Digitala projekt

Grupp 15

Daniel Fritze, d03df@student.lth.se
Rickard Törnblad, d03rt@student.lth.se

Handledare: Bertil Lindvall

31st May 2007

Abstract

For our project we decided to build an electronic timer for controlling home appliances. The timer would be programmed from a personal computer, but function independently. The project is implemented using an ATMega16 micro processor and the pc-user interface is programmed in Java.

Contents

1 Inledning

Vi valde att bygga en elektronisk timer som ska kunna sätta på och stänga av apparater i hemmet (t.ex. lampor) vid givna tidpunkter. Planen var att timern skulle programmeras via ett användargränssnitt på en hemdator, hemdatorn behöver dock inte vara inkopplad för att timern ska fungera. För en mer detaljerad beskrivning se kravspecifikationen i appendix 1.

2 Utrustning

ATMega16 är den enchipsdator vi valde att använda till våran timer. De funktioner vi använder på den är RXD och TXD för kommunikation med datorn via com-porten. Vi använder alla åtta ben på port A för att kontrollera åtta lysdioder som representerar de åtta utgångarna.

MAX232 som inverterar och konverterar signalen mellan vår ATMega16 som använder sig av 5V för logisk etta samt 0V för logisk nolla till ungefär +10V för logisk etta och -10V för nolla som är vad com-porten använder sig av.

Vi har även använt lysdioder, kondensatorer och motstånd.

3 Mjukvara

Här följer en beskrivning av de viktigaste funktionerna i mjukvaran, själva källkoden bifogas sist i rapporten.

3.1 Timer

Själva huvudprogrammet är en evighetsloop som loopar och kontrollerar om en minut har passerat. Detta kan den veta eftersom ett avbrott räknar upp ett heltal fyra gånger i sekunden. När detta heltal har blivit 240 har det alltså gått en sekund. När en minut har passerat så kontrollerar programmet om en händelsetidpunkt infaller samt nollställer heltalet. Om en händelsetidpunkt infaller så utförs händelsen i fråga (en lampa tänds eller släcks).

Det finns ytterligare ett avbrott som hanteras i vårt program, detta inträffar ifall data kommer till processorn från com-porten. Detta fungerar som en tillståndsmaskin med tre tillstånd. Om ett sådant avbrott sker så hämtas en byte och ytterligare avbrott stängs av. Sedan kontrolleras ifall byten har värdet 1, 2 eller 3.

1. Detta innebär att programmet ska skicka tillbaka alla händelser som finns sparade i EEPROM-minnet.
2. Nu ska programmet ta emot den aktuella tiden i form av fyra bytes.
3. Detta innebär att timern ska ta emot nya händelser. Först kommer det en byte som innehåller antalet händelser som ska tas emot. Efter detta kommer händelserna (varje händelse tar fyra bytes).

Utöver detta så finns det metoder för att kontrollera utgångarna, jämföra tider och konvertera mellan olika format.

3.2 Användargränssnittet

Vi går inte in på själva GUI-programmeringen här då det är överflödigt.

Utöver själva GUI:t så har dator-sidan metoder för att konvertera tal mellan de olika formaten som används, vilka beskrivs nedan. Java-delen tar även hand om övergången mellan sommar- och vintertid (och tvärtom) samt skottår.

4 Design

4.1 Tidsformat

Vi valde att representera tid som det antal minuter som passerat sedan den förste januari år 2000. Detta eftersom vi inte behöver kunna ange tid mer exakt än på minuten. Vi funderade först på att använda millisekunder sedan år 1970 (unix-tid), men detta tidsformat tar upp fyra bytes samt fungerar bara fram till år 2038, någon som köper vår timer ska kunna ha glädje av den längre än så. Vårt tidsformat använder 3,25 bytes och är således korrekt till år 2127.

4.2 Händelser

En händelse beskrivs som sagt av fyra bytes. 3,25 av dessa används alltså till tidsangivelse. Av de återstående 6 bitarna används den första till om porten ska stängas av eller sättas igång, den andra och tredje anger om händelsen är dagsvis eller veckovis återkommande. De tre sista bitarna anger vilken utgång händelsen gäller.

Eftersom vår ATMega16 har 512 bytes EEPROM-minne så kan vi spara 127 händelser, en byte går till att spara antalet sparade händelser.

5 Brister och kommentarer

5.1 Strömförsörjning

Timern är tänkt att gå på batteri, och när batteriet tar slut så räcker det inte med att byta batteri. Man måste även synkronisera klockan via datorn. Detta skulle kunna undvikas ifall man hade en extern klockkrets med egen strömförsörjning.

5.2 Klockans exakthet

Eftersom vi inte använder en extern klocka så utgår vi från processorns klocka och försöker komma så nära en minut vi kan. Detta leder till att klockan kommer att gå aningen fel. Även detta hade kunnat undvikas med en extern klockkrets eller en kristall. Eftersom detta är en prototyp så fungerar det, men vi hade fått ändra det om timern skulle användas på riktigt.

5.3 Kommunikationsprotokoll

Vi gör ingen kontroll om timern inkopplad eller om den har strömförsörjning. Detta leder till att om man försöker hämta händelser från timern och den inte

skickar något så kommer användargränssnittet att hänga sig. Om man försöker skicka händelser eller synkronisera klockan under samma förutättningar så tror användargränssnittet att operationen lyckades. Detta skulle man kunna lösa med hjälp av ACK efter att alla händelser mottagits, man behöver även en time-out för att lösa problemet med uteblivit svar.

6 Appendix 1: Kravspecifikation

Vi tänkte konstruera en timer som ska kunna sätta igång och stänga av ett antal lampor eller apparater vid olika tidpunkter. Timern ska kunna hantera minst åtta dagliga till och frånslag och även kunna ta hänsyn till sommartid/vintertid.

Timern ska programmeras via ett grafiskt användargränssnitt på en vanlig persondator. Den ska dock fungera utan att vara ihopkopplad med datorn. Programmet ska ha olika vyer: En som ser ut som en vanlig kalender där man kan välja år, månad och dag. När man valt en dag kan man lägga till och ta bort händelser för den dagen. Man kan även se en lista med alla händelser.

7 Appendix 2: Scheman och programflöde

Det var så här vi tänkte att vårt projekt skulle fungera, vi var tvungna att göra vissa förändringar för att det att fungera, vi hade ritat lite fel på kopplingsschemat.

7.1 Programflöde

7.1.1 ATMega16

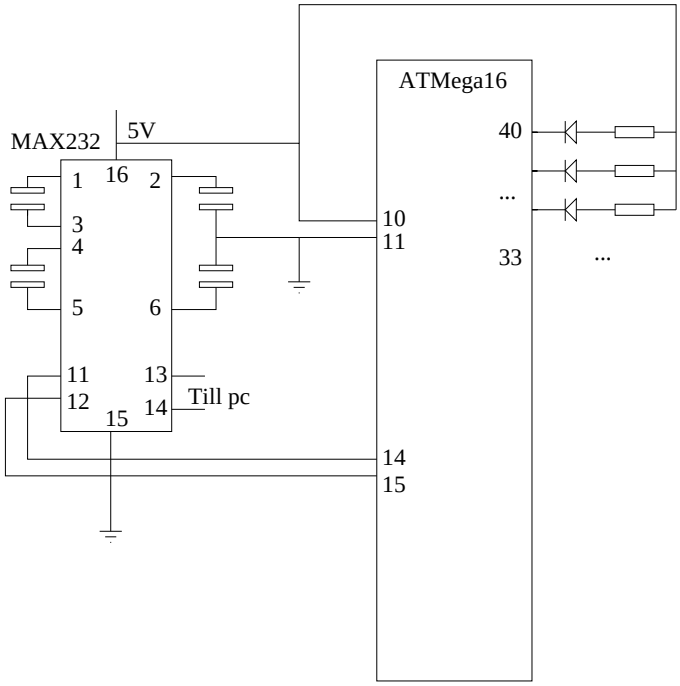
Vi har tänkte att det kan fungera så här:

1. Väntelage
2. Ett avbrott genereras med en bestämd frekvens.
3. Gå tillbaka till ett om ingen tidpunkt passerats.
4. Utför kommandot som hör ihop med tidpunkten.

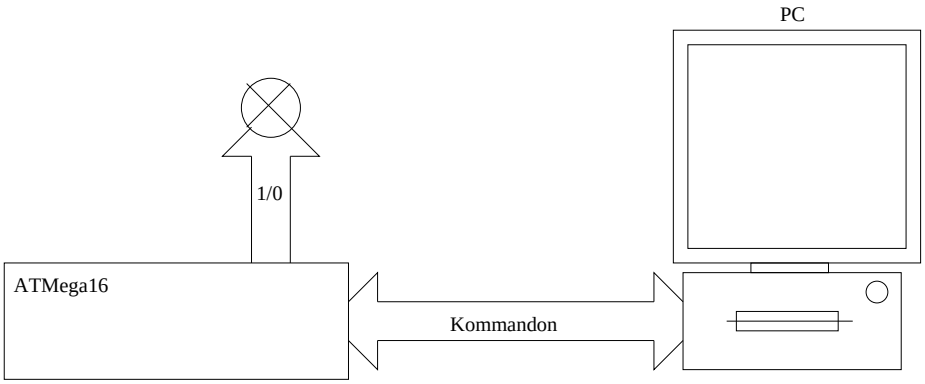
7.1.2 PC

Beskrivning av programflöde är överflödigt.

8 Kopplingschema



9 Blockschema



10 Appendix 3: Källkod till Timern

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

#define USART_BAUDRATE 2400
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

void on_off(unsigned int a, unsigned int b);
uint32_t bytes_to_int(uint8_t* numbers);
void sync_clock();
void compare_times();

uint32_t clock;
uint8_t EEMEM alarm_times[508];
uint8_t EEMEM nbr_saved;
int count =0;

int main (void) {
        DDRA = 0xFF;    //Sätter port A till utgångar
        PORTA = 0xFF;   //Sätter port A till höga

        //Ställer in rätt värden för kommunikation via comport
        UCSRB |= (1 << RXEN) | (1 << TXEN);
        UCSRC |= (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);
        UBRRL = BAUD_PRESCALE;
        UBRRH = (BAUD_PRESCALE >> 8);
        UCSRB |= (1 << RXCIE);

        TCCR0 = (1 << WGM01) | (1 << CS02) | (1 << CS00); // Skala clk_{i/O} med 1024
        OCR0 = 0xF4; // Timer1 ska matcha F4 hexadecimalt, detta för att få en någorlunda korrekt klocka
        TIMSK |= (1 << OCIE0);  //Se ovan

        sei(); //Gör interrupts möjliga
        clock=0;

        //Huvudloop som räknar minuter och varje minut kontrollerar om någon utgång ska ändras
        while (1) {
                if (count == 10) { // 240
                        count = 0;
                        compare_times();
                        clock++;
                }
        }
}



//metod för att läsa in 4 bytes och synkronisera klockan med dessa
void sync_clock() {
        uint8_t numbers[4];
        int j;
        for(j=0;j<4;j++){
                while ((UCSRA & (1 << RXC)) == 0) {};
                numbers[j] = UDR;
        }
        clock = bytes_to_int(numbers);
}

//jämför alla händelser i minnet med den aktuella tiden och utför de som sammanfaller
void compare_times() {
        uint8_t number = eeprom_read_byte(&nbr_saved);
        uint32_t alarm_time;
        int i,j;
        uint8_t numbers[4];
        uint8_t on, daily, weekly, ausgang;

        for(i=0; i<number; i++) {
                for(j=0; j<4; j++) {
                        numbers[j] = eeprom_read_byte(&alarm_times[i*4+j]);
                }

                on = numbers[0] & 0b10000000;
```

```
                daily = numbers[0] & 0b01000000;
                weekly = numbers[0] & 0b00100000;
                ausgang = (numbers[0] & 0b00011100)>>2;
                numbers[0]&=0b00000011;
                alarm_time = bytes_to_int(numbers);

                if(alarm_time == clock || (weekly && alarm_time<=clock &&
                        (alarm_time%10080 == clock%10080)) || (daily && alarm_time<=clock &&
                                (alarm_time%1440 == clock%1440)))
                        on_off(ausgang, on);
        }
}

//konverterar 4 bytes till en uint32_t
uint32_t bytes_to_int(uint8_t* numbers) {
        uint32_t result = 0;
        int i;
        for(i=0; i<4; i++) {
                result+=numbers[i];
                if(i < 3)
                        result<<=8;
        }

  return result;
}

//sätter igång eller stänger av en utgång
void on_off(unsigned int pin, unsigned int value) {

        int i;
        int p = pin;
        pin = 1;
        for(i=0; i<p; i++)
                pin *= 2;

        if(value)
                PORTA&=~pin;
        else
                PORTA|=pin;
}

//avbrott som sker 4ggr i sekunden
ISR(TIMER0_COMP_vect) {
        count++;
}

//Hanterar interrupts när man försöker kommunicera via kom-porten
ISR(USART_RXC_vect) {

        uint8_t ReceivedByte;
        ReceivedByte = UDR;             //Läser en byte
        cli();                                  //Stänger av interrupts
        switch(ReceivedByte) {

                //skickar tillbaks alla händelser som är sparade i eepromen
                case 1: {}
                        unsigned int number = eeprom_read_byte(&nbr_saved);
                        unsigned int i;
                        for(i=0; i<number*4; i++) {
                                while ((UCSRA & (1 << UDRE)) == 0) {};
                                UDR = eeprom_read_byte(&alarm_times[i]);
                        }
                        break;

                //Synkroniserar klocka
                case 2:
                        sync_clock();
                        compare_times();
                        break;

                //Tar emot ett antal händelser
                case 3: {}
                        uint8_t nbr_sav;
                while ((UCSRA & (1 << RXC)) == 0) {};
```

```
                        nbr_sav = UDR;
                        eeprom_write_byte(&nbr_saved,nbr_sav);
                        unsigned int j;
                        for(j=0;j<nbr_sav*4;j++){
                                while ((UCSRA & (1 << RXC)) == 0) {};
                                uint8_t a = UDR;
                                eeprom_write_byte(&alarm_times[j], a);
                        }
                sync_clock();
                        break;

                default:
                        break;
   };

   sei();       //Sätter ingång interrupts
}
```

11 Appendix 4: Källkod till användargränssnittet

11.1 Timer

```
package digp15;

public class Timer {
    public static void main(String[] args) {
        new TimerGUI();
    }
}
```

11.2 TimerEvent

```
package digp15;

import java.util.*;

public class TimerEvent implements Comparable<TimerEvent> {
        private long time;
        private int ausgang;
        private boolean on, recurday, recurweek;

        /*
         * Representerar en timerhändelse
         */
        public TimerEvent(long time, int ausgang, boolean on, boolean recurday, boolean recurweek) {
                this.time = time;
                this.ausgang = ausgang;
                this.on = on;
                this.recurday = recurday;
                this.recurweek = recurweek;
        }

        /*
         * Konverterar en byte-vektor till en arraylist med TimerEvents.
         */
        public static ArrayList <TimerEvent> toTimerEvent(byte[] bytes, int size) {

                ArrayList<TimerEvent> ret = new ArrayList<TimerEvent>();
                for(int j = 0;j<size/4;j++){

                        boolean on = (bytes[j*4]&128)!=0;
                        boolean daily = (bytes[j*4]&64)!=0;
                        boolean weekly = (bytes[j*4]&32)!=0;
                        int leg =(bytes[j*4]&28)>>2;

                        long time = 0;
                        bytes[j*4]&=0x3;
                        time+=bytes[j*4]&0xff;
```

```
                time<<=8;
                time+=bytes[j*4+1]&0xff;

                time<<=8;
                time+=bytes[j*4+2]&0xff;

                time<<=8;
                time+=bytes[j*4+3]&0xff;

                ret.add(new TimerEvent(time, leg, on, daily, weekly));
        }
        return ret;
}

/*
 * Returnerar tiden för timereventet i normal form
 */
public int[] normalTime() {
        long tark = time;
        int[] retVal = new int[5];
        GregorianCalendar cal = new GregorianCalendar();
        cal.setTimeInMillis(tark*60000+new GregorianCalendar(2000, 0, 1, 0 , 0).getTimeInMillis());
        retVal[0] = cal.get(Calendar.YEAR);
        retVal[1] = cal.get(Calendar.MONTH)+1;
        retVal[2] = cal.get(Calendar.DAY_OF_MONTH);
        retVal[3] = cal.get(Calendar.HOUR_OF_DAY);
        retVal[4] = cal.get(Calendar.MINUTE);

        return retVal;
}

/*
 * Returnerar antalet minuter sen år 2000
 */
public static long getMinsSince2000(){
        Calendar cal = new GregorianCalendar(2000, 0, 1);
        long ret = System.currentTimeMillis()-cal.getTimeInMillis();
        ret/=60000;
        return ret;
}

/*
 * Konverterar en arraylist med timerevents till en byte-vektor
 */
public static byte[] fromTimeEvent(ArrayList<TimerEvent> te) {

        byte[] returnArray = new byte[te.size()*4];


        for(int i = 0;i<te.size();i++){
                TimerEvent temp = te.get(i);
                returnArray[i*4] = 0;
                if(temp.isOn())
                        returnArray[i*4] += 128;
                if(temp.isRecurday())
                        returnArray[i*4] += 64;
                if(temp.isRecurweek())
                        returnArray[i*4] += 32;

                returnArray[i*4] += temp.getAusgang()<<2;
                long l = temp.getTime();
                returnArray[i*4] += l>>24;
                returnArray[i*4+1] += (l&0xFF0000)>>16;
                returnArray[i*4+2] += (l&0x00FF00)>>8;
                returnArray[i*4+3] += (l&0x0000FF);

        }
        return returnArray;
}


/*
 * Returnerar antalet minuter sen år 2000
 */
```

```
public static long minutesSince2000(int y, int m, int d, int h, int min) {
        long time = 0;
        time = new GregorianCalendar(y, m-1, d, h , min).getTimeInMillis()-
                new GregorianCalendar(2000, 0, 1, 0 , 0).getTimeInMillis();
        time/=60000;
        return time;
}

public String toString() {
        int[] ret = this.normalTime();
        return ret[0] + "-" + ret[1] + "-" + ret[2] + " " + ret[3] + ":" + ret[4];
}

public boolean equals(Object t) {
        if(t==null)
                return true;
        return ((TimerEvent) t).time == this.time && ((TimerEvent) t).ausgang == this.ausgang &&
                ((TimerEvent) t).on == this.on && ((TimerEvent) t).recurday == this.recurday &&
                        ((TimerEvent) t).recurweek == this.recurweek;
}

public int compareTo(TimerEvent t) {
        if(this.time == t.time) {
                if(this.ausgang == t.ausgang) {
                        if(this.on == t.on) {
                                if(this.recurday == t.recurday) {
                                        if(this.recurweek == t.recurweek) {
                                                return 0;
                                        } else {
                                                if(this.recurweek)
                                                        return -1;
                                                else
                                                        return 1;
                                        }
                                } else {
                                        if(this.recurday)
                                                return -1;
                                        else
                                                return 1;
                                }
                        } else {
                                if(this.on)
                                        return -1;
                                else
                                        return 1;
                        }

                } else {
                        return this.ausgang - t.ausgang;
                }

        }

        return (int) (this.time - t.time);
}


public int getAusgang() {
        return ausgang;
}

public boolean isOn() {
        return on;
}

public boolean isRecurday() {
        return recurday;
}

public boolean isRecurweek() {
        return recurweek;
}
public long getTime(){
        return time;
```

```
        }
}
```

11.3 TimerGUI

```
package digp15;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class TimerGUI {
    private JTabbedPane tabbedPane;
    private ArrayList<TimerEvent> events;
    private ComPortCommunication c;

    /*
     * Skapar ett timergui
     */
    public TimerGUI() {
        c = new ComPortCommunication();
        events = new ArrayList<TimerEvent>();

        JFrame frame = new JFrame("Timer");
        DefaultListModel nameListModel = new DefaultListModel();
        tabbedPane = new JTabbedPane();

        CalendarPane calPane = new CalendarPane(events, c, nameListModel);
        tabbedPane.addTab("Kalendervy", null, calPane, "Visa kalendervy");

        ListPane listPane = new ListPane(events, c, nameListModel);
        tabbedPane.addTab("Listvy", null, listPane, "Visa listvy");
        tabbedPane.setSelectedIndex(0);

        frame.getContentPane().add(tabbedPane,BorderLayout.CENTER);
        tabbedPane.addChangeListener(new ChangeHandler());
        frame.addWindowListener(new WindowHandler());

        frame.setSize(580, 420);
        frame.setVisible(true);
    }

    class ChangeHandler implements ChangeListener {
         public void stateChanged(ChangeEvent e) {
            BasicPane selectedPane =
                (BasicPane) tabbedPane.getSelectedComponent();
            selectedPane.entryActions();
        }
    }

    class WindowHandler extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    }
}
```

11.4 ComPortCommunication

```
package digp15;

import java.io.*;
import java.util.*;

import javax.comm.*;

public class ComPortCommunication {
    private InputStream in;
    private OutputStream ut;
    private SerialPort serialPort;
```

```
/*
 * Denna klass sköter kommunikationen över com-porten
 */
    public ComPortCommunication() {
            String driverName = "com.sun.comm.Win32Driver";
            try{
                    CommDriver commdriver = (CommDriver)Class.forName(driverName).newInstance( );
                    commdriver.initialize();
                    CommPortIdentifier c = javax.comm.CommPortIdentifier.getPortIdentifier("COM1");
                    serialPort = (SerialPort) c.open(driverName, 1000);
                    serialPort.setSerialPortParams(2400, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
                            SerialPort.PARITY_NONE);
                    in = serialPort.getInputStream();
                    ut = serialPort.getOutputStream();
            } catch (Exception e) {
                    e.printStackTrace();
            }
    }

    /*
     * Returnerar en arraylist med alla events som fanns i hårdvaran
     */
    public ArrayList<TimerEvent> getEvents(){

        byte[] readBuffer = new byte[508];
        int read=0;
        try {
            ut.write((byte) 1);

            while(in.available() == 0) { ; }
            while(in.available() != 0) {
                    readBuffer[read++] = (byte) in.read();
                    Thread.sleep(10);
            }

        } catch (Exception e) { e.printStackTrace(); }

        return TimerEvent.toTimerEvent(readBuffer,read);
    }

    /*
     * Synkronisera klockan
     */
    public boolean syncClock() {
            try {
                    ut.write((byte) 2);
                    sendClock();
            } catch (Exception e) {e.printStackTrace();}

            return true;
    }

    private boolean sendClock(){
            long l = TimerEvent.getMinsSince2000();
            byte[] b = new byte[4];

            b[0] += (l&0xFF000000)>>24;
            b[1] += (l&0x00FF0000)>>16;
            b[2] += (l&0x0000FF00)>>8;
            b[3] += (l&0x000000FF);
            try {
                    for(int i =0;i<b.length;i++){
                            ut.write(b[i]);
                    Thread.sleep(10);
            }
            } catch (Exception e) {e.printStackTrace(); return false;}
            return true;
    }

    /*
     * Skicka alla events till hårdvaran
     */
    public boolean sendEvents(ArrayList<TimerEvent> te){
```

```
                try {
                ut.write((byte) 3);
                ut.write((byte)te.size());

                byte[] b = TimerEvent.fromTimeEvent(te);
                for(int i =0;i<b.length;i++){
                        ut.write(b[i]);
                        Thread.sleep(10);
                }
                sendClock();
            } catch (Exception e) { return false; }
                return true;
        }
}
```

11.5 TimerCal

```
package digp15;

import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class TimerCal extends Canvas implements MouseListener{
        private Image i;
        private GregorianCalendar cal;
        private int sqX, sqY;
        private boolean drawSquare;
        private ArrayList<TimerEvent> events;
        private CalendarPane l;
        private int day;

        /*
         * Skapar en grafisk kalender
         */
        public TimerCal(GregorianCalendar cal, ArrayList<TimerEvent> events, CalendarPane l) {
                this.l = l;
                this.events = events;
                this.cal = cal;

            this.addMouseListener(this);
            }

        /*
         * Byter år och månad samt ritar om kalendern
         */
        public void setYM(int y, int m) {
                cal.set(y, m, 1);
                drawSquare = false;
                this.repaint();
        }

        public void paint(Graphics g) {
                g.setColor(Color.RED);
                g.drawString("Sö", 12, 20);
                g.setColor(Color.BLACK);
                g.drawString("Må", 52, 20);
                g.drawString("Ti", 92, 20);
                g.drawString("On", 132, 20);
                g.drawString("To", 172, 20);
                g.drawString("Fr", 212, 20);
                g.drawString("Lö", 252, 20);

                for(int i=0; i<=7; i++)
                        g.drawLine(i*40, 30, i*40, 270);
                for(int i=0; i<=6; i++)
                        g.drawLine(0, i*40+30, 280, i*40+30);

                        int xc = cal.get(GregorianCalendar.DAY_OF_WEEK)-1,yc=1;

                        for(int i=1; i<=cal.getActualMaximum(Calendar.DAY_OF_MONTH);i++) {
                                g.drawString(i+"", xc*40+8, yc*40+6);
                                xc++;
```

```
                        if(xc == 7) {
                                xc=0;
                                yc++;
                        }
                }
                for(int i=0; i<events.size(); i++) {

                        int[] r = events.get(i).normalTime();
                        if(l.getYear() == r[0] && l.getMonth()+1 == r[1]) {
                                Color onoff;
                                int geschussen;
                                if(events.get(i).isOn()){
                                        onoff = Color.green;
                                        geschussen = 17;
                                }
                                else{
                                        onoff = Color.red;
                                        geschussen = -3;
                                }

                                int x =((cal.get(GregorianCalendar.DAY_OF_WEEK)+r[2]-2)%7)*40+8;
                                int y = ((cal.get(GregorianCalendar.DAY_OF_WEEK)+r[2]-2)/7+1)*40+6;
                                g.setColor(onoff);
                                g.drawOval(x+geschussen, y+8, 10, 10);
                                g.setColor(Color.BLACK);

                        }
                }

                if(drawSquare) {
                        g.setColor(Color.RED);
                        g.drawRect(sqX, sqY, 40, 40);
                        g.drawRect(sqX+1,sqY+1,38,38);
                        g.drawRect(sqX+2,sqY+2,36,36);
                        g.setColor(Color.BLACK);
                }
    }

    /*
     * Lyssnar på musklick
     */
public void mousePressed(MouseEvent e) {
    int xc = e.getX()/41;
    int yc = (e.getY()-30)/41;
    if(xc+yc*7 >= cal.get(GregorianCalendar.DAY_OF_WEEK)-1 && xc+yc*7 < cal.get(GregorianCalendar.DAY_OF_WEEK)-1+
            cal.getActualMaximum(Calendar.DAY_OF_MONTH)) {

            drawSquare = true;
            sqX = xc*40;
            sqY = yc*40+30;
            repaint();
            day = xc+yc*7-cal.get(GregorianCalendar.DAY_OF_WEEK)+2;
            ArrayList<Integer> a = new ArrayList<Integer>();

            l.setDates(day);

            l.getEventsOfTheDay().removeAll();
            for(int i=0; i<events.size(); i++) {
                        int[] r = events.get(i).normalTime();
                        if(l.getYear() == r[0] && l.getMonth()+1 == r[1] && day == r[2]) {
                                a.add(i);
                                l.getEventsOfTheDay().add(events.get(i).toString());
                        }
                }
            l.setAnIndexWhyNot(a);
    }
 }

 public void mouseReleased(MouseEvent e) {
 }

 public void mouseEntered(MouseEvent e) {
 }
```

```
    public void mouseExited(MouseEvent e) {
    }

    public void mouseClicked(MouseEvent e) {
    }

}
```

11.6 BasicPane

```
package digp15;

import javax.swing.*;
import java.awt.*;
import java.util.*;

public class BasicPane extends JPanel {
        protected ArrayList<TimerEvent> events;
        protected DefaultListModel nameListModel;
        protected JTextField[] fields;
        protected JTextArea log;
        protected Choice legs;
        protected Checkbox onoff, recday, recweek;
        protected JList nameList;

        /*
         * De två panelerna ärver denna grundlayouten
         */
        public BasicPane(ArrayList<TimerEvent> te, DefaultListModel nameListModel) {
        this.events=te;
        this.nameListModel = nameListModel;

        setLayout(new BorderLayout());
        JComponent leftPanel = createLeftPanel();
        add(leftPanel, BorderLayout.WEST);

        JPanel rightPanel = new JPanel();
        rightPanel.setLayout(new BorderLayout());

        JComponent topPanel = createTopPanel();
        JComponent middlePanel = createMiddlePanel();
        rightPanel.add(topPanel, BorderLayout.NORTH);

        rightPanel.add(middlePanel, BorderLayout.CENTER);
        add(rightPanel, BorderLayout.CENTER);
    }

    /*
     * Lägg till en händelse i händelselistan
     */
    protected boolean addEvent(){
        if(events.size()>=127) {
                log.setText("Händelseminnet fullt!");
                return false;
        }

        log.setText("");
                boolean error = false;
                String message="";
                int year = 0, month = 0, day = 0, hours = 0, minutes = 0;
                try{
                year = Integer.parseInt(fields[0].getText());
                if(year<2000){
                        error = true;
                        message = "Felaktigt år!\n";
                }
                month = Integer.parseInt(fields[1].getText());
                if(month<1 || month>12){
                        message+="Felaktig månad!\n";
                        error = true;
                }
                day = Integer.parseInt(fields[2].getText());
                Calendar cal = new GregorianCalendar(year, month-1, 1);
```

```
            if(day<1 || day>cal.getActualMaximum(Calendar.DAY_OF_MONTH)){
                        error = true;
                        message+="Felaktig dag!\n";
            }
                String[] hm = fields[3].getText().split(":");
                if(hm.length!=2) {
                        error = true;
                        message+="Felaktigt antal ':' i tidpunkt!\n";
                } else {
                        hours = Integer.parseInt(hm[0]);
                        minutes = Integer.parseInt(hm[1]);
                        if(hours < 0 || hours >= 24)  {
                                error = true;
                                message+="Felaktigt timtal!\n";
                        }
                        if(minutes < 0 || minutes >= 60) {
                                error = true;
                                message+="Felaktigt minuttal!\n";
                        }
                        }

                }catch (Exception ex){message+="Endast siffror tillåtna!\n"; error = true;}

                if(!error) {
                        message = "Ny händelse sparad!";
                        TimerEvent n = new TimerEvent(TimerEvent.minutesSince2000(year, month,
                               day, hours, minutes),legs.getSelectedIndex(),onoff.getState(),
                                       recday.getState(),recweek.getState());
                        if(!events.contains(n)){
                                events.add(n);

                        }
                        else
                                message = "En likadan händelse finns redan!";
                        fillEventList();
                        if(nameList!=null)
                                nameList.setSelectedIndex(Collections.binarySearch(events, n));

                }
                log.setText(message);
                return error;
    }

    /*
     * Sortera händelselistan och lägger till händelserna i gui:t.
     */
    protected void fillEventList() {
        nameListModel.removeAllElements();

        Collections.sort(events);
        for(int i=0; i<events.size(); i++)
                nameListModel.addElement(events.get(i));

    }

    public void entryActions() {}

    public JComponent createLeftPanel() {
        return new JPanel();
    }

    public JComponent createTopPanel() {
        return new JPanel();
    }

    public JComponent createMiddlePanel() {
        return new JPanel();
    }
}
```

11.7 CalendarPane

```
package digp15;
```

```
import javax.swing.*;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class CalendarPane extends BasicPane implements ActionListener, ItemListener {
        private TimerCal c;
        private Choice months;
        private JTextField year;
        private Button update;
        private GregorianCalendar cal;
        private ComPortCommunication com;
    private Choice eventsOfDay;
    private ArrayList<Integer> indexes;
    JButton[] buttons, hwbuttons;

    /*
     * Skapa kalendervyn
     */
    public CalendarPane(ArrayList<TimerEvent> events, ComPortCommunication com, DefaultListModel nameListModel) {
        super(events, nameListModel);
        this.com = com;
    }

    /*
     * Fyll i datumet
     */
    public void setDates(int d) {
        fields[0].setText(year.getText());
        fields[1].setText(months.getSelectedIndex()+1+"");
        fields[2].setText(d+"");
        fields[3].setText("00:00");

    }


    public JComponent createMiddlePanel() {
        eventsOfDay = new Choice();
        eventsOfDay.addItemListener(this);
        eventsOfDay.add("                            ");
        JPanel p = new JPanel(new BorderLayout());
        p.add(eventsOfDay,BorderLayout.NORTH);

        String[] texts = new String[5];
        texts[0] = "Tid";
        texts[1] = "Utgång";
        texts[2] = "På";
        texts[3] = "Veckovis";
        texts[4] = "Dagsvis";

        JPanel left = new JPanel();
        JPanel timePanel = new JPanel();

        timePanel.setLayout(new FlowLayout(FlowLayout.LEFT));
        GregorianCalendar cal = new GregorianCalendar(2000,1,1);
        cal.setTimeInMillis(System.currentTimeMillis());
        fields = new JTextField[4];
        fields[0] = new JTextField(cal.get(GregorianCalendar.YEAR)+"","2000".length());
        fields[1] = new JTextField(cal.get(GregorianCalendar.MONTH)+1+"","05".length());
        fields[2] = new JTextField(cal.get(GregorianCalendar.DAY_OF_MONTH)+"","17".length());
        fields[3] = new JTextField(cal.get(GregorianCalendar.HOUR_OF_DAY)+":"+cal.get(GregorianCalendar.MINUTE),
                "00:00".length());

        timePanel.add(new JLabel(texts[0] + "      ", JLabel.LEFT));

        for (int i = 0; i < fields.length; i++) {
            timePanel.add(fields[i]);
            if(i<2)
                timePanel.add(new JLabel("-"));
            else
                timePanel.add(new JLabel(" "));
        }
```

```
left.add(timePanel);

JPanel legPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

JPanel nestledLayout1 = new JPanel(new BorderLayout());

legPanel.add(new JLabel(texts[1] + "                                          ", JLabel.LEFT));
legs = new Choice();
for(int i=1; i<=8; i++)
        legs.add("Utgång " + i);
legPanel.add(legs);
nestledLayout1.add(legPanel, BorderLayout.NORTH);

JPanel onoffPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
onoffPanel.add(new JLabel(texts[2] + "                                                    ", JLabel.LEFT));
onoff = new Checkbox();
onoffPanel.add(onoff);
nestledLayout1.add(onoffPanel,BorderLayout.CENTER);
JPanel nestledLayout2 = new JPanel(new BorderLayout());
JPanel recDayPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

recDayPanel.add(new JLabel(texts[3] + "                                        ", JLabel.LEFT));
recday = new Checkbox();
recDayPanel.add(recday);
nestledLayout2.add(recDayPanel,BorderLayout.NORTH);

JPanel recWeekPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
recWeekPanel.add(new JLabel(texts[4] + "                                          ", JLabel.LEFT));
recweek = new Checkbox();
recWeekPanel.add(recweek);
nestledLayout2.add(recWeekPanel,BorderLayout.CENTER);
nestledLayout1.add(nestledLayout2, BorderLayout.SOUTH);
left.add(nestledLayout1);

JPanel buttPanel = new JPanel();
buttPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
buttons = new JButton[3];
buttons[0] = new JButton("Spara som ny");
buttons[1] = new JButton("Spara");
buttons[2] = new JButton("Ta bort");
for(int i=0; i<buttons.length; i++) {
        buttPanel.add(buttons[i]);
        buttons[i].addActionListener(this);
}
left.add(buttPanel);


JPanel logPanel = new JPanel();
logPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
log = new JTextArea("");
log.setBackground(new Color(238,238,238));
log.setEditable(false);
log.setRows(2);
log.setText("                                                                                ");
logPanel.add(log);
logPanel.setBorder(new TitledBorder("Senaste aktivitet"));
        left.add(logPanel);

        JPanel buttPanel2 = new JPanel();
buttPanel2.setLayout(new FlowLayout(FlowLayout.CENTER));
hwbuttons = new JButton[3];
hwbuttons[0] = new JButton("Hämta");
hwbuttons[1] = new JButton("Spara");
hwbuttons[2] = new JButton("Synkronisera");
for(int i=0; i<hwbuttons.length; i++) {
        buttPanel2.add(hwbuttons[i]);
        hwbuttons[i].addActionListener(this);
}
left.add(buttPanel2);

p.add(left, BorderLayout.CENTER);

return p;
```

```
}

public JComponent createLeftPanel() {
    cal = new GregorianCalendar(2000,1,1);
    cal.setTimeInMillis(System.currentTimeMillis());
    cal.set(GregorianCalendar.DATE, 1);
    c = new TimerCal(cal, events, this);
    months = new Choice();
    year = new JTextField(cal.get(GregorianCalendar.YEAR)+"","2OXX".length());
    update = new Button("Uppdatera");
    update.addActionListener(this);
    months.add("Januari");
    months.add("Februari");
    months.add("Mars");
    months.add("April");
    months.add("Maj");
    months.add("Juni");
    months.add("Juli");
    months.add("Augusti");
    months.add("September");
    months.add("Oktober");
    months.add("November");
    months.add("December");
    months.select(cal.get(GregorianCalendar.MONTH));
    JPanel p = new JPanel(new BorderLayout());

    JPanel selYM = new JPanel(new FlowLayout(FlowLayout.CENTER));
    selYM.add(months);
    selYM.add(year);
    selYM.add(update);
    p.add(selYM,BorderLayout.NORTH);
    c.setBounds(10, 10, 281, 251);
    c.setBackground(Color.WHITE);
    p.add(c,BorderLayout.CENTER);
    c.repaint();
    return p;
}

/*
 * Lyssnar på dropdown-rutan
 */
public void itemStateChanged(ItemEvent e) {
    if(e.getSource() == eventsOfDay) {
            if(indexes != null && events.size()!=0) {
              int a = indexes.get(eventsOfDay.getSelectedIndex());
              TimerEvent sel = events.get(a);
              int[] t = sel.normalTime();
              onoff.setState(sel.isOn());
              recday.setState(sel.isRecurday());
              recweek.setState(sel.isRecurweek());
              legs.select(sel.getAusgang());
              fields[0].setText(t[0]+"");
              fields[1].setText(t[1]+"");
              fields[2].setText(t[2]+"");
              fields[3].setText(t[3]+":"+t[4]);

            } else {
                    eventsOfDay.removeAll();
            }

    }
}

/*
 * Lyssnar på knapparna
 */
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == update) {
            try {
                    c.setYM(Integer.parseInt(year.getText()), months.getSelectedIndex());
            } catch(Exception ee) {ee.printStackTrace();} {

                    }
    } else if(e.getSource() == buttons[0]) {
```

```
                addEvent();
                c.setYM(Integer.parseInt(year.getText()), months.getSelectedIndex());
        }
        else if(e.getSource() == buttons[1]) {
                if(indexes != null){
                        events.remove((int)indexes.get(eventsOfDay.getSelectedIndex()));
                        eventsOfDay.remove(eventsOfDay.getSelectedIndex());
                        fillEventList();

                        if(!addEvent()) log.setText("Händelse sparad!");
                        c.setYM(Integer.parseInt(year.getText()), months.getSelectedIndex());

                }
        else
                log.setText("Ingen händelse vald!");

        }else if(e.getSource() == buttons[2]) {
                if(indexes != null){
                        events.remove((int)indexes.get(eventsOfDay.getSelectedIndex()));
                        c.setYM(Integer.parseInt(year.getText()), months.getSelectedIndex());
                        eventsOfDay.remove(eventsOfDay.getSelectedIndex());
                        fillEventList();
                        log.setText("Händelse borttagen!");
                }
                else
                        log.setText("Ingen händelse vald!");

        } else if(e.getSource() == hwbuttons[0]) {


                events.removeAll(events);
                        events.addAll(com.getEvents());
                        fillEventList();
                        log.setText(events.size() +" händelser har lagts till!");



                        c.setYM(Integer.parseInt(year.getText()), months.getSelectedIndex());
                                indexes = null;
                                eventsOfDay.removeAll();

        } else if(e.getSource() == hwbuttons[1]){
                if(com.sendEvents(events))
                        log.setText("Skickade upp " + events.size() + " händelser!");
                else
                        log.setText("Kunde inte skicka!");

        } else if(e.getSource() == hwbuttons[2]){

                if(com.syncClock())
                        log.setText("Klockan synkroniserad!");
                else
                        log.setText("Kunde inte synkronisera!");

        }
    }


    public int getYear() {
        return Integer.parseInt(year.getText());
    }

    public int getMonth() {
        return months.getSelectedIndex();
    }

    public Choice getEventsOfTheDay() {
        return eventsOfDay;
    }

    public void setAnIndexWhyNot(ArrayList<Integer> indexes) {
        this.indexes = indexes;
    }
}
```

11.8 ListPane

```
package digp15;
import javax.swing.*;
import javax.swing.border.TitledBorder;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class ListPane extends BasicPane implements ActionListener, ListSelectionListener {
    private JButton[] buttons;
    private JButton[] hwbuttons;
    private JPanel p;
    private ComPortCommunication c;

    /*
     * Listvyn
     */
    public ListPane(ArrayList<TimerEvent>events, ComPortCommunication c, DefaultListModel nameListModel) {
        super(events, nameListModel);
        this.c = c;

    }

    public JComponent createLeftPanel() {
        nameList = new JList(nameListModel);
        nameList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        nameList.setPrototypeCellValue("2000-11-11 12:12");
        nameList.addListSelectionListener(this);
        JScrollPane p1 = new JScrollPane(nameList);
        JPanel p = new JPanel();
        p.setLayout(new GridLayout(1, 2));
        p.add(p1);
        return p;
    }

    public JComponent createTopPanel() {
        String[] texts = new String[5];
        texts[0] = "Tid";
        texts[1] = "Utgång";
        texts[2] = "På";
        texts[3] = "Återkommande veckovis";
        texts[4] = "Återkommande dagsvis";

        JPanel left = new JPanel();
        left.setLayout(new GridLayout(5, 2));

        JPanel timePanel = new JPanel();
        timePanel.setLayout(new FlowLayout(FlowLayout.LEFT));
        GregorianCalendar cal = new GregorianCalendar(2000,1,1);
        cal.setTimeInMillis(System.currentTimeMillis());
        fields = new JTextField[4];
        fields[0] = new JTextField(cal.get(GregorianCalendar.YEAR)+"","2000".length());
        fields[1] = new JTextField(cal.get(GregorianCalendar.MONTH)+1+"","05".length());
        fields[2] = new JTextField(cal.get(GregorianCalendar.DAY_OF_MONTH)+"","17".length());
        fields[3] = new JTextField(cal.get(GregorianCalendar.HOUR_OF_DAY)+":"+cal.get(GregorianCalendar.MINUTE),
                "00:00".length());

        for (int i = 0; i < fields.length; i++) {
            timePanel.add(fields[i]);
            if(i<2)
                timePanel.add(new JLabel("-"));
            else
                timePanel.add(new JLabel(" "));
        }

        left.add(new JLabel(texts[0] + "      ", JLabel.LEFT));
        left.add(timePanel);
        left.add(new JLabel(texts[1] + "      ", JLabel.LEFT));
        legs = new Choice();
        for(int i=1; i<=8; i++)
                legs.add("Utgång " + i);
        left.add(legs);
```

```
    left.add(new JLabel(texts[2] + "      ", JLabel.LEFT));
    onoff = new Checkbox();
    left.add(onoff);

    left.add(new JLabel(texts[3] + "      ", JLabel.LEFT));
    recday = new Checkbox();
    left.add(recday);

    left.add(new JLabel(texts[4] + "      ", JLabel.LEFT));
    recweek = new Checkbox();
    left.add(recweek);

    JPanel p1 = new JPanel();
    p1.setLayout(new FlowLayout(FlowLayout.LEFT));
    p = new JPanel();
    p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
    p.add(left);
    p.add(p1);

    JPanel buttPanel = new JPanel();
    buttPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
    buttons = new JButton[3];
    buttons[0] = new JButton("Spara som ny händelse");
    buttons[1] = new JButton("Spara");
    buttons[2] = new JButton("Ta bort");
    for(int i=0; i<buttons.length; i++) {
            buttPanel.add(buttons[i]);
            buttons[i].addActionListener(this);
    }
    p.add(buttPanel);


    JPanel logPanel = new JPanel();
    logPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
    log = new JTextArea("");
    log.setBackground(new Color(238,238,238));
    log.setEditable(false);
    log.setRows(6);
    logPanel.add(log);
    logPanel.setBorder(new TitledBorder("Senaste aktivitet"));
            p.add(logPanel);

            JPanel buttPanel2 = new JPanel();
    buttPanel2.setLayout(new FlowLayout(FlowLayout.CENTER));
    hwbuttons = new JButton[3];
    hwbuttons[0] = new JButton("Hämta händelser");
    hwbuttons[1] = new JButton("Spara händelser");
    hwbuttons[2] = new JButton("Synkronisera klockan");
    for(int i=0; i<hwbuttons.length; i++) {
            buttPanel2.add(hwbuttons[i]);
            hwbuttons[i].addActionListener(this);
    }
    p.add(buttPanel2);

    return p;
}

public void entryActions() {
    fillEventList();
}

/*
 * Lyssnar på knapparna
 */
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == buttons[0]) {
            addEvent();
    }
    else if(e.getSource() == buttons[1]) {
            if(nameList.getSelectedIndex()!=-1){
                    events.remove(nameList.getSelectedIndex());
                    if(!addEvent()) log.setText("Händelse sparad!");
            }
```

```
        else
                log.setText("Ingen händelse vald!");

        }else if(e.getSource() == buttons[2]) {
                if(nameList.getSelectedIndex()!=-1){
                        events.remove(nameList.getSelectedIndex());

                        fillEventList();
                        log.setText("Händelse borttagen!");
                }
                else
                        log.setText("Ingen händelse vald!");

        } else if(e.getSource() == hwbuttons[0]) {


                        events.removeAll(events);
                                events.addAll(c.getEvents());
                                fillEventList();
                                log.setText(events.size() +" händelser har lagts till!");

        } else if(e.getSource() == hwbuttons[1]){
                if(c.sendEvents(events))
                        log.setText("Skickade upp " + events.size() + " händelser!");
                else
                        log.setText("Kunde inte skicka!");

        } else if(e.getSource() == hwbuttons[2]){
                if(c.syncClock())
                        log.setText("Klockan synkroniserad!");
                else
                        log.setText("Kunde inte synkronisera!");

        }
    }

    /*
     * Lyssnar på listan
     */
    public void valueChanged(ListSelectionEvent e) {
        if (nameList.isSelectionEmpty())
                return;
        int a = nameList.getSelectedIndex();
        TimerEvent sel = events.get(a);
        int[] t = sel.normalTime();
        onoff.setState(sel.isOn());
        recday.setState(sel.isRecurday());
        recweek.setState(sel.isRecurweek());
        legs.select(sel.getAusgang()); // tog bort -1
        fields[0].setText(t[0]+"");
        fields[1].setText(t[1]+"");
        fields[2].setText(t[2]+"");
        fields[3].setText(t[3]+":"+t[4]);

    }
}
```