LAMPIRAN A
PROGRAM PADA MIKROKONTROLER ATMEGA16

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 11/9/2011
Author  : F4CG
Company : F4CG
Comments:

Chip type           : ATmega16
Program type        : Application
Clock frequency     : 11.059200 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>

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

// Declare your global variables here
void main(void)
{
// Declare your local variables here
unsigned char d,p;

// 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=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
PORTB=0x00;
DDRB=0x00;

// 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=0xFF;

// 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=0x18;
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;
printf("\n Password");


while (1)
      {
      // Place your code here
      p=getchar();
      if( p=='2')
      {  printf("\n Password Benar");
           d=getchar();
           if (d=='A' )
              {
              PORTC.0=1;
                    if(PORTD.4 ==0)
                      {
                      delay_ms(200);
                      printf("Lampu 1 Off");
                      }
                    else
                    {
                     delay_ms(200);
                     printf("Lampu 1 On");
                     }
             }
           if (d=='D')
              {
              PORTC.0=0;
                     if(PORTD.4 ==0)
                     {
                     delay_ms(200);
                     printf("Lampu 1 Off");
                     }
                    else
                    {
                     delay_ms(200);
                     printf("Lampu 1 On");
                     }



              }
if (d=='G' )
              {
              PORTC.1=1;
                    if(PORTD.5 ==0)
```

```
                {
                delay_ms(200);
                printf("Lampu 2 Off");
                }
              else
              {
               delay_ms(200);
               printf("Lampu 2 On");
               }
          }
        if (d=='J')
           {
           PORTC.1=0;
                 if(PORTD.5 ==0)
                 {
                 delay_ms(200);
                 printf("Lampu 2 Off");
                 }
              else
              {
               delay_ms(200);
               printf("Lampu 2 On");
               }
           }
        if (d=='M' )
           {
           PORTC.2=1;
              if(PORTD.6 ==0)
                 {
                 delay_ms(200);
                 printf("Lampu 3 Off");
                 }
              else
              {
               delay_ms(200);
               printf("Lampu 3 On");
               }



}
        if (d=='P')
           {
           PORTC.2=0;
                 if(PORTD.6 ==0)
                 {
```

```
                    delay_ms(200);
                    printf("Lampu 3 Off");
                    }
                  else
                  {
                   delay_ms(200);
                   printf("Lampu 3 On");
                   }
            }
    if (p!='2')
    {
    printf("\n Password Salah");
    }
    };
}
```

LAMPIRAN B
PROGRAM PADA *HANDPHONE* (J2ME)

BluetoothDeviceDiscovery

```
import java.io.IOException;
import java.util.Vector;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;

public class BluetoothDeviceDiscovery implements DiscoveryListener
{

	public static Object lock=new Object();

	public static Vector vecDevices=new Vector();

	public static void main() throws IOException
	{
		BluetoothDeviceDiscovery bluetoothDeviceDiscovery=new
BluetoothDeviceDiscovery();
		LocalDevice localDevice = LocalDevice.getLocalDevice();
		DiscoveryAgent agent = localDevice.getDiscoveryAgent();
		agent.startInquiry(DiscoveryAgent.GIAC,
bluetoothDeviceDiscovery);
		try
		{
			synchronized(lock)
			{
				lock.wait();
			}
		}
		catch (InterruptedException e)
		{
			e.printStackTrace();
		}
	}

	public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
	{
		if(!vecDevices.contains(btDevice))
```

```
                {
                        vecDevices.addElement(btDevice);
                }
        }

        public void servicesDiscovered(int transID, ServiceRecord[] servRecord)
        {
        }

        public void serviceSearchCompleted(int transID, int respCode)
        {
        }

        public void inquiryCompleted(int discType)
        {
                synchronized(lock)
                {
                        lock.notify();
                }
        }
}
```

TerminalIOStream

```
import javax.microedition.midlet.MIDlet;

public class TerminalIOStream
{
   StringBuffer line;
   private boolean full;

   public TerminalIOStream()
        {
      reset();
   }

   public void append(char c)
        {
      line.append(c);
   }

   public void append(char [] l)
        {
      line.append(l);
   }
```

```
public synchronized String read()
        {
    try
                        {
        while (full == false)
                                {
                                wait();
                                }
                        }
                catch (Exception e)
                        {
                        e.printStackTrace();
                        }
    String s = line.toString();
    reset();
    return s;
}

public void write(String s)
        {
    line.append(s);
    flush();
}

public void reset()
        {
    line = new StringBuffer();
    full = false;
}

public synchronized boolean isFull()
        {
    return full;
}
```

Lampu

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
import javax.bluetooth.*;
import javax.microedition.io.Connector;
import java.io.IOException;
import java.util.Stack;
```

```
public class lampu extends MIDlet implements Runnable,
javax.microedition.lcdui.CommandListener {

        public String output;
    data data;

    public lampu()
        {
    }

    private void initialize()
        {
    }

    public void commandAction(javax.microedition.lcdui.Command command,
javax.microedition.lcdui.Displayable displayable)
        {
        if (command == exitCommand1)
                {
            exitMIDlet();
        }
    }

    private javax.microedition.lcdui.Command get_exitCommand()
        {
        if (exitCommand == null) {
            exitCommand = new javax.microedition.lcdui.Command("Exit",
javax.microedition.lcdui.Command.EXIT, 1);
        }
        return exitCommand;
    }

        protected void exitMIDlet()
        {
                data.Closeconnection();
        destroyApp(true);
        notifyDestroyed();
    }

    private javax.microedition.lcdui.Command get_okCommand()
        {
        if (okCommand == null) {
            okCommand = new javax.microedition.lcdui.Command("Ok",
javax.microedition.lcdui.Command.OK, 1);
```

```
        }
        return okCommand1;
    }

    private javax.microedition.lcdui.Form get_form1()
            {
        if (form1 == null) {
            form1 = new javax.microedition.lcdui.Form(null, new
javax.microedition.lcdui.Item[] {get_stringItem1()});
            form1.addCommand(get_exitCommand1());
            form1.setCommandListener(this);
        }
        return form1;
    }

    private javax.microedition.lcdui.StringItem get_stringItem1()
            {
        if (stringItem1 == null)
                    {
            stringItem1 = new javax.microedition.lcdui.StringItem("Thank You","");
        }
        return stringItem1;
    }

    public void startApp()
            {
      initialize();
      data = new data(Display.getDisplay(this), get_form1());
      Display.getDisplay(this).setCurrent(data);
      new Thread(this).start();
    }

    public void pauseApp()
            {
    }

    public void destroyApp(boolean unconditional)
            {
    }

    public void run()
            {
        try {
                                        data.write(" L A M P U " + "\n");
                                        while (true)
                                        {
```

```
                                        output = "" ;
                                        String tmp = "";
                                        int input;
                                                if (data.reader != null)
                                                {
                                                        input = data.reader.read();
                                                        output += (char) input;
                                                                if (input == 10)
                                                                {

        data.write("\r\n");
                                                                }
                                                }
                                        data.write(output);
                                }
                        }
                catch (Exception e)
                        {
                        e.printStackTrace();
                        }
   }
}
```

Data

```
import java.io.IOException;
import java.util.Vector;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import java.io.IOException;
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.TextBox;
import javax.microedition.io.StreamConnection;
import java.io.InputStreamReader;
```

```
import javax.microedition.io.Connector;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import javax.bluetooth.*;
import java.io.*;

public class data extends Canvas implements CommandListener, Runnable
{
   InputStreamReader reader;
         public StreamConnection connection;
         public  List lamp = null;
         public String URL = "";
         StringBuffer strbuf = new StringBuffer();
         List list = null;
         Vector two = new Vector();
         private Command show;
         private Command conne;
         private Command discon;
         private Command back;
         public String output;
         private Command cmdExit;
         StringBuffer sb = new StringBuffer(0);
         BluetoothDeviceDiscovery disc;
   private Command cmdDiscover;
   private Command cmdStopDiscover;
         public String col[]= null;
   private LocalDevice devLocal;
   private DiscoveryAgent discoverAgent;
         public String BUG = "";
   lampu midi;
   public OutputStreamWriter ow = null;
         public  boolean what;
         public OutputStream dos = null;
   private char[][] content;
   private short numberOfRows, numberOfColumns;
   private int cursorRow, cursorCol;
   private int width, height;
   private short charHeight, charWidth;
   private Font font;
   private int color = 0x00FFFFFF;
   private Image offscreenT;
   private Image offscreenG;
   private boolean isWaiting;
   private char[] typeAheadBuffer = new char[20];
```

```
    boolean timerActive = false, upperCase = false, userInput = false, starInput =
false;
    private String keys[] = { " 0=", "1.,'", "abc2", "def3", "ghi4", "jkl5", "mno6",
"pqrs7", "tuv8", "wxyz9", "?!\"-()@/:_;+&%*=<>$[]{}\\~^#|",  "#" };
    public TerminalIOStream istream = null;
    public TerminalIOStream ostream = null;
    Display display = null;
    Displayable parent = null;
    Command backCommand;
    Command Connect;
    TextBox tb = null;
    Command BackCommand = null;
    Command TextboxCommand = null;
    Command Connection = null;
    Command okCommand = null;

    public void setColor(int color)
          {
       this.color = color;
    }

    public data(Display display, Displayable parent)
          {
       super();
       this.display = display;
       this.parent = parent;
       initialize();
    }

    public TextBox get_StarTextBox()
          {
       if (tb == null)
                   {
          tb = new TextBox("", "", 20, 0);
          okCommand = new Command("Ok", Command.OK, 1);
          tb = new TextBox(null, "", 20, 0x0);
          tb.addCommand(okCommand);
          tb.setCommandListener(this);
       }
       return tb;
    }

          public void Connect ()
          {
    try {
```

```
        connection = (StreamConnection) Connector.open ("btspp://"+URL+":1",
Connector.READ_WRITE);
        reader = new InputStreamReader(connection.openInputStream());
        dos = connection.openOutputStream();
        ow = new OutputStreamWriter(dos, "iso-8859-1");
      }
                catch (IOException ex)
                {
        ex.printStackTrace();
                }
        }

        public void Closeconnection()
        {
   try {
                        if (reader != null)
                        {
                        reader.close();
                        reader = null;
                        }

                        if (dos != null)
                        {
                        dos.close();
                        dos = null;
                        }

                        if (connection !=null)
                        {
                        connection.close();
                        connection = null;
                        }
                }
        catch (Exception e)
                {
                e.printStackTrace();
                }
        }

        public List getlamp()
        {
                if (lamp==null)
                {
                        lamp = new List( "Device list", Choice.IMPLICIT );
                        show = new Command("Search", Command.OK, 1);
                        back= new Command("Back", Command.OK, 1);
```

```
                        conne = new Command("Connect", Command.OK, 1);
                        discon = new Command("Disconnect", Command.OK, 1);
                        lamp.addCommand( show );
                        lamp.setCommandListener( this );
                }
                return lamp;
        }

   public void initialize()
        {
         this.addCommand(Connection = new Command("Connect",
Command.SCREEN, 2));
         this.addCommand(TextboxCommand = new Command("Text input",
Command.SCREEN, 2));
    this.addCommand(BackCommand = new Command("Exit",
Command.SCREEN, 2));

      istream = new TerminalIOStream();
      ostream = new TerminalIOStream();
      setCommandListener(this);
      width = getWidth();
      height = getHeight();
      offscreenT = Image.createImage(width, height);
      offscreenG = Image.createImage(width, height);
      font = Font.getFont( Font.FACE_MONOSPACE, Font.STYLE_PLAIN,
Font.SIZE_SMALL);
      charHeight = (short) font.getHeight();
      charWidth = (short) font.stringWidth("w");
      numberOfColumns = (short) ( width / charWidth );
      numberOfRows = (short) ( height / charHeight );
      content = new char[numberOfRows][];
      for (int i = 0; i < numberOfRows; i++)
                {
         content[i] = new char[numberOfColumns];
      }
     cls();
      new Thread(this).start();
   }

   public Graphics getGraphics()
        {
      return offscreenG.getGraphics();
   }

  public void cls()
        {
```

```
    clearGraphixScreen();
    clearTextScreen();
}

public void clearTextScreen()
        {
    for (int i = 0; i < numberOfRows; i++)
                {
        for (int j = 0; j < numberOfColumns; j++)
                        {
            content[i][j] = ' ';
        }
    }
    cursorRow = 0;
    cursorCol = 0;
    Graphics g = offscreenT.getGraphics();
    g.setColor(0, 0, 0);
    g.fillRect(0, 0, width, height);
    repaint();
}

public void clearGraphixScreen()
        {
    clearTextScreen();
    Graphics g = offscreenG.getGraphics();
    g.setColor(0, 0, 0);
    g.fillRect(0, 0, width, height);
    repaint();
}

private void processLineFeed()
        {
    if (cursorRow == numberOfRows - 1)
                {
    }
                else
                {
        cursorRow++;
    }
    if (starInput == true)
                {
                ostream.append(content[cursorRow - 1]);
                }
}

public void update(Graphics g)
```

```
        {
    paint(g);
}

public void paint(Graphics g)
        {
    Graphics g1 = offscreenT.getGraphics();
    g1.setFont(font);
    g1.setColor(0);
    g1.fillRect(0, 0, width, height);
    g1.setColor(255, 255, 255);
    g1.drawImage(offscreenG, 0,0,0 );
    int y = 0;
    content[cursorRow][cursorCol] = '_';
    for (int i = 0; i < numberOfRows; i++)
                {
        g1.drawChars(content[i], 0, numberOfColumns, 0, y, 0);
        y += charHeight;
    }
    g.drawImage(offscreenT, 0, 0, 0);
}

public void run()
        {
    String s = null;
    int i = 0;
    while (true)
                {
        s = istream.read();
        for (i = 0; i < s.length(); i++)
                        {
            userInput = false; writeChar(s.charAt(i));
        }
    }
}

public void writeChar(char c)
        {
    if (content[cursorRow][cursorCol] == '_')
                { content[cursorRow][cursorCol] = ' '; }
    switch (c)
                {
        case '\r': cursorCol = 0; break;
        case '\n':  /* CR+LF */
            cursorCol = 0;
            processLineFeed();
```

```
            break;
                        case '\f': clearTextScreen(); break;
          case '\b':
            if (--cursorCol < 0)
                                {
              cursorCol = numberOfColumns - 1;
              cursorRow = (cursorRow - 1 + numberOfRows) % numberOfRows;
              while (content[cursorRow][cursorCol] == ' ' && cursorCol > 0) {
cursorCol--; }
              if (content[cursorRow][cursorCol] != ' ')
                                        {
                try
                                                {
                  System.out.print(content[cursorRow][cursorCol+1]);
                  cursorCol++;
                } catch (ArrayIndexOutOfBoundsException e) { }
              }
            }
            break;
          case '\t':
            do {
              writeChar(' ');
            } while ((cursorCol % 8) != 7);
            break;
          default:
            if (c >= ' ') {
              content[cursorRow][cursorCol++] = (upperCase == true) ?
Character.toUpperCase(c) :
                c;
            }
        }
        if (cursorCol >= numberOfColumns)
                {
          cursorCol = 0;
          processLineFeed();
          if (userInput == true)
                        { ostream.append(content[cursorRow - 1]); }
        }
        repaint();
    }


    public void axis()
          {
        Graphics g = offscreenG.getGraphics();
        g.setColor(255, 255, 255);
```

```
    g.drawLine(width/2, 0, width/2, height);
    g.drawLine(0, height/2, width, height/2);
    repaint();
}

public void destroyApp(boolean unconditional) {}

public void commandAction(Command command, Displayable displayable)
        {
    if (command == BackCommand)
                {
       display.setCurrent(parent);
    }
    if (command == back)
                {
       display.setCurrent(this);
    }
    if (command == TextboxCommand)
                {
          display.setCurrent(get_StarTextBox());
    }
    if (command == Connection)
                {
    display.setCurrent(getlamp());
    }
    if (command == conne)
    {
      URL = "";
      URL = lamp.getString(lamp.getSelectedIndex());
      Connect();
      lamp.removeCommand(conne);
      lamp.addCommand(discon);
      display.setCurrent(this);
    }
    if (command == discon)
                {
      Closeconnection();
      if (reader == null && dos == null)
                    {
         lamp.addCommand(conne);
         lamp.removeCommand(discon);
      }
    }
    else if (command == okCommand)
                {  BUG = tb.getString();
       try
```

```
                        {
            ow.write(BUG+"\r\n");
            ow.flush();
          }
                        catch (IOException ex)
                        {
            ostream.write("No connection"+"\n");
            ostream.flush();
          }
          starWrite(tb.getString());
          tb.setString("");
          display.setCurrent(this);
        }
        else if (command == show)
                  {
            String str = "Device";
            Ticker t = new Ticker(str);
            lamp.removeCommand(show);
            lamp.addCommand(back);
            lamp.setTicker(t);
           try
                        {
            BluetoothDeviceDiscovery.main();
          } catch (IOException ex) {
            ex.printStackTrace();
          }
            int deviceCount = BluetoothDeviceDiscovery.vecDevices.size();
            for (int i = 0; i < deviceCount; i++) {
              RemoteDevice remoteDevice = (RemoteDevice)
BluetoothDeviceDiscovery.vecDevices.elementAt(i);
              lamp.append(remoteDevice.getBluetoothAddress(), null);
                }
              lamp.addCommand(conne);
              display.setCurrent(lamp);
            }
    }

    public TerminalIOStream getInputstream()
          {
        return ostream;
    }

    public TerminalIOStream getOutputstream()
          {
        return istream;
    }
```

```
    public void write(String s)
        {
        istream.write(s);
    }

    public void starWrite(String s)
        {
        starInput = true; userInput = false;
        for (int i = 0; i < s.length(); i++)
                {
            writeChar(s.charAt(i));
        }
        starInput = false;
    }
}
```

LAMPIRAN C

SKEMATIC DAN FOTO

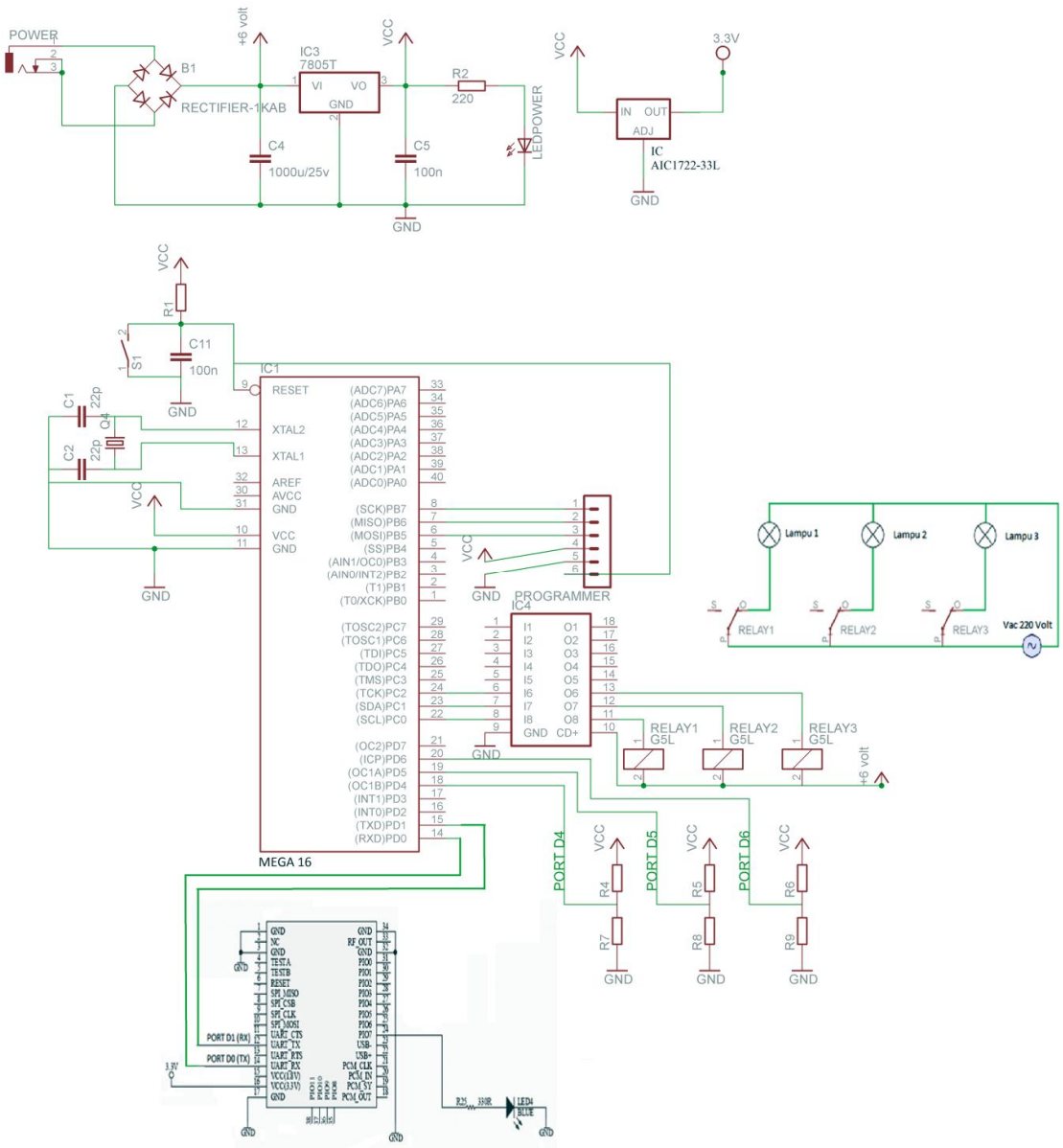
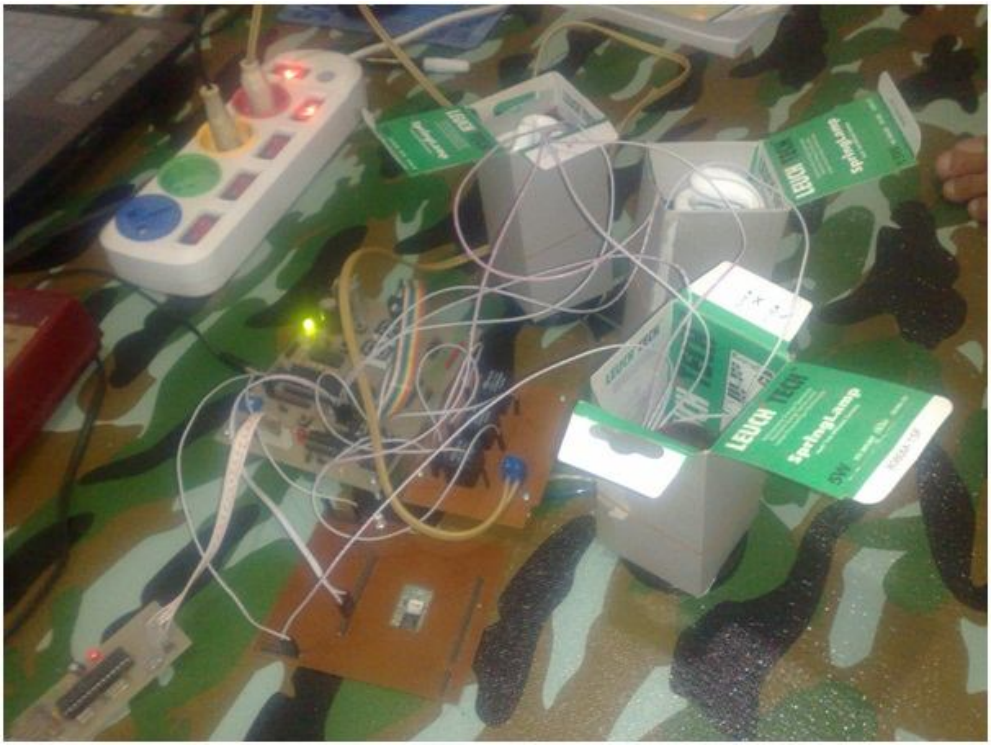


FOTO ALAT


N95 8GB
Lampu
LAMPU
sword 2
assword Benara
Lampu 1 Ont
Password Salah2
Password Benard
Lampu 1 Off_
Pilihan
NOKIA


LEUCH TECH
SpringLamp

LAMPIRAN D

AIC 1722

■ ORDERING INFORMATION

AIC1722-XXCXX

- PACKAGING TYPE
  - X : SOT-89
  - ZT : TO-92
  - ZL : TO-92
- TEMPERATURE RANGE
  - C: 0°C~70°C
- OUTPUT VOLTAGE
  - 33: 3.3V
  - 35: 3.5V
  - 37: 3.7V
  - 38: 3.8V
  - 50: 5.0V
  - 52: 5.2V

ORDER NUMBER	PIN CONFIGURATION
AIC1722-33CX AIC1722-35CX AIC1722-37CX AIC1722-38CX AIC1722-50CX AIC1722-52CX (SOT-89)	FRONT VIEW 1: VOUT 2: GND 3: VIN 
AIC1722-33CZT AIC1722-35CZT AIC1722-37CZT AIC1722-38CZT AIC1722-50CZT AIC1722-52CZT (TO-92 T TYPE)	TOP VIEW 1: GND 2: VIN 3: VOUT 
AIC1722-33CZL AIC1722-35CZL AIC1722-37CZL AIC1722-38CZL AIC1722-50CZL AIC1722-52CZT (TO-92 L TYPE)	TOP VIEW 1: VIN 2: GND 3: VOUT 

■ ABSOLUTE MAXIMUM RATINGS

Input Supply Voltage ..... -0.3~12V

Operating Junction Temperature Range ..... -40°C~ 125°C

Storage Temperature Range ..... -65°C~150°C

Power Dissipation
- SOT-89 Package 0.5W
- TO-92 Package 0.5W

■ TEST CIRCUIT

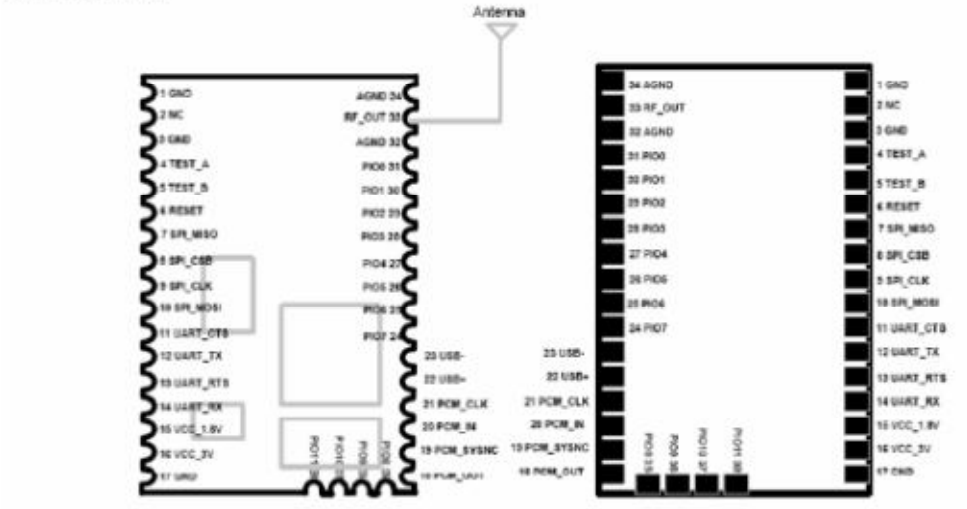
Refer to the TYPICAL APPLICATION CIRCUIT

■ ELECTRICAL CHARACTERISTICS (Ta=25°C, C_{IN}=1μF, C_{OUT}=10μF, unless otherwise specified.)

PARAMETER	TEST CONDITIONS		MIN.	TYP.	MAX.	UNIT
Output Voltage	No Load					V
	AIC1722-52	V_{IN}=5.5~12V	5.100	5.2	5.300	
	AIC1722-50	V_{IN}=5.5~12V	4.900	5.0	5.100	
	AIC1722-38	V_{IN}=4.1~12V	3.725	3.8	3.875	
	AIC1722-37	V_{IN}=4.0~12V	3.625	3.7	3.775	
	AIC1722-35	V_{IN}=4.0~12V	3.430	3.5	3.570	
	AIC1722-33	V_{IN}=3.6~12V	3.235	3.3	3.365	

1. Hardware & Technical Information

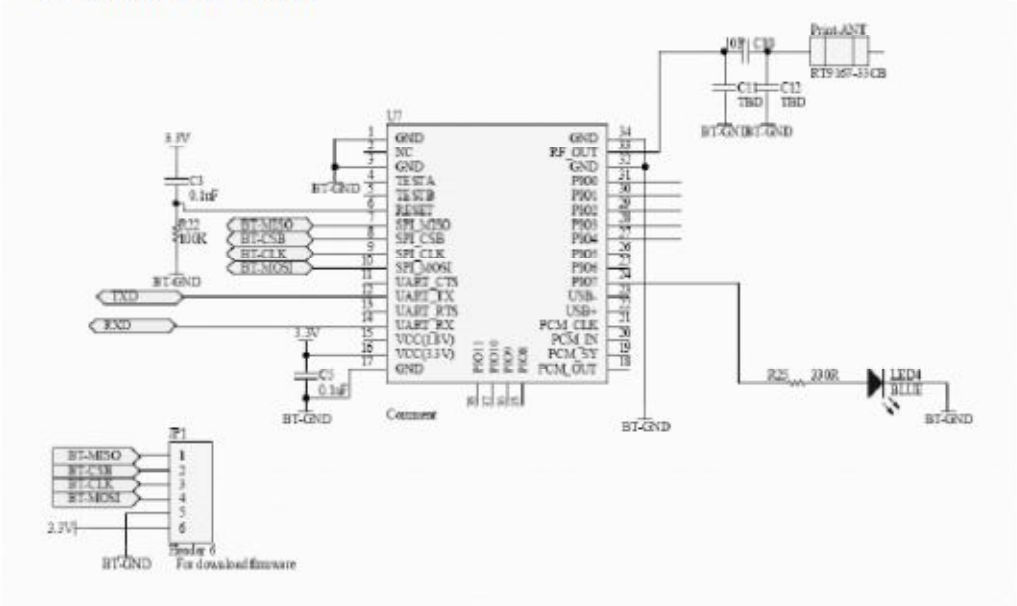
1.1 Pin definition



Pin No.	Name	Type	Note	Pin No.	Name	Type	Note
1	Ground			34	Ground		
2	NC			33	RF_OUT	Out	
3	Ground			32	Ground		
4	TestA		Internal testing using	31	PIO0	In/Out	
5	TestB		Internal testing using	30	PIO1	In/Out	
6	Reset	Input		29	PIO2	In/Out	
7	SPI_MISO		Internal testing using	28	PIO3	In/Out	
8	SPI_CSB		Internal testing using	27	PIO4	In/Out	
9	SPI_CLK		Internal testing using	26	PIO5	In/Out	
10	SPI_MOSI		Internal testing using	25	PIO6	In/Out	
11	UART_CTS	Input		24	PIO7	Out	Driving LED
12	UART_TX	Out		23	USB-	In/Out	
13	UART_RTS	Out		22	USB+	In/Out	
14	UART_RX	Input		21	PCM_CLK	In/Out	
15	VCC_1.8V	Out		20	PCM_IN	In	
16	VCC_3.3V	Input		19	PCM_SYNC	In/Out	
17	Ground			18	PCM_OUT	Out	

Pin No.	Name	Type	Note	Pin No.	Name	Type	Note
35	PIO8	In/Out		37	PIO10	In/Out	
36	PIO9	In/Out		38	PIO11	In/Out	

1.5 Reference Schematics



4. Standard Setup Information

	Parameter		Value
1	Baud Rate		9600
2	Pin Code Prompt		Disable
3	Local Name		SPP
4	LED PIN24(PIO 7)	Power on	Flash 26 times[ON time frame: 80ms, OFF time frame: 140ms]
		Connect	Flash with ON one time within 1 second,[ON time frame: 35ms]
		Disconnected	Flash with ON one time within 3 seconds,[ON time frame : 35ms]

5. Customization Information

	Parameter		Value
1	Baud Rate		
2	Pin Code Prompt		
3	Local Name		
4	LED	Connected	
	PIN24(PIO 7)	Disconnected	

RELAY

PCB RELAY www.chinarelay.com

JQC-3F(T73)

JQC-3F(T73)

Contact specification

Contact form	1H, 1Z
Contact resistance	≤100mΩ (1A 6VDC)
Contact material	AgCdO AgSnO2
Contact capacity	1a 240VAC/30VDC 10A 240VAC/30VDC

Technical specification

Insulation resistance	≥1000MΩ
Dielectric strength	Between coil and contact 2kV50/60HZ 1min
	Between open contacts 2kV50/60HZ 1min
Operate / Release time	≤10ms/5ms
Terminal type	PCB mounting

Coil specification

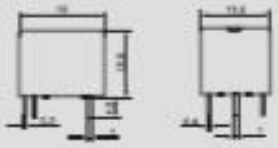
Rated coil power	160mW/240mW/DC 0.61W

Coil data

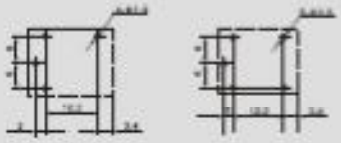
Nominal voltage	Operate voltage	Release voltage	Coil resistance
VDC	VDC	VDC	Ω ± 10%
3	2.25	0.3	25
5	3.75	0.5	70
6	4.5	0.6	100
9	6.8	0.9	225
12	9.0	1.2	400
18	13.5	1.8	900
24	18.0	2.4	1600
48	36.0	4.8	4900

Outline dimension、Mounting holes (Bottom view)、Wiring diagram (Bottom view)

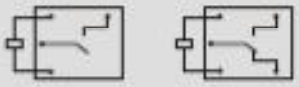
JQC-3F(T73) Outline dimension



JQC-3F(T73) Mounting hole (Bottom view)



JQC-3F(T73) Wiring diagram (Bottom view)



Order this document by ULN2803/D

MOTOROLA

Octal High Voltage, High Current Darlington Transistor Arrays

ULN2803
ULN2804

OCTAL PERIPHERAL DRIVER ARRAYS

SEMICONDUCTOR TECHNICAL DATA

The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open–collector outputs and free wheeling clamp diodes for transient suppression.

The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 6 to 15 volt high level CMOS or PMOS.

A SUFFIX
PLASTIC PACKAGE
CASE 707

MAXIMUM RATINGS (T_A = 25°C and rating apply to any one device in the package, unless otherwise noted.)

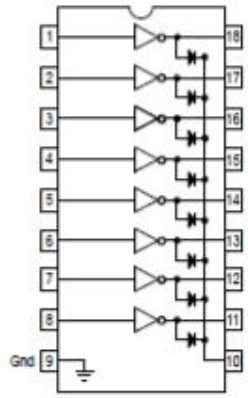
Rating	Symbol	Value	Unit
Output Voltage	V_O	50	V
Input Voltage (Except ULN2801)	V_I	30	V
Collector Current – Continuous	I_C	500	mA
Base Current – Continuous	I_B	25	mA
Operating Ambient Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C
Junction Temperature	T_J	125	°C

$R_{\theta JA}$ = 55°C/W
Do not exceed maximum current limit per driver.

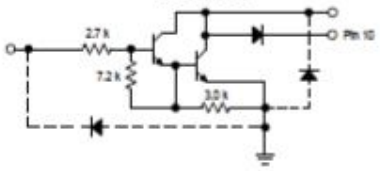
ORDERING INFORMATION

Device	Characteristics		
	Input Compatibility	V_{CE}(Max)/I_C(Max)	Operating Temperature Range
ULN2803A ULN2804A	TTL, 5.0 V CMOS 6 to 15 V CMOS, PMOS	50 V/500 mA	T_A = 0 to + 70°C

PIN CONNECTIONS



1/8 ULN2803



Rev 1

5 J2ME and JABWT programming

This chapter will give a thorough description on how to program with JABWT and J2ME. Code samples are provided throughout the chapter, aiming to show the reader how a complete Bluetooth MIDlet is built. This chapter is partly based on [7]. The code samples in this chapter are simplified as much as possible, highlighting the Bluetooth specific Java code instead of describing both J2ME and JABWT issues simultaneously. The event-driven nature of J2ME applications tend to raise the complexity of the source code, making it difficult to understand the structure of the application. The simplified code samples provided in this chapter should make it easier for the reader to understand JABWT programming. It is assumed that the reader is familiar with J2ME programming. Readers who lack general knowledge about J2ME programming should consult [38].

The first code samples provided in this chapter show the structure of a Bluetooth MIDlet. Functionality will be added to these code samples as new functionality is explained. Note that these code samples are not fully functional MIDlets. However, they can be used as a starting point for a complete application. Method declarations have font typeface bold to increase the readability of the code.

5.1 Structure of Bluetooth MIDlet

This section shows the structure in a Java/Bluetooth MIDlet. Several MIDlet examples are available on Sun Microsystems' Mobility website [39].

Usually an event-driven MIDlet with no Bluetooth support looks like this:

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class YourMidlet extends MIDlet implements CommandListener {

        public void startApp() {

        }

        public void pauseApp() {
```

```
        }

        public void destroyApp(boolean unconditional) {

        }

        public void commandAction(Command c, Displayable d) {

        }
}
```

The first three methods, `startApp()`, `pauseApp()` and `destroyApp()` are needed for any MIDlet. They come from extending the MIDlet class. The next method, `commandAction()` comes from the `CommandListener` interface. This is needed to catch command events. The MIDlet is extended to support Bluetooth communication in the next code sample.

During device discovery and service discovery, events will be delivered to a `DiscoveryListener` object when devices or services are found or the device discovery or service discovery is completed. An object implementing the `DiscoveryListener` interface is used to catch these events. The MIDlet will then look like this:

```
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class YourMidlet extends MIDlet implements CommandListener,
                                                   DiscoveryListener {

    public void startApp() {

    }

    public void pauseApp() {

    }

    public void destroyApp(boolean unconditional) {

    }

    public void commandAction(Command c, Displayable d) {

    }
```

```
    public void deviceDiscovered(RemoteDevice remoteDevice,
                                 DeviceClass deviceClass) {

    }

    public void inquiryCompleted(int param) {

    }

    public void serviceSearchCompleted(int transID, int respCode) {

    }

    public void servicesDiscovered(int transID,
                                   ServiceRecord[] serviceRecord) {

    }
}
```

The last four methods, `deviceDiscovered()`, `inquiryCompleted()`, `serviceSearchCompleted()` and `servicesDiscovered()` are used to catch events during device discovery and service discovery. Device discovery and service discovery will be outlined in the two next sections.

5.2 Device discovery (Inquiry)

Device discovery, introduced in Section 2.4, is the first step required when browsing nearby Bluetooth devices. When we have discovered nearby devices we can find out which services they offer. Note that no UI specific code is included in the following examples, only Bluetooth specific code.

To use any Bluetooth related methods you need to obtain a reference to the `LocalDevice` object by calling the `LocalDevice.getLocalDevice()` method. The obtained `LocalDevice` object gives access to Bluetooth properties for the device, such as the Bluetooth address, friendly name and discovery mode. We will use the `LocalDevice` object to obtain a `DiscoveryAgent` object. The `DiscoveryAgent` object is used for device discovery and service discovery.

The MIDlet now looks like this:

```
import java.util.Vector;
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
```

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class yourMIDlet extends MIDlet implements CommandListener,
                                                   DiscoveryListener {


    private LocalDevice local = null;
    private DiscoveryAgent agent = null;

    private Vector devicesFound = null;

    public void startApp() {

        /*  Add your MIDlet specific code here.
         *  You probably want to show the user
         *  a welcome screen.
         *  The call to doDeviceDiscovery() is
         *  here for the example's sake. You
         *  should call doDeviceDiscovery() when
         *  the user actively asks for it.
         */


        doDeviceDiscovery();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {

    }

    public void deviceDiscovered(RemoteDevice remoteDevice,
                                 DeviceClass deviceClass) {

    }

    public void inquiryCompleted(int param) {

    }

    public void servicesDiscovered(int transID,
                                   ServiceRecord[] serviceRecord) {
    }

    public void serviceSearchCompleted(int transID, int respCode) {
    }
```

```
    private void doDeviceDiscovery() {

        try {
            local = LocalDevice.getLocalDevice();
        }catch (BluetoothStateException bse) {

            // Error handling code here
        }

        agent = local.getDiscoveryAgent();

        devicesFound = new Vector();

        try {

            if(!agent.startInquiry(DiscoveryAgent.GIAC,this)) {

                // Inquiry not started, error handling code here
            }
        }catch(BluetoothStateException bse) {

            // Error handling code here
        }
    }

}
```

Device discovery is started using the `LocalDevice` and `DiscoveryAgent` objects. Observe that the `doDeviceDiscovery()` method is called in the `startApp()` method. Searching with the GIAC parameter will find devices which are general discoverable (see Section 2.4). The `DiscoveryListener` parameter is `this`, meaning our MIDlet. When devices are discovered or the search is complete, events will be delivered to our MIDlet. Note that the `startInquiry()` method returns immediately, returning `true` if the device discovery was initiated or `false` if the device discovery process was not started. An event will be delivered to the MIDlet when the device discovery is completed. This is important to take into account when designing the flow of execution in the MIDlet.

The `deviceDiscovered()` and `inquiryCompleted()` methods are used to catch events related to the device discovery process. When a device is discovered the `deviceDiscovered()` method of the object `this` will be called. The parameter `remoteDevice` will be the discovered device, it is up to us to decide what to do with it. Note that we do not know how many devices will be discovered. A `Vector` will therefore be an appropriate data structure to save discovered devices.

The `inquiryCompleted()` method is called when the inquiry ends. The status code supplied in the parameter `param` should always be checked. The complete code for device discovery follows:

```
import java.util.Vector;
```

```
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class yourMIDlet extends MIDlet implements CommandListener,
                                                   DiscoveryListener {

    private LocalDevice local = null;
    private DiscoveryAgent agent = null;

    Vector devicesFound = null;

    public void startApp() {

        /* Add your MIDlet specific code here.
         * You probably want to show the user
         * a welcome screen.
         * The call to doDeviceDiscovery() is
         * here for the example's sake. You
         * should call doDeviceDiscovery when
         * the user actively asks for it.
         */


        doDeviceDiscovery();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {

    }

    public void deviceDiscovered(RemoteDevice remoteDevice,
                                 DeviceClass deviceClass) {

        devicesFound.addElement(remoteDevice);
    }

    public void inquiryCompleted(int param) {

        /* We should give the user an alert based on the
         * inquiry status code
         */
```

```
        switch (param) {

            case DiscoveryListener.INQUIRY_COMPLETED:

                /* Inquiry completed normally, add appropriate code
                 * here
                 */

                break;

            case DiscoveryListener.INQUIRY_ERROR:

                // Error during inquiry, add appropriate code here.

                break;

            case DiscoveryListener.INQUIRY_TERMINATED:

                /* Inquiry terminated by agent.cancelInquiry()
                 * Add appropriate code here.
                 */

                break;
        }
    }

    public void servicesDiscovered(int transID,
                                   ServiceRecord[] serviceRecord) {
    }

    public void serviceSearchCompleted(int transID, int respCode) {
    }

    private void doDeviceDiscovery() {

        try {
            local = LocalDevice.getLocalDevice();
        }catch (BluetoothStateException bse) {

            // Error handling code here
        }

        agent = local.getDiscoveryAgent();

        devicesFound = new Vector();

        try {

            if(!agent.startInquiry(DiscoveryAgent.GIAC,this)) {

                // Inquiry not started, error handling code here
            }
        }catch(BluetoothStateException bse) {

            // Error handling code here
```

```
        }
    }
}
```

Discovered devices are kept in the `DevicesFound` vector by adding them as they are discovered. When our search ends, we check if everything went as expected and can alert the user by adding appropriate code in our switch-statement.

5.3 Service search

After device discovery is completed it is time to find out which services are offered by the discovered devices. This is accomplished by doing a service discovery on the device of interest.

The `servicesDiscovered()` and `serviceSearchCompleted()` methods must be implemented. They will handle the events occuring when services are found or the service discovery completes. In addition, the `doServiceSearch()` method has been added to show how a service discovery is initiated. This method will start a service discovery on the `RemoteDevice` supplied as a parameter, and is called in the `inquiryCompleted()` method.

The complete example Bluetooth MIDlet looks like this:

```
import java.util.Vector;
import javax.bluetooth.UUID;
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;


public class YourMIDlet extends MIDlet implements CommandListener,
                                                   DiscoveryListener {

    private LocalDevice local = null;
    private DiscoveryAgent agent = null;

    private Vector devicesFound = null;
    private ServiceRecord[] servicesFound = null;

    public void startApp() {
```

```
        /*  Add your MIDlet specific code here.
         *  You probably want to show the user
         *  a welcome screen.
         *  The call to doDeviceDiscovery() is
         *  here for the example's sake. You
         *  should call doDeviceDiscovery() when
         *  the user actively asks for it.
         */

        doDeviceDiscovery();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {

    }

    public void deviceDiscovered(RemoteDevice remoteDevice,
                                 DeviceClass deviceClass) {

        devicesFound.addElement(remoteDevice);
    }

    public void inquiryCompleted(int param) {

        /* We should give the user an alert based on the
         * inquiry status code
         */

        switch (param) {

            case DiscoveryListener.INQUIRY_COMPLETED:

                /* Inquiry completed normally, so we
                 * initiate a service discovery on the first
                 * device found. That is, if we actually
                 * found any devices.
                 */

                 if (devicesFound.size() > 0){

                       doServiceSearch( (RemoteDevice)
                                   devicesFound.firstElement());
                 }
                break;

            case DiscoveryListener.INQUIRY_ERROR:

                //Error during inquiry, add appropriate code here

                break;
```

```
            case DiscoveryListener.INQUIRY_TERMINATED:

                /*
                 * Inquiry terminated by agent.cancelInquiry()
                 * Add appropriate code here
                 */

                break;
        }
    }

    public void servicesDiscovered(int transID,
                                   ServiceRecord[] serviceRecord) {

        /* Services discovered, keep reference to the ServiceRecord
         * array
         */

        servicesFound = serviceRecord;
    }

    public void serviceSearchCompleted(int transID, int respCode) {

            switch(respCode) {

            case DiscoveryListener.SERVICE_SEARCH_COMPLETED:

                /*
                 * Service search completed successfully
                 * Add appropriate code here
                 */

                break;

            case
               DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE:

                // device not reachable, add appropriate code here

                break;

            case DiscoveryListener.SERVICE_SEARCH_ERROR:

                // Service search error, add appropriate code here

                break;

            case DiscoveryListener.SERVICE_SEARCH_NO_RECORDS:

                // No records found, add appropriate code here

                break;

            case DiscoveryListener.SERVICE_SEARCH_TERMINATED:
```

```
                /*
                 * Search terminated by agent.cancelServiceSearch()
                 * Add appropriate code here
                 */

                break;
        }

    }


    private void doDeviceDiscovery() {

        try {
            local = LocalDevice.getLocalDevice();
        }catch (BluetoothStateException bse) {

            // Error handling code here
        }

        agent = local.getDiscoveryAgent();

        devicesFound = new Vector();

        try {

            if(!agent.startInquiry(DiscoveryAgent.GIAC,this)) {

                // Inquiry not started, error handling code here
            }
        }catch(BluetoothStateException bse) {

            // Error handling code here
        }
    }

    private void doServiceSearch(RemoteDevice device) {

        /*
         * Service search will always give the default attributes:
         * ServiceRecordHandle (0x0000), ServiceClassIDList (0x0001),
         * ServiceRecordState (0x0002), ServiceID (0x0003) and
         * ProtocolDescriptorList (0x004).
         *
         * We want additional attributes, ServiceName (0x100),
         * ServiceDescription (0x101) and ProviderName (0x102).
         *
         * These hex-values must be supplied through an int array
         */

        int[] attributes = {0x100,0x101,0x102};

        /*
         * Supplying UUIDs in an UUID array enables searching for
         * specific services. PublicBrowseRoot (0x1002) is used in
```

```
         * this example. This will return any services that are
         * public browseable. When searching for a specific service,
         * the service's UUID should be supplied here.
         */

        UUID[] uuids = new UUID[1];
        uuids[0] = new UUID(0x1002);

        try {
            agent.searchServices(attributes,uuids,device,this);
        } catch (BluetoothStateException e) {

            // Error handling code here
        }
    }
}
```

The `searchServices()` method will return immediately, returning a transaction ID for the service discovery. The transaction ID is used to identify the particular service discovery. When the service discovery completes, an event will be delivered to the MIDlet.

This example Bluetooth MIDlet will hopefully be of help to J2ME application developers getting started with Bluetooth programming. Study the Bluetooth Browser source code in Appendix A to see what a fully functional Bluetooth MIDlet looks like.