec 16 2005	00:49:12	B9000800BA49AB10	31.0
112	Dec 16 2005	00:50:12	B9000800BA49AB10	31.0

Figure 4.5 Timestamp integrity

Other data we had, we can counted 10 seconds change in timestamp for period of four days measurements.

4.3 Error Analysis

In this section, we compare our measurements data with the thermometer Hg reading. Since the minimum scale on the thermometer is 0.5 °C, then we get the same resolution between our temperature measurement system and the calibrator. In figure 4.6 shown that the relationship requires the equation of $y = 0.8943x + 3.2252$ with the coefficient correlation of 0.8306 between our system compare to the calibrator, which means we have a faulty measurement around 16.94%. The temperature measurement system detected by the DS19B20 sensor has an integrated chip including the signal conditioning which relatively give an accurate measurement in a stable condition. A simple test has done to

observe the response of the old thermometer which shows it has a slow transition measurement reading. In the experiment, we obtain the time response from the 29.0 °C to 29.5 °C in the range of three to ten seconds for the calibrator. Meanwhile for our system, the transition will need about maximum three seconds. In respect to these data we believe that our measurement system is better than the calibrator.

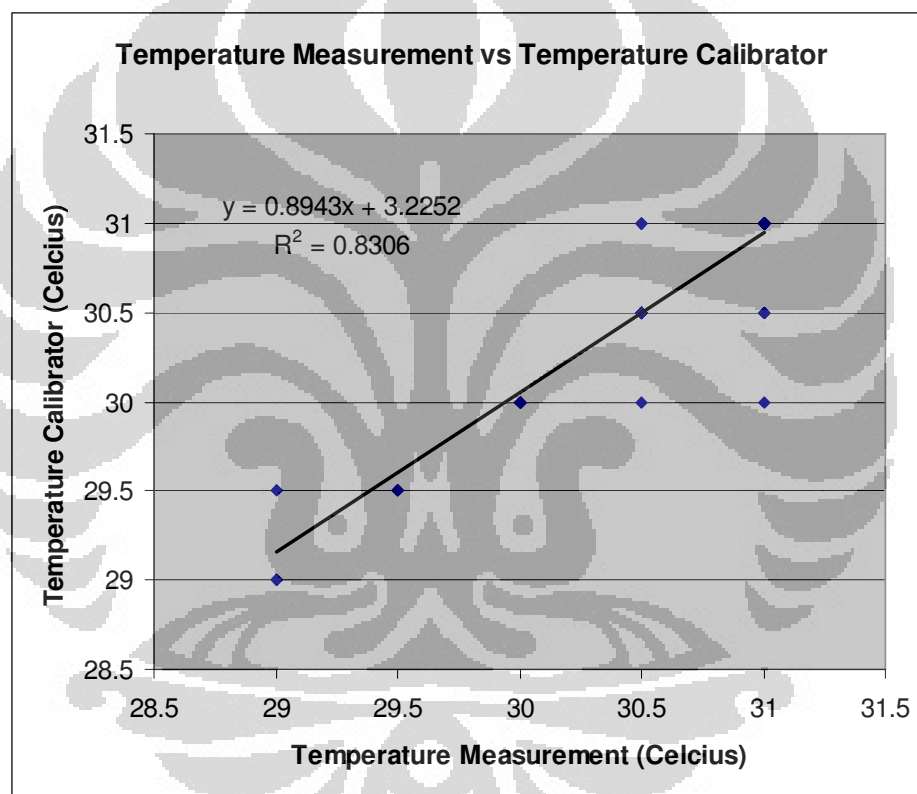
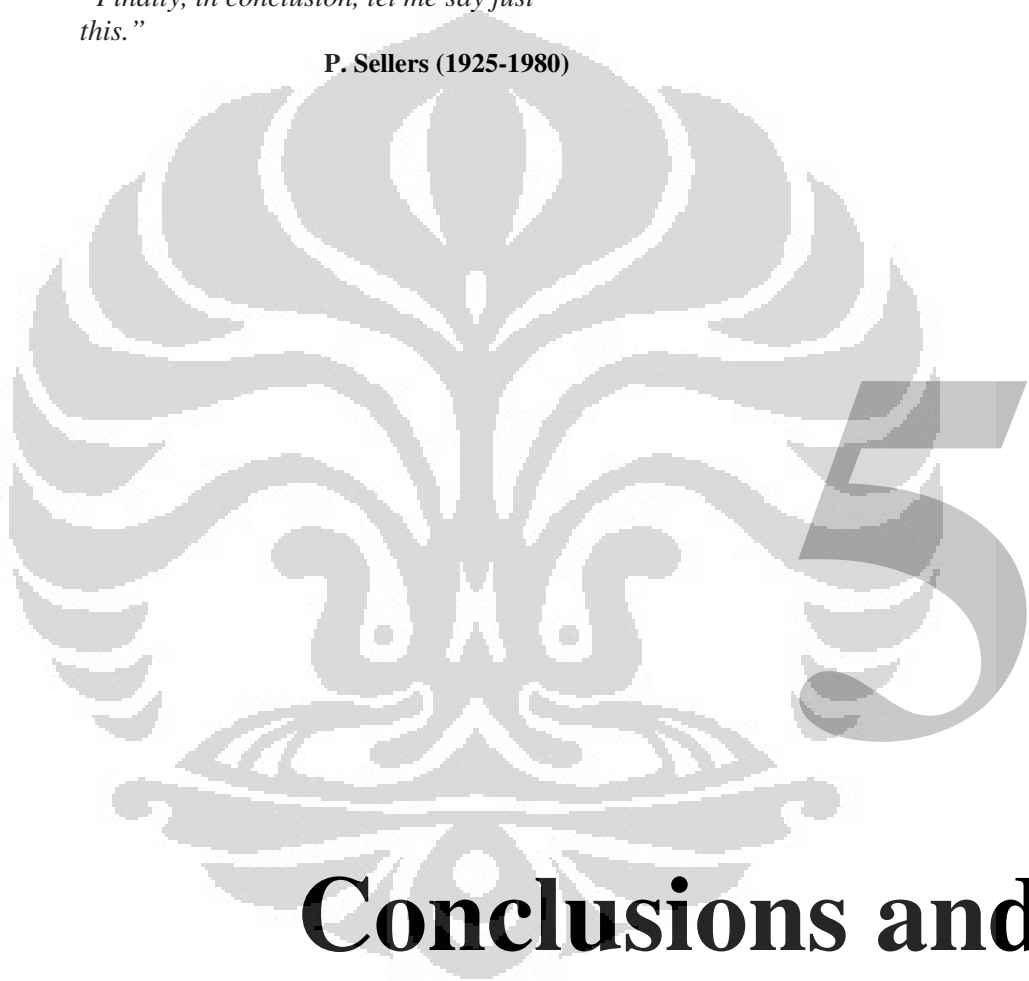


Figure 4.6 Temperature measurements versus Temperature Calibrator

“Finally, in conclusion, let me say just this.”

P. Sellers (1925-1980)



Conclusions and Future Works

In the previous chapters we have seen the experiments prepared to set up a distributed temperature measurements. Also we have seen what sort of equipments we need, the method adapted, the software and hardware

implementation, the data obtain, error and analysis might occurs during the process. In this chapter as it is the last part of this book, we conclude the experiments set up and the implementation. We also give some remarks that might be useful for further improvements for the next experiments.

As we can see from the explanation that the architecture proposed can be considered as a simple distributed measurements system. We can get a good and consistent temperature reading using the 1-Wire iButton temperature sensor to observe the change in the big environment. Since the response of the iButton is a little bit slow around 3 seconds, than we are not recommending it to be used in a critical application. But somehow the sensor can be very useful for a moderate measurement. According to the physical sensor makes we believe that the reliability measurement of the system can be gained in a rough environment. But it is still need to be installed in a good packaging for certain environments.

In the next experiments, we hope that we can focus on the architecture: which fit the best for certain applications and which method is efficient to use? We also suggest using several of sensors to search which one fit and best for some range of requirements of temperature measurements. Also the power supplies suppose to be stable and good with no much flicker when powering the systems.

REFERENCES

- [01]. Andrew S. Tanenbaum, Maarten van Steen, *Distributed Systems: Principles and Paradigms*, International Edition, Prentice Hall – Pearson Education International, New Jersey 07468, 2002.
- [02]. The AVR318, Dallas 1-Wire® master, ATMEL Application Note, Rev: 2579A-AVR-09/04
- [03]. The AT90S8535 Datasheets, ATMEL Supporting CD, 2002.
- [04]. the TINI Guide, Dallas Semiconductor – MAXIM, Ref. 0; 7/04, 2004
- [05]. B. Hancock, *Designing and Implementing Ethernet Networks*, New York: Wiley, 1993.
- [06]. P. Arpaia, F. Cennamo, P. Daponte, and M. Savastano, *A distributed laboratory based on object-oriented systems*, *Measurement*, vol 19, pp.207-215, 1996.
- [07]. G. Fortino, D. Grimaldi and L. Nigro, *Distributed measurement patterns based on Java and web tools*, *Proc. IEEE Autotestcon*, pp.624-628, 1997.
- [08]. M. Bertocco, F. Ferraris, C. Offelli and M. Parvis, *A client-server architecture for distributed measurement systems*, *IEEE Trans. Instrum. Meas.*, vol. 47, pp 1143-1148, Oct 1998.
- [09]. G. Held, *Ethernet Networks: Design, Implementation, Operation, Management*, 3rd ed. New York: Wiley, 1998.
- [10]. A. Ferrero and V. Piuri, *A simulation tool for virtual laboratory experiments in a WWW environment*, *IEEE Trans. Instrum. Meas.*, vol. 48, pp 741-746, June 1999.
- [11]. G. Bucci and C. Landi, *on-line digital measurement for the quality analysis of power systems under nonsinusoidal conditions*, *IEEE Trans. Instrum. Meas.*, vol 48, pp 853-857, Aug 1999.
- [12]. L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, C. Offelli, M. Parvis, and V. Piuri, *A web-based distributed virtual laboratory*, *IEEE Trans. Instrum. Meas.*, vol. 49, pp 349-356, No. 3, Apr. 2000.

- [13]. D. Buhler, W. Kuchlin, G. Grubler, and G. Nusser, *The virtual automation lab-web based teaching of automation engineering concepts*, in Proc. Computing Based Systems (ECBS), pp.156-164, Apr. 2000.
- [14]. T. Saito, I. Tomoda, Y. Takabatake, K. Teramoto, and K. Fujimoto, *Gateway technologies for home network and their implementations*, in Proc. Workshop Distributed Computing System (DCS), pp.175-180, Apr. 2001.
- [15]. G.Bucci and C. Landi, *A Distributed Measurement Architecture for Industrial Applications*, IEEE Trans. Instrum. Meas., vol. 52, pp 165-174, No. 1, Feb 2003.
- [16]. F. Pianegiani, D. Macii, and P. Carbone, *An Open Distributed Measurement System Based on an Abstract Client-Server Architecture*, IEEE Trans. Instrum. Meas., vol. 52, pp 686-692, No. 3, June 2003.
- [17]. A. Ferrero, S. Silicone, C. Bonora and M. Parmigiani, *ReMLab: A Java-Based Remote, Didactic Measurement Laboratory*, IEEE Trans. Instrum. Meas., vol. 52, pp 710-715, No. 3, June 2003.



Appendix A

The ds1820 library file (ds1820.lib)

```

#define DS1820_SEARCH_ROM_CMD 0xf0
#define DS1820_ALARM_SEARCH_CMD 0xec

#include <delay.h>

#if funcused ds1820_read_spd || funcused ds1820_select || funcused
ds1820_temperature_10 || funcused ds1820_set_alarm
unsigned char ds1820_select(unsigned char *addr)
{
  unsigned char i;
  if (w1_init()==0) return 0;
  if (addr)
  {
    w1_write(0x55);
    i=0;
    do
      w1_write(*(addr++));
    while (++i<8);
  }
  else w1_write(0xcc);
  return 1;
}
#endif
#if funcused ds1820_read_spd || funcused ds1820_temperature_10 || funcused
ds1820_set_alarm
unsigned char ds1820_read_spd(unsigned char *addr)
{
  unsigned char i;
  unsigned char *p;
  if (ds1820_select(addr)==0) return 0;
  w1_write(0xbe);
  i=0;
  p=(char *) &__ds1820_scratch_pad;
  do
    *(p++)=w1_read();
  while (++i<9);
  return !w1_dow_crc8(&__ds1820_scratch_pad,9);
}
#endif
#if funcused ds1820_temperature_10
int ds1820_temperature_10(unsigned char *addr)
{
  if (ds1820_select(addr)==0) return -9999;
  w1_write(0x44);
  delay_ms(550);
  if (ds1820_read_spd(addr)==0) return -9999;
  w1_init();
  return (((int)__ds1820_scratch_pad.temp_msb<<8)|
    __ds1820_scratch_pad.temp_lsb)*5;
}
#endif
#if funcused ds1820_set_alarm
unsigned char ds1820_set_alarm(unsigned char *addr,signed char temp_low,signed
char temp_high)
{
  if (ds1820_select(addr)==0) return 0;
  w1_write(0x4e);
  w1_write(temp_high);
  w1_write(temp_low);
  if (ds1820_read_spd(addr)==0) return 0;
  if ((__ds1820_scratch_pad.temp_low!=temp_low) ||
    (__ds1820_scratch_pad.temp_high!=temp_high)) return 0;
  if (ds1820_select(addr)==0) return 0;
  w1_write(0x48);
  delay_ms(15);
  return w1_init();
}
#endif

```

The I2C headers file (i2c.h)

```

/*
CodeVisionAVR C Compiler
(C) 1998-2000 Pavel Haiduc, HP InfoTech S.R.L.

Prototypes for Dallas Semiconductor
1 Wire protocol functions

BEFORE #include -ING THIS FILE YOU
MUST DECLARE THE I/O ADDRESS OF THE
DATA REGISTER OF THE PORT AT WHICH
THE 1 WIRE BUS IS CONNECTED AND
THE DATA BIT USED

EXAMPLE FOR PORTB:

    #asm
        .equ __w1_port=0x18
        .equ __w1_bit=3
    #endasm
    #include <i2c.h>
*/

#ifndef _I2C_INCLUDED_
#define _I2C_INCLUDED_

#pragma used+
unsigned char i2c_init(void);
unsigned char i2c_read(void);
unsigned char i2c_write(unsigned char data);
unsigned char i2c_search(unsigned char cmd,void *p);
unsigned char i2c_dow_crc8(void *p,unsigned char n);
#pragma used-

#endif

```

The AT90S8535 headers file (90s8535.h)

```

// I/O registers definitions for the AT90S8535

#ifndef _90S8535_INCLUDED_
#define _90S8535_INCLUDED_

#pragma used+
sfrb ADCL=4;
sfrb ADCH=5;
sfrw ADCW=4;           // 16 bit access
sfrb ADCSR=6;
sfrb ADMUX=7;
sfrb ACSR=8;
sfrb UBRR=9;
sfrb UCR=0xa;
sfrb USR=0xb;
sfrb UDR=0xc;
sfrb SPCR=0xd;
sfrb SPSR=0xe;
sfrb SPDR=0xf;
sfrb PIN0=0x10;
sfrb DDR0=0x11;
sfrb PORT0=0x12;
sfrb PIN1=0x13;
sfrb DDR1=0x14;
sfrb PORT1=0x15;
sfrb PIN2=0x16;

```

```

sfrb DDRB=0x17;
sfrb PORTB=0x18;
sfrb PINA=0x19;
sfrb DDRA=0x1a;
sfrb PORTA=0x1b;
sfrb EECR=0x1c;
sfrb EEDR=0x1d;
sfrb EEARL=0x1e;
sfrb EEARH=0x1f;
sfrw EEAR=0x1e; // 16 bit access
sfrb WDTCR=0x21;
sfrb ASSR=0x22;
sfrb OCR2=0x23;
sfrb TCNT2=0x24;
sfrb TCCR2=0x25;
sfrb ICR1L=0x26;
sfrb ICR1H=0x27;
sfrw ICR1=0x26; // 16 bit access
sfrb OCR1BL=0x28;
sfrb OCR1BH=0x29;
sfrw OCR1B=0x28; // 16 bit access
sfrb OCR1AL=0x2a;
sfrb OCR1AH=0x2b;
sfrw OCR1A=0x2a; // 16 bit access
sfrb TCNT1L=0x2c;
sfrb TCNT1H=0x2d;
sfrw TCNT1=0x2c; // 16 bit access
sfrb TCCR1B=0x2e;
sfrb TCCR1A=0x2f;
sfrb TCNT0=0x32;
sfrb TCCR0=0x33;
sfrb MCUSR=0x34;
sfrb MCUCR=0x35;
sfrb TIFR=0x38;
sfrb TIMSK=0x39;
sfrb GIFR=0x3a;
sfrb GIMSK=0x3b;
sfrb SPL=0x3d;
sfrb SPH=0x3e;
sfrb SREG=0x3f;
#pragma used-

// Interrupt vectors definitions

#define EXT_INT0 2
#define EXT_INT1 3
#define TIM2_COMP 4
#define TIM2_OVF 5
#define TIM1_CAPT 6
#define TIM1_COMPA 7
#define TIM1_COMPB 8
#define TIM1_OVF 9
#define TIM0_OVF 10
#define SPI_STC 11
#define UART_RXC 12
#define UART_DRE 13
#define UART_TXC 14
#define ADC_INT 15
#define EE_RDY 16
#define ANA_COMP 17

#endif

```

The delay headers file (Delay.h)

```
// CodeVisionAVR C Compiler
// (C) 1998-2000 Pavel Haiduc, HP InfoTech S.R.L.

#ifndef _DELAY_INCLUDED_
#define _DELAY_INCLUDED_

#pragma used+

void delay_us(unsigned int n);
void delay_ms(unsigned int n);

#pragma used-

#endif
```

The 1820 headers file (Ds1820.h)

```
#ifndef _DS1820_INCLUDED_
#define _DS1820_INCLUDED_

#include <lwire.h>

#define DS1820_FAMILY_CODE 0x10

#pragma used+
struct __ds1820_scratch_pad_struct
{
    unsigned char temp_lsb,temp_msb,
                 temp_high,temp_low,
                 res1,res2,
                 cnt_rem,cnt_c,
                 crc;
} __ds1820_scratch_pad;

unsigned char ds1820_select(unsigned char *addr);
unsigned char ds1820_read_spd(unsigned char *addr);
int ds1820_temperature_10(unsigned char *addr);
unsigned char ds1820_set_alarm(unsigned char *addr,signed char temp_low,signed
char temp_high);
#pragma used-
#pragma library ds1820.lib

#endif
```

The standard input output library file (stdio.lib)

```
#include <ctype.h>
#include <stdarg.h>
#include <string.h>

#ifndef NULL
#define NULL 0
#endif

#ifndef EOF
#define EOF -1
#endif

#ifndef _DEBUG_TERMINAL_IO_
#ifndef _ALTERNATE_GETCHAR_
char getchar(void)
{
```



```

#asm
    sbis usr, rxc
    rjmp _getchar
    in    r30, udr
#endasm
}
#endif

#ifndef _ALTERNATE_PUTCHAR_
void putchar(char c)
{
#asm
    sbis usr, udre
    rjmp _putchar
    ld   r30, y
    out  udr, r30
#endasm
}
#endif
#endif

#if funcused printf | funcused sprintf | funcused scanf | funcused sscanf
#pragma used+
static char *pp;
#pragma used-
#endif

#if funcused printf | funcused sprintf
static void _put(char k)
{
#asm("put:")
if (pp) *pp++=k;
else putchar(k);
}

#define TEST_FORMAT 0
#define GET_FLAGS 1
#define GET_PAD_CHAR 2
#define GET_WIDTH 3
#define GET_PRECISION 4
#define DO_PRINT 5

#if defined _PRINTF_INT_ | defined _PRINTF_INT_WIDTH_
static flash unsigned tbl10[]={10000,1000,100,10,1};
static flash unsigned tbl16[]={0x1000,0x100,0x10,1};
#endif

#if defined _PRINTF_INT_

#define F_SIGNED 1
#define F_CAP_HEX 2
#define F_PLUS_PLUS 4 // plus char is '+'
#define F_PLUS_SPACE 8 // plus char is ' '
#define F_PAD_CHR0 0x10 // pad char is '0'

static void _print(char flash *fmtstr, va_list argptr)
{
    register unsigned char l=TEST_FORMAT, //R16
    flags, //R17
    k, //R18
    s; //R19
    register unsigned n; //R20,R21
    unsigned i;
    unsigned flash *pi;
    char *p;

    while (k=*fmtstr++)
        switch (l)
        {
            case TEST_FORMAT: if (k=='%') l=GET_FLAGS; else _put(k);
                                break;

```

```

case GET_FLAGS: if (k=='%') {_put(k); l=TEST_FORMAT; break;};
                l=GET_PAD_CHAR;
                s=0;
                flags=0;
                if (k=='+') {s='+'; break;};
                if (k==' ') {s=' '; break;};
case GET_PAD_CHAR:
                if (k=='0') {flags|=F_PAD_CHR0; l=DO_PRINT; break;}
case DO_PRINT:
                switch (k)
                {
                case 'c':
                    _put(va_arg(argptr, char));
                    goto next;

                case 's':
                    p=va_arg(argptr, char *);
                    while (k=*p++) _put(k);
                    goto next;

                case 'p':
                    pi=va_arg(argptr, char flash *);
                    while (k=* (char flash *) pi++) _put(k);
                    goto next;

                case 'd':
                case 'i':
                    flags|=F_SIGNED;
                case 'u':
                    pi=tbl10;
                    goto get_arg;

                case 'X':
                    flags|=F_CAP_HEX;
                case 'x':
                    pi=tbl16;
                    get_arg:
                    if (flags & F_SIGNED)
                    {
                        n=va_arg(argptr, int);
                        if ((int)n<0)
                        {
                            n=- (int) n;
                            s='-';
                        };
                        if (s) _put(s);
                    }
                    else n=va_arg(argptr, unsigned);
                    do
                    {
                        k='0';
                        i=*pi++; //R30, R31=i
                        #asm
                        calc_digit:
                            cp    r20, r30
                            cpc   r21, r31
                            brlo  calc_digit_done
                        #endasm
                        ++k;
                        #asm
                            sub   r20, r30
                            sbc   r21, r31
                            brne  calc_digit
                        #endasm
                        calc_digit_done:
                        #endasm
                        if ((flags & F_PAD_CHR0) || (k>'0') || (i==1))
                        {
                            flags|=F_PAD_CHR0;
                            if (k>'9')
                            {
                                if (flags & F_CAP_HEX) k+=7;
                            }
                        }
                    }
                }

```

```

        else k+=0x27;
        };
        _put(k);
        };
    }
    while (i>1);
default:
    next:
    l=TEST_FORMAT;
};
};
}
#elif defined _PRINTF_INT_WIDTH_

#define F_LEFT_JUSTIFY 1
#define F_STRING 2
#define F_SIGNED 4
#define F_CAP_HEX 8
#define F_STRING_FLASH 8
#define F_NON_ZERO 0x10
#define F_PLUS_PLUS 0x20 // plus char is '+'
#define F_PLUS_SPACE 0x40 // plus char is ' '
#define F_PAD_CHR0 0x80 // pad char is '0'

static void _print(char flash *fmtstr, va_list argptr)
{
register unsigned char l=TEST_FORMAT, //R16
flags, //R17
j, //R18
k, //R19
width, //R20
s; //R21
char *p;
unsigned n, //Y+10
i; //Y+8;
unsigned flash *pi;

while (k=*fmtstr++)
    switch (l)
    {
    case TEST_FORMAT: if (k=='%') l=GET_FLAGS; else _put(k);
        break;
    case GET_FLAGS: if (k=='%') {_put(k); l=TEST_FORMAT; break;};
        l=GET_PAD_CHAR;
        s=0;
        flags=0;
        if (k=='-') {flags=F_LEFT_JUSTIFY; break;};
        if (k=='+') {s='+'; break;};
        if (k==' ') {s=' '; break;};
    case GET_PAD_CHAR:
        width=0;
        l=GET_WIDTH;
        if (k=='0') {flags|=F_PAD_CHR0; break;}
    case GET_WIDTH:
        if ((k>='0') && (k<('9'+1)))
        {
        #ifdef _ENHANCED_CORE_
        width*=10;
        #else
        j=width;
        width<<=2;
        width+=j;
        width<<=1;
        #endif
        width+=k-'0';
        break;
        };

        switch (k)
        {
        case 'c':

```

```

        _put(va_arg(argptr, char));
        goto next;

    case 's':
        p=va_arg(argptr, char *);
        l=strlen(p);
        goto disp_string;

    case 'p':
        pi=va_arg(argptr, char flash *);
        l=strlenf((char flash *) pi);
        flags|=F_STRING_FLASH;
    disp_string:
        flags|=F_STRING;
        flags&=~F_PAD_CHR0;
        j=0;
        goto pad_left;

    case 'd':
    case 'i':
        flags|=F_SIGNED;
    case 'u':
        pi=tbl10; l=5;
        goto get_arg;

    case 'X':
        flags|=F_CAP_HEX;
    case 'x':
        pi=tbl16; l=4;
    get_arg:
        if (flags & F_SIGNED)
        {
            n=va_arg(argptr, int);
            if ((int)n<0)
            {
                n=- (int) n;
                s='-';
            };
            if (s) ++l;
            else flags&=~F_SIGNED;
        }
        else n=va_arg(argptr, unsigned);
    pad_left:
        if ((flags & F_LEFT_JUSTIFY)==0)
            while (width>l)
            {
                if (flags & F_PAD_CHR0)
                {
                    if (flags & F_SIGNED)
                    {
                        flags&=~F_SIGNED;
                        k=s;
                        --l;
                    }
                    else k='0';
                }
                else k=' ';
                _put(k);
                --width;
            };

        j=1;

        if (flags & F_STRING)
        {
            while (j)
            {
                if (flags & F_STRING_FLASH) _put(* (char
flash *) pi++);

                else _put(*p++);
                if (width) --width;
                --j;
            }
        }

```

```

        };
    }
    else
    do
    {
        k='0';
        i=*pi++; //R30,R31=i
        #asm
            ldd r26,y+10 ;R26,R27=n
            ldd r27,y+11
        calc_digit:
            cp r26,r30
            cpc r27,r31
            brlo calc_digit_done
        #endasm
        ++k;
        #asm
            sub r26,r30
            sbc r27,r31
            brne calc_digit
        calc_digit_done:
            std Y+10,r26 ;n=R26,R27
            std y+11,r27
        #endasm
        if (k>'9')
        {
            if (flags & F_CAP_HEX) k+=7;
            else k+=0x27;
        };
        if (flags & F_NON_ZERO) goto print_digit;
        if ((k>'0') || (i==1))
        {
            flags|=F_NON_ZERO;
            goto print_sign;
        };
        if ((width>=j) && ((flags & F_LEFT_JUSTIFY)==0))
        {
            k=' ';
            if (flags & F_PAD_CHR0)
            {
                k='0';
                flags|=F_NON_ZERO;
            }
            print_sign:
            if (flags & F_SIGNED)
            {
                flags&=~F_SIGNED;
                _put(s);
                if (width) --width;
            };
        };
        print_digit:
        _put(k);
        if (width) --width;
    };
    --j;
}
while (i>1);
if (flags & F_LEFT_JUSTIFY)
while (width)
{
    --width;
    _put(' ');
};
default:
next:
l=TEST_FORMAT;
};
};
}
#endif defined _PRINTF_LONG_WIDTH_

```

```

#define F_LEFT_JUSTIFY 1
#define F_LONG 2
#define F_SIGNED 4
#define F_CAP_HEX 8
#define F_STRING_FLASH 8
#define F_NON_ZERO 0x10
#define F_PLUS_PLUS 0x20 // plus char is '+'
#define F_PLUS_SPACE 0x40 // plus char is ' '
#define F_PAD_CHR0 0x80 // pad char is '0'

static void _print(char flash *fmtstr,va_list argptr)
{
register unsigned char l=TEST_FORMAT,flags,j,k,width,s;
unsigned char base;
unsigned long n,i;
char *p;
char flash *pf;

while (k=*fmtstr++)
switch (l)
{
case TEST_FORMAT: if (k=='%') l=GET_FLAGS; else _put(k);
break;
case GET_FLAGS: if (k=='%') {_put(k); l=TEST_FORMAT; break;};
l=GET_PAD_CHAR;
s=0;
flags=0;
if (k=='-') {flags=F_LEFT_JUSTIFY; break;};
if (k=='+') {s='+'; break;};
if (k==' ') {s=' '; break;};
case GET_PAD_CHAR:
width=0;
l=GET_WIDTH;
if (k=='0') {flags|=F_PAD_CHR0; break;}
case GET_WIDTH:
if ((k>='0') && (k<('9'+1)))
{
#ifdef _ENHANCED_CORE_
width*=10;
#else
j=width;
width<<=2;
width+=j;
width<<=1;
#endif
width+=k-'0';
break;
};
if (k=='l')
{
flags|=F_LONG;
l=DO_PRINT;
break;
};
case DO_PRINT:
switch (k)
{
case 'c':
_put(va_arg(argptr,char));
goto next;

case 's':
p=va_arg(argptr,char *);
l=strlen(p);
goto disp_string;

case 'p':
pf=va_arg(argptr,char flash *);
l=strlenf(pf);
flags|=F_STRING_FLASH;
disp_string:

```

```

        flags&=~F_PAD_CHR0;
        base=0;
        j=0;
        goto pad_left;

    case 'd':
    case 'i':
        flags|=F_SIGNED;
    case 'u':
        base=10;
        if (flags & F_LONG)
            {i=1000000000L; l=10; goto get_arg;};
        i=10000; l=5;
        goto get_arg;

    case 'X':
        flags|=F_CAP_HEX;
    case 'x':
        base=16;
        if (flags & F_LONG)
            {i=0x10000000L; l=8; goto get_arg;};
        i=0x1000; l=4;
    get_arg:
        if (flags & F_LONG) n=va_arg(argptr, long);
        else
            if (flags & F_SIGNED) n=va_arg(argptr, int);
            else n=va_arg(argptr, unsigned);
        if (flags & F_SIGNED)
            {
                if ((long)n<0)
                    {
                        n=- (long) n;
                        s='-';
                    };
                if (s) ++l;
                else flags&=~F_SIGNED;
            };
    pad_left:
        if ((flags & F_LEFT_JUSTIFY)==0)
            while (width>l)
                {
                    if (flags & F_PAD_CHR0)
                        {
                            if (flags & F_SIGNED)
                                {
                                    flags&=~F_SIGNED;
                                    k=s;
                                    --l;
                                }
                            else k='0';
                        }
                    else k=' ';
                    _put(k);
                    --width;
                };

        j=l;
        if (base==0)
            {
                while (j)
                    {
                        if (flags & F_STRING_FLASH) _put(*pf++);
                        else _put(*p++);
                        if (width) --width;
                        --j;
                    };
            }
        else
            do
                {
                    k=(unsigned char) (n/i);
                    if (k>9)

```

```

        {
        if (flags & F_CAP_HEX) k+=0x37;
        else k+=0x57;
        }
else k+='0';

if (flags & F_NON_ZERO) goto print_digit;
if ((k>'0') || (i==1)) goto print_sign;
if ((width>=j) && ((flags & F_LEFT_JUSTIFY)==0))
{
k=' ';
if (flags & F_PAD_CHR0)
{
k='0';
print_sign:
flags|=F_NON_ZERO;
if (flags & F_SIGNED)
{
flags&=~F_SIGNED;
_putchar(s);
if (width) --width;
};
};
print_digit:
_putchar(k);
if (width) --width;
};
--j;
n%=i;
i/=base;
}
while (i);
if (flags & F_LEFT_JUSTIFY)
while (width)
{
--width;
_putchar(' ');
};
default:
next:
l=TEST_FORMAT;
};
};
}
}
#elif defined _PRINTF_LONG_WIDTH_PRECISION_

#define F_LEFT_JUSTIFY 1
#define F_LONG 2
#define F_SIGNED 4
#define F_CAP_HEX 8
#define F_STRING_FLASH 8
#define F_NON_ZERO 0x10
#define F_PLUS_PLUS 0x20 // plus char is '+'
#define F_PLUS_SPACE 0x40 // plus char is ' '
#define F_PAD_CHR0 0x80 // pad char is '0'

static void _print_sign(void)
{
#asm
;flags&=~F_SIGNED
andi r17,~4
;_put(s)
ldd r30,y+17
st -y,r30
rcall put
;if (width) --width
cpi r20,0
breq width0
dec r20
width0:
#endasm

```



```

}

static void _print(char flash *fmtstr,va_list argptr)
{
register unsigned char l=TEST_FORMAT, //R16
flags, //R17
j, //R18
k, //R19
width, //R20
precision; //R21
unsigned char s, //Y+17
base; //Y+16
unsigned long n,i;
char *p;
char flash *pf;

while (k=*fmtstr++)
switch (l)
{
case TEST_FORMAT: if (k=='%') l=GET_FLAGS; else _put(k);
break;
case GET_FLAGS: if (k=='%') {_put(k); l=TEST_FORMAT; break;};
l=GET_PAD_CHAR;
s=0;
flags=0;
if (k=='-') {flags=F_LEFT_JUSTIFY; break;};
if (k=='+') {s='+'; break;};
if (k==' ') {s=' '; break;};
case GET_PAD_CHAR:
width=0;
l=GET_WIDTH;
if (k=='0') {flags|=F_PAD_CHR0; break;}
case GET_WIDTH:
if ((k>='0') && (k<('9'+1)))
{
#ifdef _ENHANCED_CORE_
width*=10;
#else
j=width;
width<<=2;
width+=j;
width<<=1;
#endif
width+=k-'0';
break;
};
precision=0;
if (k=='.') {l=GET_PRECISION; break;}
goto test_long;
case GET_PRECISION:
if ((k>='0') && (k<('9'+1)))
{
#ifdef _ENHANCED_CORE_
precision*=10;
#else
j=precision;
precision<<=2;
precision+=j;
precision<<=1;
#endif
precision+=k-'0';
break;
};
test_long:
if (k=='l')
{
flags|=F_LONG;
l=DO_PRINT;
break;
};
case DO_PRINT:

```

```

switch (k)
{
case 'c':
    _put(va_arg(argptr, char));
    goto next;

case 's':
    p=va_arg(argptr, char *);
    l=strlen(p);
    goto disp_string;

case 'p':
    pf=va_arg(argptr, char flash *);
    l=strlenf(pf);
    flags|=F_STRING_FLASH;
    disp_string:
    flags&=~F_PAD_CHR0;
    if (precision && (l>precision))
        l=precision;
    precision=0;
    base=0;
    j=0;
    goto pad_left;

case 'd':
case 'i':
    flags|=F_SIGNED;
case 'u':
    base=10;
    if (flags & F_LONG)
        {i=1000000000L; l=10; goto get_arg;};
    i=10000; l=5;
    goto get_arg;

case 'X':
    flags|=F_CAP_HEX;
case 'x':
    base=16;
    if (flags & F_LONG)
        {i=0x10000000L; l=8; goto get_arg;};
    i=0x1000; l=4;
get_arg:
    if (precision) flags&=~F_PAD_CHR0;
    else precision=1;
    if (flags & F_LONG) n=va_arg(argptr, long);
    else
        if (flags & F_SIGNED) n=va_arg(argptr, int);
        else n=va_arg(argptr, unsigned);
    if (flags & F_SIGNED)
        {
        if ((long)n<0)
            {
            n=- (long) n;
            s='-';
            };
        if (s) {++l; ++precision;}
        else flags&=~F_SIGNED;
        };
    j=precision;
pad_left:
    if ((flags & F_LEFT_JUSTIFY)==0)
        while ((width>l) && (width>j))
            {
            if (flags & F_PAD_CHR0)
                {
                if (flags & F_SIGNED)
                    {
                    flags&=~F_SIGNED;
                    k=s;
                    --l;
                    }
                }
            }

```

```

        else k='0';
    }
    else k=' ';
    _put(k);
    --width;
};
while (precision>1)
{
    flags|=F_NON_ZERO;
    if (flags & F_SIGNED)
    {
        _print_sign();
        --l;
        --precision;
    };
    _put('0');
    if (width) --width;
    --precision;
};
j=1;
if (base==0)
{
    while (j)
    {
        if (flags & F_STRING_FLASH) _put(*pf++);
        else _put(*p++);
        if (width) --width;
        --j;
    };
}
else
do
{
    k=(unsigned char) (n/i);
    if (k>9)
    {
        if (flags & F_CAP_HEX) k+=0x37;
        else k+=0x57;
    }
    else k+='0';
    if (flags & F_NON_ZERO) goto print_digit;
    if ((k>'0') || (i==1)) goto print_sign;
    if (precision>=j)
    {
        k='0';
        goto print_sign;
    };
    if ((width>=j) && ((flags & F_LEFT_JUSTIFY)==0))
    {
        k=' ';
        if (flags & F_PAD_CHR0)
        {
            k='0';
        };
        print_sign:
        flags|=F_NON_ZERO;
        if (flags & F_SIGNED)
            _print_sign();
    };
    print_digit:
    _put(k);
    if (width) --width;
};
--j;
n%=i;
i/=base;
}
while (i);
if (flags & F_LEFT_JUSTIFY)
while (width)
{
    --width;
}

```

```

        _put(' ');
    };
    default:
        next:
            l=TEST_FORMAT;
    };
};

}
#endif
#endif

#if funcused sprintf
void sprintf(char *str,char flash *fmtstr,...)
{
    va_list argptr;
    va_start(argptr,fmtstr);
    pp=str;
    _print(fmtstr,argptr);
    *pp=0;
}
#endif

#if funcused printf
void printf(char flash *fmtstr,...)
{
    va_list argptr;
    va_start(argptr,fmtstr);
    pp=NULL;
    _print(fmtstr,argptr);
}
#endif

#if funcused scanf | funcused sscanf
#pragma used+
static char cc;
#pragma used-

static char _get(void)
{
    char k;
    if (cc)
    {
        k=cc;
        cc=0;
    }
    else if (pp)
    {
        if (k=*pp) ++pp;
    }
    else k=getchar();
    return k;
}

static signed char _scanf(char flash *fmtstr,va_list argptr)
{
    void *parg;
    unsigned char k,b,width;
    signed char ns,s;
    unsigned int n;

    cc=ns=0;
    while (k=*fmtstr++)
    {
        if (isspace(k))
        {
            while ((k=_get()) && isspace(k));
            cc=k;
        }
        else if (k=='%')
        {
            width=0;

```

```

while (1)
{
    k=*fmtstr++;
    if ((k<'0')|| (k>'9')) break;
    width*=10;
    width+=k-'0';
};
if (k==0) break;
while (isspace(b=_get()));
if (b==0) goto _scan_end;
cc=b;
if (width==0) width=255;
switch (k)
{
    case 'c':
    parg=va_arg(argptr, char *);
    *(char *)parg=_get();
    break;

    case 's':
    parg=va_arg(argptr, char *);
    while (width--)
    {
        if ((k=_get())==0 || isspace(k)) break;
        *(char *)parg++=k;
    };
    *(char *)parg=0;
    break;

    default:
    s=1;
    switch (k)
    {
        case 'd':
        case 'i': s=0;
        case 'u': b=10; break;
        case 'x': b=16; break;
        case '%': goto _scan_cmp;
        default: return ns;
    };

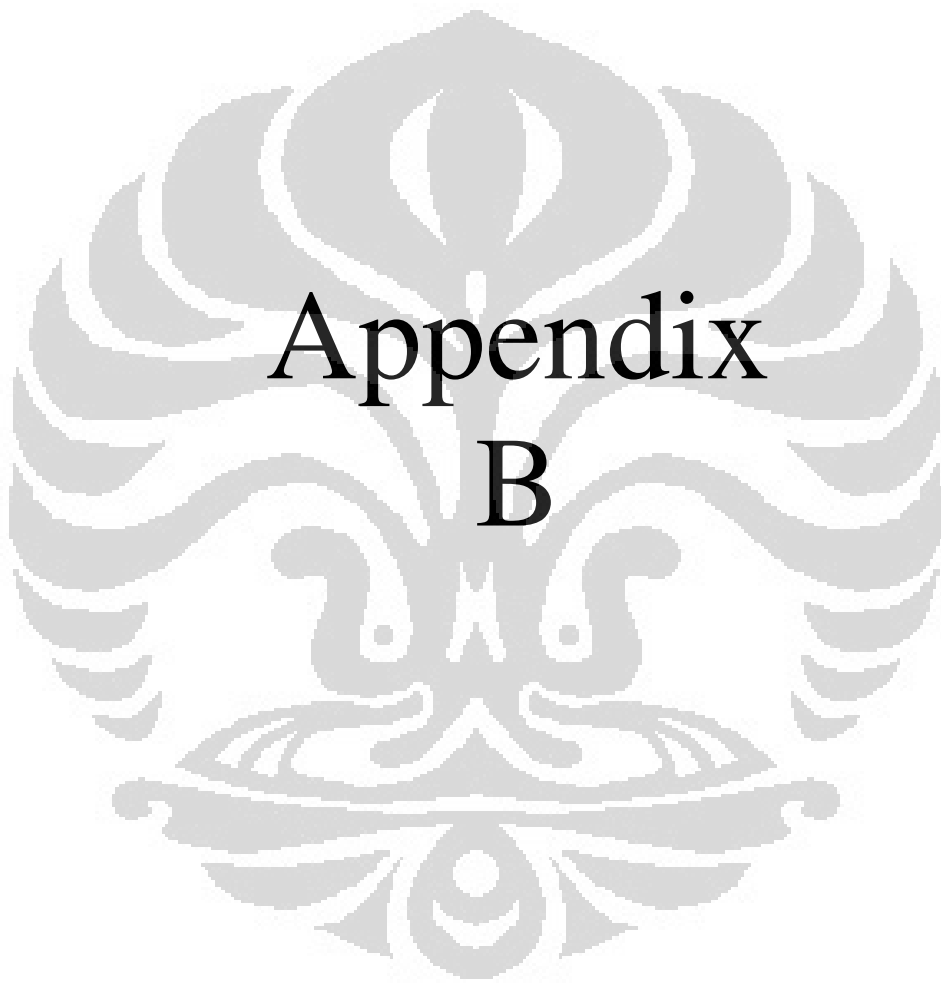
    n=0;
    while (width--)
    {
        if ((k=_get())<=' ') goto _scan_next;
        if (s==0)
        {
            if (k=='-') {s=-1; continue;}
            else s=1;
        };
        if (k<'0') goto _scan_next;
        if (k>='a') k-=0x57;
        else if (k>='A') k-=0x37;
        else k--;'0';
        if (k>=b)
        {
            _scan_next:
            cc=k;
            break;
        };
        n=n*b+k;
    };
    parg=va_arg(argptr, int *);
    *(int *)parg=n*s;
};
++ns;
}
else
{
    _scan_cmp:
    if (k!=_get())
    {

```

```
        _scan_end:
        if (ns==0) return EOF;
        break;
        );
    };
};
return ns;
}
#endif

#if funcused sscanf
signed char sscanf(char *str,char flash *fmtstr,...)
{
    va_list argptr;
    va_start(argptr,fmtstr);
    pp=str;
    return _scanf(fmtstr,argptr);
}
#endif

#if funcused scanf
signed char scanf(char flash *fmtstr,...)
{
    va_list argptr;
    va_start(argptr,fmtstr);
    pp=NULL;
    return _scanf(fmtstr,argptr);
}
#endif
```



Appendix B

The surya.c AVR main program

```

#asm
    .equ __wl_port=0x15
    .equ __wl_bit=0
#endasm

#include <90s8535.h>
#include <stdio.h>
#include <ds1820.h>
#include <delay.h>

/* Mendefinisikan konstanta */
#define xtal 3690000L           //quartz-crystal frequency [Hz]
#define baud 9600              // Baud rate
#define DS1990_FAMILY_CODE 1
#define SEARCH_ROM 0xF0
#define MAX_DEVICES 2

/*variabel global didefinisikan*/
void init_serial(void);
void get_command(void);
char command;
unsigned char rom_code[MAX_DEVICES][9];

/* initialize the UART's baud rate */
void init_serial(void){
    UBRR=xtal/16/baud-1;       //set baud rate
    UCR=0x18; }               //RX&TX enabled, no int, 8 data bits

void get_command(void){
    unsigned char i;
    i = USR;                   //Ambil status register UART
    i&= 0x80;                  //Apakah ada data di buffer ?
    if (i == 0x80){
        command=getchar();
        // putchar(command);
    }
    else
        command = -1;
    }

main() {
    unsigned char i,j,devices;
    int temp;
    init_serial();

    while (1) {
        get_command();
        if (command == 'S')
        {

// detect how many 1 Wire devices are present on the bus
devices=wl_search(SEARCH_ROM,&rom_code[0][0]);
for (i=0;i<devices;i++){
    for (j=1;j<=8;j++){
        printf("%02X",rom_code[i][j]);
    };
};

/* measure and display the temperature(s) */

        for (i=0;i<devices;)
        {
            temp=ds1820_temperature_10(&rom_code[i][0]);
            j='+';
            if (temp<0)
            {
                j='-';
                temp=-temp;
            };
};

```



```

        printf("%c%i.%i", ++i, temp/10, temp%10);
        delay_ms(800);
    };
    }
    delay_ms(1000);}
}

```

The Tini_Client.java user interface main program

```

/*
 * Tini_Client.java
 *
 * Created on December 13, 2005, 2:32 AM
 */

import java.awt.*;
import java.awt.event.*;
import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Enumeration;
import java.util.TooManyListenersException;
import javax.swing.*;
import java.sql.*;

import java.util.Date;
import java.util.GregorianCalendar;
import java.util.Calendar;
import java.text.SimpleDateFormat;

import java.io.*;
import java.net.*;
/**
 *
 * @author Surya Darma
 */
public class Tini_Client extends JFrame implements ActionListener {

    Connection connection;
    String DB_NAME;
    int i;
    int delay, nomor, delay2;
    double DC;
    Timer timer;
    Timer timer2;
    int Baris;
    int No;
    int n;

    /** Creates new form Tini_Client */
    public Tini_Client() {
        initComponents();
        //Bagian untuk inisialisasi Timer-----
        delay = 1000;
        timer = new Timer(delay, this);
        timer.setInitialDelay(delay);
        timer.setCoalesce(true);
        timer.start();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code ">
    private void initComponents() {

```

```

jLabel5 = new javax.swing.JLabel();
jLabel11 = new javax.swing.JLabel();
Txt_MySQL_Host = new javax.swing.JTextField();
Txt_MySQL_User = new javax.swing.JTextField();
jLabel12 = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
Txt_MySQL_Password = new javax.swing.JPasswordField();
jButton1 = new javax.swing.JButton();
Text5 = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
jScrollPane = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
jButton3 = new javax.swing.JButton();
List1 = new java.awt.List();
Label_Tanggal = new javax.swing.JLabel();
Label_Jam = new javax.swing.JLabel();
Text_Temperature = new javax.swing.JTextField();
Label_Ctr = new javax.swing.JLabel();
jButton7 = new javax.swing.JButton();
jLabel9 = new javax.swing.JLabel();
Text_Wkt = new javax.swing.JTextField();
jLabel10 = new javax.swing.JLabel();
Label_ID = new javax.swing.JLabel();
jButton2 = new javax.swing.JButton();
Text_SQL = new javax.swing.JTextArea();

getContentPane().setLayout(null);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jLabel5.setText("MySQL Database");
getContentPane().add(jLabel5);
jLabel5.setBounds(60, 0, 110, 14);

jLabel11.setText("Host Name");
getContentPane().add(jLabel11);
jLabel11.setBounds(10, 20, 80, 14);

Txt_MySQL_Host.setText("localhost");
getContentPane().add(Txt_MySQL_Host);
Txt_MySQL_Host.setBounds(90, 20, 120, 19);

Txt_MySQL_User.setText("root");
Txt_MySQL_User.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        Txt_MySQL_UserActionPerformed(evt);
    }
});

getContentPane().add(Txt_MySQL_User);
Txt_MySQL_User.setBounds(90, 40, 120, 19);

jLabel12.setText("User Name");
getContentPane().add(jLabel12);
jLabel12.setBounds(10, 40, 80, 14);

jLabel13.setText("Password");
getContentPane().add(jLabel13);
jLabel13.setBounds(10, 60, 70, 14);

getContentPane().add(Txt_MySQL_Password);
Txt_MySQL_Password.setBounds(90, 60, 120, 18);

jButton1.setText("Login");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

getContentPane().add(jButton1);
jButton1.setBounds(10, 80, 200, 23);

```

```

getContentPane().add(Text5);
Text5.setBounds(50, 110, 160, 19);

jLabel6.setText("Status");
getContentPane().add(jLabel6);
jLabel6.setBounds(10, 110, 40, 14);

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
    },
    new String [] {
        "No", "Tanggal", "Jam", "ID", "Temperature"
    }
));
jScrollPane1.setViewportViewView(jTable1);

getContentPane().add(jScrollPane1);
jScrollPane1.setBounds(10, 170, 630, 140);

jButton3.setText("Go");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

getContentPane().add(jButton3);
jButton3.setBounds(10, 320, 80, 23);

getContentPane().add(List1);
List1.setBounds(380, 20, 260, 140);

Label_Tanggal.setText("Date");
getContentPane().add(Label_Tanggal);
Label_Tanggal.setBounds(220, 20, 140, 20);

Label_Jam.setText("Time");
getContentPane().add(Label_Jam);
Label_Jam.setBounds(220, 40, 140, 20);

getContentPane().add(Text_Temperature);
Text_Temperature.setBounds(220, 80, 150, 19);

Label_Ctr.setText("0");
getContentPane().add(Label_Ctr);
Label_Ctr.setBounds(10, 135, 200, 20);

jButton7.setText("Start");
jButton7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton7ActionPerformed(evt);
    }
});

getContentPane().add(jButton7);

```

```

jButton7.setBounds(220, 110, 150, 23);

jLabel9.setText("Interval (S)");
getContentPane().add(jLabel9);
jLabel9.setBounds(220, 140, 80, 14);

Text_Wkt.setText("0");
getContentPane().add(Text_Wkt);
Text_Wkt.setBounds(300, 140, 70, 20);

jLabel10.setText("TINI");
getContentPane().add(jLabel10);
jLabel10.setBounds(470, 0, 40, 14);

Label_ID.setText("ID");
getContentPane().add(Label_ID);
Label_ID.setBounds(220, 60, 150, 14);

jButton2.setText("Execute");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

getContentPane().add(jButton2);
jButton2.setBounds(10, 350, 80, 23);

Text_SQL.setText("SELECT * FROM Tini_Table WHERE No <10");
getContentPane().add(Text_SQL);
Text_SQL.setBounds(100, 320, 540, 50);

java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
setBounds((screenSize.width-654)/2, (screenSize.height-415)/2, 654, 415);
}
// </editor-fold>

private void Txt_MySQL_UserActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
try{
    Statement statement = connection.createStatement();
    String SQL_1=Text_SQL.getText();
    String sql= SQL_1 + ";";
    ResultSet rs=statement.executeQuery(sql);
    rs.last();

    n=rs.getRow();
    Baris=0;
    //Untuk Nambah Baris Tabel
    jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [n][5],
    new String [] { "No", "Tanggal", "Jam", "ID",
"Temperature" }));

    //Tulis ke Tabel
    rs.beforeFirst();

    while (rs.next()){
        jTable1.setValueAt(rs.getString(1), Baris,0);
        jTable1.setValueAt(rs.getString(2), Baris,1);
        jTable1.setValueAt(rs.getString(3), Baris,2);
        jTable1.setValueAt(rs.getString(4), Baris,3);
        jTable1.setValueAt(rs.getString(5), Baris,4);
        Baris++;
    }
    statement.close();
}

```

```

    }
    catch(Exception e){
        System.out.println("Error :"+e);
    }
}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    String Wkt= Text_Wkt.getText();
    int Wkt1= Integer.parseInt(Wkt);
    delay2=Wkt1*1000;
    //delay2 = 5000;
    timer2 = new Timer(delay2, this);
    timer2.setInitialDelay(delay);
    timer2.setCoalesce(true);
    timer2.start();

    Login();
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    Select_SQL();
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    Login();
}

public void Kirim (){
    try {
        // Socket and streams
        m_socket = new Socket("152.118.167.83",50);
        m_in = new BufferedReader(new InputStreamReader(m_socket.getInputStream()));
        m_out = new PrintWriter(new OutputStreamWriter(m_socket.getOutputStream()));

        List1.add(""+m_in);

        //Terima Text dari Server
        // Memulai Threding
        m_tini_Thread = new Tini_Thread(m_in, this);
        m_tini_Thread.start();

    }
    catch(Exception e) {
        List1.add("error : " + e.toString());
    }
}

public void Login(){
    try{
        Class.forName("org.gjt.mm.mysql.Driver");
        String MySQL_Host = Txt_MySQL_Host.getText();
        String MySQL_User = Txt_MySQL_User.getText();
        String MySQL_Password = Txt_MySQL_Password.getText();
        connection = DriverManager.getConnection("jdbc:mysql://" +
MySQL_Host + "/"TINI_DB", MySQL_User , MySQL_Password );
        // connection.close();
        Text5.setText("You Are Connected");
    }
    catch(Exception e){
        Text5.setText("Your not connected !!! :"+e);
    }
}

public void Select_SQL(){
    try{
        Statement statement = connection.createStatement();
        String sql=("SELECT * FROM Tini_Table;");
    }
}

```

```

ResultSet rs=statement.executeQuery(sql);
rs.last();
n=rs.getRow();
Baris=0;
//Untuk Nambah Baris Tabel
jTable1.setModel(new javax.swing.table.DefaultTableModel(
new Object [n][5],
new String [] { "No", "Tanggal", "Jam", "ID",
"Temperature" }));

//Tulis ke Tabel
rs.beforeFirst();

while (rs.next()){
    jTable1.setValueAt(rs.getString(1), Baris,0);
    jTable1.setValueAt(rs.getString(2), Baris,1);
    jTable1.setValueAt(rs.getString(3), Baris,2);
    jTable1.setValueAt(rs.getString(4), Baris,3);
    jTable1.setValueAt(rs.getString(5), Baris,4);
    Baris++;
}
statement.close();
}
catch(Exception e){
    System.out.println("Error :"+e);
}
}
public void Add_Data(){
    try{
        String Tanggal=Label_Tanggal.getText();
        //No++;
        String Waktu=Label_Jam.getText();
        String Temperature=Text_Temperature.getText();
        String ID = Label_ID.getText();
        Statement statement = connection.createStatement();
        String sql="insert into Tini_Table
('"+No+"', '"+Tanggal+"', '"+Waktu+"', '"+ID+"', '"+Temperature+"')";
statement.executeUpdate(sql);
statement.close();
Text5.setText(nomor + ". Data Added" );
Select_SQL();
}
catch(Exception e){
    System.out.println("Error :"+e);
}
}
public void log(java.lang.String strMsg) {
List1.add(strMsg + "\n");
String iButton_ID = strMsg;
String iButton_ID_Pars = strMsg.substring(0,16);

String waktu = strMsg.substring(28,36);
Label_Jam.setText(waktu);

String Tahun=strMsg.substring(41,45);
String tanggal=strMsg.substring(21,27);
Label_Tanggal.setText(tanggal + " " + Tahun);

String temperature =strMsg.substring(46);
Text_Temperature.setText(temperature);

Label_ID.setText(iButton_ID_Pars);
Add_Data();
Select_SQL();

}
//Sub Timer ()
public void actionPerformed(ActionEvent e) {
/*
if (e.getSource()==timer)
{

```

```

//Untuk Tanggal
String timeTxt = formatter.format(gCal.getTime());
Label_Tanggal.setText(timeTxt);

//Untuk Jam
gCal.add(Calendar.SECOND,1);
String timeTxt2 = formatter2.format(gCal.getTime());
Label_Jam.setText(timeTxt2);
}
*/
if(e.getSource()==timer2 )
{
//Counter 2
nomor++;
Label_Ctr.setText(""+nomor);
Kirim();
}
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            JFrame.setDefaultLookAndFeelDecorated(true);
            new Tini_Client().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JLabel Label_Ctr;
private javax.swing.JLabel Label_ID;
private javax.swing.JLabel Label_Jam;
private javax.swing.JLabel Label_Tanggal;
private java.awt.List List1;
private javax.swing.JTextField Text5;
private javax.swing.JTextArea Text_SQL;
private javax.swing.JTextField Text_Temperature;
private javax.swing.JTextField Text_Wkt;
private javax.swing.JTextField Txt_MySQL_Host;
private javax.swing.JPasswordField Txt_MySQL_Password;
private javax.swing.JTextField Txt_MySQL_User;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton7;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel19;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
// End of variables declaration
//Untuk Kalender
private GregorianCalendar gCal = new GregorianCalendar();
private String timeFormat = "dd MMM yyyy";
private String timeFormat2 = "hh:mm:ss ";
private SimpleDateFormat formatter = new SimpleDateFormat(timeFormat);
private SimpleDateFormat formatter2 = new SimpleDateFormat(timeFormat2);
/** Socket */
private Socket m_socket = null;
/** Socket's input stream */
private BufferedReader m_in = null;
/** Socket's output stream */
private PrintWriter m_out = null;
/** Client thread */
private Tini_Thread m_tini_Thread = null;
}

```

The Tini_Thread.java
Threads program at user interface main program

```

/*
 * ClientThread.java
 * @author Surya Darma
 * Created on 22. October 2005, 22:36
 */

import java.io.*;
import java.net.*;

public class Tini_Thread extends Thread {

    public Tini_Thread(BufferedReader in, Tini_Client tini_Client) {
        m_in = in;
        m_tini_Client = tini_Client;
    }

    /**
     * Start thread */
    public void run() {
        String strBuffer;
        for(;;) {
            try {
                strBuffer = m_in.readLine();
                if(strBuffer != null) {
                    m_tini_Client.log(strBuffer);
                }
                else {
                    System.out.println("Connection closed");
                    break;
                }
            }
            catch(IOException e) {
                System.out.println("IOException by server");
                break;
            }
        }
        //m_tini_Client.closeConnection();
        System.out.println("Bye");
    }

    /** Input stream from socket */
    private BufferedReader m_in;
    /** Tini_Client for callbacks */
    private Tini_Client m_tini_Client;
}

```


The SerialComm.java TINI® main program

```

/*
 * SerialComm.java
 * @author surya
 * Created on December 9, 2005, 8:37 PM
 */
package serialcomm;

import java.net.ServerSocket;
import javax.comm.*;
import java.io.*;
import java.net.*;
import java.util.*;

import com.dalsemi.tininet.*;
import com.dalsemi.onewire.OneWireAccessProvider;
import com.dalsemi.onewire.*;
import com.dalsemi.onewire.adapter.*;
import com.dalsemi.onewire.container.*;
//import com.dalsemi.onewire.container.OneWireContainer10;

public class SerialComm
{
    public static void main(String[] args) {
        ServerSocket s;
        SerialComm ct = new SerialComm();
        OneWireContainer10 dev = null;
        ct.run();
    }

    private static OneWireContainer10 getOneWireDevice()
    throws OneWireException {
        // get access provider for 1-Wire devices
        DSPortAdapter pa = OneWireAccessProvider.getDefaultAdapter();
        // get exclusive use of adapter
        pa.beginExclusive(true);
        // clear any previous search restrictions
        pa.setSearchAllDevices();
        pa.setSpeed(pa.SPEED_REGULAR);
        // target family 10
        pa.targetFamily(0x10);
        // get devices
        Enumeration e = pa.getAllDeviceContainers();
        OneWireContainer10 a[] = new OneWireContainer10[10];
        // at least one device?
        if (e.hasMoreElements() == false) {
            System.out.println("Error: cannot find the 1-Wire
device");
            System.exit(1);
        }
        // return it
        return (OneWireContainer10)e.nextElement();
    }
}

```

```

public void run()
{
    OneWireContainer10 dev = null;
    try {
        CommPortIdentifier cpi =
CommPortIdentifier.getPortIdentifier("serial0");
        SerialPort sp = (SerialPort)cpi.open("SerialComm", 5000);

sp.setSerialPortParams(9600,SerialPort.DATABITS_8,SerialPort.STOPB
ITS_1,SerialPort.PARITY_NONE);
        while (true)
        {
            OutputStream os = sp.getOutputStream();
            byte tes = 'S' ;
            os.write(tes) ;
            InputStream in = sp.getInputStream();
            sp.enableReceiveThreshold(1024);
            sp.enableReceiveTimeout(1000);

            // Connect to the host machine for sending data.
            ServerSocket s;
            s = new ServerSocket(50);

            // Array to be used for data read.
            byte[] data = new byte[1024];

            Socket incoming = s.accept();
            PrintWriter bos;
            bos = new
PrintWriter(incoming.getOutputStream(),true);
            int num = in.read(data);

            // send it
            // data configuration (from AVR)= String address -
space - temperature
            String address = new String(data,0,16) ;
            String suhu = new String(data,17,19) ;
            bos.println(address + " " +""+ new Date() + " " +
suhu + "");

            // Dari iButton2
            try {
            // get 1-Wire device
                dev = getOneWireDevice();
                } catch (OneWireException e) {
                    System.out.println("TemperatureServer error: "
+ e) ;
                }

            byte[] state = dev.readDevice();
            // start temperature conversion
            dev.doTemperatureConvert(state);
            dev.writeDevice(state);
            // get the temperature
            double temperature = dev.getTemperature(state);
            // send it

```

```
        bos.println(dev.getAddressAsString() + " " + "" +  
new Date() + " " + temperature);  
        bos.close();  
        incoming.close();  
        s.close() ;  
    } //end while  
    } catch(Exception e)  
    {  
        System.out.println("GOT AN EXCEPTION = " +  
e.getMessage());  
    }  
    } //end of run()  
}
```

