LAMPIRAN A

FOTO ALAT

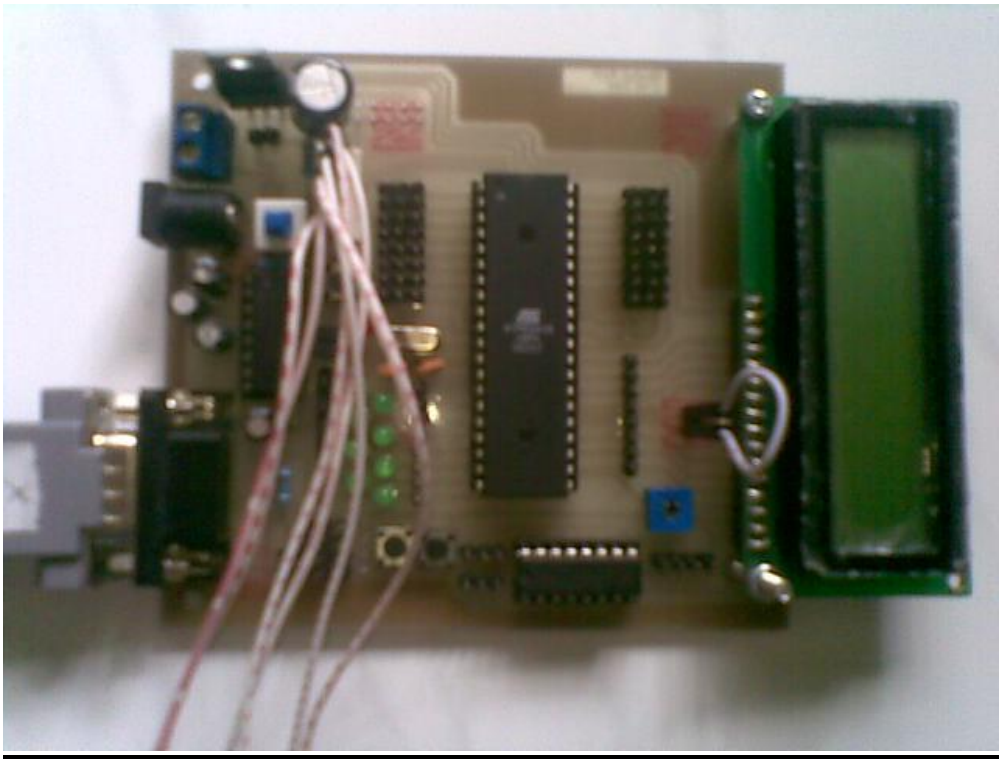

Foto 1 – Modul Mikrokontroler ATmega16 & LCD 16x2

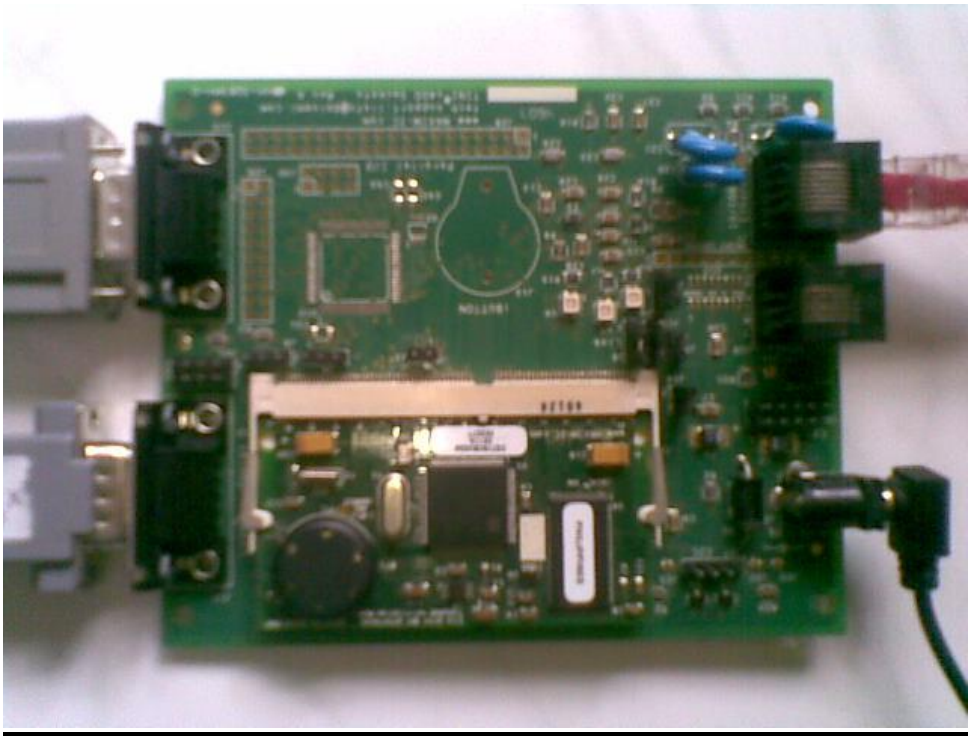

Foto 2 - Modul TINI400

Foto 3 - Modem GPRS

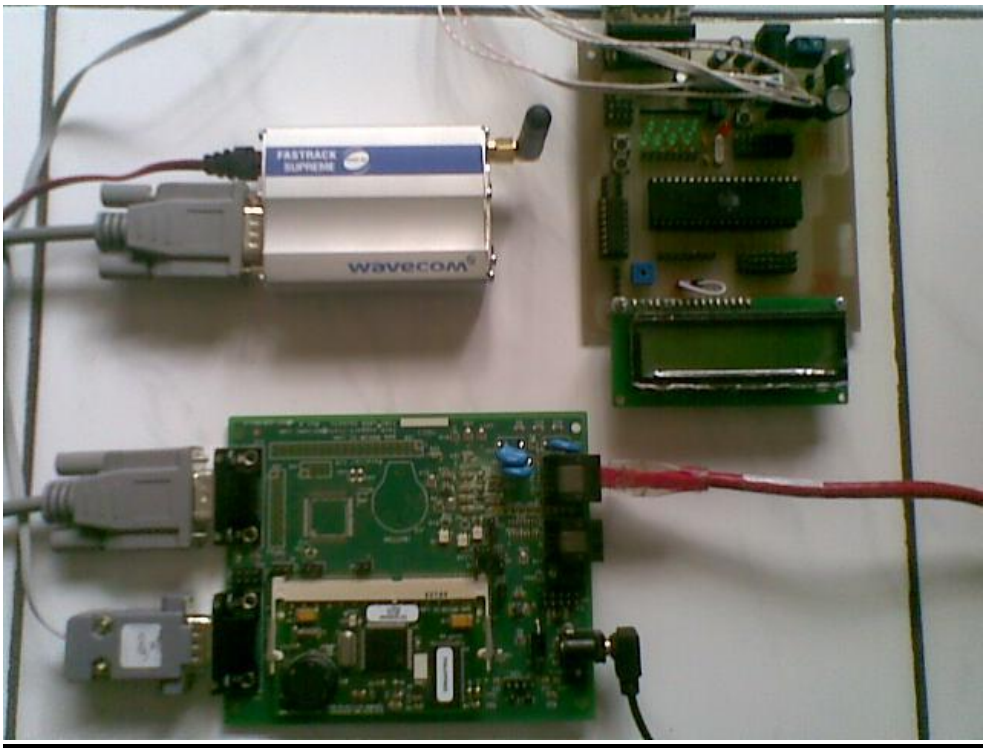

Foto 4 - Model Sistem Keseluruhan

LAMPIRAN B

FOTO TAMPILAN HALAMAN WEB

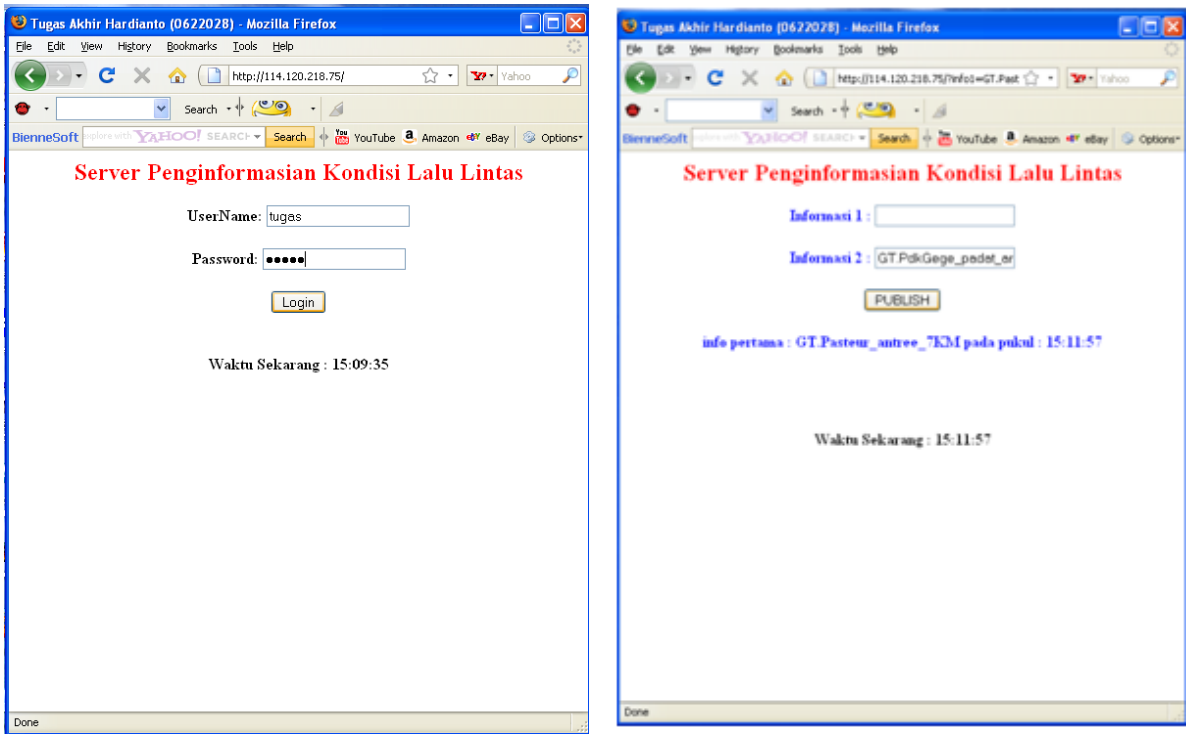

Foto 1 - Tampilan Halaman *Web* pada *Web Browser* PC

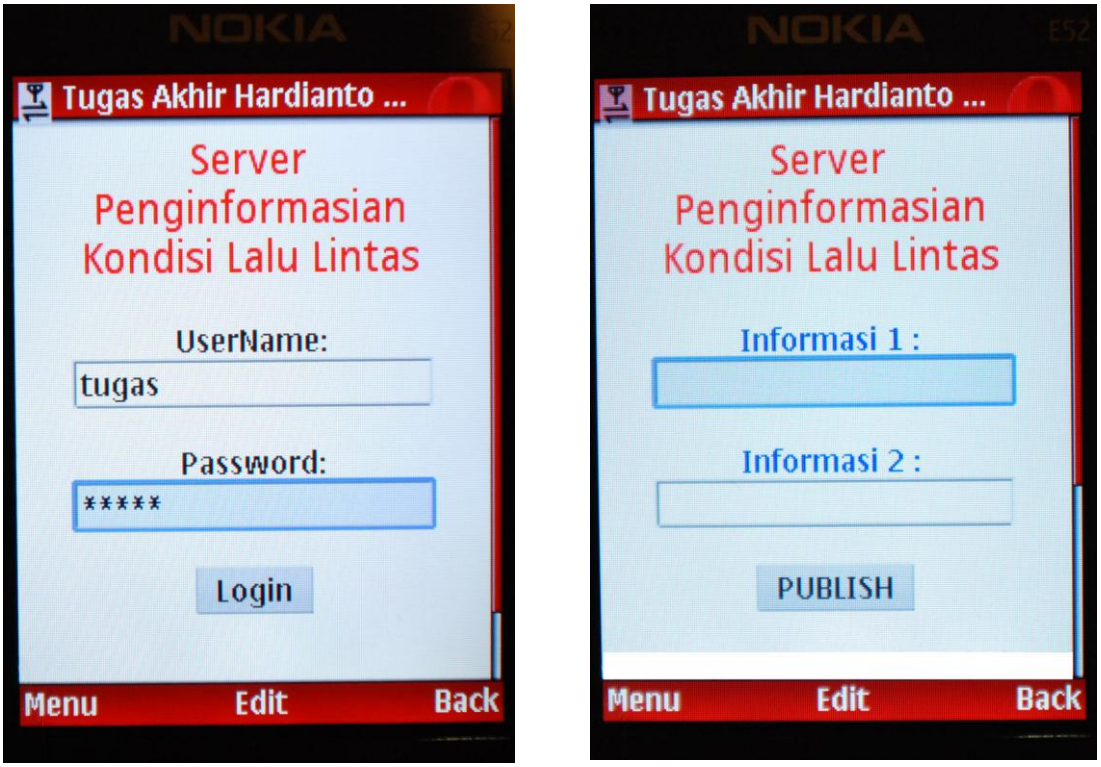

Foto 2 - Tampilan Halaman *Web* pada *Web Browser* Telepon Seluler

LAMPIRAN C

KODE PROGRAM PADA TINI400

Utama.java

```
import java.util.*;
import java.io.*;
import java.net.*;
import java.lang.*;
import com.dalsemi.system.*;
import com.dalsemi.shell.server.*;

public class Utama
{
Sistem sistem;
Modem modem;

int port                    = 80;
int info1;
int info2;

boolean IsiPertama          =false;
boolean IsiKedua            =false;
static boolean login        =false;

Object lock;

static String user = "";
static String pass = "";
static String variabel1 = "";
static String variabel2 = "";

public Utama(){}

public static void writePage(BufferedWriter wr)
{
try
{
Calendar calendar = Calendar.getInstance();
int hour    = calendar.get(Calendar.HOUR_OF_DAY);
int minute  = calendar.get(Calendar.MINUTE);
int second  = calendar.get(Calendar.SECOND);
String timestring = new String("Waktu Sekarang : " + hour + ":");
if(minute < 10) timestring += "0";
timestring += Integer.toString(minute);
timestring += ":";
if(second < 10) timestring += "0";
timestring += Integer.toString(second);
```

```
String jam = new String("pukul : " + hour + ":");
if(minute < 10) jam += "0";
jam += Integer.toString(minute);
jam += ":";
if(second < 10) jam += "0";
jam += Integer.toString(second);
wr.write("HTTP/1.0 200 OK\n");
wr.write("Content-type: text/html\n\n");
wr.write("<HTML>\n");
wr.write("<HEAD>\n");
wr.write("<TITLE>Tugas Akhir Hardianto (0622028)</TITLE>\n");
wr.write("</HEAD>\n");
wr.write("<BODY COLOR='green' BGCOLOR='white'>\n");
wr.write("<h2 ALIGN='CENTER'><font COLOR='red'>Server Penginformasian
          Kondisi Lalu Lintas </font></h4>\n");
wr.write("<form>");
if(!login)
{
wr.write("<h4 ALIGN='CENTER'> Name:  <INPUT TYPE='text'
          NAME='Name'>\n");
wr.write("<h4 ALIGN='CENTER'> Password:  <INPUT TYPE='password'
          NAME='Password'>\n");
wr.write("<h4 ALIGN='CENTER'><INPUT TYPE='submit' NAME='login'
          VALUE='OK'><br>\n");
}
else
{
wr.write("<font color='blue'>\n");
wr.write("<h4 ALIGN='CENTER'> Informasi 1 :    <INPUT  TYPE='TEXT'
          NAME='info1' MAXLENGTH='32'>\n");
wr.write("<h4 ALIGN='CENTER'> Informasi 2 :    <INPUT  TYPE='TEXT'
          NAME='info2' MAXLENGTH='32'>\n");
wr.write("<h4 ALIGN='CENTER'><INPUT TYPE='submit' NAME='tombol1'
          VALUE='PUBLISH'><br>\n");
wr.write("<h4 ALIGN='CENTER'>info pertama : " + variabel1 + " <br>\n");
wr.write("<h4 ALIGN='CENTER'>info kedua : " + variabel2 + " <br>\n");
wr.write("</font>\n");
}
wr.write("<form><br>\n\n");
wr.write("</font></h4>\n");
wr.write("<h4 ALIGN='CENTER'>"+timestring+"</h4>\n");
wr.write("</CENTER></BODY></HTML>\n");
wr.flush();
}
catch (IOException e) {}
}
```

```
public void Running()
{
try
{
modem.setSerialPort(new Integer("4"));
modem.setUsername("wap");
modem.setPassword("wap123");
modem.setPhoneNumber("*99***1#");
Thread sistemClass = new Thread(sistem);
sistemClass.setName("Sistem");
Thread modemClass = new Thread(modem);
modemClass.setName("Modem");
sistemClass.start();
modemClass.start();
Thread.sleep(2000);
ServerSocket srv = new ServerSocket(port);
while(true)
{
try
{
Socket mySocket = srv.accept();
BufferedWriter serverResponse =
newBufferedWriter(newOutputStreamWriter(mySocket.getOutputStream()));
BufferedReader clientRequest =
newBufferedReader(newInputStreamReader(mySocket.getInputStream()));
String str, temp, result = "";
str=clientRequest.readLine();
System.out.println(str);
sistem.setConnectionWeb(4);
Calendar calendar = Calendar.getInstance();
int hour    = calendar.get(Calendar.HOUR_OF_DAY);
int minute  = calendar.get(Calendar.MINUTE);
String jam = new String("_pk." + hour + ":");
if(minute < 10) jam += "0";
jam += Integer.toString(minute);
if (str.startsWith("GET /?info1"))
{
temp = str;
int i=0;
temp = temp.substring(12);
char[] tempChar = temp.toCharArray();
while(tempChar[i] != '&'){
result += tempChar[i];
i++;
}
variabel1 = result;
```

```
if(variabel1 != "")
{
variabel1 = result + jam;
}
while(tempChar[i]!='=')
{
i++;
}
i++;
result = "";
while(tempChar[i] != '&')
{
result += tempChar[i];
i++;
}
variabel2 = result;
if(variabel2 != "")
{
variabel2 = result + jam;
}
System.out.println("variabel1 : "+ variabel1);
System.out.println("variabel2 : "+ variabel2);
sistem.setDataInfo(variabel1, variabel2);
writePage(serverResponse);
}
else if(str.startsWith("GET /?Name"))
{
temp = str;
int i=0;
temp = temp.substring(11);
char[] tempChar = temp.toCharArray();
while(tempChar[i] != '&')
{
result += tempChar[i];
i++;
}
user = result;
while(tempChar[i]!='=')
{
i++;
}
i++;
result = "";
while(tempChar[i] != '&')
{
result += tempChar[i];
```

```
i++;
}
pass = result;
System.out.println("user : "+ user);
System.out.println("pass : "+ pass);
if(user.equals("root")&&pass.equals("tini")){
login = true;
}
else
{
login = false;
}
writePage(serverResponse);
}
clientRequest.close();
mySocket.close();
}
catch(Exception e)
{
System.out.println("Connection Socket Error !!");
System.out.println(e.toString());
}
}
}
catch(Exception e)
{
System.out.println("Error to Initial Thread !!");
System.out.println(e.toString());
}
}

public static void main(String[] args)
{
try
{
System.loadLibrary("mod_ppp_400.tlib");
Utama utama             = new Utama();
utama.sistem            = new Sistem();
utama.modem             = new Modem();
utama.Running();
}
catch(Throwable t){}
}
}
```

Sistem.java

```
import java.io.*;
import javax.comm.*;
import java.util.*;

public class Sistem implements Runnable
{
String info1                    =  "";
String info2                    =  "";
int WebConnect                  = 0;
boolean plant                   = false;
boolean WebConnection           = false;
InputStream                 in;
OutputStream                output;
SerialPort                  sp;
static Enumeration          portList;
static CommPortIdentifier   portIdTemp;
public Sistem()
{
try
{
portList = CommPortIdentifier.getPortIdentifiers();
while (portList.hasMoreElements())
{
portIdTemp = (CommPortIdentifier) portList.nextElement();
System.out.println(portIdTemp.getName());
}
CommPortIdentifier portId = CommPortIdentifier.getPortIdentifier("serial0");
sp = (SerialPort)portId.open("PLANT", 2000);
sp.setSerialPortParams(9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
            SerialPort.PARITY_NONE);
sp.setFlowControlMode(sp.FLOWCONTROL_NONE);
System.out.println();
System.out.println();
System.out.println("Open Serial with System....");
}
catch(Exception e){System.out.println(e.toString());}
}
public void initwritetoport()
{
try
{
output = sp.getOutputStream();
}
catch(IOException e){
```

```
System.out.println(e.toString());}
return;
}
public void setData()
{
try
{
String str;
if(info1 != "")
{
str = "1" + info1 ;
if(str.length()  < 32)
{
int i = str.length();
while(i<32)
{
str += "#";
str += info1;
i += info1.length()+1;
}
}
byte[] data = str.getBytes();
output.write(data);
output.flush();
}
if(info2 != "")
{
str = "2" + info2;
if(str.length()  < 32)
{
int i = str.length();
while(i<32)
{
str += "#";
str += info2;
i += info2.length()+1;
}
}
byte[] kata = str.getBytes();
output.write(kata);
output.flush();
}
}
catch(IOException e){
System.out.println(e.toString());
return;
```

```
}
}
public void setConnectionWeb(int Web)
{
WebConnection=true;
WebConnect=Web;
}
public void setDataInfo(String info1, String info2)
{
this.info1=info1;
this.info2=info2;
}

public void run()
{
plant=false;
int x = 0;
initwritetoport();
while(true)
{
try
{
Thread.sleep(100);
if (!plant)
{
plant    =         true;
System.out.println("System Model Connected");
}
else
{
Thread.sleep(20);
if (WebConnection)
{
Thread.sleep(20);x=0;WebConnection=false;
}
setData();
Thread.sleep(20);
}
x=x+1;
if (x==75) {Thread.sleep(20);x=0;
}
}
catch(Throwable t){}//System.out.println(t.toString())
}
}
}
```

Modem.java

```
import java.io.*;
import java.util.*;
import javax.comm.*;
import com.dalsemi.system.*;
import com.dalsemi.tininet.ppp.*;

public class Modem extends Thread implements PPPEventListener,
                    CommPortOwnershipListener
{
private CommPortIdentifier portId;
static SerialPort serialPort                = null;
private InputStream modemInputStream;
private OutputStream modemOutputStream;
private int portNumber;
PPP ppp                                     = null;
private boolean connected                   = false;
private boolean running                     = true;
private byte[] localAddress                 = new byte[] {0, 0, 0, 0};
private byte[] remoteAddress                = new byte[] {0, 0, 0, 0};
private String interfaceName                = "pppClient";
public String username;
public String password;
public String phoneNumber;
private boolean haveControlLines    = false;
private boolean carrierLost                 = false;
private boolean interfaceActive             = false;

private ModemCommand[] dialSequence =
{
new ModemCommand("AT\r",          "OK",      5),
new ModemCommand("ATZ\r",         "OK",      5),
new ModemCommand("ATM1L0\r",      "OK",      5),
};

public void run()
{
boolean callServer = false;
boolean closeConnection = false;
String optionParam;
System.out.println("Starting PPP....");
ppp = new PPP();
if (openSerialPort(portNumber) == false)
{
closeSerialPort();
```

```
return;
}
try
{
ppp.addEventListener(this);
}
catch(Exception e)
{
System.out.println("Exception adding event listener");
}
ppp.setLocalAddress(localAddress);
ppp.setRemoteAddress(remoteAddress);
ppp.setRemoteAccm(0x00000000);
ppp.setLocalAccm(0x00000000);
ppp.setAuthenticate(false, true);
try
{
ppp.setUsername(username);
ppp.setPassword(password);
}
catch(Exception e)
{
System.out.println("Exception setting username/password");
}
System.out.println("Dialing ISP....");
for (int i = 0; i < dialSequence.length; ++i)
{
if (!atCommand(dialSequence[i]))
{
System.out.println("Modem did not respond to command " +
                    dialSequence[i].command);
System.out.println();
System.out.println("Please Shutdown Modem GPRS and Restart TINI400 Module
                    !!");
System.out.println();
System.out.println();
return;
}
}

if (!atCommand(new ModemCommand("ATDT" + phoneNumber + "\r",
                    "CONNECT", 45)))
{
closeSerialPort();
System.out.println("Modem unable to establish connection !!");
```

```
System.out.println();
System.out.println("Please Shutdown Modem GPRS
                    and Restart TINI400 Module !!");
System.out.println();
System.out.println();
return;
}
System.out.println("Connected to ISP");
connected = true;
ppp.up(serialPort);
while (running)
{
try { Thread.sleep(1000); } catch (Exception e){}
if (connected && haveControlLines && !carrierLost)
{
if (!serialPort.isCD())
{
ppp.close();
carrierLost = true;
}
}
if (!connected)
{
System.out.println("closePPP");
closePPP();
closeSerialPort();
running = false;
}
}
}

public void pppEvent(PPPEvent ev)
{
switch (ev.getEventType())
{
case PPPEvent.UP:
System.out.println("PPP IS UP");
System.out.println();
System.out.println();
System.out.println("System Model Ready to Control and Monitor");
System.out.println();
System.out.println();
ppp.addInterface(interfaceName);
ppp.setDefaultInterface(true);
interfaceActive = true;
break;
```

```
case PPPEvent.CLOSED:
System.out.println("PPP IS DOWN");
if (interfaceActive)
{
interfaceActive = false;
ppp.removeInterface(interfaceName);
}
connected = false;
break;
default:
break;
}
}
private void closePPP()
{
ppp.freePort();
ppp.finish();
try{Thread.sleep(2000);}
catch (Exception _){}
ppp.removeEventListener(this);
}

private void closeSerialPort()
{
resetModem();
serialPort.close();
portId.removePortOwnershipListener(this);
}

private boolean openSerialPort(int portNum)
{
try
{
if ( portNum > 0 && portNum < 4 )
TINIOS.setSerial(TINIOS.SERIAL_SET_ENABLE, portNum, true);
}
catch (UnsupportedCommOperationException _)
{
System.out.println("Exception enabling serial port " + portNum);
return false;
}
try
{
portId = CommPortIdentifier.getPortIdentifier("serial" + portNum);
serialPort = (SerialPort)portId.open(interfaceName, 5000);
serialPort.setSerialPortParams(115200,
```

```
                    SerialPort.DATABITS_8,
                    SerialPort.STOPBITS_1,
                    SerialPort.PARITY_NONE);
portId.addPortOwnershipListener(this);
if (haveControlLines)
{
TINIOS.setSerial(TINIOS.SERIAL_SET_RTSCTS_FLOW_CONTROL, 4,
                    false);
TINIOS.setSerial(TINIOS.SERIAL_SET_RTSCTS_FLOW_CONTROL, 1,
                    false);
TINIOS.setSerial(TINIOS.SERIAL_SET_RTSCTS_FLOW_CONTROL, 0,
                    false);
TINIOS.setSerial(TINIOS.SERIAL_SET_RTSCTS_FLOW_CONTROL,
                    portNum, true);
serialPort.setFlowControlMode(serialPort.FLOWCONTROL_RTSCTS_IN |
                    serialPort.FLOWCONTROL_RTSCTS_OUT);
}
else
{
serialPort.setFlowControlMode(serialPort.FLOWCONTROL_NONE);
}
modemInputStream = serialPort.getInputStream();
modemOutputStream = serialPort.getOutputStream();
long available = modemInputStream.available();
if (available > 0)
modemInputStream.skip(available);
modemOutputStream.flush();
resetModem();
}
catch (Exception e)
{
System.out.println(e.toString());
return false;
}
return true;
}

private boolean waitFor(String expect, String response, int timeout)
{
boolean done = false;
boolean returnValue = false;
int numBytes;
byte[] readBuffer = new byte[40];
int bytesAvailable;
String buffer = new String();
timeout *= 2;
```

```
while (done == false)
{
try
{
Thread.sleep(500);
if ((bytesAvailable = modemInputStream.available()) > 0)
{
numBytes = modemInputStream.read(readBuffer, 0, bytesAvailable);
buffer = buffer.concat(new String(readBuffer, 0, numBytes));
if (buffer.indexOf(expect) != -1)
{
try
{
if (response != null)
modemOutputStream.write(response.getBytes());
}
catch(Exception e)
{
System.out.println(e.toString() + " during write");
}
returnValue = true;
done = true;
}
}
else
{
if (timeout != 0)
{
if (--timeout == 0)
{
returnValue = true;
done = true;
}
}
}
}
catch (Exception e)
{
System.out.println(e.toString());
}
}
return returnValue;
}

public boolean atCommand(ModemCommand mc)
{
```

```
boolean returnValue = false;
String response = new String();
int timeout = mc.timeout * 2;
try
{
try
{
modemOutputStream.write(mc.command.getBytes());
}
catch(Exception e)
{
System.out.println("atCommand write " + e.toString());
return false;
}
boolean done = false;
while (done == false)
{
Thread.sleep(500);
int bytesAvailable = modemInputStream.available();
if (bytesAvailable > 0)
{
byte[] readBuffer = new byte[bytesAvailable];
int numBytes = modemInputStream.read(readBuffer, 0, bytesAvailable);
response = response.concat(new String(readBuffer, 0, numBytes));
if (response.indexOf(mc.response) != -1)
{
returnValue = true;
done = true;
}
}
else
{
// If not waiting forever
if (timeout > 0)
{
// Decrement seconds counter
if (--timeout == 0)
{
returnValue = false;
done = true;
}
}
}
}
}
catch (Exception e)
```

```
{
System.out.println("atCommand all " + e.toString());
}
return returnValue;
}

private void resetModem()
{
if (haveControlLines)
{
serialPort.setDTR(false);
serialPort.setRTS(false);
try { Thread.sleep(2000);
}
catch(Exception _){}
serialPort.setDTR(true);
serialPort.setRTS(true);
try
{ Thread.sleep(2000);
}
catch(Exception _){}
}
else
{
try { Thread.sleep(2000); }
catch(Exception _){}
atCommand(new ModemCommand("+++", "", 2));
try { Thread.sleep(2000); }
catch(Exception _){}
atCommand(new ModemCommand("ATH\r", "OK", 2));
}
}

public void ownershipChange(int ev)
{
switch (ev)
{
case CommPortOwnershipListener.PORT_OWNED:
break;
case CommPortOwnershipListener.PORT_OWNERSHIP_REQUESTED:
ppp.close();
closePPP();
try
{
CommPortIdentifier portId = CommPortIdentifier.getPortIdentifier("serial" +
                    portNumber);
```

```
portId.removePortOwnershipListener(this);
}
catch(Exception _){}
break;
case CommPortOwnershipListener.PORT_UNOWNED:
break;
default:
break;
}
}

public void setUsername(String username)
{
this.username = username;
}

public void setPassword(String password)
{
this.password = password;
}

public void setPhoneNumber(String phoneNumber)
{
this.phoneNumber = phoneNumber;
}

public void setSerialPort(Integer portNumber)
{
this.portNumber = portNumber.intValue();
}
}
class ModemCommand
{
String command;
String response;
int timeout;
private ModemCommand(){};
public ModemCommand(String command, String response, int timeout)
{
this.command = command;
this.response = response;
this.timeout = timeout;
}
}
```

LAMPIRAN D

KODE PROGRAM PADA ATMEGA16

```
#include <mega16.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
unsigned int j;
signed char rf1[32],rf2[32],isi[2];

int z,y;

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
```

```
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR |
DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      };
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

void main(void)
{
    char text[32];

    // Declare your local variables here
```

```
    // Input/Output Ports initialization
    // Port A initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTA=0x00;
    DDRA=0x00;

    // Port B initialization
    // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
    // State7=1 State6=1 State5=1 State4=1 State3=1 State2=1 State1=1 State0=1
    PORTB=0xFF;
    DDRB=0xFF;

    // Port C initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTC=0x00;
    DDRC=0x00;

    // Port D initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
```

```
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
```

```
UBRRL=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// LCD module initialization
lcd_init(16);
// Global enable interrupts
#asm("sei")

while (1)
{
 // Place your code here
scan:
lcd_clear();
PORTB=0xFF;
for(j=0;j<1;j++)
{
scanf("%c",&isi[j]);
}
if (isi[0]=='1')goto lcd1;
if (isi[0]=='2')goto lcd2;
else
{
lcd_putsf("Tunggu Info");
delay_ms(100);
goto scan;
}

lcd1:
for(j=0;j<32;j++)
{
scanf("%c",&rf1[j]);
}
for (z=0;z<17;z++)
{
lcd_gotoxy(0,0);
sprintf(text,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",rf1[z],rf1[z+1],
        rf1[z+2],rf1[z+3],rf1[z+4],rf1[z+5],rf1[z+6],rf1[z+7],rf1[z+8],rf1[z+9],
        rf1[z+10],rf1[z+11],rf1[z+12],rf1[z+13],rf1[z+14],rf1[z+15];
lcd_puts(text);
delay_ms(500);
lcd_clear();
}
```

```
goto scan;

lcd2:
for(j=0;j<32;j++)
{
 scanf("%c",&rf2[j]);
}
for (y=0;y<17;y++)
{
lcd_gotoxy(0,1);
sprintf(text,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",rf2[y],rf2[y+1],
          rf2[y+2],rf2[y+3],rf2[y+4],rf2[y+5],rf2[y+6],rf2[y+7],rf2[y+8],rf2[y+9],
          rf2[y+10],rf2[y+11],rf2[y+12],rf2[y+13],rf2[y+14],rf2[y+15]);
lcd_puts(text);
delay_ms(500);
lcd_clear();
}
goto scan;
};
}
```