UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

“Development of a wireless humidity and temperature sensor device with low power consumption itegrated into a home automation system.”

TRABAJO FINAL DE GRADO

Autor/a:
Alejandro Faus Atienza

Tutor/a:
Dr. Fernando Boronat Seguí

GANDIA, 2014

FH D
Fachhochschule Düsseldorf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS DE GANDIA

Praxisprojekt

Development of a wireless humidity and temperature sensor device with low power consumption integrated into a home automation system.

Created by:

Alejandro Faus Atienza

Matr.-No. 664940

Supervisor:

Prof. Dr.-Ing. Ulrich G. Schaarschmidt
B.Sc. Oliver de Fragstein

Düsseldorf, 17.07.14

Summary

Developing and production of a wireless humidity and temperature sensor device. This prototype is part of a low power home automation system which is based on Atmel ATmega128RFA1 microcontrollers using the OpenSource OS Contiki. The device communicates with the central node via Wifi 802.11 and 6LoWPAN based on IEEE 802.15.4 with IPv6.

Index

1. Introduction

Today the world of home automation is having an exponential growth. When we talk about home automation, we understand it as a set of systems capable of automating a home, providing energy management services, safety, welfare and communication, and which can be integrated through internal and external communication networks, wired or wireless.

The number of domotic homes is still relatively low compared to the total homes, however the interest in its adoption is steadily growing. On the other hand, the reduction of prices of home automation products contributes to the growth of these networks.

This project is located as part of a home automation network developed by the different components of the communications department.

The global project consists of a home automation network where we have different devices as a window open detector, a temperature sensor, water detector, and definitely all the different sensors devices that we want to include depending on the purpose and objective of the project.

The initial objective pursued in this project is to facilitate the lives of people in their homes. Especially thinking about the elderly and those who need special care and a dedicated staff. This project should therefore be very easy and intuitive for the final user.

Our goal within the project will be the developing and production of a wireless humidity and temperature sensor device. This prototype is part of a low power home automation system which is based on Atmel ATmega128RFA1 microcontrollers using the Open Source Contiki OS. The device communicates with the central node via WiFi 802.11 and 6LoWPAN based on IEEE 802.15.4 with IPv6.

Moreover, the learning objective of this project is to enhance knowledge in electronics and communications area, previously acquired.

Specifically my work within this overall project will cover from design, to printing, assembly and programming of a sensor device that can take and send temperature and humidity data to the central module (Router), which is responsible for receiving frames data from each sensor/detector, interpret and display as interest on an interface.

Regarding the content of the document, in the second point the entire process of project implementation is explained. Then in the third point, some problems that we have experienced are discussed. In the fourth point, an improvement of the prototype is exposed. In the fifth point we have the conclusion of the document. In the sixth point, the annexes are added, as the full code of the program or as some settings, necessary for the proper functioning of the programs. Finally the list of abbreviations, figures and references are included in the seventh, eighth and ninth points.

2. Implementation

In order to meet the objective we had to learn to manage software tools for PCB, layout for device programming and for testing the correct operation and also learn to handle various tools for developing PCBs available laboratory.

To achieve full alignment with the home automation system installed in the laboratory, we used the components that have been used in other devices connected to the network (such as the microcontroller, the radio transmission circuit or the programming interface), changing and adding, other necessary components for our purpose.

2.1. Schematic circuit

The first step in the project was the design of it. In our project we used the EAGLE as CAD tool to design the schematic and PCB.

The EAGLE is a powerful and flexible PCB design software offering high level functionality of expensive commercial circuit board design software at a fraction of the cost for over 20 years. EAGLE is easy to learn, easy to use and easy to buy [1].

We can also find a lot of free libraries and it has a very efficient editor.

The first task was to build a circuit that can be implemented with the objectives set. The key elements here are:

- The microcontroller
- The programming interface
- The humidity and temperature sensor
- The radio communication

We can observe the Eagle schematic in the Figure 1 below.

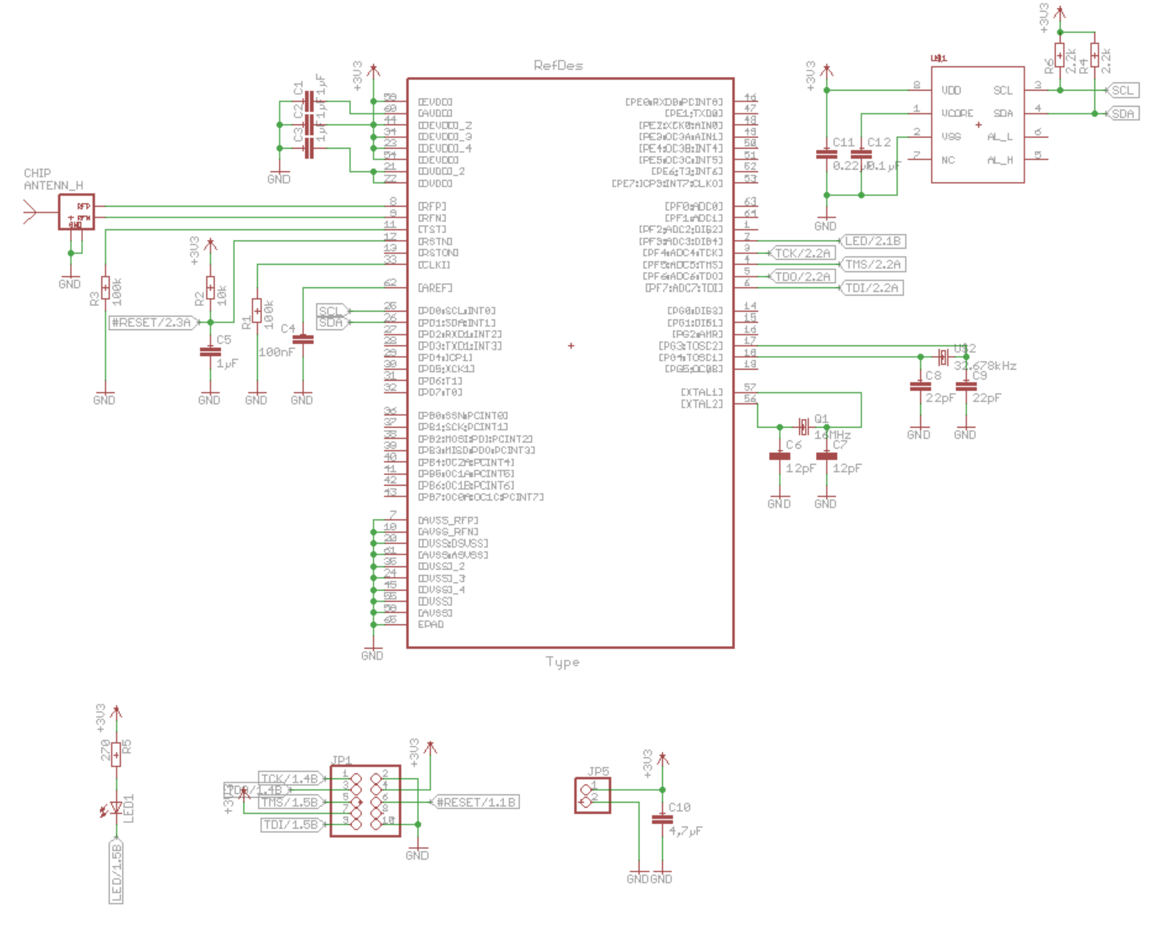


Figure 1. Eagle schematic

The individual components are described in more detail.

2.1.1. Microcontroller

The ATmega128RFA1 is a low-power CMOS 8 bit microcontroller based on the AVR enhanced RISC architecture combined with a high data rate transceiver for the 2.4 GHz ISM band. It is derived from the ATmega1281 microcontroller and the AT86RF231 radio transceiver.

By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The radio transceiver provides high data rates from 250 kb/s up to 2 Mb/s, frame handling, outstanding receiver sensitivity and high transmit output power enabling a very robust wireless communication [2].

The Figure 2 shows the ATmega128RFA1 block diagram.

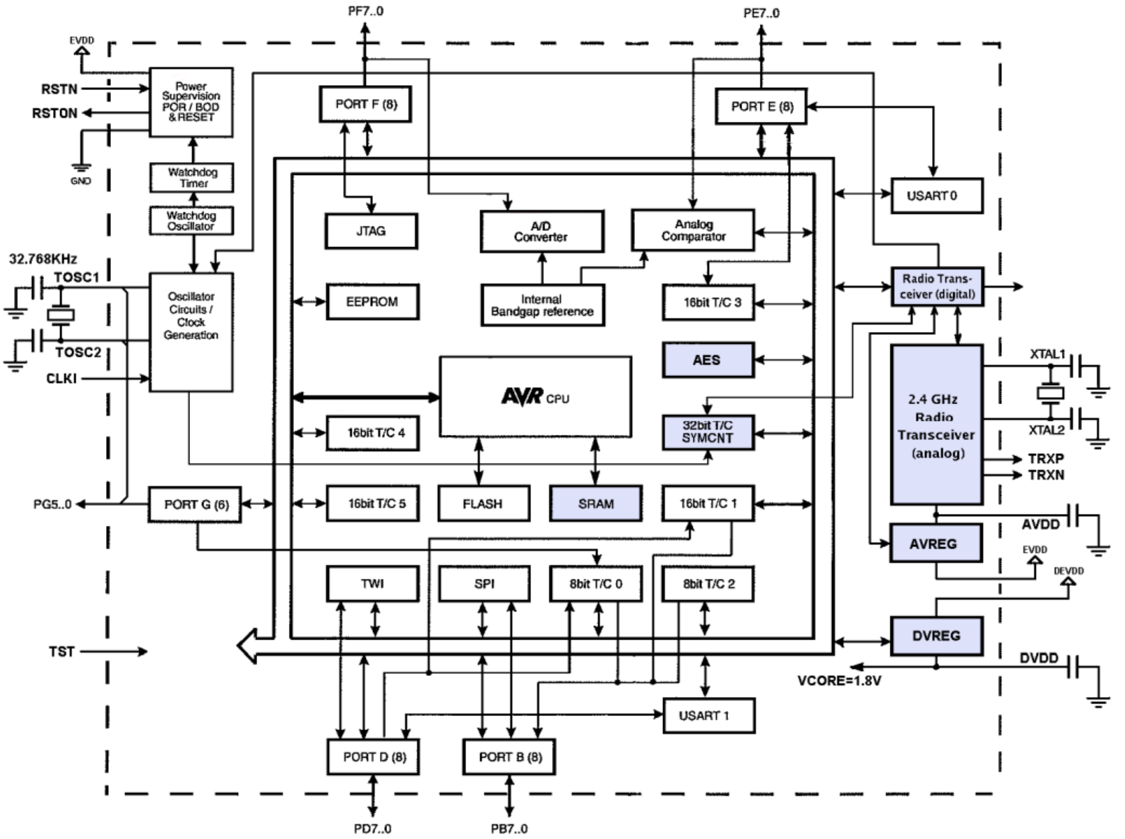


Figure 2. Atmega128RFA1 block diagram

2.1.2. Programming Interface

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

- Testing PCBs by using the JTAG Boundary-scan capability
- Programming the non-volatile memories, Fuses and Lock bits
- On-chip debugging

This programming interface has greatly facilitated the programming and testing of our programs.

Programming through the JTAG interface requires control of the four JTAG specific pins: TCK, TMS, TDI, and TDO. Control of the reset and clock pins is not required.

To be able to use the JTAG interface, the JTAGEN Fuse must be programmed. The device is default shipped with the fuse programmed. In addition, the JTD bit in MCUCR must be cleared. Alternatively, if the JTD bit is set, the external reset can be forced low.

Then, the JTD bit will be cleared after two chip clocks, and the JTAG pins are available for programming. This provides a means of using the JTAG pins as normal port pins in running mode while still allowing In-System Programming via the JTAG interface. Note that this technique cannot be used when using the JTAG pins for Boundary-scan or

On-chip Debug. In these cases the JTAG pins must be dedicated for this purpose.

During programming the clock frequency of the TCK Input must be less than the maximum frequency of the chip. The System Clock Prescaler cannot be used to divide the TCK Clock Input into a sufficiently low frequency [2].

The Figure 3 shows the Atmel128RFA1 JTAG connections in our scheme.

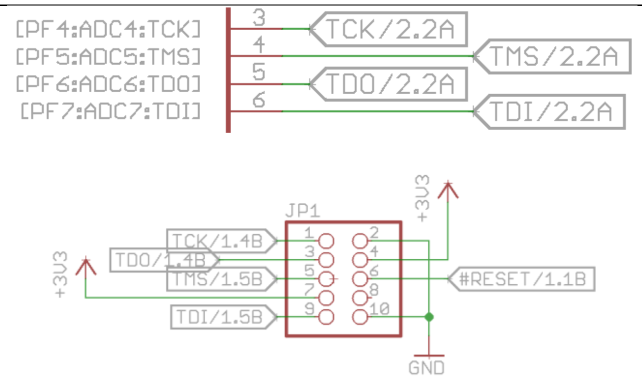


Figure 3. JTAG Atmel128RFA JTAG connections

2.1.3. Humidity & temperature sensor

We have chosen the HIH6130 sensor in our project for the following main reasons:

- Low supply voltage: Can operate down to 2.3 Vdc, which allows use in low energy and wireless-compatible applications to enhance energy savings and prolong system battery life. In our case we will use around 3.3 Vdc for the sensor and the microcontroller.

- Low power consumption: The sensor goes into sleep mode when not taking a measurement within the application, consuming only 1 µA of power versus 650µA in full operation in a battery operated system. Sleep mode helps maximize battery life, reduces power supply size, and reduces the application's overall weight

- High 14-bit humidity sensor resolution and 14-bit temperature sensor resolution within the application help the user's system detect the smallest relative humidity or temperature change.

- The package is ultra-small. Allows for flexibility of use within the application, occupies less space on the PCB, and typically simplifies placement on crowded PCBs or in small devices.

2.1.4. Radio communication

The ATmega128RFA1 features a low-power 2.4 GHz radio transceiver designed for industrial and consumer ZigBee/IEEE 802.15.4, 6LoWPAN, RF4CE and high data rate 2.4 GHz ISM band applications. The radio transceiver is a true peripheral block of the AVR microcontroller. All RF-critical components except the antenna, crystal and de-coupling capacitors are integrated on-chip. Therefore, the ATmega128RFA1 is particularly suitable for wireless sensor networks applications.

This radio transceiver is part of a system-on-chip solution with an AVR® microcontroller. It comprises a complex peripheral component containing the analog radio, digital modulation and demodulation including time and frequency synchronization and data buffering. The number of external components for the transceiver operation is minimized such that only the antenna, the crystal and decoupling capacitors are required. The bidirectional differential antenna pins (RFP, RFN) are used for transmission and reception, thus no external antenna switch is needed.

Unlike all other AVR I/O modules, the transceiver module can operate asynchronously to the controller. The transceiver requires an accurate 16MHz crystal clock for operation, but the controller can run at any frequency within its operating limits.

The following figure (Figure 4) [5] shows the recommended mounting considerations with matching circuits:

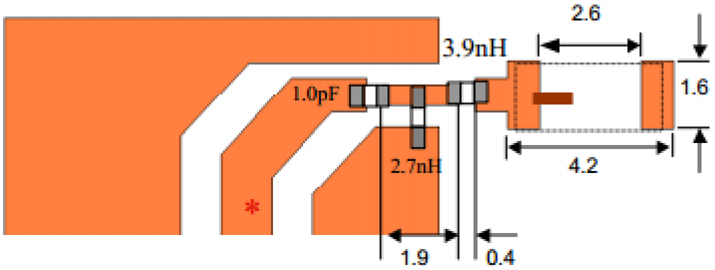


Figure 4. Mounting considerations with matching circuits

We need to have impedance matching to prevent reflections at the desired frequency. In the graph below (Figure 5) we can see the response that we get with the filter of the previous figure (Figure 4). It is observed as we get a gain of -16 dB at the point with frequency equal to 2.45 GHz, this will make all frequencies, except frequencies that are close to the center frequency (point m3), are reflected.

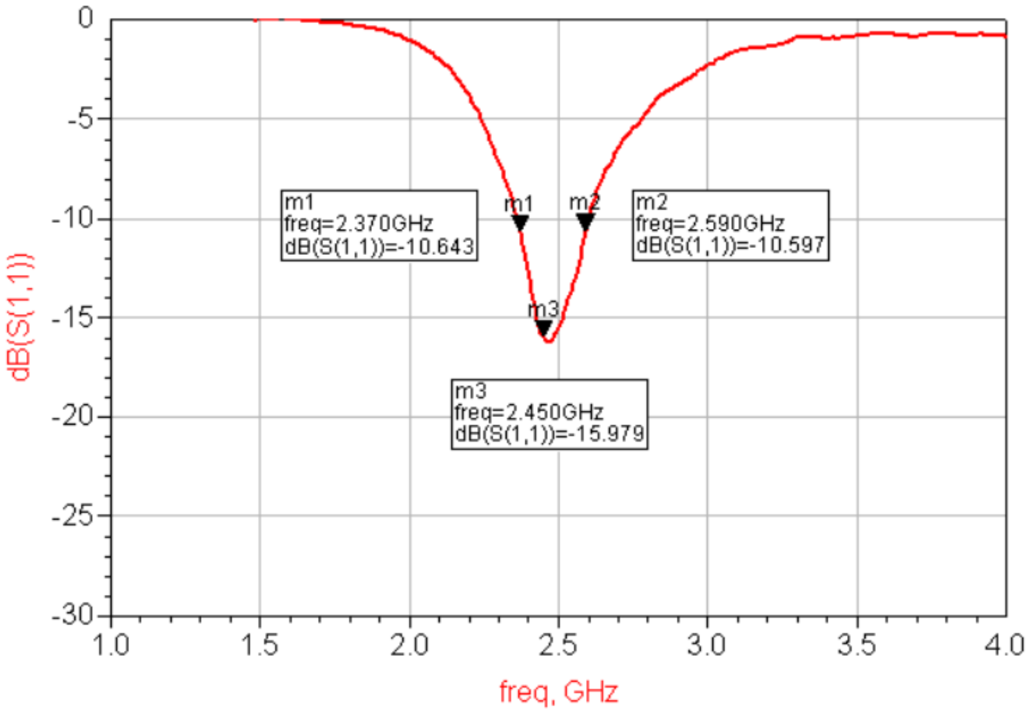


Figure 5. Reflection factor

$$fc(GHz) = \frac{fmax + fmin}{2} = \frac{2.370 + 2.590}{2} = 2.48\ GHz$$

Equation 1. Central frequency

We also note the input impedance on the Smith chart (Figure 6) that shows the m3 point is very close to the unit circumference, so definitely we will have a good impedance match with this filter.

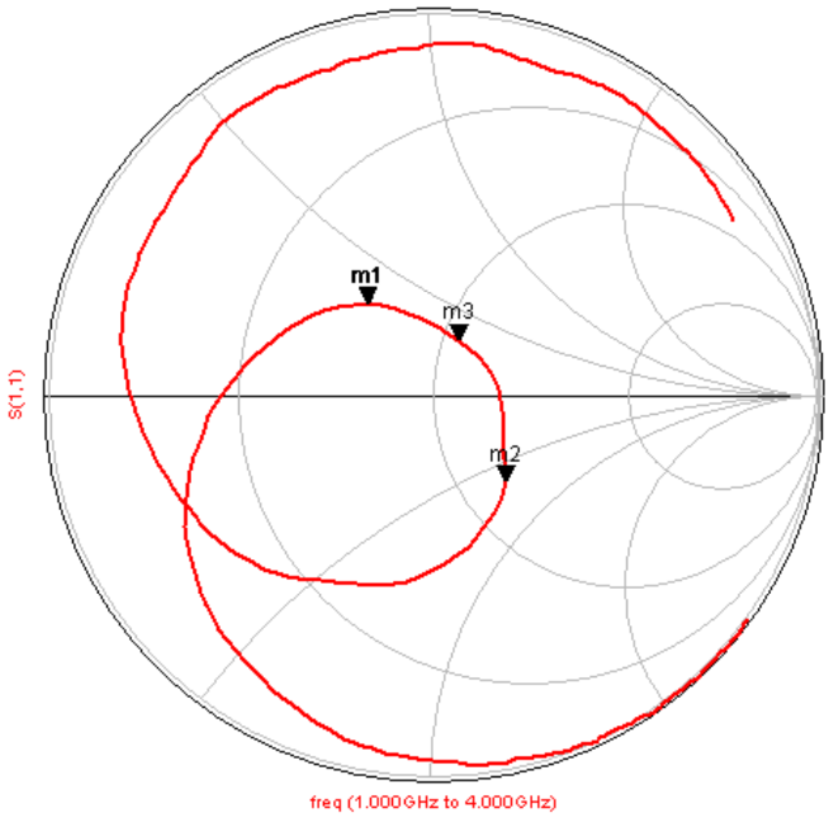


Figure 6. Input impedance

The radiation pattern of the antenna is also shown in the following figure (Figure 7):

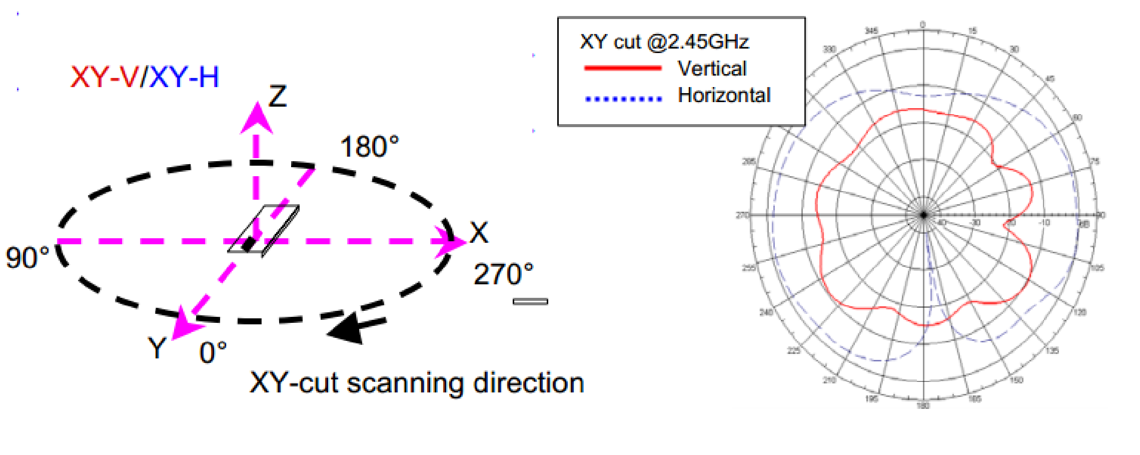


Figure 7. Typical radiation patterns

In addition the 50Ω single-ended RF input is transformed to the 100Ω differential RF port impedance using a Balun.

A block diagram for an application using an antenna switch is shown in the following figure (Figure 8):

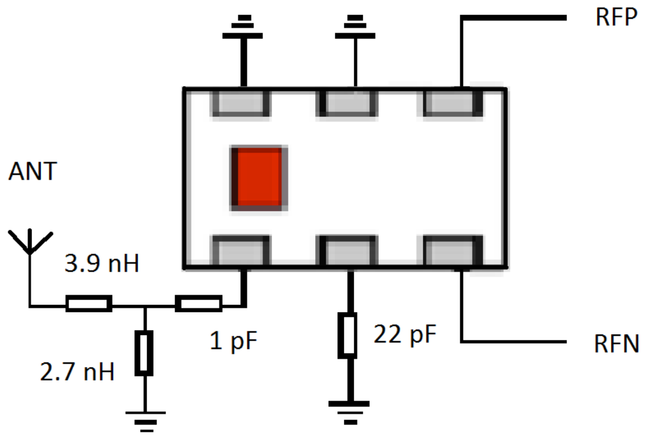


Figure 8. Radio communication module

In our scheme, we include the antenne, the filter and the balun in the same module (Figure 9):

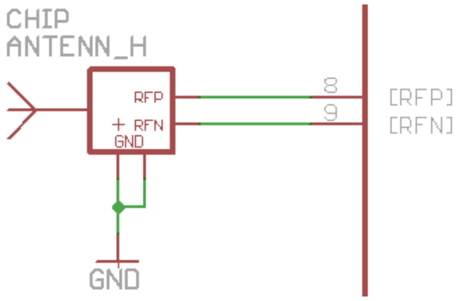


Figure 9. Radio communication scheme module

2.2. Layout

Once we have the schematic with all the components that make up our PCB, we began to design the layout. Logically we use the same program as above, this allows us to add capsules to schema components and go directly to the layout scheme preserving the connections between components.

The layout of the prototype is shown on next figure (Figure 10), which you can see both sides of the PCB along with the drawing of the components in the positions that they will occupy once welded.

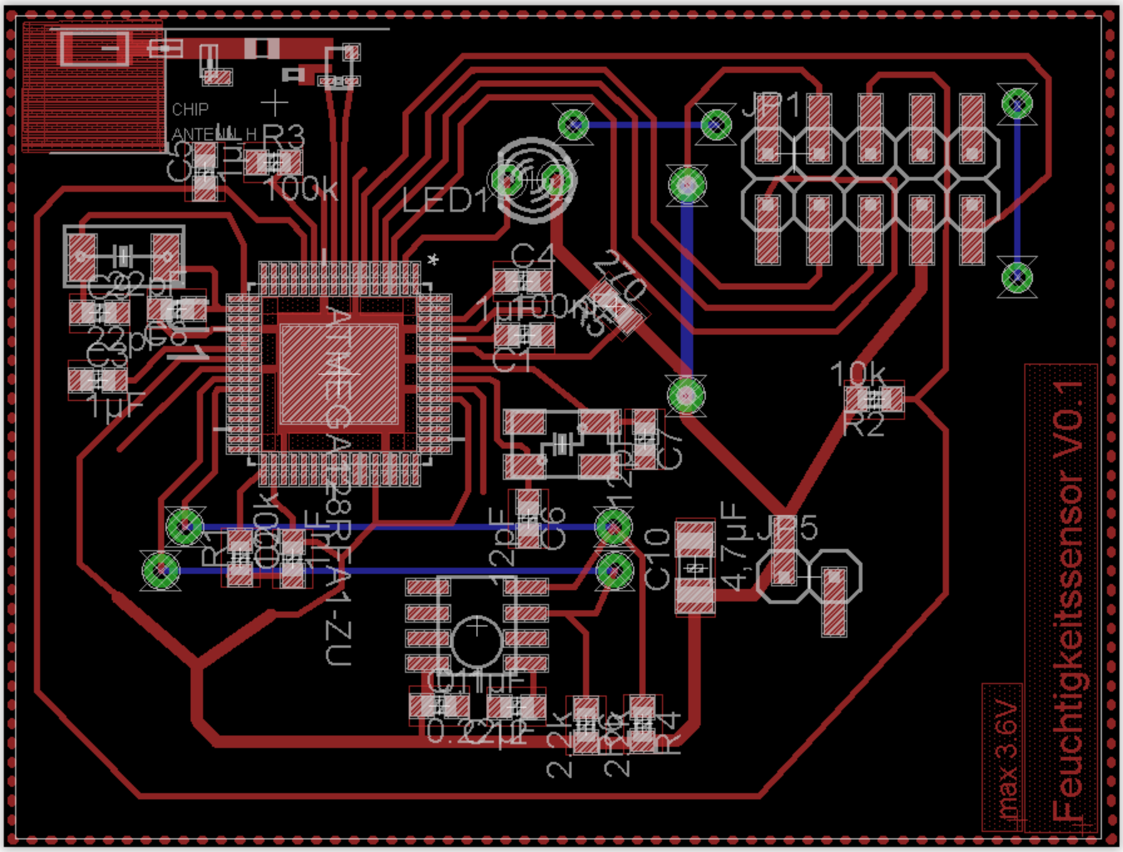


Figure 10. Layout

To make this layout we have taken into account several features and recommendations, and apply lessons learned from previous tests conducted in the laboratory. Here are some tips to consider for various components that we needed to be careful.

2.2.1. Microcontroller

After a bad experience with the first encapsulation that I used in the first test, we decided to use another which improved the length of the tracks compared to the previous. In the first of these were so short that they could not be welded well.

We must be careful with the microcontroller because if you consider the size of it, it will be the most difficult part of welding and therefore the more critical point in our layout. We have to be careful with the microcontroller massa below and make sure that the outer tracks reach the center of each pin.

The following figure (Figure 11) shows the microcontroller packaging in more detail:

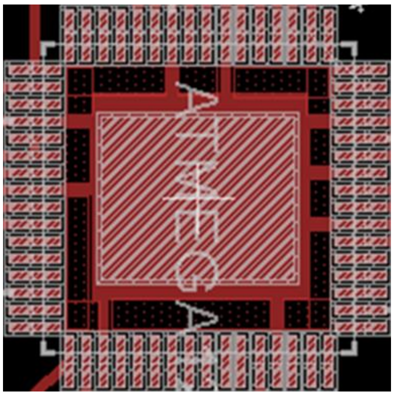


Figure 11. Microcontroller packaging

2.2.2. Radio communication

We must be very careful with that part of the transmission in the layout, any failure working at high frequencies can be very serious. We must consider two main precautions. The first is to try to have the minimum possible distance from the microcontroller to the balun. And second, get that the angle between the two tracks be less abrupt as possible.

Another issue to keep in mind is to apply a restriction on the area around the antenna to avoid a possible reflection of the PCB material. This is because we are working with large ground planes.

The following figure (Figure 12) shows the radio communication packaging in more detail:

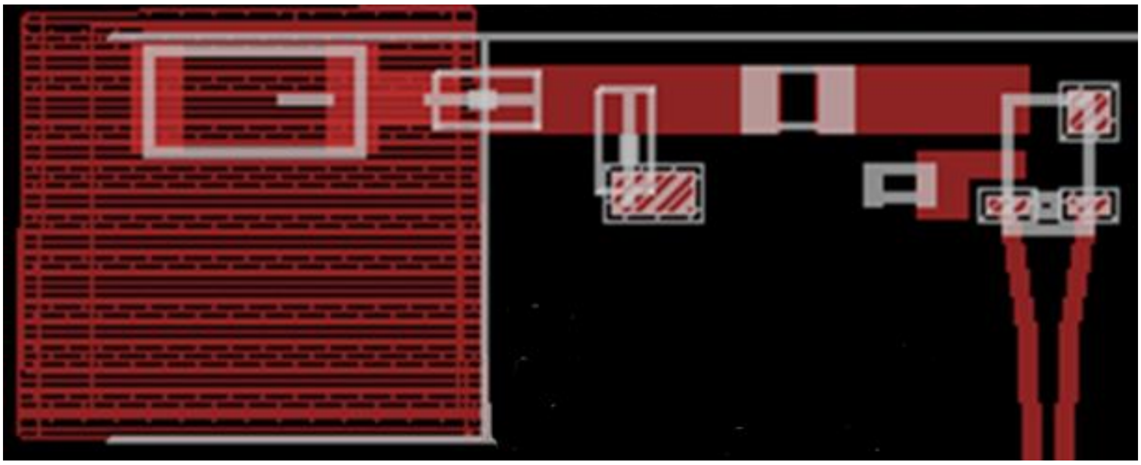


Figure 12. Radio communication packaging

2.2.3. PCB

Once the design is complete, we print the PCB mask on transparent paper or greaseproof paper. You can use a spray to achieve greater density in the mask. Photo etching process (1 minute and 14 seconds in our prototype) is performed. Now we should start seeing the mask on our PCB. It's time to make the process of copper corrosion by acid. It should be careful to narrow due process to the dangerousness of the materials used in it. Once the process is complete, check that everything is as it should be in the PCB. Now we can use acetone to remove the photoresist film that is still covering the copper. Now we can use another spray to protect that copper and also facilitate welding on it.

We are now ready to weld all components. We must also be very careful because we are working with SMD components and this increases the difficulty. We use flux to help the solder adhere better to the surface.

Once we finished the assembly process, the result can be seen in the following figure (Figure 13):

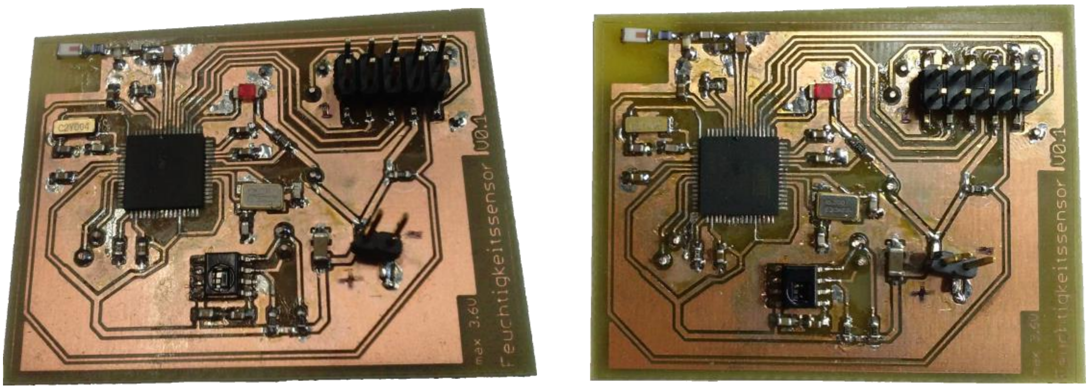


Figure 13. PCB

2.3. Software

After finishing the EAGLE circuit design, we print, assemble and weld the circuit to operate it. The next step is to implement the program using the Atmel Studio 6 program.

Atmel Studio 6 is the integrated development platform (IDP) for developing and debugging Atmel ARM Cortex-M and Atmel AVR microcontroller (MCU) based applications. The Atmel Studio 6 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code.

2.3.1. Contiki OS

Contiki is an open-source multitasking event-driven operating system designed for networked embedded devices. Its lightweight footprint makes it suitable for memory constraint microcontrollers.

Contiki gathers several independent modules such as an event-driven thread-like multitasking environment with theprotothread library, the uIP TCP/IP (v4 and v6) stack, the wireless sensor network set of protocols: the Rimestack.

Contiki is primarily designed for networking applications, but can be used for any other application using only its event driven kernel.

- Events

The Contiki kernel is event-driven. The idea of such a system is that every execution of a part of the application is a reaction to an event. The entire application (kernel + libraries + user code) may contain several processes that will execute concurrently.

The different processes usually execute for some time, then wait for events to happen. While waiting, a process is said to be blocked. When a event happen, the kernel execute the process passing it information about the event. The kernel is responsible for activating the processes when the events they're waiting for happen.

Events can be classified in three kinds:

timer events: a process may set a timer to generate an event after a given time, it will block until the timer expires and then continue its execution. This is useful for periodic actions, or for networking protocols e.g. involving synchronization;

external events: peripheral devices connected to IO pins of the microcontroller with interrupt capabilities may generate events when triggering interruptions. A push-button, a radio chip or a shock detector accelerometer are a few examples of devices that could generate interruptions, thus events. Processes may wait for such events to react accordingly.

internal events: any process has the possibility to address events to any other process, or itself. This is useful for interprocess communication as informing a process that data is ready for computation.

Events are said posted. An interrupt service routine will post an event to a process when it is executed. Events have the following information:

process: the process addressed by the event. It can be either one specific process or all the registered processes;

event type: the type of event. The user can define some event types for the processes to differentiate them, such as one when a packet is received, one when a packet is sent;

data: additionally, some data may be provided along with the event for the process.

This is the main principle of the Contiki Operating System: events are posted to processes, these execute when they receive them until they block waiting for another event.

- Processes

Processes are the task-equivalent of Contiki. The process mechanism uses the underlying protothread library which in turn uses the local continuation library.Refer to the given links for more information.

A process is a C function most likely containing an infinite loop and some blocking macro calls. Since the Contiki event-driven kernel is not preemptive, each process when executed will run until it blocks for an event. Several macros are defined for the different blocking possibilities. This allows programming state-machines as a sequential flow of control.

- uIP TCP/IP stack

Contiki contains a lightweight TCP/IP stack called uIP (uIP website). It implements RFC-compliant IPv4, IPv6, TCP and UDP (the latter two compatible with IPv4 and IPv6). uIP is very optimized, only the required features are implemented. For instance there is a single buffer for the whole stack, used for received packets as well as for those to send.

In the Annex (Point 6) can be found an example program that is like the basic skeleton of Contiki OS and from which we can begin programming [7].

2.3.2. Wireshark

Once the program is completed, we want to test the prototype. We need to check that the information provided in the program. How we work connected to the central node (router) via WI-FI IEEE 802.11, we can connect the PC to the same network and using a "sniffer" for monitoring the data transmitted in this network.

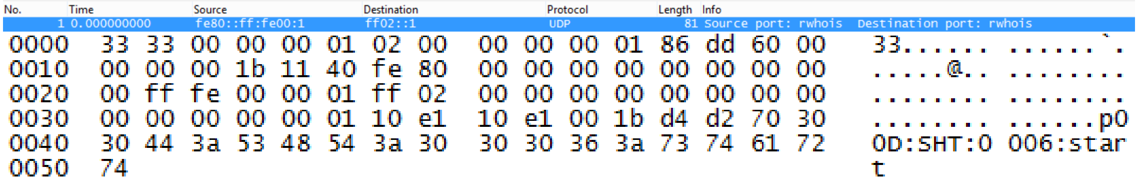
To do this, we used the freeware version of WireShark program.

What is Wireshark?

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

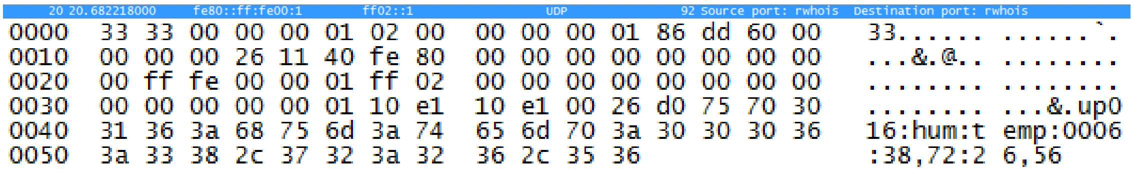
You could think of a network packet analyzer as a measuring device used to examine what's going on inside a network cable, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course) [3].

If everything works correctly, should be able to see the data that we tried to convey. What we see in the following figures (Figure 15, 16, 17, 18 and 19).



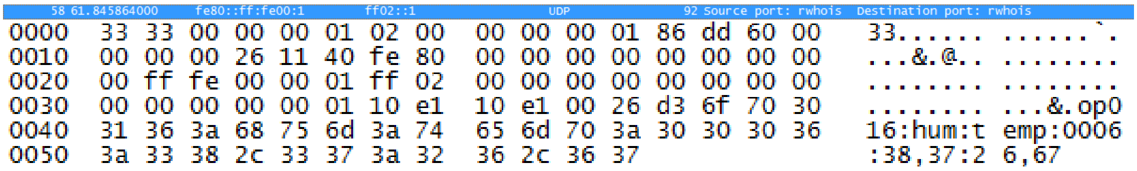
```
No.   Time          Source              Destination   Protocol   Length  Info
    1 0.000000000   fe80::ff:fe00:1     ff02::1       UDP            81  Source port: rwhois  Destination port: rwhois
0000  33 33 00 00 00 01 02 00  00 00 00 01 86 dd 60 00   33...... ......`.
0010  00 00 00 1b 11 40 fe 80  00 00 00 00 00 00 00 00   .....@.. ........
0020  00 ff fe 00 00 01 ff 02  00 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 01 10 e1  10 e1 00 1b d4 d2 70 30   ........ ......p0
0040  30 44 3a 53 48 54 3a 30  30 30 36 3a 73 74 61 72   0D:SHT:0 006:star
0050  74                                                 t
```

Figure 14. Start data



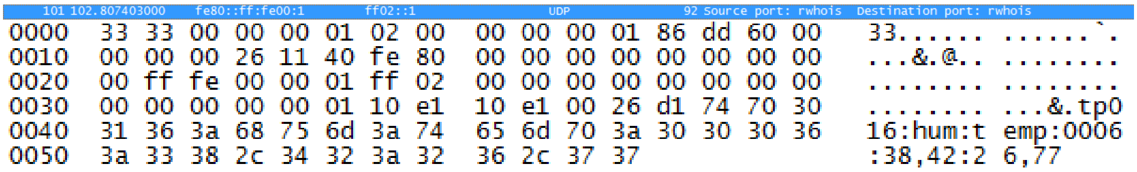
```
   20 20.682218000  fe80::ff:fe00:1     ff02::1       UDP            92  Source port: rwhois  Destination port: rwhois
0000  33 33 00 00 00 01 02 00  00 00 00 01 86 dd 60 00   33...... ......`.
0010  00 00 00 26 11 40 fe 80  00 00 00 00 00 00 00 00   ...&.@.. ........
0020  00 ff fe 00 00 01 ff 02  00 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 01 10 e1  10 e1 00 26 d0 75 70 30   ........ ...&.up0
0040  31 36 3a 68 75 6d 3a 74  65 6d 70 3a 30 30 30 36   16:hum:t emp:0006
0050  3a 33 38 2c 37 32 3a 32  36 2c 35 36               :38,72:2 6,56
```

Figure 15. Humidity and temperature data 1



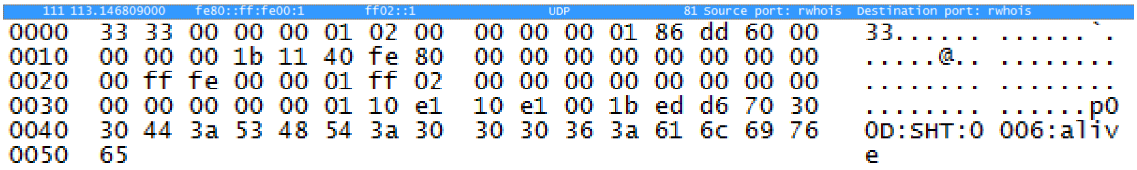
```
     58 61.845864000     fe80::ff:fe00:1         ff02::1                 UDP               92 Source port: rwhois  Destination port: rwhois
0000   33 33 00 00 00 01 02 00  00 00 00 01 86 dd 60 00   33...... ......`.
0010   00 00 00 26 11 40 fe 80  00 00 00 00 00 00 00 00   ...&.@.. ........
0020   00 ff fe 00 00 01 ff 02  00 00 00 00 00 00 00 00   ........ ........
0030   00 00 00 00 00 01 10 e1  10 e1 00 26 d3 6f 70 30   ........ ...&.op0
0040   31 36 3a 68 75 6d 3a 74  65 6d 70 3a 30 30 30 36   16:hum:t emp:0006
0050   3a 33 38 2c 33 37 3a 32  36 2c 36 37               :38,37:2 6,67
```

Figure 16. Humidity and temperature data 2



```
    101 102.807403000    fe80::ff:fe00:1         ff02::1                 UDP               92 Source port: rwhois  Destination port: rwhois
0000   33 33 00 00 00 01 02 00  00 00 00 01 86 dd 60 00   33...... ......`.
0010   00 00 00 26 11 40 fe 80  00 00 00 00 00 00 00 00   ...&.@.. ........
0020   00 ff fe 00 00 01 ff 02  00 00 00 00 00 00 00 00   ........ ........
0030   00 00 00 00 00 01 10 e1  10 e1 00 26 d1 74 70 30   ........ ...&.tp0
0040   31 36 3a 68 75 6d 3a 74  65 6d 70 3a 30 30 30 36   16:hum:t emp:0006
0050   3a 33 38 2c 34 32 3a 32  36 2c 37 37               :38,42:2 6,77
```

Figure 17. Humidity and temperature data 3



```
    111 113.146809000    fe80::ff:fe00:1         ff02::1                 UDP               81 Source port: rwhois  Destination port: rwhois
0000   33 33 00 00 00 01 02 00  00 00 00 01 86 dd 60 00   33...... ......`.
0010   00 00 00 1b 11 40 fe 80  00 00 00 00 00 00 00 00   .....@.. ........
0020   00 ff fe 00 00 01 ff 02  00 00 00 00 00 00 00 00   ........ ........
0030   00 00 00 00 00 01 10 e1  10 e1 00 1b ed d6 70 30   ........ ......p0
0040   30 44 3a 53 48 54 3a 30  30 30 36 3a 61 6c 69 76   0D:SHT:0 006:aliv
0050   65                                                 e
```

Figure 18. Alive data

3. Problems

Throughout the implementation process we have had several problems explained in the following points.

3.1. PCB process

We had a problem in the process of the acid to remove copper in the PCB. After the process could not get a well-defined mask, so we had to repeat the process several times to get a good result. This problem was solved by changing the material of the PCB.

3.2. Microcontroller layout

The problem that we find when we mount the first PCBs was that the microcontroller package had the pins too shorts, thus increasing the difficulty of welding.

Once experienced the problem we decided to redesign the PCB with an encapsulated with longer pins and thus we get solve the problem.

The following figure (Figure 21) shows the difference between the two packages.

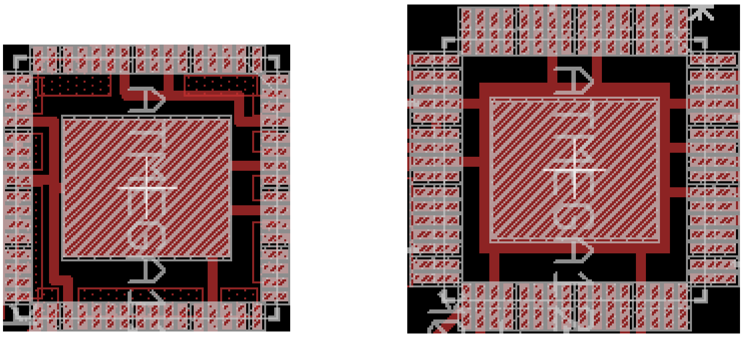


Figure 19. Microcontroller packaging comparison

3.3. Humidity & temperature sensor

The second issue where I have spent more time has been implementing the program for sensor data. The first reason was that I had never previously worked with I2C. Not was a problem finding general examples for I2C operation. However the instructions to get sensor data correctly and so that they will update were a bit confusing in the sensor datasheet.

3.4. Antenna & balun

It has been another critical part regarding the part of the layout. We can get in trouble for reasons like having welds too thick, too abrupt angles on the slopes or some kind of reflection on the material.

The good part is that thanks to the microcontroller that we use, we can make the transmission of the few components and fully integrated into the PCB form, so reduce costs compared to a commercial external module and we also saves space.

3.5. Contiki

Another problem we have found is to work with Contiki OS. There is not much information or examples where use Contiki OS. Add to that there is no possibility to simulate or test step by step our program with Contiki OS. So the best way to investigate has been taking the skeleton of the basic example they offer us at the official site of Contiki OS and make our tests from there. For this reason, the first thing that we had to do is make sure that the transmission working properly to start monitoring the data being sent.

4. Improvements

Looking at the layout of the prototype, a clear improvement that we can make is to bring the sensor to the microcontroller I2C pins. Rearranging the components got some valuable space to include, for example, a voltage regulator. In this way, we ensure that you get the correct voltage or a peak of voltage can not damage the microcontroller or any other component.

In this design I used the LP2985 voltage regulator for its small size and its availability and reliability. Also it had been used in previous laboratory projects.

The schematic design has had to add to the new distribution is shown in the following figure (Figure 21).

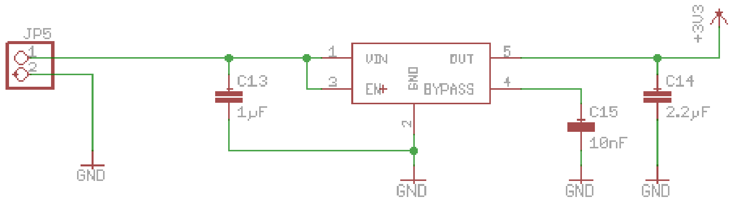


Figure 20. Modified schema

Of course we've also changed the layout (Figure 22).

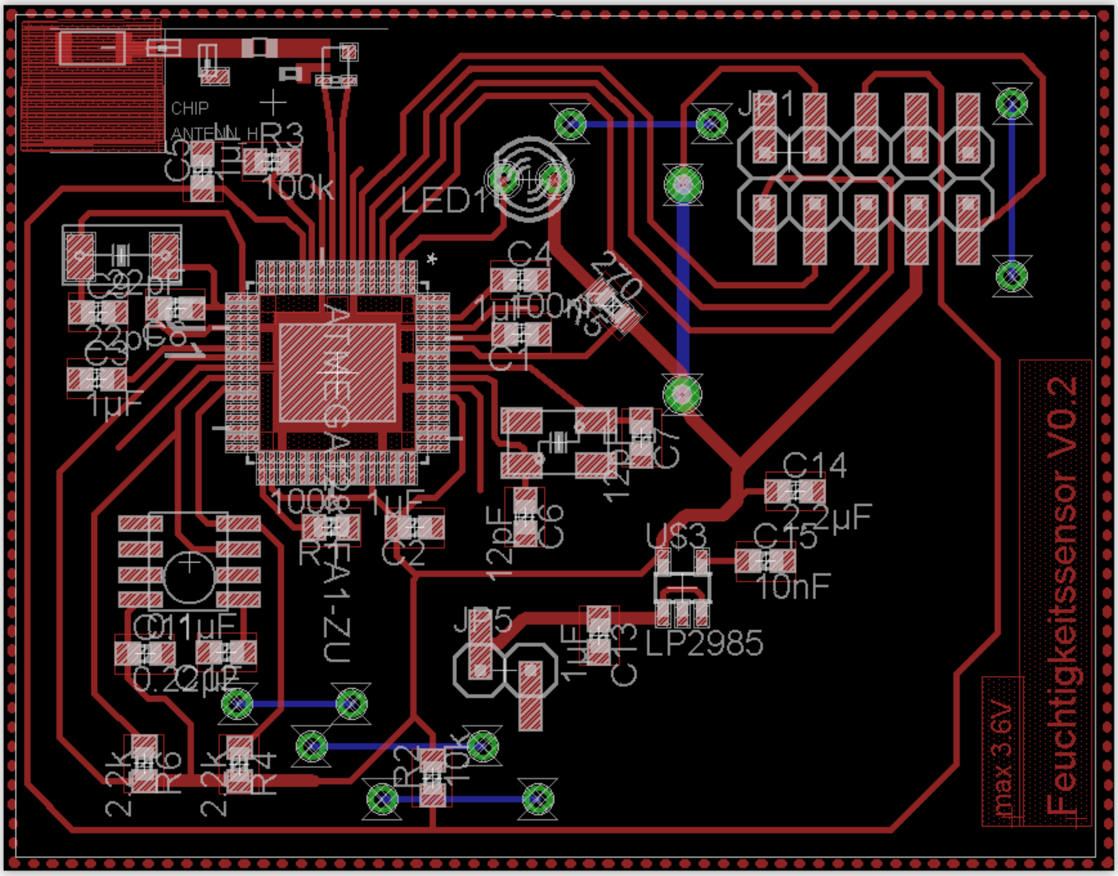


Figure 21. Modified layout

5. Conclusion

In conclusion, this has been a complete project where I could reinforce previously acquired knowledge in the University.

Beginning with the design and the problems it brings, making modifications and optimizations. I learned at every step and especially from the errors.

Also has been very rewarding knowledge retake c and c ++. In addition, working with the I2C standard which is used as a sensing element and digital devices on the market.

In addition to adapt a device to a network, previously installed, you have to follow some rules and standards that make you should arrange and plan the steps to take.

Ultimately it was a great opportunity to observe the work that is carried out in a laboratory and to work in a global project with several departments. Also, I liked to work on a current topic such as home automation low power consumption and on the ipv6 (6LoWPAN) protocol.

6. List of abbreviations

6LoWPAN	IPv6 over Low Power Wireless Personal Area Network
ACK	Acknowledgement Packet
ARM	Advanced RISC Machine
AVR	Advanced Virtual RISC
CAD	Computer-Aided Design
CMOS	Complementary Metal-Oxide
I2C	Interface to Communicate
IDP	Integrated Development Platform
IPv6	Internet Protocol version 6
ISM	Industrial, Scientific and Medical
JTAG	Join Test Action Group
JTD	JTAG Interface Disabled
MCU	Microcontroller Unit
MCUCR	MCU Control Register
MIPS	Mega Instructions Per Second
MR	Measurement Request
OS	Operating System
PCB	Process Control Block
RF	Radio Frequency
RF4CE	Radio Frequency for Consumer Electronics
RFC	Request For Comments
RFN	Radio Frequency Negative
RFP	Radio Frequency Positive
RH	Relative Humidity
RISC	Reduced Instruction Set Computer
SMD	Surface Mount Devices
TCK	Test ClocK
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
TWI	Two Wire Interface
WiFi	Wireless Fidelity

7. List of figures

8. References

[1] http://www.cadsoftusa.com/ (©2011 CadSoft. CadSoft Computer in the USA is a trading division of Newark Corporation, a Premier Farnell company).

[2] http://www.atmel.com/ (© 2014 Atmel Corporation).

[3] http://www.wireshark.org/ (Wireshark and the "fin" logo are registered trademarks of the Wireshark Foundation).

[4] ATmega128rfa1 Datasheet (© 2009 Atmel Corporation. All rights reserved).

[5] “High Frequency Ceramic Solutions” 2.45 GHz Antenna P/N 2450AT18A100 (2003 Johanson Technology, Inc. All Rights Reserved).

[6] “High Frequency Ceramic Solutions” 2.45 GHz Harmonic Filter-Balun Optimized for AT86RF230/231 and ATmega128RFA1 P/N 2450FB15L000 (2012 Johanson Technology, Inc. All Rights Reserved).

[7] http://senstools.gforge.inria.fr/doku.php?id=os:contiki

[8] Digital Humidity/Temperature Sensors Datasheet (Honeywell HumidIcon ™).

[9] I2C Communication with the Digital Humidity/Temperature Sensors Technical Note (Honeywell HumidIcon™)

[10] 150-mA Low-noise low-dropout regulator with shutdown (Copyright © 2004–2011, Texas Instruments Incorporated).

[11] EAGLE easily applicable graphical layout editor Tutorial Version 5 (Copyright © 2010 CadSoft Computer All Rights Reserved).