GEORG-SIMON-OHM
HOCHSCHULE NÜRNBERG

UVIC Universitat de Vic
Escola Politècnica Superior

Final Career Project

Development of a microprocessor controlled energy harvester using a Peltier element

Carolina Moreno Rodriguez

Technical Industrial Engineering
Director: Prof. Dr. Peter Urbanek

Georg-Simon-Ohm-Hochschule. Nürnberg
July 2009

Index of contents:

Resume of the Project 4
1. Introduction 5
2. Objectives 6
3. Project exposure 7
3.1. Harvester 7
3.2. Seebeck 8
3.2.1. Seebeck effect 9
3.3. Thermoelectric coefficients 10
3.4. Peltier element 11
3.5. Hardware development 14
3.5.1. Microprocessor 15
3.5.1.1. Power management and sleep mode 17
3.5.1.2. Timer/Counter 2 21
3.5.1.3. Analog to Digital Converter.....24
3.5.2. Temperature sensor 27
3.5.3. Circuit board 33
3.6. Software development 37
3.6.1. Analog Converter 39
3.6.2. Timer2 39
3.6.3. Sleep mode.....39
4. Results 41
5. Conclusions 49
6. Future guidelines 50
7. Bibliography 51
8. Annex 53

Index of figures:

Figure 3.2.1.1: Seebeck effect 9
Figure 3.4.1: Peltron GmbH sample 11
Figure 3.4.2: Voltage depending on the temp. 13
Figure 3.4.3: Current depending on the temp. 14
Figure 3.5.1.1: Pins configuration Atmega16 16
Figure 3.5.2.1: Temperature sensor 27
Figure 3.5.2.2: Sensor circuit 30
Figure 3.5.3.1: Power supply circuit 33
Figure 3.5.3.2: Voltage regulator circuit 34
Figure 3.5.3.3: Microcontroller I/O 35
Figure 3.6.1: Flow chart of the application 38
Figure 3.6.2: Flow chart of the sleep mode 40
Figure 4.1: Resistor warming system 41
Figure 4.2: Voltage depending on the temp. 42
Figure 4.3: Voltage depending on the temp. results 44
Figure 4.4: Current depending on the temp. results 44
Figure 4.5: Sensor R depending on the temp. results 45
Figure 4.6: Power consumption 47

Universitat de Vic
Escola Politècnica Superior

Resume of the Final Career Project
Technical Industrial Engineering

Title: Development of a microprocessor controlled energy harvester using a Peltier element.

Keyword: Seebeck effect, harvester, temperature sensor, microcontroller Atmega 16

Author: Carolina Moreno Rodriguez

Director: Prof. Dr. Peter Urbanek (Georg-Simon-Ohm-Hochschule. Nürnberg)

Co director: Ramon Reig

Date: September 2009

Resume

Nowadays it is necessary to research other types of energy alternatives and find the way to supply and save the energy we waste.

The aim of the project consist of programming a microprocessor to measure if an oven radiates heat to the exterior, for the measure It is used a Peltier element that generates a voltage depending of the temperature difference between the oven and the air of the place where the oven is situated; The energy generated by the oven will be recollected in a condensor. A sensor will be used to know the exact measure.

The second part of the project the main propose, is the development of a harvester. The microprocessor will use the voltage produced by the Peltier element to supply the electricity that it needs to work. A low power circuit and the appropriate software are needed to save the voltage generated.

1. Introduction

It is of importance to look for alternative energy sources, for that reason I had studied Peltier elements, specifically the opposite consequence is known as Seebeck effect. These elements have a property which contact with different temperatures in each side of the element creates a voltage proportional to the temperature difference.

This project deals with how with a Peltier element the temperature of an oven can be controlled as well as to create a harvester system that with the energy generated by the element makes a microcontroller work and control the process of determinate the temperature. The project is divided in two different parts, the Peltier system that allow measuring the temperature and the second part the design of the software and hardware for the harvester.

The harvester is built to obtain an energy efficient system and an independent system without needing electrical energy of the network. The part of controlling all the process is done by the microcontroller Atmega16, the software and hardware is created especially for saving power and the components consume little power.

2. Objectives

The purpose of the project is designing a system to control a Peltier element that generates a voltage depending on the temperature of an oven, it will work between 0ºC and 200ºC. With a sensor the temperature of the oven can be measured and a microprocessor in power save mode will control the process. All the system will use the energy generated by itself.

The project requires different tasks:

- Research of the element Peltier, looking for a company manufacturing components with the characteristics required (0 – 200ºC).
- Test the Peltier element to know the real characteristics and the limits of work.
- Programming a microcontroller to control the measure of temperature.
- Design of the circuit to the part of storage energy system.
- Suit the software for the final application.

3. Project exposure

3.1 Harvester

The power consumption of electronic circuits must consume low power around microwatts; it becomes possible to power these devices by using ambient energy harvested from the environment. The heat can be one of the possible sources of energy. Everything in this world creates heat, natural phenomena such as lava volcano or the heat created by a motor, etc.

An energy harvester transforms energy of the element around and transforms it in electrical, magnetic or piezoelectric energy. These three are the main strategies for energy transduction.

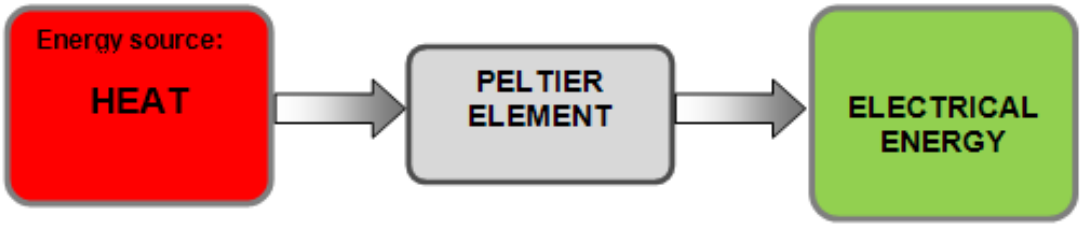
- Electric harvesting uses either a time-varying capacitor or moving permanent electrets.
- Magnetic harvesting can be further categorized into systems with a time-varying inductor and systems that employ moving permanent magnets.
- Piezoelectric materials, such as quartz and barium titanate, contain permanently polarized structures that produce an electric field when the materials deform due to imposed mechanical forces.

In that project the concept of harvester is different; it will convert the heat of an element into electrical energy. That will be possible with a Peltier element that transforms heat energy into electrical energy .The heat is provided by an oven with temperatures from 0ºC to 200ºC. The electrical energy transformed will be used to powering a microcontroller and the temperature control circuit. The name of the phenomenon caused by the Peltier element is the Seebeck effect.

It will be explained in the following point.

The voltage associated with the electrical current increases when the Peltier element increases the temperature, the electrical current will be stored for powering the microprocessor.
The power generation capability of the designated energy harvester will be calculated in that project.

Energy harvesting process:



3.2. Seebeck

Seebeck observed already 1822; that a closed circle of two different metallic conductors, which are connected in an electrical circuit, generate a magnetic field when between the two contact points exist a temperature difference. The magnetic field has the effect of current flow through the head. The voltage is a measure of the temperature difference that the junctions are exposed to. This principle is still used today for temperature measurement using thermocouples.

A thermocouple is a converter device; it transforms heat energy into electrical energy. The thermocouples are used as temperature sensors. The most advantageous choice of type of thermocouple is based on the temperature of application, the service life, accuracy and cost required.

3.2.1 Seebeck effect

The voltage difference “dV” produced across the terminals of an open circuit made up of a pair of dissimilar metals, A and B, whose two junctions are held at different temperatures, is directly proportional to the difference of the hot and cold junction temperatures “Kh – Kc”, and does not depend in any way on the distribution of temperature along the metals between the junctions. The factor of proportionality, SAB, is called the relative Seebeck coefficient, thermoelectric power, or just thermopower, of the bi-metallic couple, and in general this coefficient also varies with the level of the temperature at which the temperature difference occurs. If the circuit is closed, a current will flow in the metals, which can be detected by the magnetic field produced around the wires, or by the joule heating produced by the resistance in the wires, or closing the circuit with a capacitor or condensor of sufficient capacity to accumulate a measurable charge for the transient current which will flow in this case, or by a galvanometer or ammeter placed in the circuit to measure the current, or by measuring the amount of chemical substance deposited at the positive and/or negative electrodes in an electrochemical cell.

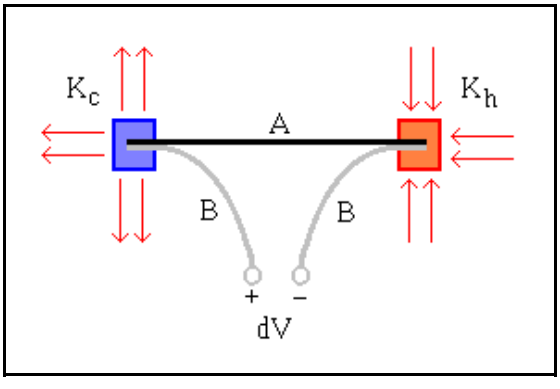


Figure 3.2.1.1: Seebeck effect

3.3 Thermoelectric Coefficients

Seebeck coefficient is α_{ab}, under open circuit conditions is defined as:

$$\alpha_{ab} = \frac{dV}{dT}$$

If Kh is hotter than Kc, the thermocouple generates current with positive α .

By contrast, if the thermocouple liberates heat at the Kh part the thermocouple has a negative Π . The rate of heat exchange at the junctions is:

$$Q = \Pi_{ab} \cdot I$$

If current is flowing and there is a temperature gradient, there is also heat generation or absorption within each segment of the thermocouple because α is temperature dependent.

The following equation provides a link between thermoelectric cooling (π) and thermoelectric power generation (α).

A good thermoelectric material must combine a large α with low electrical resistivity ρ and low thermal conductivity. Further, because Joule heating is proportional to the square of the electric current and the Peltier effect is only linear in current, one cannot increase the temperature gradient indefinitely simply by increasing the current.

3.4 Peltier element

The research of Peltier element companies finished when *PELTRON GmbH* sent me a sample of the Peltier element model: 128 A 0020H 150.

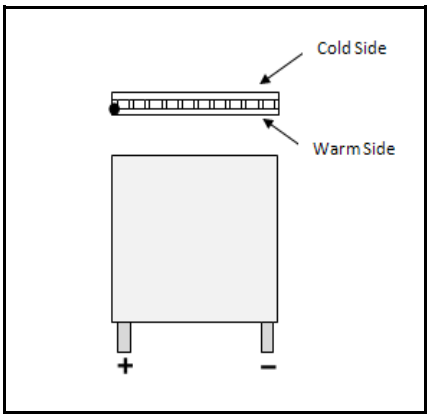


Figure 3.4.1: Peltron GmbH sample

Characteristics of the Peltier element form 128 A 0020H 150 Peltron:

.

Dimensions	40x40x4 mm
Maximum refrigerating capacity	31 W
Maximum difference temperature	66 K
Maximum voltage	15 V
Maximum current	3,5 A
Deviation	15%
Maximum temperature	160 ºC
Internal resistance	3,6 Ω
Seebeck coefficient	49 mV/K
Thermal conductivity	250 mW/K

Test with the Peltier test:

The measure of the properties of the element was done with a large block which guarantees a constant of 30ºC in the cold side of the element. The other side is warmed from 0 to 90ºC.

The result of testing the Peltier element by measuring the voltage generated and the current is shown in the following tables:

Cold side	Warm side	Difference	Voltage	alfa=dV/dT
33	90	57	1,70	0,03
33	74	41	1,40	0,03
33	68	35	1,24	0,04
33	65	32	1,13	0,04
33	60	27	0,95	0,04
33	55	22	0,80	0,04
33	50	17	0,65	0,04
33	48	15	0,56	0,04
32	45	13	0,45	0,03
32	40	8	0,28	0,04
32	38	6	0,23	0,04

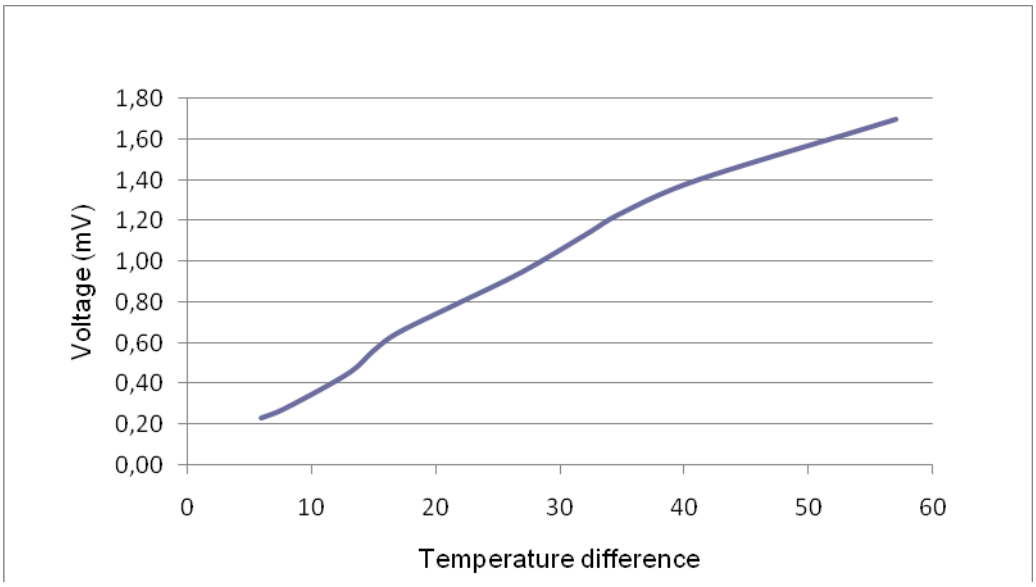


Figure 3.4.2: Voltage depending on the temperature

***Voltage Results**: a difference in one degree up is the result of 30mV more. With 60 degrees of difference the voltage generated is almost 2V.*

Cold side	Warm side	Difference	Current (mA)
34	80	46	300
34	75	41	272
34	70	36	256
34	65	31	221
34	60	26	187
34	55	21	152
34	50	16	123
34	46	12	98
34	44	10	82
34	42	8	67
33	40	7	53
33	38	5	40
33	36	3	28
33	35	2	21

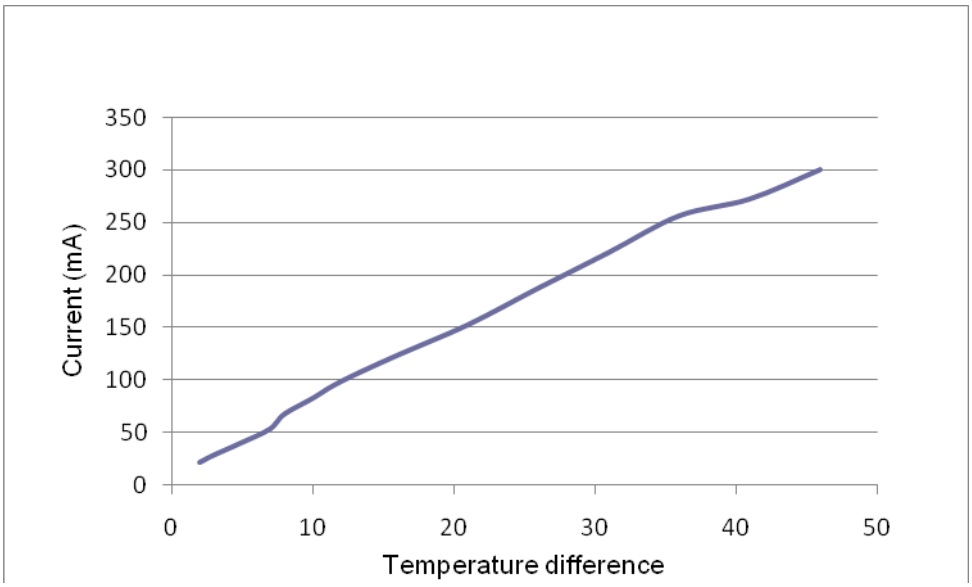


Figure 3.4.3: Current depending on the temperature

***Current Results**: a difference in one degree up is the result of 6,35mA more. With 50 degrees of difference the current generated is 300mA.*

3.5 Hardware development

The project needs a particular design of circuit thinking especially about energy saving.

The Peltier element works for a while and then the system has to work with the energy storaged. The microprocessor, the voltage converter and the rest of the circuit will work with the energy that is storaged in the condensor.

The capacity of the condensor depends of what we need, if the system has to works for long a period of time it will be bigger but I will be charge slowly.

Other important point is the election of the sensor, it has to support high temperatures and it must consume little power and it will also need a circuit annex to make a constant current.

The system will work between 0ºC and 200ºC, and it needs to be functioning for long times. The hardware is development following the patterns above.

3.5.1 Microprocessor

The microprocessor used in this project is the Atmega16.The following table shows a resume of the technical characteristics:

Flash (bytes)	16
EEPROM (bytes)	512
SRAM (bytes)	1024
Max I/O Pins	32
F.max (MHz)	16
Vcc (V)	2.7-5.5
16-bit Timers	1
PWM (channels)	4
RTC	Yes
Master/Slave SPI	1
USART	1
TWI	Yes
ISP	Yes
10-but A/D (channels)	8
Analog Comparator	Yes
Watchdog	Yes
On Chip Oscillator	Yes
Hardware Multiplier	Yes
Interrupts	21
Ext Interrupts	3
Self Program Memory	Yes
Packages	PDIP 40 TQFP 44 MLF 44
JTAG Interface	Yes

Pins configuration:

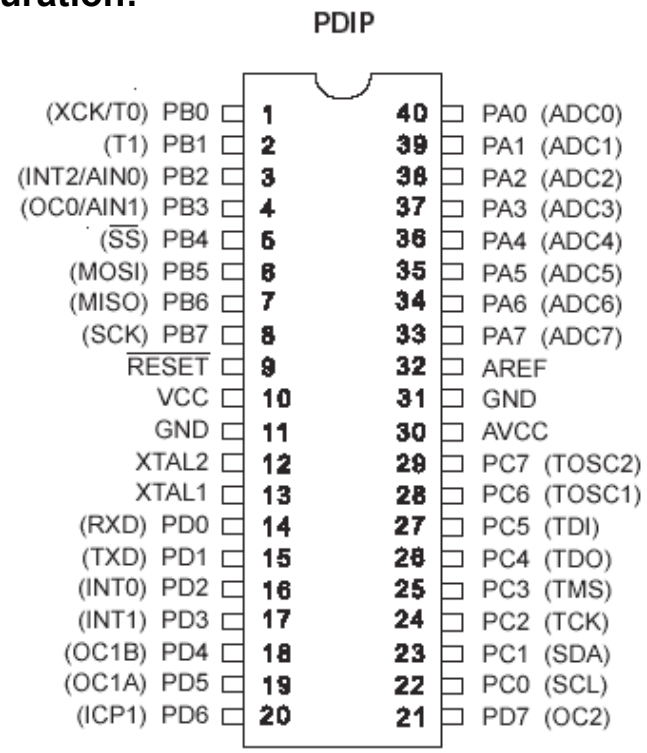


Figure 3.5.1.1: Pins configuration Atmega16

The pins and ports connected in the microprocessor are:

- **Port A:**

  PA[0] - in - ADC0 (measure 1, Sensor voltage)

  PA[0] - in - ADC1 (measure 2, Vsupply)

  PA[4] - out - Red led 2

  PA[6] - out - Green led1

- **Port B:**
  Not used in that program.

- **Port C :**

  PC[2] - in - TMS
  PC[3] - in - TM0

PC[4] - in - TDI
PC[6] - in - OSC1
PC[7] - in - OSC2

- **Port D :**
  Not used in that program.

PIN 9 : Reset
PIN 10: VCC
PIN 11: GND
PIN 30: AVCC
PIN 31: AGND
PIN 32: AREF

3.5.1.1 Power management and sleep mode

The microprocessor needs a configuration of the sleep mode to work with the less energy is possible. That part of the hardware will be used to set the program in C language
Sleep mode enable the application to shut down unused modules in the MCU, thereby saving power. There are various sleep modes depending on the application´s requirements. To enter any of the six sleep modes that allow the microprocessor Atmega16 there are the following steps:

- The SE bit in MCUCR must be written to logic one.
- The SLEEP instruction must be executed.
- The SM2, SM1, SM0 bits in the MCUCR select which sleep mode will be activated.

The MCU control register contains the bit for power management:

7	6	5	4	3	2	1	0
SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00

SE: Sleep Enable

The SE must be written to one, that makes the MCU sleep during the sleep mode execution, it must clear it just before finishig the execution.

Sleep mode Select

The six bits can select between the six available sleep modes. This is shown in the following table:

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power down
0	1	1	Power save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby
1	1	1	Extended Standby

- **Idle mode**

When the SM2..0 bits are written to 000, the Sleep instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, USART, Analog Comparator, ADC, Two wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to continue operating (if enabled). This sleep mode halts clk CPU and clk FLASH, while allowing the other clocks to run.

- **ACD Noise Reduction mode**

When the SM2..0 bits are written to 001, the Sleep instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the External Interrupts, the Two-wire Serial Interface address watch, Timer/Counter2 and the Watchdog to continue operating (if enabled). This sleep mode halts clk I/O,clk CPU and clk FLASH, while allowing the other clocks to run.

- **Power down mode**

When the SM2..0 bits are written to 010, the Sleep instruction makes the MCU enter Power-down mode. In this mode the External Oscillator is stopped, while the External Interrupts, the Two-wire Serial Interface address watch, Timer/Counter2 and the Watchdog to continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface address match interrupt, an External level interrupt on INT0 or INT1, or an External interrupt on INT2 can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

- **Power-save mode**

When the SM2..0 bits are written to 011, the Sleep instruction makes the MCU enter Power-save mode. This mode is identical to power-down, with one exception:
If Timer/Counter2 is clocked asynchronously, i.e., the AS2 bit in ASSR is set, Timer/Counter2 will run during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK, and the Global Interrupt Enable bit in SREG is set.

- **Standby Mode**

When the SM2..0 bits are written to 110, and an external crystal/resonator clock option is selected the Sleep instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.
For that application of the project is needed to select the bits 7, 5, 4 with the power save mode.

The sleep mode used in the program is the ***Power save mode*:**
It is needed to count the time of the microprocessor while sleeping and when it will show the measurement of the temperature. The power save mode has the option of setting the Timer/Counter 2 while the CPU is sleeping, it continues working, the rest is in sleep mode, so the only part that consumes energy is the Timer/Counter2.

- Active Clock domains and wake up source in the Power save mode:

Active Clock domains					Oscillators	
clk CPU	clk FLASH	clk IO	clk ADC	clk ASY	Main Clock Source Enabled	Timer Osc. Enabled
				x (I)		x (I)

Wake-up Sources					
INT2 INT1 INT0	TWI Address Match	Timer 2	SPM/EEPROM Ready	ADC	Other I/O
x (II)	x	x (I)			

(I) If AS2 bit in ASSR is set.

(II) Only INT2 or level interrupts INT1 and INT0.

3.5.1.2 Timer/Counter 2

The timer is necessary to count the time the microprocessor is in the power save mode and also to enable the interruption that wakes up the microprocessor.

To set the Timer2 is required to designate the next registers:

1. Asynchronous Status Register - ASSR
2. Timer/Counter Register - TCNT2
3. Output Compare Register - OCR2
4. Timer/Counter 2 Control Register - TCCR2
5. TIMSK

1. Asynchronous Status Register – ASSR

7	6	5	4	3	2	1	0
-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB

The program must have the AS2 (bit 3) written to one to make the timer enable, the Timer/Counter2 is clock from a Crystal Oscillator connected to the Timer Oscillator 1 (TOSC1) pin.

2. Timer/Counter Register – TCNT2

That register gives the start value counter, in the program it will be zero. The Timer/counter (TCNT2) and Output Compare Register (OCR2) are 8 bit registers.

7	6	5	4	3	2	1	0
TCNT2(7:0)							

3. Output Compare Register – OCR2

The Control Register contains an 8-bit value that is compared continually with the counter value of TCN2. A match will be use to generate an output compare interrupt.

7	6	5	4	3	2	1	0
OCR2(7:0)							

4. Timer/Counter 2 Control Register - TCCR2

7	6	5	4	3	2	1	0
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

In the program only three bits will be selected CS22, CS21 and CS20, it sets the timer used and the frequency of the counting.

CS22	CS21	CS20	Description
0	0	0	Timer disable
0	0	1	T2S-Takt/1
0	1	0	I/O-Takt/8
0	1	1	I/O-Takt/32
1	0	0	I/O-Takt/64
1	0	1	I/O-Takt/128
1	1	0	I/O-Takt/256
1	1	1	I/O-Takt/1024

I will use the T2S-Takt/1, so I only is needed to set the CS20 bit.

5. TIMSK

When the AS2 bit in ASSR is set, Timer/Counter2 will run during sleep. The device can wake up from the Timer Over-flow or Output Compare event from Timer/Counter 2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK, and the Global Interrupt Enable bit in SREG is set.

7	6	5	4	3	2	1	0
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OEIE0	TOIE0

Calculation of the Timer overflow:

The microprocessor works at 32768Hz, therefore:

1 Timer overflow: $\frac{256}{32768} = 0{,}0078125$ seconds

The waiting time will be 10s, therefore:

$\frac{10s}{0{,}0078125s} = 1280$ Timer interrupts (that result is used in the main program function).

3.5.1.3 Analog to Digital Converter

The application for the program needs the ADC. The measures taken in the input are analog and to be processed must be in digital format.
The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents zero and the maximum value represents the voltage on the AREF pin minus 1 LSB. On this occasion the maximum value is the internal reference 2.56V, the AREF pin is written to the REFSn bits in the ADMUX register.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. The ADC does not consume power when ADEN is cleared, so it will be switch off the ADC before entering the power save mode.
A conversion is started by writing a logic one to the ADC start conversion bit, ADSC. I need to select to inputs that will be set in the ADMUX register.

The result of the conversion is:

$$ADC = \frac{Vin \cdot 1024}{VREF}$$

To set the ADC is necessary to designate the next registers:

1. ADC Multiplexer Selection Register – ADMUX
2. ADC Control Status Register A – ADCSRA

1. ADC Multiplexer Selection Register – ADMUX

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

The bits 7:6 (REF1:0) are the reference selection bits. These bits select the voltage reference for the ADC. The other pins are not used for the application.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56 Voltage Reference with external capacitor at AREF pin

I will use the Internal 2,56V Voltage therefore the REFS1 and the REFS0 will be set.

2. ADC Control Status Register A – ADCSRA

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

The bits used in the application are:

- **ADEN**

  Writing this bit to one enables the DC. By writing it to zero, the ADC is turned off.
- **ADIF**

  This bit is set when an ADC conversion completes and the Data Registers are updates. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set; ADIF is cleared by hardware when executing the corresponding interrupt handling vector.
- **ADIE**

  When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

3.5.2 Temperature sensor

The measuring of the temperature is done by platinum -chip - temperature sensor, model PCA 1.2005 10 from JUMO:

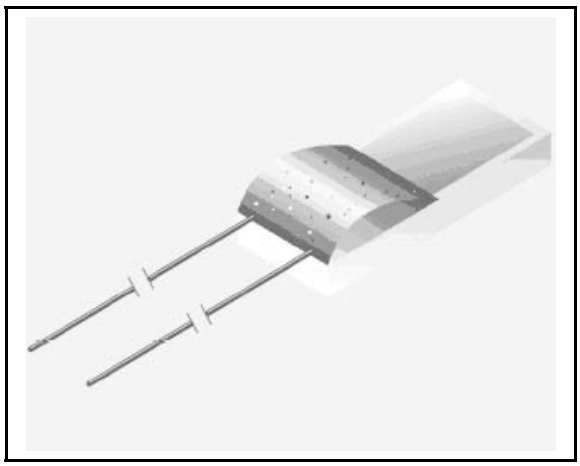


Figure: 3.5.2.1. Temperature sensor

Technical data:

- Range of measurement: -70ºC…+250ºC
- Resistance: 1KΩ
- Tolerance class: B
- Size: 2,0 x 5,0 x 1,3(0,64)mm
- Range of resistance: 20…5KΩ
- Linear working
- $\alpha = 3{,}850 \cdot 10^{-3} ºC^{-1}$

The working of the sensor is lineal, it needs a current maximum 2mA and the optimal current is 0,1mA.

The sensor needs a special circuit to have the suitable current; it will need a dimension operation of the circuit. Before the sizing of the circuit; it's needed to test the resistance values with the temperature, this possible with the following formula:

$R(T2)= \alpha (T2-T)* R(T1) + R(T1)$

$\alpha = 3,85 E-03$

$T1= 25ºc$

Results of the resistance working with the temperature:

T2 (°C)	R (T2)
25	1090,00
30	1110,98
35	1131,97
40	1152,95
45	1173,93
50	1194,91
55	1215,90
60	1236,88
65	1257,86
70	1278,84
75	1299,83
80	1320,81
85	1341,79
90	1362,77

T2 (°C)	R (T2)
95	1383,76
100	1404,74
105	1425,72
110	1446,70
115	1467,69
120	1488,67
125	1509,65
130	1530,63
135	1551,62
140	1572,60
145	1593,58
150	1614,57
155	1635,55
160	1656,53

Calculation for the sensor circuit:

1. Voltage divider

The second measurement is done in the voltage divider, that result will be useful to know the resistance of the sensor therefore the temperature. The voltage divider is necessary to obtain a voltage less than the Vcc, because the ADC can't work with voltage higher than 2,56V. I use two resistors one of 620KΩ and the second 1200KΩ, that divider will result in the Vcc divider in three.

2. Constant current source with fixed Vcc

It will be use a Mosfet with a diode device known as BSS131 that component will provides the sensor with the current in the R5.

$$R5 = \frac{VGS}{Id} \approx \frac{1{,}8V}{0{,}1mA} = 18K\Omega$$

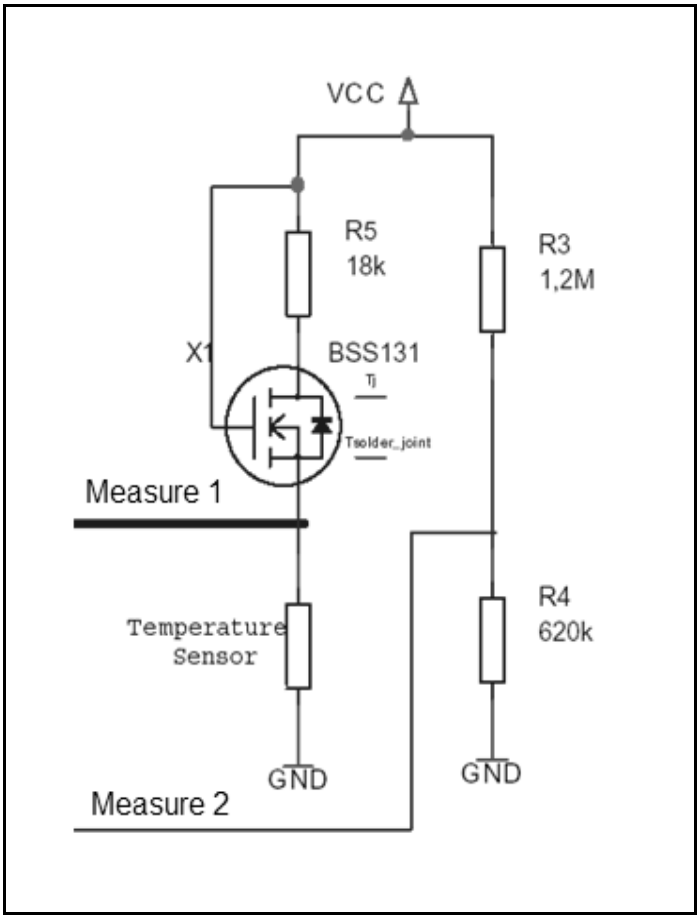


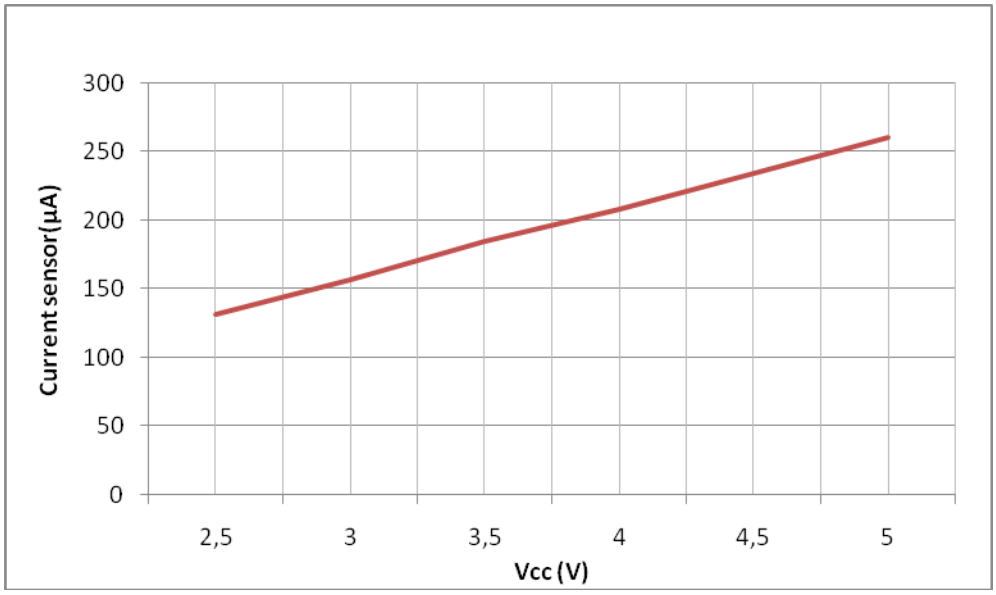
Figure 3.5.2.2: Sensor Circuit

To be sure the measurement of the Temperature is correct the temperature sensor must have a constant current, when the current in the Temperature sensor and the Voltage are known, is easy to calculate the resistance. With the value of resistance is possible to know the Temperature.

In the circuit before there are two points of measurement, the first is the voltage in the sensor. There is also a BSS131 device it provides a constant current when the Input Voltage is constant. The problem is that the system doesn't have a constant voltage; it depends on the Peltier element and the temperature of the oven, that means the BSS131 won't have a constant current. With a voltage divider connected to the input voltage the problem is fixed.
The second measurement is a third part of the input voltage, it has a range from 0 to 5 V.

By testing the circuit It is known that the current increase lineally when the voltage increase.

Vcc	Current sensor(μA)
2,5	132
3	157,3
3,5	185
4	208,7
4,5	234,5
5	260,42



With the table over, a line is necessary o calculating the gradient:

Gradient =Voltage / Current = 0,019

The gradient doesn't change if the variables change because is a proportional constant.

Calculation of the sensor resistance:

$$Rsensor = \frac{Measure1}{Measure2 * Gradient * \left(\frac{R3 + R4}{R4} \right)}$$

3.5.3 Circuit board

The circuit includes the microprocessor, the sensor circuit, the Outputs (leds), the power supply, the JTAG connection and the voltage regulator.
The last one is the work of another student and it's assigned to this project.

Power supply:

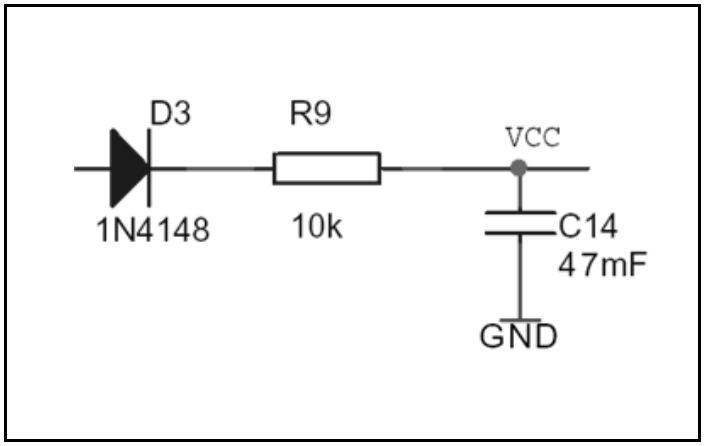


Figure 3.5.3.1: Power supply circuit

The capacitor value can be changed depending on how much time we need the system working. This capacitor will be charged fast, useful to test the system. The diode prevents the return of voltage when the Peltier element is not working but the capacitor is charged. The resistor protects the life of the capacitor.

Voltage regulator:

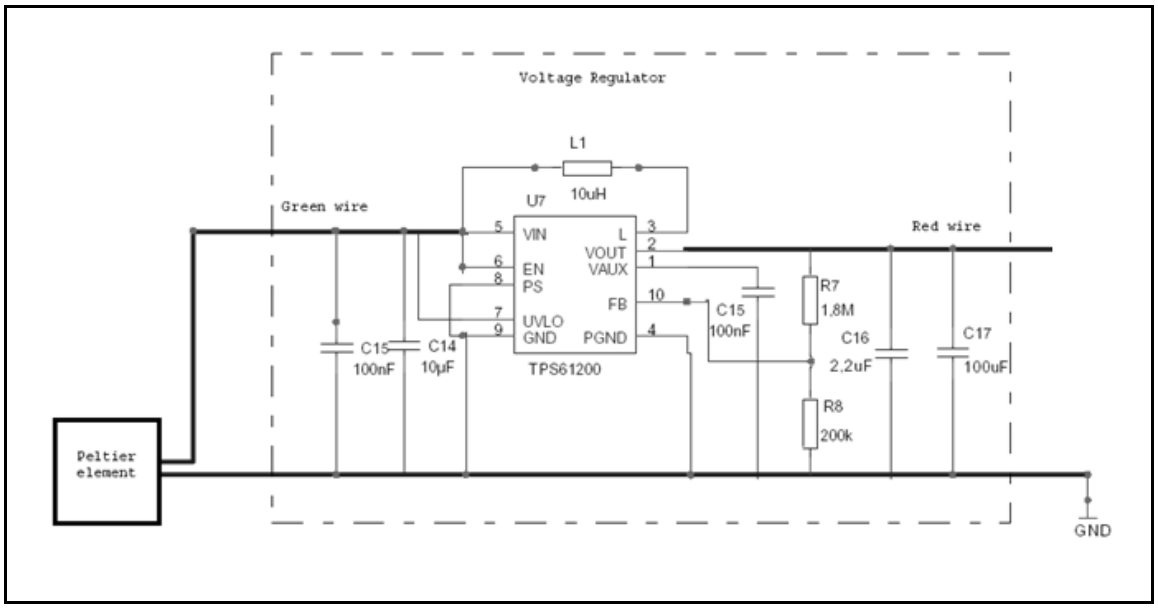


Figure 3.5.3.2: Voltage regulator circuit

I wrote before that part is not my work; it will transform the voltage generated by the Peltier element in a regular voltage to get into the microprocessor.

The reason to use that Voltage regulator is because it has a step up with 0,3mV, it means that when the Peltier element has just started running, only with 0,3mV the voltage regulator makes up the voltage to prepare it for powering the microprocessor. It is not needed too much time to start running the microprocessor.

Microprocessor Atmega16, JTAG connection and the Leds circuit:

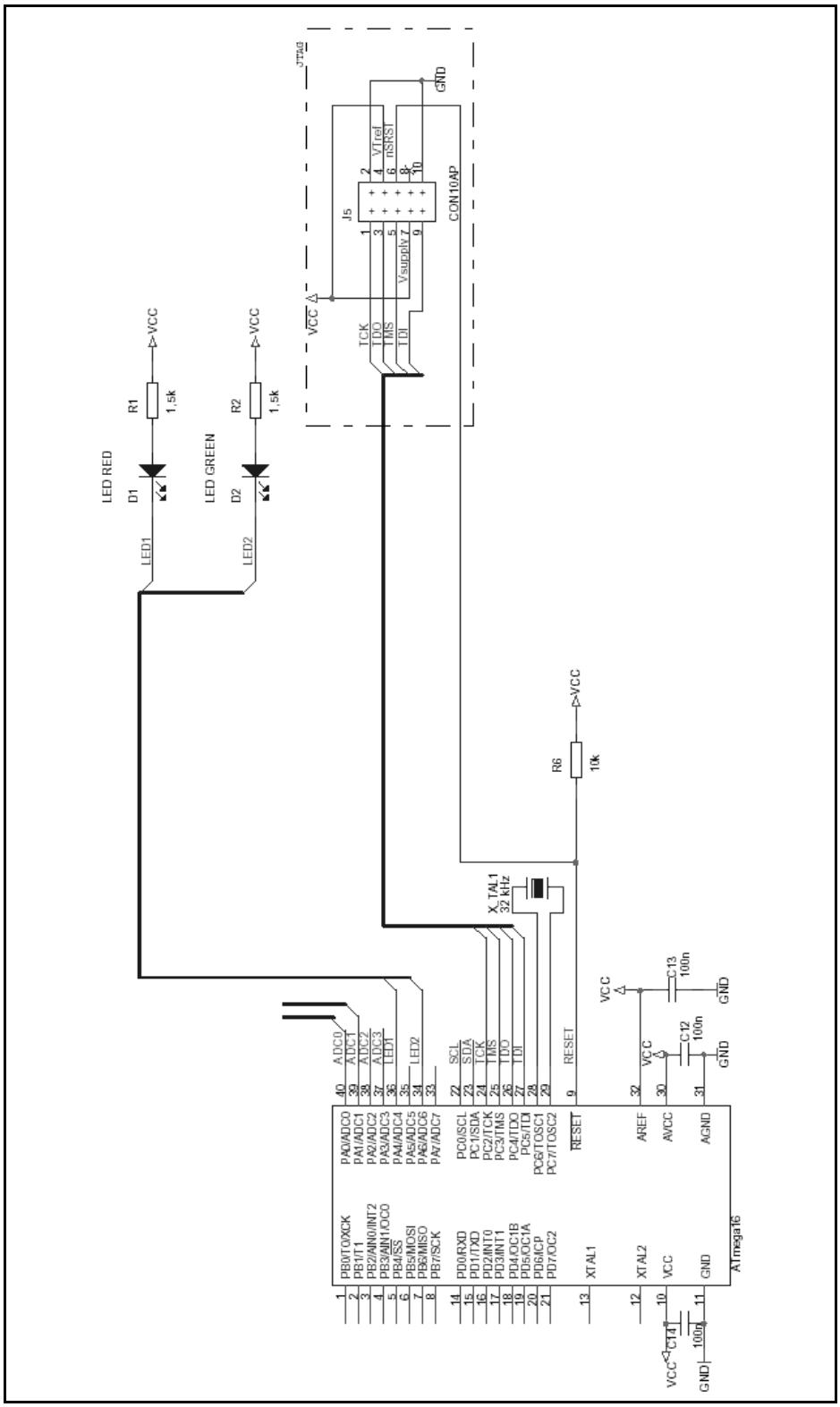
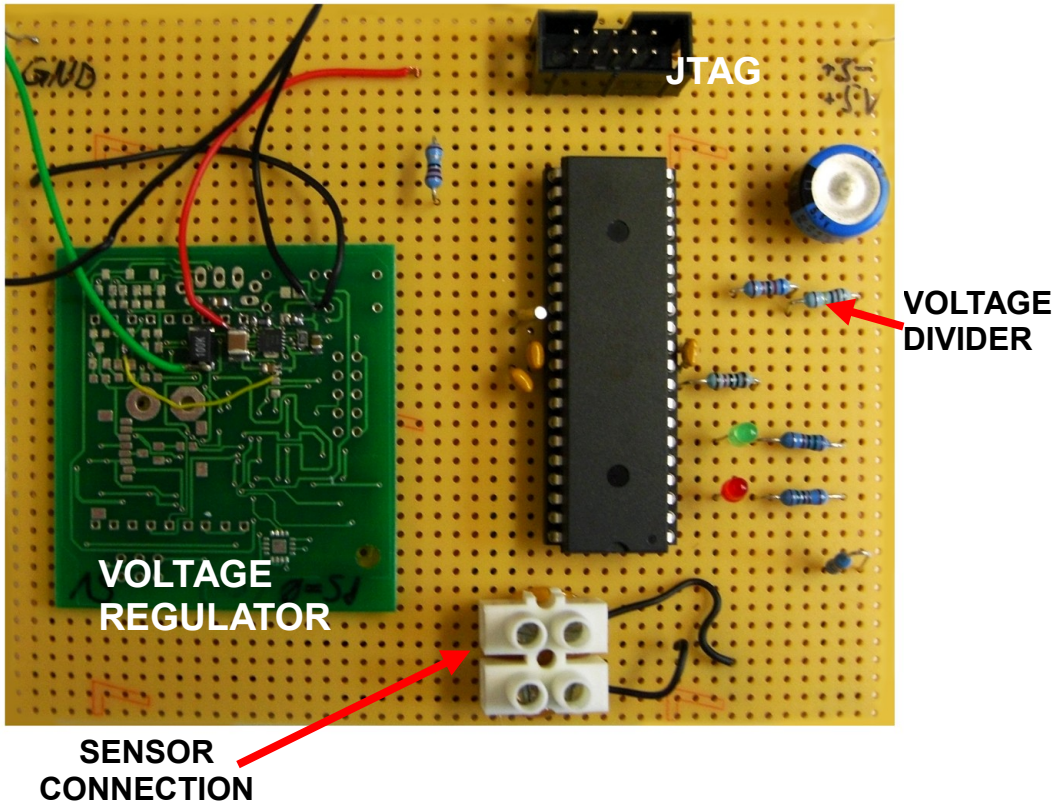


Figure 3.5.3.3: Microcontroller I/O circuit

Real Circuit board:



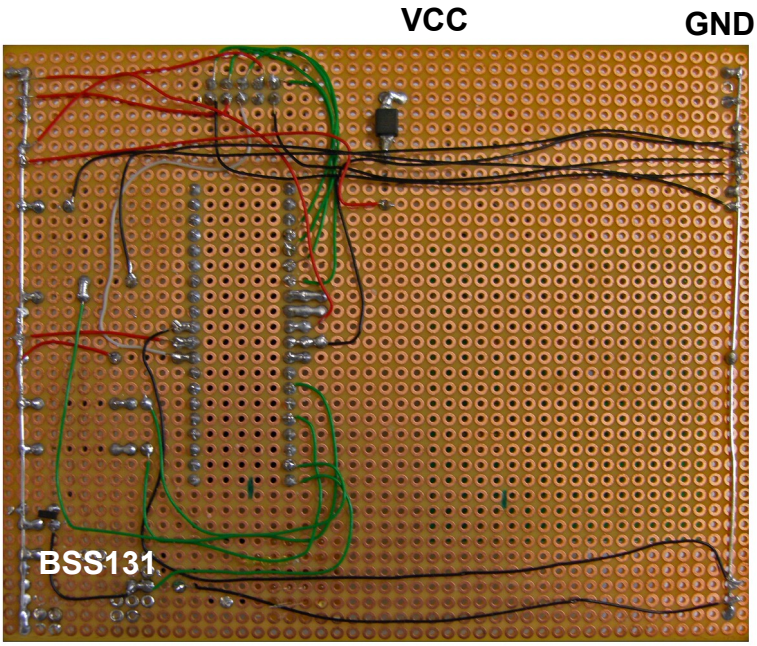


Figure 3.5.3.4: Board ahead and behind

3.6 Software development

The application consists on several functions developed in C language.

- Main program
- Adc converter
- Adc initialization
- Port initialization
- Sleep mode initialization
- Timer 2
- Timer 2 initialization

The program works with the two measures of the microcontroller, one is the voltage in the sensor, and the other is the voltage in the voltage divider.
First it converts the data with the ADC converter, that information is used to know the resistance of the sensor so that we know the temperature of the Peltier element.
The program uses 1300 as a constant to define 50 degrees. More than 50 degrees the Led will be red, and less than 50 degrees the Led will be green. After the measure and the conversion the microprocessor goes to Sleep Mode during 10s that will save enough energy to work.

Flow diagram of the measurement:

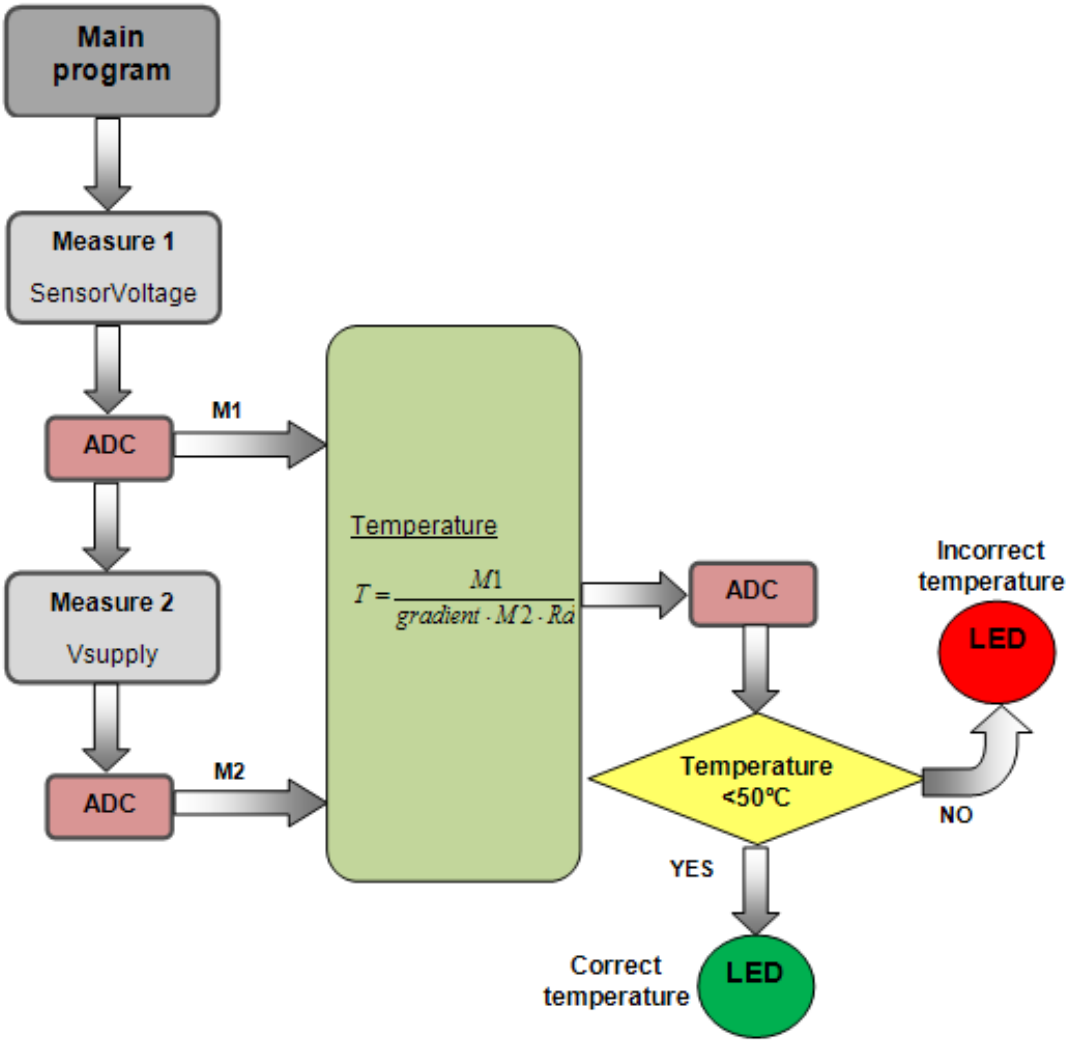


Figure 3.6.1: Flow chart of the application

3.6.1 Analog Digital Converter

- This is the **ISR** for the Analog Digital Converter. When an interrupt occurs, conversion is ready, the global variable is set.
- **Adc initialization:** Enable ADC chose the one shot mode and interrupt when finished. It use ADC0 and ADC1.

3.6.2 Timer 2

- The Timer/Counter2 uses a subroutine ISR to count the time.
- In the Initialization, it enables the timer and starts the counter2 with I/O Clock/1

Enable timer 2:	*ASSR = (1<<AS2);*
Start value counter:	*TCNT2= 0;*
Counter 2 I/O Clock 1:	*TCCR2|= (1<<CS20);*
Enable interrupt:	*TIMSK |= (1<<TOIE2);*

3.6.3 Sleep mode

Initialization of the sleep mode, in that case the Power-save mode.

Power save mode selection:	*MCUCR = (1<<SM1)|(1<<SM0);*
Turn on sleep mode:	*MCUCR|= (1<<SE);*

Flow diagram of the sleep mode:

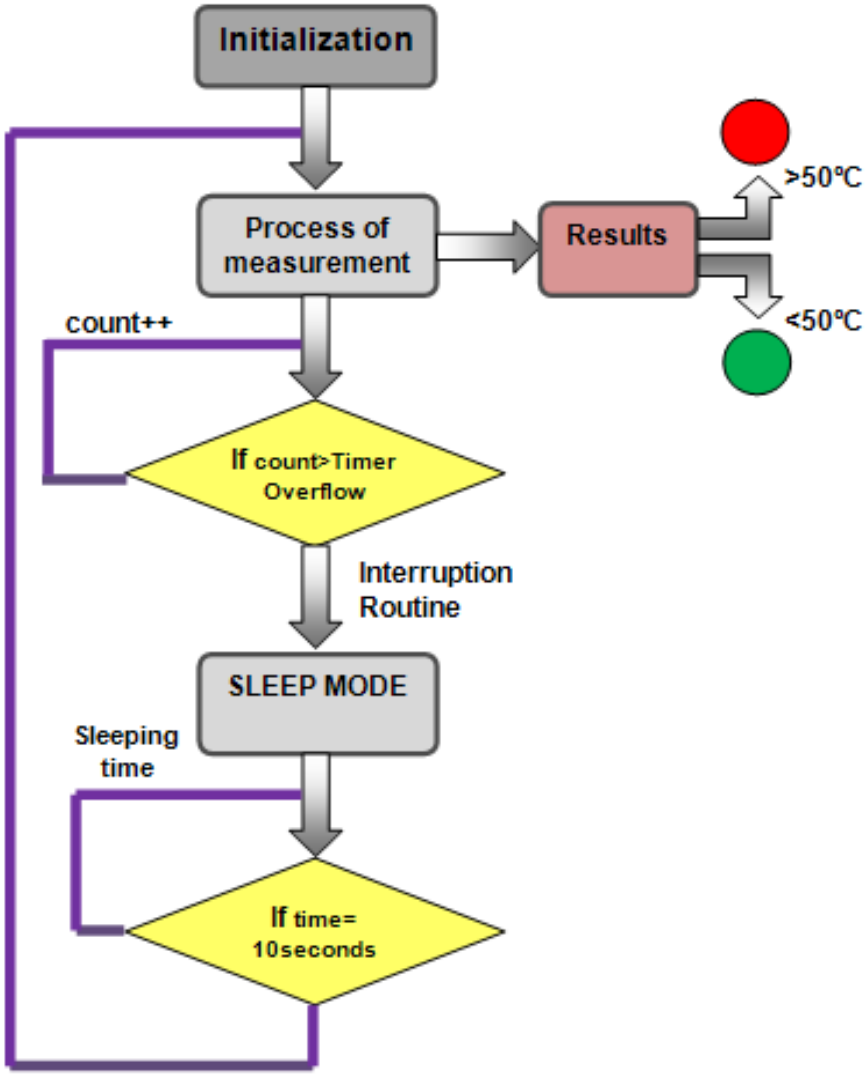


Figure 3.6.2: Flow chart of the sleep mode

4. Results

The last test with the device is the measurement with more than 60ºC. It is difficult to find a system to simulate an oven, for testing the Peltier element I used a resistor system that warmed till 60ºC.

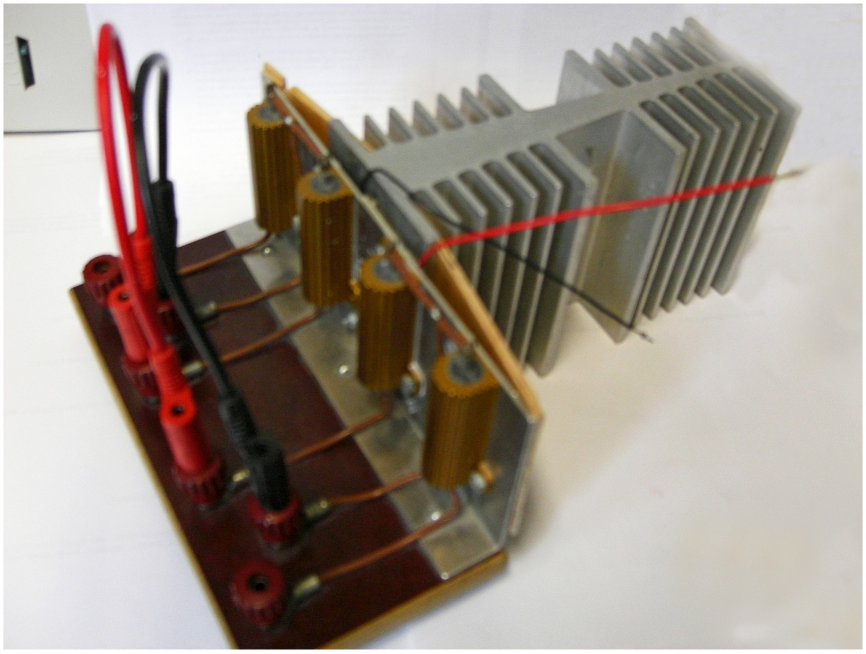


Figure 4.1: Resistor warming system

The resistors warmed the Peltier element, on the other side there is the cooling element that kept the cold side of the Peltier element in 25ºC.
All the graphics and tables in the project are done with that system, the inconvenient is the maximum of temperature is 60ºC, anyway is easy to predict the line of work because the device is lineal.

Although the test with the resistor system is enough to predict the potential of the element; I analyze the system with other warm generator, a fuse.

A fuse warmed a board where the sensor was in; the system was powered with 4V. That is the last demonstration but it gives all the information necessary to evaluate the system. The fuse allows till 140ºC, the following table is calculated from data obtained from the last test.

Temperature	Voltage (mV)
50	265
70	280
88	290
103	304
115	315
120	320
125	324
132	326
137	330
140	333

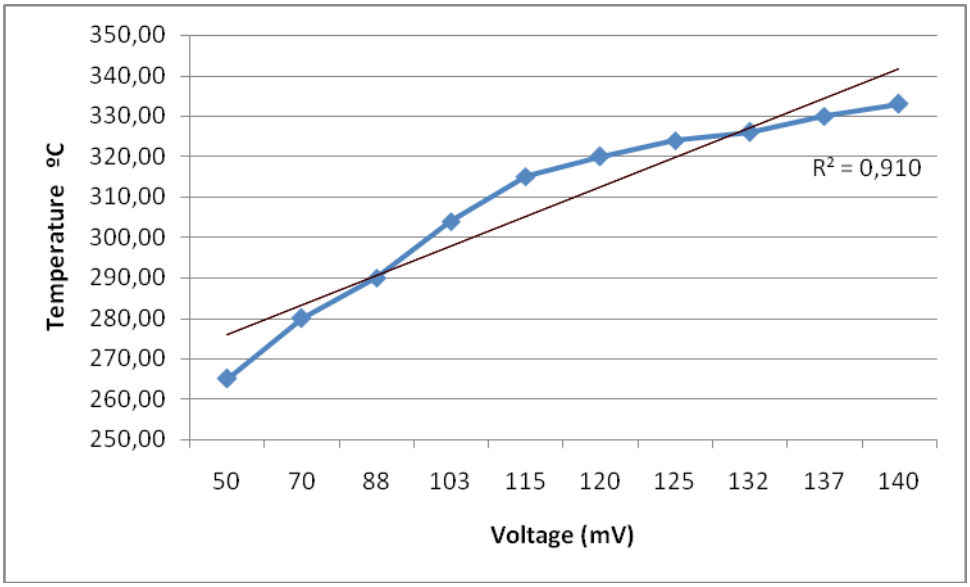


Figure 4.2: Voltage depending on the temperature

The idea was test the element till 200ºC but the characteristics of the Peltier element I use in the project don't allow more than 160ºC, there are other Peltier elements that can work with high temperatures.

The last table shows that the voltage generated changes lineally to the temperature. The information obtained in the last test admits the calculation of the deviation gradient to predict the other measures for more than 140ºC.

Temperature	Voltage (mV)	Current (mA)	Sensor Resist-ance(Ω)
50	268,00	123,0	1205,9
55	271,78	152,5	1227,1
60	275,56	182,0	1248,2
65	279,33	211,5	1269,4
70	283,11	241,0	1290,6
75	286,89	270,5	1311,8
80	290,67	300,0	1332,9
85	294,44	329,5	1354,1
90	298,22	359,0	1375,3
95	302,00	388,5	1396,5
100	305,78	418,0	1417,6
105	309,55	447,5	1438,8
110	313,33	477,0	1460,0
115	317,11	506,5	1481,2
120	320,89	536,0	1502,3
125	324,66	565,5	1523,5
130	328,44	595,0	1544,7
135	332,22	624,5	1565,9
140	336,00	654,0	1587,0
145	339,77	683,5	1608,2
150	343,55	713,0	1629,4
155	347,33	742,5	1650,6
160	351,11	772,0	1671,7

Graphics:

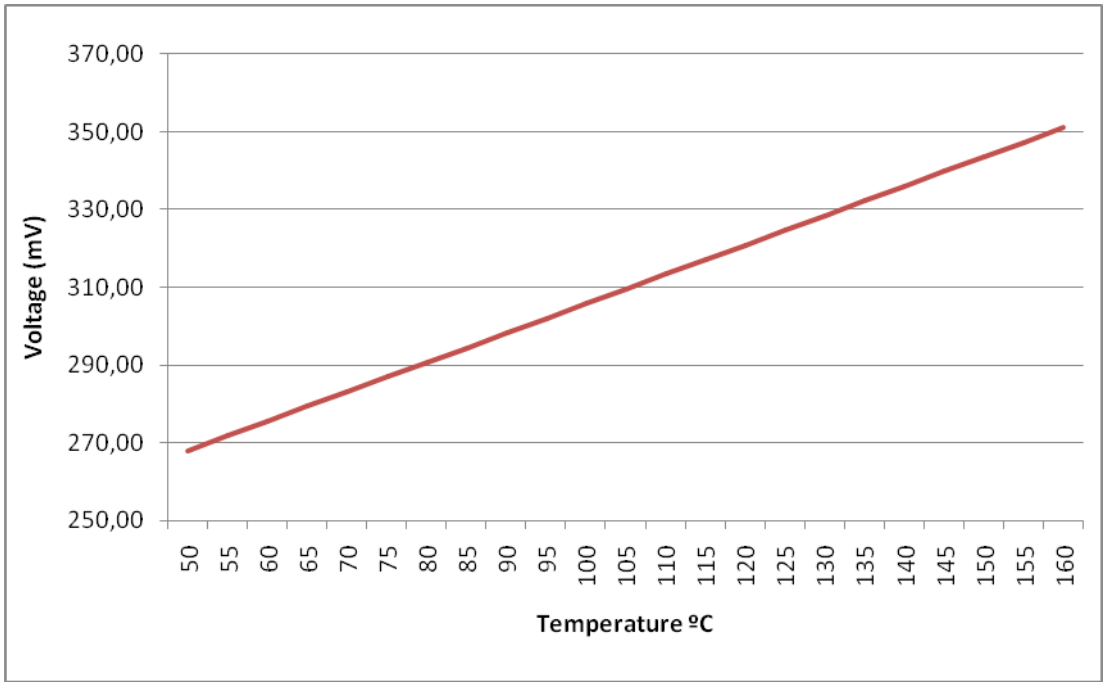


Figure 4.3: Voltage depending on the temperature results

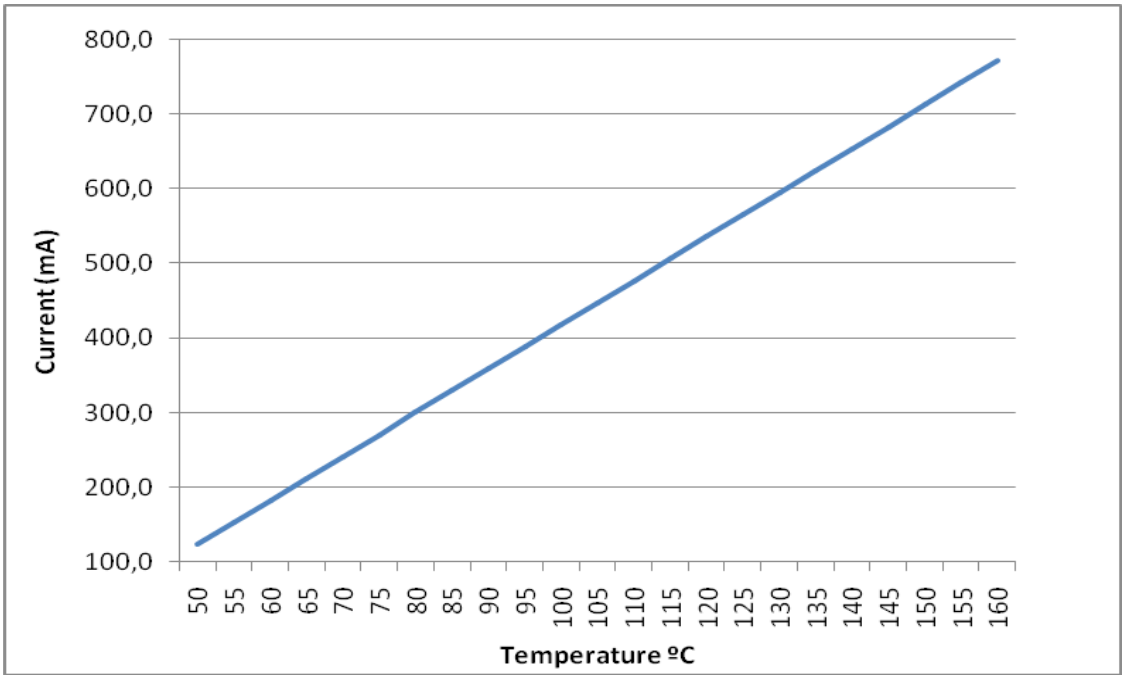


Figure 4.4: Current depending on the temperature results

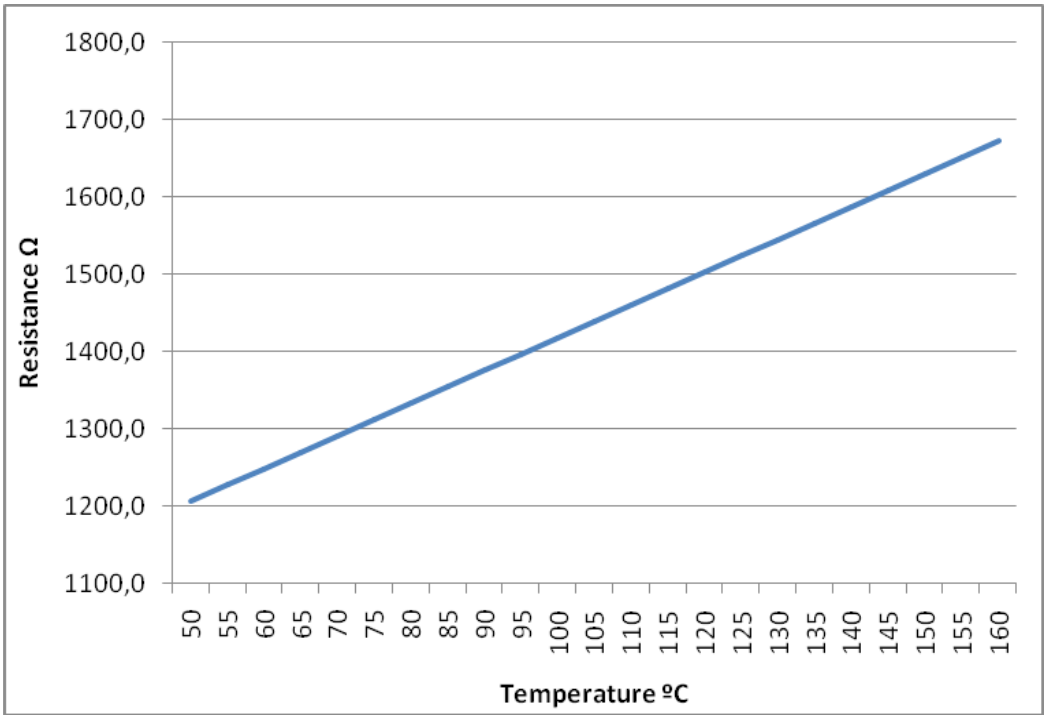


Figure 4.5: Sensor resistance depending on the temperature results

The results associated with the table shows that with a temperature of 160ºC the element provides until 770mA and 350mV that is sufficiently energy to work for a while.

It is difficult to calculate the time the capacitor will be powering the system, because it depends on the energy generated by the element Peltier and the time is working the oven. I have counted the time of working with a capacitor of 47mF, with the “oven” functioning during 4 minutes, then turning off the oven, the microcontroller was still working during 15 minutes more, that time can varied if the capacitor is bigger and the system use a real oven with higher temperatures; It is possible to works for hours.

Power generation

The following formula allows knowing the quantity of heat in the warm side of the element. The Qw variable provides the power generation of the Peltier element.

$$Qw = T \cdot \alpha \cdot I + \frac{I^2 \cdot Ri}{2} - \lambda \cdot \Delta T$$

$\alpha = 49mV/K$
$Ri = 3{,}6$
$\lambda = 250mW/K$

Temp.(ºC)	Temp.(K)	Current (mA)	Qw (KW)/Kelvin	Qw(KW)/ºC
50	323	123	1968	7,21
55	328	152,5	2485	9,10
60	333	182	3021	11,06
65	338	211,5	3573	13,09
70	343	241	4144	15,18
75	348	270,5	4732	17,33
80	353	300	5337	19,55
85	358	329,5	5961	21,83
90	363	359	6601	24,18
95	368	388,5	7260	26,59
100	373	418	7936	29,07
105	378	447,5	8629	31,61
110	383	477	9340	34,21
115	388	506,5	10069	36,88
120	393	536	10815	39,62
125	398	565,5	11579	42,41
130	403	595	12360	45,28
135	408	624,5	13160	48,20
140	413	654	13976	51,19
145	418	683,5	14810	54,25
150	423	713	15662	57,37
155	428	742,5	16532	60,56
160	433	772	17419	63,80

Power consumption

When the Peltier element works as generator, there is a consumption of power resulting in the variable Qe.

$$Qe = \alpha \cdot \Delta T \cdot I + I^2 \cdot Ri$$

Temp. ºC	Temp.(K)	Current (mA)	Qe (W)
50	323	123	154,29
55	328	152,5	227,80
60	333	182	315,76
65	338	211,5	418,18
70	343	241	535,06
75	348	270,5	666,40
80	353	300	812,19
85	358	329,5	972,44
90	363	359	1147,14
95	368	388,5	1336,31
100	373	418	1539,92
105	378	447,5	1758,00
110	383	477	1990,53
115	388	506,5	2237,52
120	393	536	2498,97
125	398	565,5	2774,87
130	403	595	3065,23
135	408	624,5	3370,05
140	413	654	3689,32
145	418	683,5	4023,05
150	423	713	4371,23
155	428	742,5	4733,88
160	433	772	5110,98

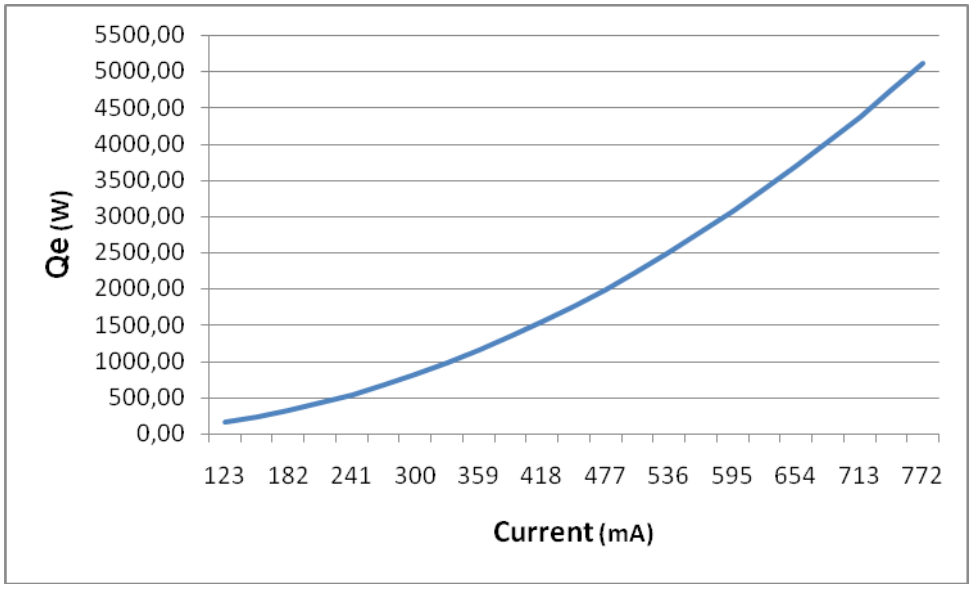


Figure 4.6: Power consumption

The table shows the increasing consumption of power when the current is higher. It varies exponentially the value.

With all the information calculated about the power consumption is possible to expand the project to calculate the exact time that the microprocessor can be working. Although the part of consume of the other elements in the circuit is not included.

5. Conclusions

The sample from Peltron GmbH admit temperatures from 0 ºC to 160ºC, another sample will be most useful for the application in an oven; that works from 0ºC to 250ºC.

The measure application is simple and easy working, it is possible to change the temperature limits using another Peltier element.
The circuit consumption is very low power, when the power save mode makes the microcontroller sleeps the power consumed is practically null.

Peltier elements provide energy alternative, supplies and save the energy wasted in the system. It is necessary to mention the capacity of the capacitor for a real simulation is better to use another with large capacity to make the harvester working hours.

The software can be modify to make the time between measures bigger, making the microprocessor sleep for more time, that will be useful to save power. Maybe a period of 30 seconds between measures is enough to know the temperature of the oven.

The conclusions of the study define the Peltier element as a great power generation when it is follows by a power saving circuit.

6. Future guidelines

This project could be expanded in several ways:

- Optimized the time between measures in the software of the system.
- Looking for other Peltier element that could work with high temperatures.
- Optimized the software of the system making the time of conversions smaller.
- Test the circuit with others capacitors looking for the best option for the application in an oven.
- Provides the laboratory with an oven simulator to test high temperatures.
- Change the led reading system to a low power led screen.

All this improvements have not been made at this project by time restrictions.

7. Bibliography

Dr.Peter Urbanek “Embedded Systems” Laserdruck im HSU-Verlag (2007).

G.S.Golas,J Sharp, H.J.Goldsmid: Thermoelectrics “Basics principles and New Materials Developments”. Ed. Springer (2001)

Harvesters and thermocouples article:
Perpetuum by Enocean 2009 ISSN 186206716 jahrgang Heft 01/ 2009 (April 2009)

Webs:

Information Thermocouples “Peltier element”:

http://tcdirect.es/deptprod.asp?deptid=180/22 (15/03/09)

http://www.isabellenhuette.de/de/das-unternehmen/?id=8&L=0 (15/03/09)

http://www.metas.com.mx/guiametas/La-Guia-MetAs-02-07-TC.pdf (18/03/09)

http://www.wamister.ch/arbeitsbl/ (19/03/09)

http://www.omega.com/temperature/z/pdf/z021-032.pdf (19/03/09)

Thermal converter Information:

http://www.enocean.com/en/energy-harvesting/ (20/03/09)

Temperature sensor:

http://www.reichelt.de (B400_t90-6121d) (27/05/09)

Microcontroller Information:

http://www2.tech.purdue.edu/ecet/courses/referencematerial/atmel/ (10/04/09)

http://www.datasheetcatalog.net/de/datasheets_pdf/A/T/M/E/ATMEGA16.shtml (14/04/09)

DC convertor:

Datasheets of the TPS 61200:
http://focus.ti.com/lit/ds/symlink/tps61200.pdf (10/06/09)

Constant current:
http://www.datasheetcatalog.net/de/datasheets_pdf/B/S/S/1/BSS131.shtml (30/06/09)

8. Annex

Software:

- Adc init function:

```
/* Includes */
#include <avr/io.h>
#include <inttypes.h>

/* Function */
void adc_init(void)
{
   ADMUX = 0;
   ADCSRA = (1 << ADEN) | (1<<ADIF)| (1 << ADIE);   /* write "1" to ADIF to clear the bit */
}
*****************************************************************************
```

- Adc ISR function:

```
/* Includes */
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <inttypes.h>

/* Global variables */
extern uint8_t conversion_finished;
/* Interrupt Service Routine */
SIGNAL(SIG_ADC)                  /* ADC interrupt occured */
{
   conversion_finished = 1;       /* set global variable for indication */
}
*****************************************************************************
```

- Timer2 init function:

```
/* Includes */
#include <avr/io.h>
#include <inttypes.h>
```

```
/* Function */
void timer2_init(void)
{
   /* Initialization timer2 */
   ASSR = (1<<AS2);    // enable timer 2
   TCNT2=0 ;          //start value counter2
   TCCR2|=(1<<CS20);   //counter2 I/O-Clock/1
   TIMSK |= (1<<TOIE2); // enable interrupt
}
*****************************************************************************
```

- Timer2 ISR function:

```
/* Includes */
#include <avr/io.h>
#include <avr/interrupt.h>

/* Variables */
volatile uint16_t up = 1;
volatile uint16_t down = 0x80;
extern volatile uint16_t count;

ISR(TIMER2_COMP_vect)
{
    count ++;
}

ISR(TIMER2_OVF_vect)
{
    count ++;
}
*****************************************************************************
```

- **Sleep mode init function:**

```
/* Includes */
#include <avr/io.h>
#include <inttypes.h>

/* Function */
void sleep_mode_init(void)

{
   /* Initialization sleep mode */
   MCUCR = (1<<SM1)|(1<<SM0);     //ON Power-save Mode
   MCUCR|= (1<<SE);              //ON  Sleep Mode
}
*****************************************************************************
```

- **Main program:**

```
/* Includes */
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/delay.h>
#include <inttypes.h>
#include <avr/sleep.h>

#define VoltageReference 2.56
#define Gradient 0.019
#define ResistenceDivider 2.935  //Result of the Resistence equivalent in the voltage divider

extern void port_init(void);
extern void adc_init(void);
extern void timer2_init(void);
extern void sleep_mode_init(void);
volatile uint16_t count;

/* Main */
int main(void)
{  volatile uint16_t SensorVoltage;
   volatile uint16_t Vsupply;
   volatile uint16_t Temperature;

    count = 0;
    cli();                  /* no interrupts during initialization */
    port_init();              /* init ports */
    adc_init();                   /* init adc */
    timer2_init();
    sleep_mode_init();
    sei();              /* accept interrupts now */

    while (1)
    {
       if (count >= 1280)
       {
          ADMUX = (1<<REFS1)|(1<<REFS0);
          ADCSRA |= 1<<ADSC;                    /* start conversion*/
          while (ADCSRA & (1 << ADSC));
          SensorVoltage = ADCW;
          ADMUX |= (1 <<MUX0);
          ADCSRA |= 1<<ADSC;                    /* start conversion*/
          while (ADCSRA & (1 << ADSC));
          Vsupply = ADCW;

          /* Calculation of the Sensor´s temperature */

          Temperature = (SensorVoltage/(Gradient*Vsupply*ResistenceDivider));
          Temperature= ((float)(Temperature *1024)/VoltageReference);
```

```
            if (Temperature<1300)                /* Switch On Green LED */
            {
              if (PORTA==0xF0)
                  PORTA &= ~(1 << PA6);
              else if (PORTA==0xE0)  /* before the led was Red */
              {
                  PORTA |= (1 << PA4);
                  PORTA &= ~(1 << PA6);
              }
             }
             else                                /* Switch On Red LED */
            {
              if (PORTA==0xF0)
                   PORTA &= ~(1 << PA4);
              else if (PORTA==0xB0)  /* before the led was Green */
              {      PORTA |= (1 << PA6);
                     PORTA &= ~(1 << PA4);
              }
            }
            count = 0;
          }
          _delay_ms(1000);  /* watch the leds */

          /* switch off the leds*/
          if ( PORTA==0xE0)
                PORTA |= (1 << PA4);

          else if (PORTA==0xB0)
                PORTA |= (1 << PA6);


          /* sleep while unused */
          sleep_mode();

        }
       return 1;
}
```

****