Physics Simulation with Java (5A14219):

A Task in Java Embedded Circuit Programming with Javelin Stamp®

Jaakko Pajunen, 800110-A891
jaakko.pajunen@hut.fi
17 March 2004

Summary

1. Introduction

Due to growing demand of employees understanding embedded programming an alternative way to pass course Physics Simulation with Java (5A14219) was arranged. This path required a task with e.g. Javelin circuit breadboard (copyright Parallax Inc. www.parallax.com). The Javelin Stamp is a Java-based integrated circuit, which uses a subset of Sun Java 1.2® and several built-in classes and methods optimised for this stamp alone. The breadboard allows many simultaneous sensors and devices and grants a possibility to AD/DA conversions.

The original idea was to make the Javeling Stamp act as a greenhouse regulator with sensors for temperature, luminosity, pressure and / or humidity. Due to lack of appropriate sensors, however, the task was slightly rearranged so that in use is thermometer for temperature, photoresistor for dim light intensity sensing and solar panel for daylight intensity measures. These sensors are used to register daily light changes in the amount of light and basic properties of solar cells. The solar panel is also used to measure some basic laws of light intensity.

2. Equipment

The task was divided into two parts. In the first part, a long data is collected to see how the sunlight luminosity and room temperature change as a function of time. Secondly, the properties of the solar panel are studied more closely and some basic laws like squared distance are shown to be true. Before the experiments we make a quick survey on the components used on the Javalin breadboard and overall features of Javelin Stamp.

2.1 Javelin Stamp

The Javelin Stamp is a single board computer that's designed to function as an easy-touse programmable brain for electronic products and projects. It is programmed using software on a PC and a subset of Sun Microsystems Java® programming language. After the program is downloaded to the Javelin, it can run the program without any further help from the PC. The programs are written and debugged on Javelin Stamp Integrated Development Environment provided with the stamp. The Javelin can be programmed and re-programmed up to one million times. The Javelin has 32k of RAM/program memory with a flat architecture and built-in Virtual Peripherals (VPs) that take care of serial communication, pulse width modulation and tracking time in the background. Also delta-sigma A/D conversion is included and D/A conversion is accomplished as a continuous pulse train delivered by I/O pin.

The Javelin Stamp requires a DV power source between 6 and 24 V and makes 5 V available for the Javelin Stamp with a total current budget of 150 mA. Javelin itself consumes approximately 60 mA so that 90 mA is left for other uses. An external power source for other devices is also possible to connect with the stamp.

On the breadboard are 17 rows of connection sites in two discrete columns (not connected to each other) and 16 I/O pins for Javalin. Each I/O pin can work as a output or input for voltage and current. There are also four 5V supply voltage connections (Vdd) as also for ground (Vss). Other connections for external power source/ battery pack and servos are included. Figure 1. clarifies these connections.

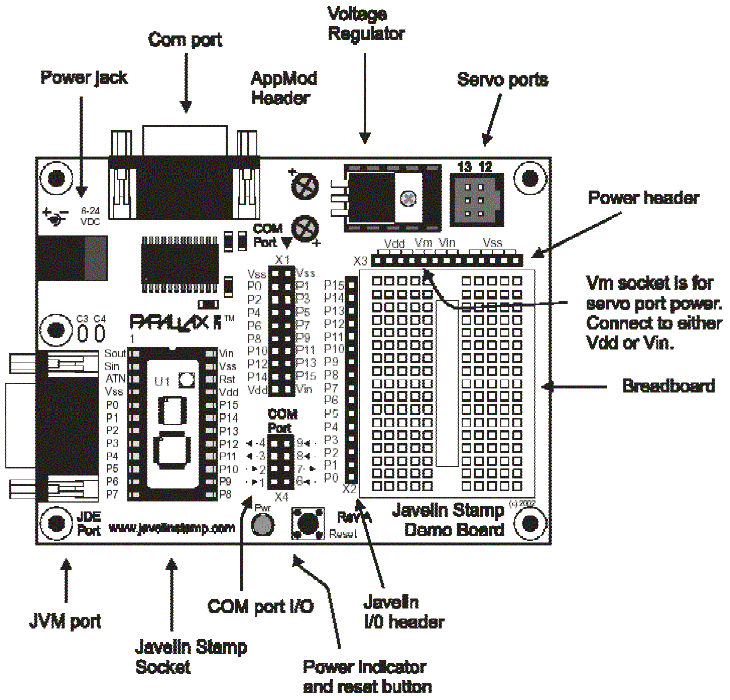
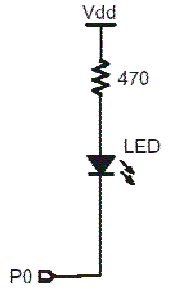


Figure 1. Javelin Stamp and its main components.

2.2 Passive components

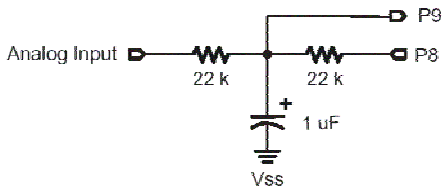
The following components and connections were used on the Javelin breadboard.

A) Light emiting diode, LED, is connected in between an I/O pin and voltage supply with a resistor so that when the LED is on state, the I/O pin is an input.



B) Analog to Digital conversion is done with the aid of two external resistors and one external capacitor. This conversion requires two I/O pins, outPin pins sends pulses and inPin monitors the voltage across the capacitor. Analog input voltage is determined indirectly by keeping the capacitor at a constant 2.5 V. Since the input voltage will affect how many pulses must be sent to keep the capacitor charged at 2.5 V, input level can be computed. The exact formulas are left out from the Javelin manual, but basically the AD converter counts how many 5 V pulses lasting τ parts of an epoch must be sent in, so that the voltage between inPin and Vss is constant.

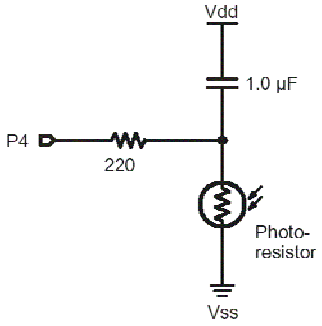
$$V_{\text{Analog}} = \tau \cdot 5\,\text{V} \tag{2.1}$$



C) The luminosity of smaller light intensities is measured with a photoresistor, a resistor that changes its resistance in proportion to absorbed (sun) light intensity. The connection below forms a RC-circuit. Time constant

$$\tau = R \cdot C \tag{2.2}$$

allows characterization of the circuit by the components. If one of the components is unknown, it can be derived indirectly by measuring or knowing the two other. In this case resistance varies so that with a known capacitance and measured time constant it can be computed.



A capacitor is charged when there is voltage difference between its plates (Vdd >< Vss). While measuring the capacitor discharge, the I/O pin 4 in figure is changed from output to input. When the capacitor voltage reaches logical CMOS threshold of 2.5 V timing stops. Circuit above is characterized by eq. (2.3)

$$\frac{dq(t)}{dt} + \frac{q(t)}{RC} = \frac{V(t)}{R} \tag{2.3}$$

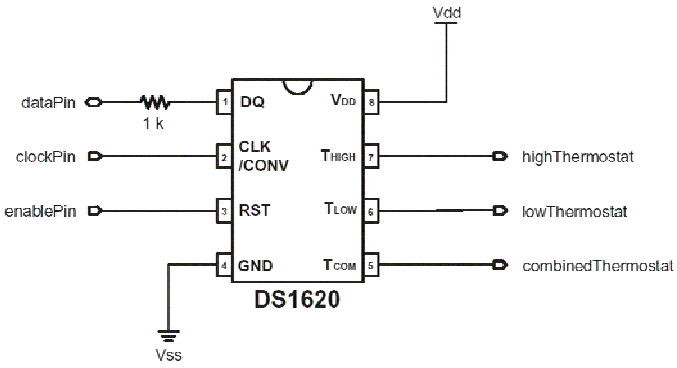
and its solution on a discharging capacitor

$$V_C(t) = V_0 e^{-t/RC} . \tag{2.4}$$

Time constant follow easily from the time that the capacitor uses to reach the CMOS threshold

$$t = \tau \ln \frac{V_0}{V_{CMOS}} . \tag{2.5}$$

D) Temperatures are measured with an integrated circuit processor (DS1620), for which Javelin Stamp has class included in the programming language. Details of the class methods are in the attached code section. Exact details of how such a processor works are beyond the capacity of this work.



E) Solar panel with 6 circular cells. Maximum DV output 3 volts.

3. Sunlight intensity and room temperature

Temperature and light intensity were measured once in six minutes (10 times / hour) for approximately 41 hours. The measurement started at 00.53 on the 11th March and ended at approximately 18.00 on the 12th March 2004. The solar panel and Javelin Stamp were sat on a window-sill pointing to north-west so that there was no direct sunlight on the sensors. Light sensitive photoresistor was covered with an antistatic back that absorbs some of the light so that not all the resolution at very dark light intensities. Under the window-sill located a central air-conditioning ventilation so that the room temperature was there somewhat higher. Then again even indirect sunlight was predicted to be able to rise and lower the temperature near the window.

Measurement was carried out as the Javelin Stamp was a greenhouse regulator or a weather station. If the temperature or lightning was not in given limits an alarm arose. For insufficient light alarm was a green led and for too cold a temperature a red led. Limits for light are 150 in Luminosity Index (from 1 to 32768) and temperature outside the interval in between 22.5 and 27 degrees Celsius. Solar panel is here considered only as a battery or external power which voltage is measured. Solar panel is not as sensitive to light intensity changes and it gives a better overall picture of sunlight but does not have such good resolution with dim light. All the measured values were considered as output, which is printed to Javelin Stamp Output Terminal from where it's copied to a file. Here follows the code.

Main class: WheatherStation.class

```
import stamp.peripheral.sensor.temperature.DS1620;
import stamp.core.*;

/**
 * WheatherStation for Javalin breadboard.
 * <p>
 * WheatherStation is main class for wheather station.
 * It runs the main -method.
 *
 * @version 1.0 13.2.2004
 * @author Jaakko Pajunen
 */

class WheatherStation {

  // Some static parameters
  static int temp_status, light, temp, volts, i;
  static boolean luminosity_status;
//  final static char HOME = 0x01;
  static String tmp;
  static StringBuffer msg = new StringBuffer(18);

  public static void main() {

      //Minimum and maximum allowed temperature in degrees Celcius
      int min_temp=22, max_temp=27;   //Actual min_temp=min_temp+0.5

      //Minimum amount of light in pseudounits (1-Integer.MAX_VALUE)
      int min_light=150;

      //Time to make measurements in seconds.
      int measure_freq = 360; //6 Minutes >> 10 times / hour

      //DS1620 connections on test board
      int data = CPU.pins[8], clock = CPU.pins[7], enable = CPU.pins[6];

      //Photoresistor connection on test board
      int photopin = CPU.pins[15];

      // Alarm connections on testboard.
      int green_led_pin=CPU.pins[3], red_led_pin=CPU.pins[5];

      //Create different sensor objects
      Thermometer warmth = new Thermometer(data,clock,enable,min_temp,max_temp);
      Luminosity luminosity = new Luminosity(min_light,photopin);
      PowerSource voltage = new PowerSource(CPU.pins[1],CPU.pins[0]);

      //Create alarm objects for sensors
      Alarm heater = new Alarm(red_led_pin);
      Alarm lamps = new Alarm(green_led_pin);


      //Loop until forever
      while(true){
          //  Measures luminosity and returns int
          light=luminosity.measure();

          //Returns temperature in (degrees Celcius)*10.
          temp=warmth.measure();

          //Returns pressure in xxx.

          volts=voltage.measure();

          // If temperature is too cold, heater (red led) turns on.
          // If there is not enough light, lamps (green led) turn on.
          regulate(warmth,luminosity,heater,lamps);

          //Print the readings on the terminal
          message(temp,temp_status,light,msg,volts);

          //Wait for measure_freq seconds
          wait(measure_freq);
    }//End while
```

```
  }//End main

  //Method for regulating observables.
  public static void regulate(Thermometer warmth, Luminosity luminosity, Alarm heater,
Alarm lamps)
  {
    temp_status = warmth.status();
    luminosity_status=luminosity.status();

    //Handles the status got from warmth object.
    if (temp_status == 1)
        heater.onAlarm();
      else if (temp_status == 0)
        heater.offAlarm();
      else if (temp_status == -1)
        heater.onAlarm();
      else
        System.out.println("Malfunction in Thermometer");

    //Handless the status got from luminosity object.
    if (luminosity_status == false)
        lamps.onAlarm();
    else
        lamps.offAlarm();
  }

  //Method for printing the status quo on the terminal.
  public static void message(int temp,int temp_status,int light, StringBuffer msg, int
bar)
  {

      tmp=Integer.toString(temp);

      msg.clear();
      for(i=0;i<tmp.length()-1;i++)
          msg.append(tmp.charAt(i));
      msg.append(".");
      msg.append( tmp.charAt(tmp.length()-1) );
      msg.append(" ");
      msg.append(Integer.MAX_VALUE/light);
      msg.append(" ");
      msg.append(bar);
      msg.append(" "); msg.append(temp_status);
      if (luminosity_status)
          msg.append(" 1");
      else
          msg.append(" 0");
      System.out.println(msg.toString());

  }

  //Wait method for int sek seconds.
  public static void wait(int sek){
      for (i=0;i<sek;i++)
        CPU.delay(10473); // Delay of 95.48 us x 10473 = 0.999962 s.
  }
}  //END CLASS
```

Alarm.class

```
import stamp.core.*;
/**
 * Alarm object for warnings.
 * <p>
 * Construction of alarm object requires knowledge of the pin where the
 * alarming devise is connected, that is Alarm(int CPU.pinnumber);
 *
 * Two methods onAlarm() and offAlarm() handle wheter the alarm is on of not.
 *
 * @version 1.0 13.02.2004
 * @author Jaakko Pajunen
 */

public class Alarm {
```

```
  private int pinN;
  final static boolean ON = false;
  final static boolean OFF = true;

  public Alarm(int pinN)
  {
      this.pinN=pinN;
  }

  public void onAlarm()
  {
   CPU.writePin(pinN,ON);
  }

  public void offAlarm()
  {
   CPU.writePin(pinN,OFF);
  }

}//END CLASS


import stamp.peripheral.sensor.temperature.DS1620;
import stamp.core.*;
/**
 * Thermometer does not extend DS1620 class but is near the same with some extra
methods.
 * <p>
 * Create new Thermometer object with constructor Thermometer(datapin,
 * clockpin, enablepin,minimum allowed temperature, maximum temperature);
 *
 * Class has two mehtods, measure() that returns 10*temperature in
 * degrees Celcius and status() that returns int depending on the warmth and
 * the minimum/maximum allowed values.
 *
 *
 * @version 1.0 13.02.2004
 * @author Jaakko Pajunen
 */
```

Thermometer.class

```
public class Thermometer{

    private static int degrees;
    private static DS1620 temperature;

    public Thermometer(){
        temperature = new DS1620(CPU.pins[8],CPU.pins[7],CPU.pins[6]);
        temperature.setTempHi(20,'C');
        temperature.setTempLo(26,'C');
    }

    public Thermometer(int data, int clock, int enable, int min, int max ){
        temperature = new DS1620(data,clock,enable);
        temperature.setTempHi(max,'C');
        temperature.setTempLo(min,'C');
    }

    //Measure method. Saves measurement to degrees variable.
    //Returns 10*degrees so that all the resolution of the temperature
    //sensor can be used (otherwise 0.5 degrees wouldn't count).
    public int measure()
    {
        try{
              do{
                  this.degrees = temperature.getTempRaw();
              }while(this.degrees==0);
            return 5*this.degrees;
            }

        catch (Exception e){
            System.out.println("Degrees virhe");
            return 10;
```

```
        }

    }
    //Returns the status value of temperature depending on the set min/max values
    public int status()
    {
      if (temperature.tempLo())
        return -1;
      else if (temperature.tempOk())
        return 0;
      else if (temperature.tempHi())
        return 1;
      else
        return 2;
    }

}
```

Luminosity.class

```
import stamp.core.*;

/**
 * A luminositymeasurer with photoresistor.
 * <p>
 * Call on object with Luminosity(int minimumlight, int photopin)
 * Class has two methods. measure() returns xxx int value and staus
 * boolean value if there is enough light compared with minimumlight.
 *
  *
 * @version 1.0 13.02.2004
 * @author Jaakko Pajunen
 */

public class Luminosity{
    private int pin, min, dischargeTime;

    public Luminosity(){
    }

    public Luminosity(int min,int pin){
      this.pin = pin;
      this.min = min;
    }

    //Light amount can be derived from /tau = RC, where capacity C is constant
    //and dischargetime /tau measured so that R can be calculated.
    public int measure()
    {
      try{
            CPU.writePin(pin,true);
            CPU.delay(10);
            this.dischargeTime = CPU.rcTime(5000,pin,false);
            if(this.dischargeTime==-1) this.dischargeTime=Integer.MAX_VALUE;
          return this.dischargeTime;
      }
      catch (Exception e){
          System.out.println("DischargeTime error");
          return 0;
      }
    }

    public boolean status()
    {
        if (dischargeTime > min)
          return false;   //Return false when there is not enough light
        else
          return true;
    }
}
```

PowerSource.class

```
import stamp.core.*;
/**
 * A solarpanel with DC voltage.
 * <p>
 * Call on object with PowerSource(int CMOSpin, int AnalogPin)
 * Class has two methods. measure() returns int value 0-255 and status
 * boolean value if there is enough light compared with minimumlight.
 * Value 255 corresponds to 5 volts voltage at AnalogPin (0 = 0 V).
 *
  *
 * @version 1.0 13.02.2004
 * @author Jaakko Pajunen
 */

public class PowerSource {

  static int ADCvalue,i;
  static ADC voltMeasurer;

  public PowerSource(int pinCMOS, int pinSupply)
  {
      voltMeasurer = new ADC(pinCMOS, pinSupply);
      ADCvalue=voltMeasurer.value();
  }

  public int measure()
  {
      i=0;
      try{
            do{
                  ADCvalue = voltMeasurer.value();
                  i++;
                  if(i==4) break;
              }while(this.ADCvalue==0);
          return this.ADCvalue;
          }
      catch (Exception e){
            System.out.println("Power source malfunction");
            return 10;
      }
  }
}
```

4. Solar panel: Law of Square of Distance and Law of Cosine

The light intensity of a point source should be inversely proportional to the squared distance as eq. (4.1) states. For this experiment two light sources were available: an IKEA table light and UK 4AA Army Model Flashlight with a parabolic reflector. Measurements were done in a dark room by variating the distance between the light source and solar panel. Twenty samples were averaged on each distance.

$$I \propto \frac{1}{r^2} \tag{4.1}$$

By plotting t $I^{½}$ as a function of distance r, the plot should be a straight line if the intensity follow equation (4.1).

The light intensity of a point source at a given distance is directly proportional to the cosine of the angle between the normal of the detector surface and point source.

$$I_{\measuredangle}(\alpha) = I_0 \cos\alpha \tag{4.2}$$

Rotating the light source at a constant distance from 0 to π continuously and then in fixed angles tested this law. At least twenty samples were averaged on each fixed angle. The ladder was done only with the flashlight due to lack of accurate means of measuring the angle and appropriate holding equipment. Since the light source was rotated also by hand only a graphical study is thought to be accurate enough.

5. Results

5.1 Sunlight intensity and room temperature study

From the first figure we can see how the light intensity changes in about 41 hours time. After sunrise around 6 o'clock the luminosity begins to rise and reaches its maximum in an our or so. One can easily see that the second day (11.3.2004) was cloudier than the first but on the afternoon clouds step away around 12-13 o'clock, thus the peak in the intensity.

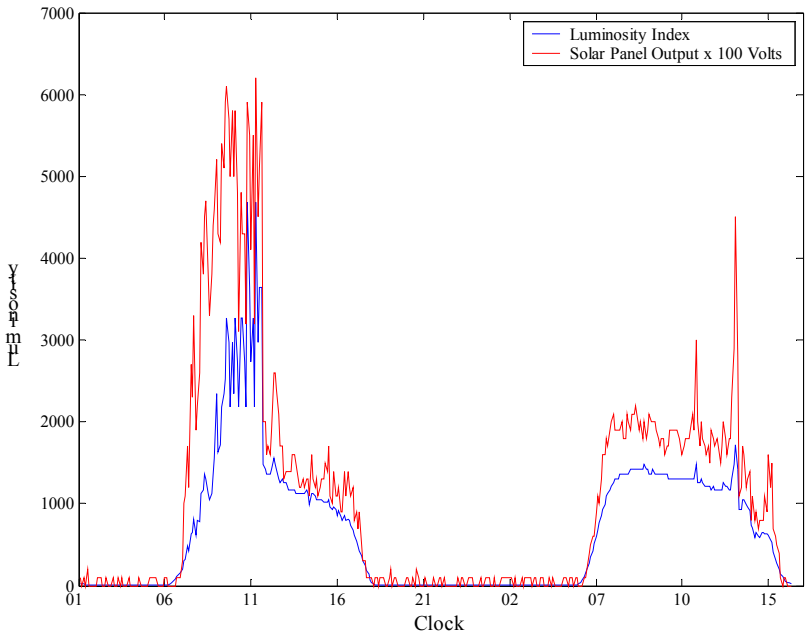


Figure 2. Luminosity changes as a function of daytime.

From the next figures we can see how the regulating system works. As obvious, light regulation is turned on in the nighttime when there is not enough natural light. Temperature regulation is turned on when the limits set in advance are exceeded. However, due to the central air-conditioning, temperature changes are minimal and only once the temperature is too low because of airing the room with fresh air from outdoors (at 01.15 on the first day).

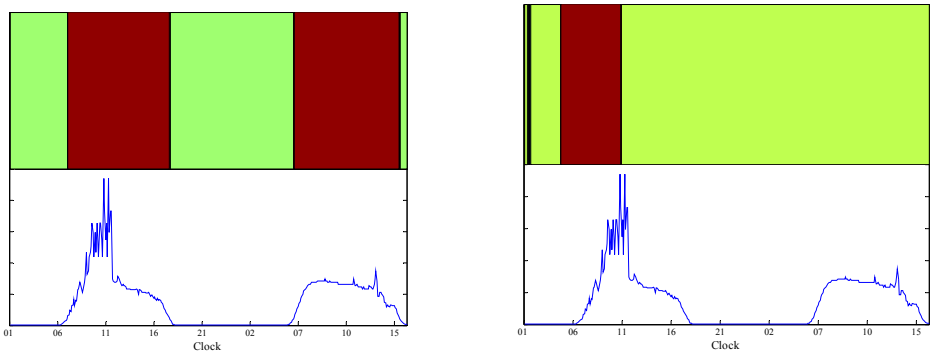


Figure 3. Luminosity and temperature warnings respectively.

The correlation between sunlight and temperature changes was also studied more closely to make an assumption if the sunlight warms the measuring device and its surroundings. For this a correlation efficient was computed in two cases: from all data and from sunset parts of the data. Almost no correlation was found when comparing temperature values to luminosity with all data. The correlation coefficient was 0.15, which stands for almost no correlation. Then again some correlation was found in the sunset data when the correlation coefficient was 0.68.

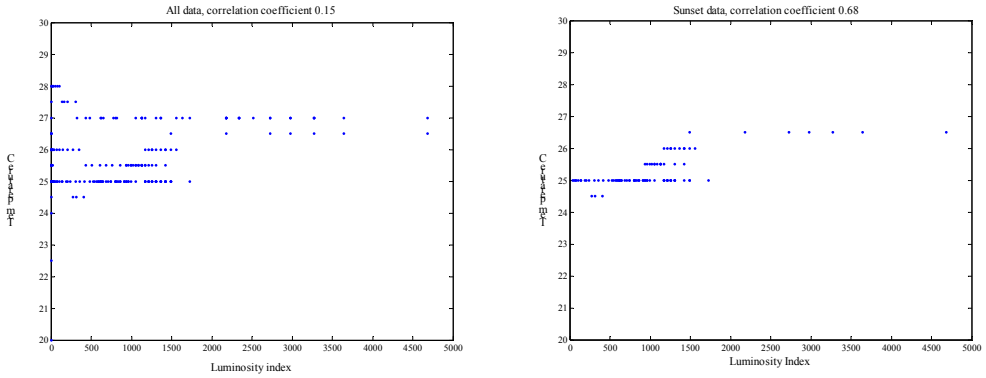


Figure 4a and 4b. Temperature as a function of luminosity with all data (a) and sunset data (b).

This correlation is not very obvious when one looks at the picture nor does correlation coefficient 0.68 stand for strong correlation between values. It stands for some correlation between values so that it is likely that the sunlight affects the temperature although the central air-conditioning. The fresh air and thus lowered temperature is easily seen now in the early part of the temperature graph.

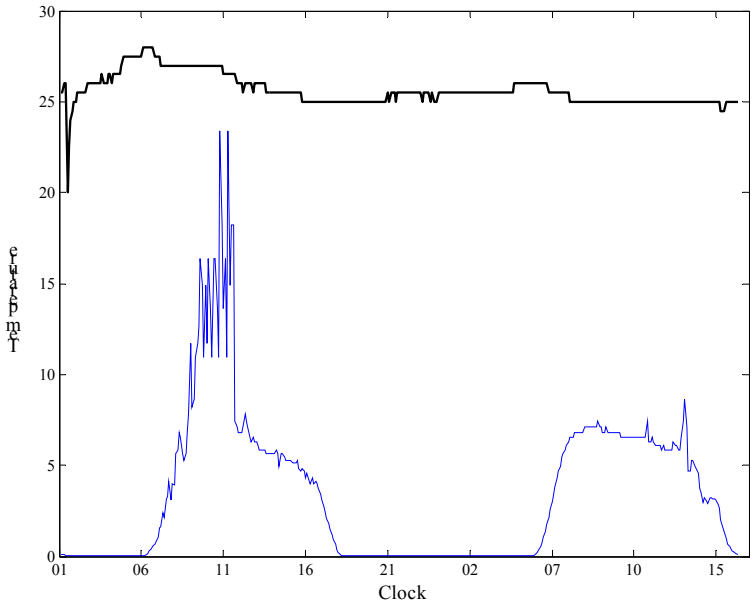


Figure 5. Temperature changes.

5.2 Solar Panel Study

Because of the parabolic reflector the flashlight didn't act like a point source on short measured short distances. Ikea table light works somewhat better as a point source and obeys squared distance law better. The errors of the measurement are not studied here.

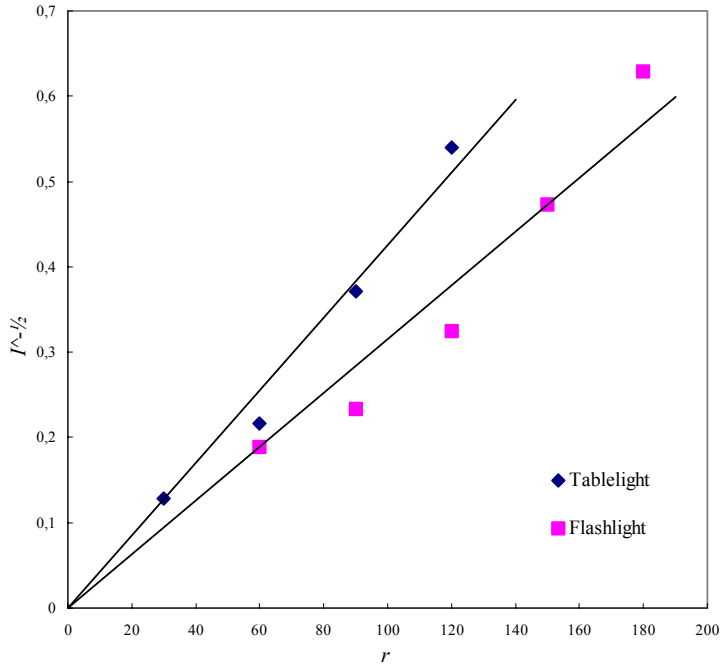


Figure 6. The inverse square root of intensity plotted as a function of distance.

The next figure shows the result of a flashlight sweep of π radians and the moving average of three nearest samples. One can't certainly be assured from this figure alone that the graph is of cosine form, but with the following figure and some imagination it seems to be reasonable.

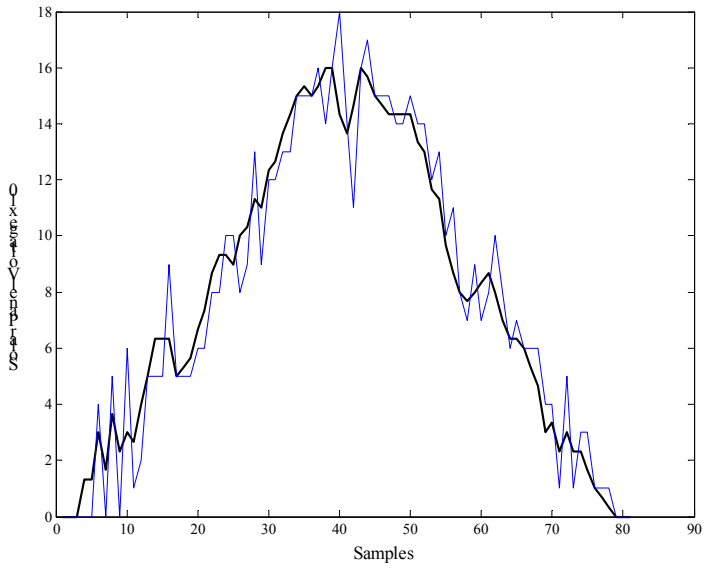


Figure 7. The sweep with UK 4AA flashlight.

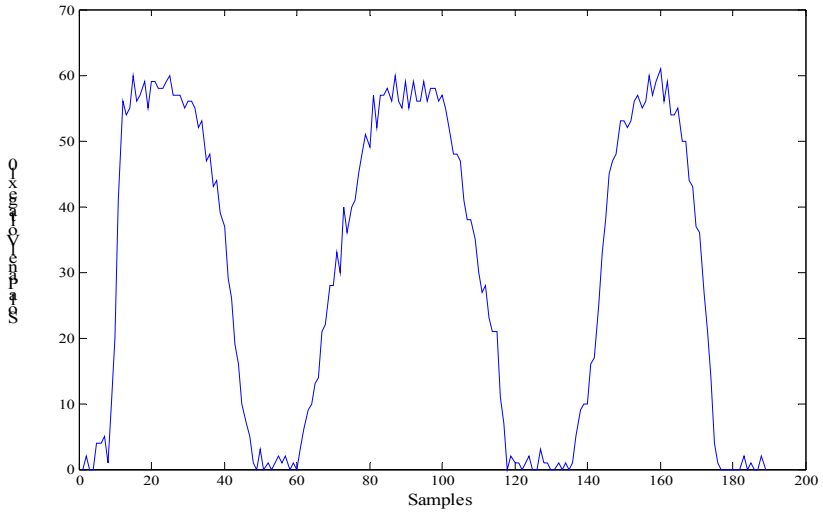


Figure 8. The sweeps with IKEA table lamp.

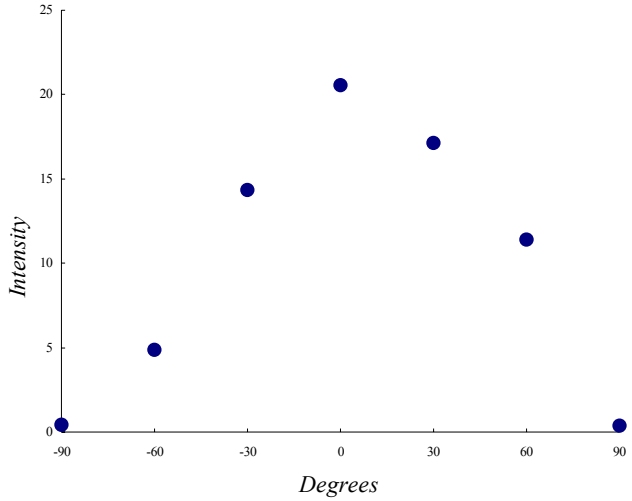


Figure 9. Intensities as a function of angle.

As it is physically relatively easy to grasp that the intensity fades as the point source is turned away from the normal, these figures support the cosine law in that way. But from these pictures alone one should make the conclusion of eq. (4.2).

6. Conclusions

As the final words we can no state that the sunlight affects at least to some extent the temperature of the window-sill although there is an air-conditioning just under the sill. The changes are most obvious when the luminosity gradient is the greatest, that is, during the sunsets and brake of dawns.

We weren't able to convince without doubt that a point source follows the square distance and cosine laws of (electromagnetic) intensities. However the experiments couldn't state any other assumptions either, noticing especially that the point source wasn't even a close to ideal one.

What it comes to Javelin Stamp, one must pay attention to the fact that not many analogical sensors needing other components (like OP-amps or voltage divides) can be fit to the breadboard. The Stamp is optimal when using an external device that can communicate with Javelin through a serial port and connecting this device to those few ones on the breadboard.

7. Reference

Javelin Stamp Users Manual Version 1.0 2002, Parallax Inc. (www.parallax.com)