Smart Fridge

'Science Fiction' in the 'Smart Kitchen' - Sesam

Tom Mampaey
Embedded system developer

Bachelor Electronics – ICT
OAMK - University of Applied Sciences - Finland
Erasmus student – Artesis Plantijn Antwerp - Belgium

January – May 2015

Table of contents:

1. Introduction

Allow me to introduce myself. I'm Tom Mampaey, a third-year student from 'Artesis Plantijn Antwerp in Belgium. When I saw the opportunity to go on Erasmus to Oulu in Finland, to do my Bachelor's project there I did everything to make it possible. I worked on this project in cooperation with Bernd Verhofstadt. We are very thrilled about what we have accomplished.

2. The project

The project we worked on was a daughter-project of a bigger project (Smart Kitchen - Sesam). The premise was to put some intelligence into the ordinary kitchen, within a time frame of two years. We have had the privilege to be the first two students to contribute to this project.

As stated in the title, our share of the project related specifically to the smart refrigerator. Imagine the possibilities when your refrigerator knows more about you, your habits and your day-to-day products. You could find recipes for the specific ingredients in your refrigerator, or learn more about the healthy choices, and wouldn't it be great to have the contents of your fridge in your pocket when grocery shopping? This and more is what we envisioned for the kitchen of the future. The general idea was to make a refrigerator that will know who's using it, what he may or may not eat and even what's in the refrigerator at any given moment. The refrigerator will base its knowledge on the current properties of the user, such as weight and length, through User-input, Readers, a BMI-calculator, etc.

This project was set up with the purpose of having a realistic and practical end result.

3. Acknowledgements

Project managers:
Mr. Jussi Kangasoja
✉ Jussi.Kangasoja@oamk.fi

Ms. Kaisa Orajärvi
✉ Kaisa.orajarvi@oamk.fi

Hardware subvention:
Mr. Henry Hinkula
✉ *Henry.Hinkula@oamk.fi*

Belgian Lectors:
Mr. Jeroen Doggen
✉ *Jeroen.doggen@ap.be*

Mr. Tim Dams
✉ *Tim.dams@ap.be*

4. Github

To maintain the structure, workflow and to easily monitor the project we have made a repository on Github for this project. As requested by our project managers the Git is public for further enhancements in the future by other international students.
You can find the repository on:
https://github.com/OAMK-Smart-Kitchen/Smart-Fridge-OAMK

5. Personal information

Curriculum Vitae

Tom Mampaey

Electronics & IT

Ambitious person with an eye for perfection.
Can operate under stressful situations and always wants to learn more.
Likes to work on projects and in a responsible position.

EDUCATION

2015	OAMK UNIVERSITY OF APPLIED SCIENCES - OULU ERASMUS in Finland - Electronics ICT
2012 - 2015	ARTESIS PLANTIJN - ANTWERP Professional bachelor's degree - Electronics ICT
2010 - 2012	PROVINCIAL TECHNICAL SCHOOL - BOOM Electronics
2008 - 2010	PROVINCIAL TECHNICAL SCHOOL - BOOM Electro mechanics
2005 - 2008	PROVINCIAL TECHNICAL SCHOOL - BOOM Basic technical years

COURSES & CERTIFICATES

2015	International Seesam Project: Smart Fridge - Certificated
2015	English language test to participate Erasmus - C1
2012	Smart wireless lightcontrol for trailer lights - i-DEPOTED
2012	JINTRO-course: Group and independent education
2012	Driver license - B
2010	Driver license - A3
2010	VCA-proof: Safety checklist course - Certificated

JOB EXPERIENCE

2012 - July	BELGOCONTROL - Brussels Airport Internship for secondary school. Responsible for the systems and communication for all airplanes in Belgium. Instructive and interesting experience to be able to do my internship in such a secure and professional company.
2011 - 2015	ZAMPASS - Digital Crowd Managment - Job student Professional service for scanning and managing entrance tickets of big events and festivals with barcodes and RFID. Refs: Tomorrowland, Rock Werchter, TW Classic, ...
2010 - 2015	BERMACO BVBA & GTV BVBA - Job student Garden/Ground contructions. Planting forests and shearing hedges to constructing terraces and driveways.

SIDELINES & INTERESTS

- SanLuce Summer-Festival (coordinator: 2012 - 2014)
- Comenius projects (Spain, Italy, Germany)
- Scout leader - Totem: 'Persistent Markhor'
- Sports: Kayak polo, rock and wall climbing, Ultimate Frisbee

CONTACT INFORMATION

's Herenbaan 44a
2840 Rumst BE
25/03/1994
+32 49354 60 51
contact@mampaeytom.be

PROFESSIONAL SKILLS

PCB DESIGN
PIC - EMBEDDED
SMD
SENSORS
WIRELESS
3D- PRINTING

OFFICE 365
C, C#, C++
NETWORKING
JAVA/JAVASCRIPT

GIT
SCRUM

LANGUAGES

DUTCH
ENGLISH
FRENCH

6. The idea's and brainstorm

Because we only got a title, and a few start-ideas, of the project we were totally free implement the technologies and features in the fridge that we wanted. This gave us a lot of freedom, but also took a lot of time for research , brainstorming, achievable/serviceable technologies and thinking about what would be handy and what the user would like to have extra in his/her fridge at home.

6.1. User recognition:

6.1.1. Fingerprint reader in handheld:

PRO	CONTRA
Very unique (for each person)	You have to touch something
Reliability	Difficult with wet fingers
	Waiting-time

6.1.2. Active RFID key-hanger or card:

PRO	CONTRA
Detection on distance	Physical object
Multiple use and handy	Can mix up multiple users
Cheap	

6.1.3. Kinect face-recognition:

PRO	CONTRA
Challenging	Delay
Automatic recognition	Height of camera
International team-work	Complex

6.1.4. Face-recognition Raspberry-Pi:

PRO	CONTRA
Automatic recognition	Delay
Cheap	Height of camera
	Reliability

6.1.5. Voice-recognition:

PRO	CONTRA
Accessibility	Complex
Cheap	Reliability
Fun	

6.1.6. Motion-detection:

PRO	CONTRA
Energy-saving (only *on* when motion)	No direct user-recognition
Cheap	Reliability
Easy	

6.2. Product recognition and recommendation:

Barcode-reader:

PRO	CONTRA
Existing databases	Finding a fast way to scan it
	By hand
	Takes time

6.2.1. NFC-tag:

PRO	CONTRA
Wireless	Attach on the products/tray
Stickers are cheap	Reader on every place
Place-independent	

6.2.2. Mechanical detection/hatches:

PRO	CONTRA
Overview and organization	Moving parts
Reliability	Expensive
Not possible to take other product	

6.2.3. LED's/ LED-strip:

PRO	CONTRA
Indications of properties of products	User can take other products.
Colors: healthy/Unhealthy/reserved	
Fancy	

6.3.Health-detection:

6.3.1. Integrated scale:

PRO	CONTRA
User-assuredness	Big surface-area
Possibility to calculate BMI	View
No extra action required (for user)	

6.3.2. Personal point-system (based on earlier snacks):

PRO	CONTRA
Motivating for users	Requires many properties of user
Competitive	
Possibility to turn off feature	

6.3.3. BMI-calculator:

PRO	CONTRA
Easy calculation	Requires (many) properties of user
User can compare to average	Can make user depressed
Option to turn off	

6.4.Communication/Platform:

6.4.1. Raspberry Pi:

PRO	CONTRA
Performance	Limited hardware ports
All-in one computer	
Internet-connection	

6.4.2. Arduino:

PRO	CONTRA
Perfect test-device	No internet-connection
Many possibilities	Limited hardware-set
Cheap	

6.4.3. Wi-Fi and Ethernet (LAN):

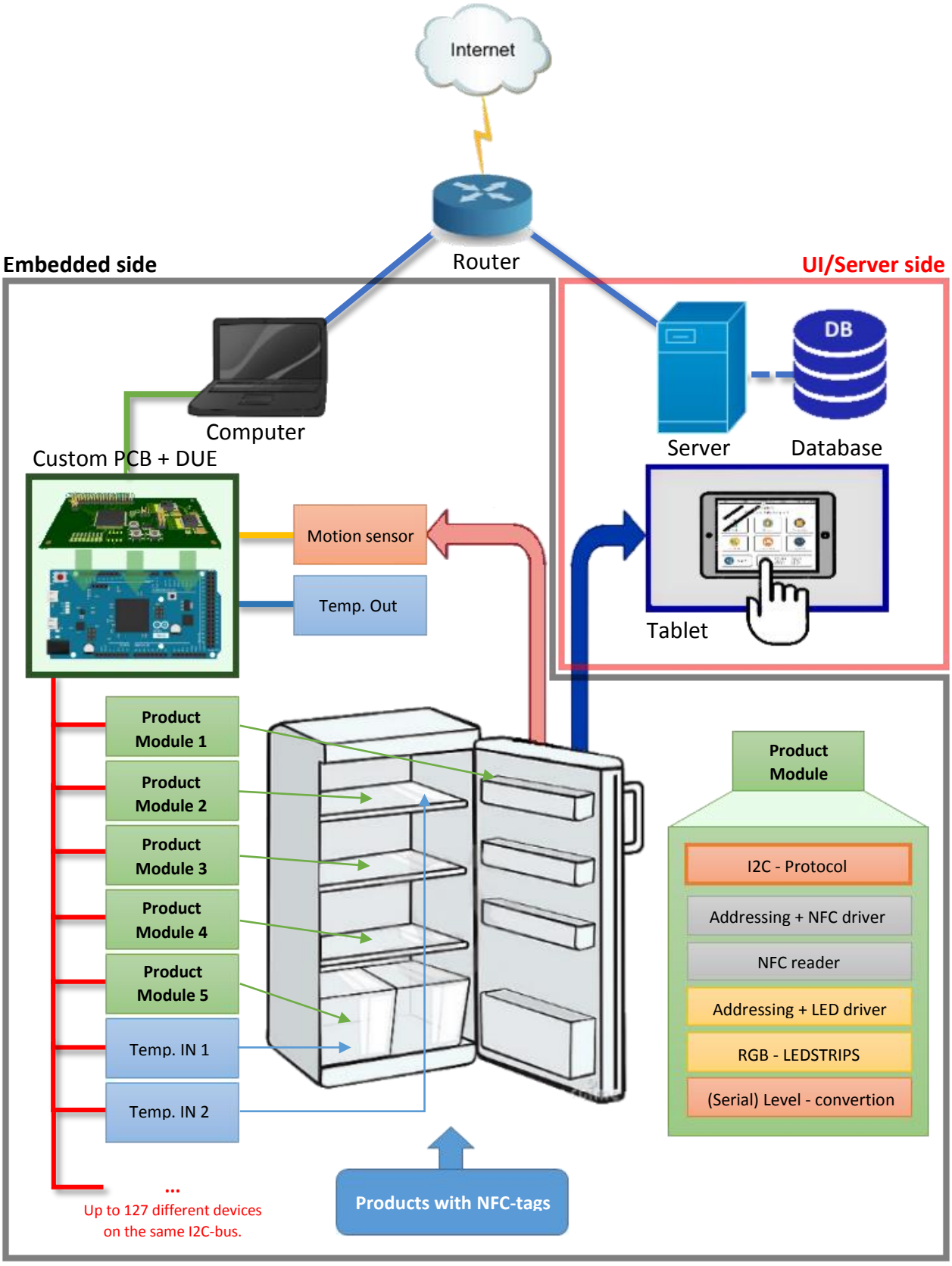
PRO	CONTRA
Connect different devices	Security
Future-oriented	
Monitoring	

6.4.4. Bluetooth:

PRO	CONTRA
Peer-to-peer (safer)	Not always reliable
Not connected to the internet	Possible delays
	Can only connect 2 devices at once

7. Conclusion in high-level schematic

After lots of research and discussing about what are the best technologies, and what is realistic in this amount of time we came up with this as the best starting idea:



8. Estimated Gantt-chart:

Subject	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Documentation:										
Concept report										
Gantt chart / Project planning										
Brainstorming										
Comparing possibilities										
Hardware:										
Research (target)										
Experimenting										
Features										
Collecting components										
Power supply										
Communication										
Arduino										
Raspberry Pi										
User recognition										
Testing										
Schematic design										
PCB desgin										
PCB manufacture										
Prototyping										
Error analysis										
Software:										
Research										
Choosing Language(s)										
User recognition										
Raspberry Pi										
Features										
Server side										
GUI										
Final:										
Final integration Software - Hardware										
Finishing hardware										
Final report										
Finishing touch Final Report										

Subject	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18	Week 19	Week 20
Documentation:										
Concept report										
Gantt chart / Project planning										
Brainstorming										
Comparing possibilities										
Hardware:										
Research (target)										
Experimenting										
Features										
Collecting components										
Power supply										
Communication										
Arduino										
Raspberry Pi test										
User recognition										
Testing										
Schematic design										
PCB desgin										
PCB manufacture										
Prototyping										
Error analysis										
Software:										
Research										
Choosing Language(s)										
User recognition										
Raspberry Pi										
Features										
Server side										
GUI										
Final:										
Final integration Software - Hardware										
Finishing hardware										
Final report										
Finishing touch Final Report										

9. Embedded systems

As the Embedded Systems developer I had to figure out which components, set up, technologies and systems were most appropriate for this application and for future extensions.

9.1.NFC-tags

After a great deal of research regarding the different possibilities, and hours of scouring the Internet for useful information I found some articles that managed to inspire me. Innovators have been proposing to replace the barcodes on products by NFC tags.
NFC tags offer increased efficiency and would make matters much more user friendly.
The more I read into the subject the more I was convinced of the potential of this approach.

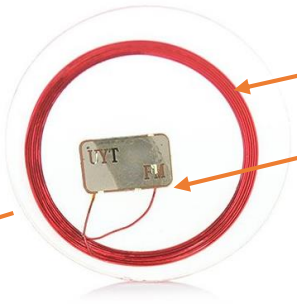
9.1.1. What is NFC and how does it work

NFC stands for **N**ear **F**ield **C**ommunication is an upcoming technology that already is widely in use by phones, tablets, laptops and custom devices. Over 50% of smartphones are already provided by such a reader/writer. The applications go from hitting a like or joining events in on Facebook/Twitter to advertisements, bus schedules and even paying by digital wallet.

It's also possible to write to an NFC tag yourself and set almost all options of your phone, just by swiping over the tag. Next to your bed; to put in mute and lower down brightness, in the doc station inside your car; to automatically open your GPS, log in into a secured Wi-Fi network without entering a password, on your bike; to start playing music, etc.
The options really are countless.

NFC technology evolved from radio-frequency identification (RFID) technology and has, like the name suggest, a short-range radio transmission. An NFC tag has no battery or what so ever. The tag consists of a thin coil, which operates as an antenna, and a small chip in the center that can store data (between 48 and 4094 bytes) that can become really powerful quite fast by writing for example a specific link to this tag.
The tag is powered by the electromagnetic field of the reader/writer by radiating the coil of the tag. This is a very low-power, compact (max 10 cm) and harmless radiation which also adds extra security. The tag can be set up for one- or two-way communications.

Because such stickers, cards, keychains, etc. are getting cheaper every day the use of NFC is getting larger and larger. This is only the beginning.
In the memory all properties of products can be stored (expiration date, calories …)



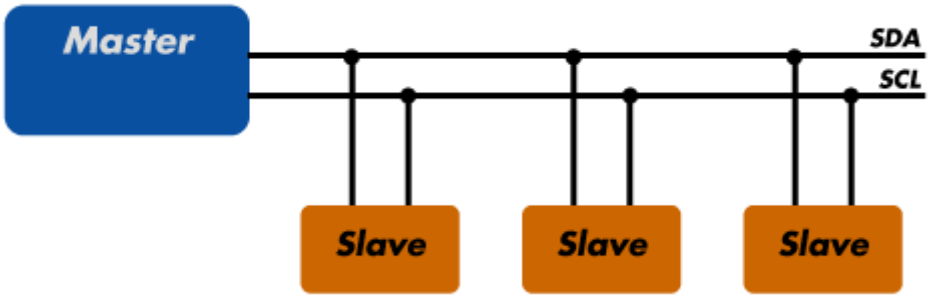
9.2.I2C Serial Bus-System

Because I'm working with multiple smart objects connected to each other at different places I had to find way to maintain explicit control and overview of the complete system. It soon became clear I was going to use a serial bus system. In this way I have a clear control (by address) over every object in the serial circuit consisting of a minimum of connection wires. I also considered SPI (Serial Peripheral Interface) but I preferred to use the I²C serial bus interface.

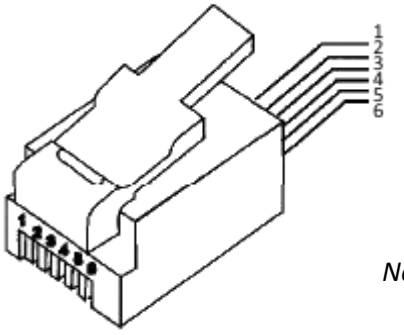
SPI	I2C
High speed communication	Low cost
Uses 4 signal lines (and one for each slave)	Only two signal lines (SDA and SCL)
No control if something is connected	Compatible with different manufacturers
More sensitive to noise	One broken IC in circuit can block serial line

9.2.1. I2C protocol

With the I2C protocol I can control all the devices in the network by using a master and multiple slave modules. The communication and acknowledgment will be by a specific (hard coded) address and over only two wires the Serial Data Line (SDA and the Serial Clock Line (SCL).



To connect them to each other I used RJ-11 connectors and six wired flat cable.



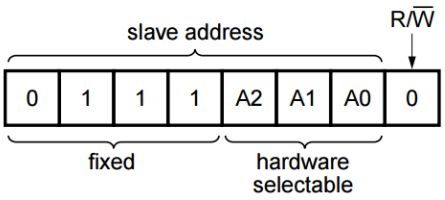
1. 12V power line towards slave modules
2. GND
3. Power support line
4. RX line for receiving NFC-ID's
5. Serial Clock Line (SCL)
6. Serial Data Line (SDL)

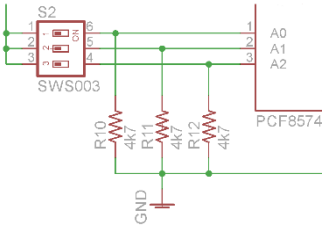
Note: The connection cable towards the sending slave has to be a crossover. Connection cables between the readers may not be longer than 1m.

To communicate and control each module independently, addresses are used.
These can be varied depending on the manufacture. After lots of research, testing and ordering different samples I came out using the PCF8574 8-bit I/O expander.
Of course IC's of different manufactures can communicate together over the same protocol.

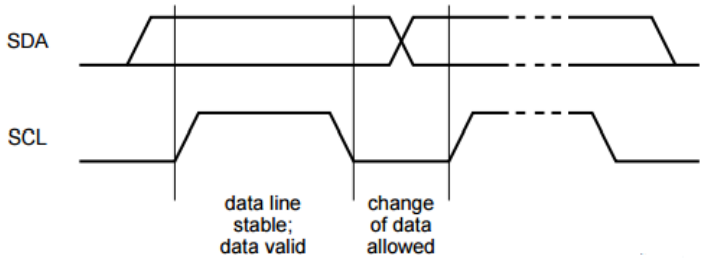
Up to 127 devices can be connected on the same serial line.
That's even more than the amount of unique products that fit in your fridge.

The addressing for the PCF 8574 8-bit I/O expander is as follows:

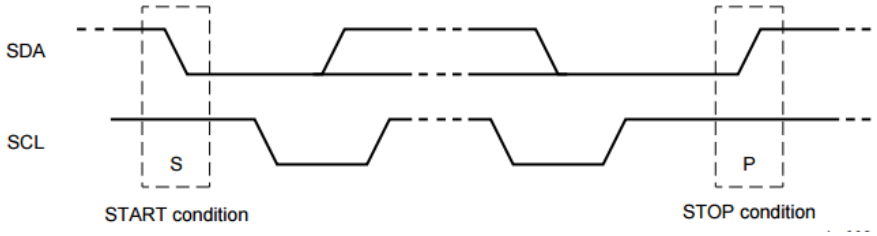




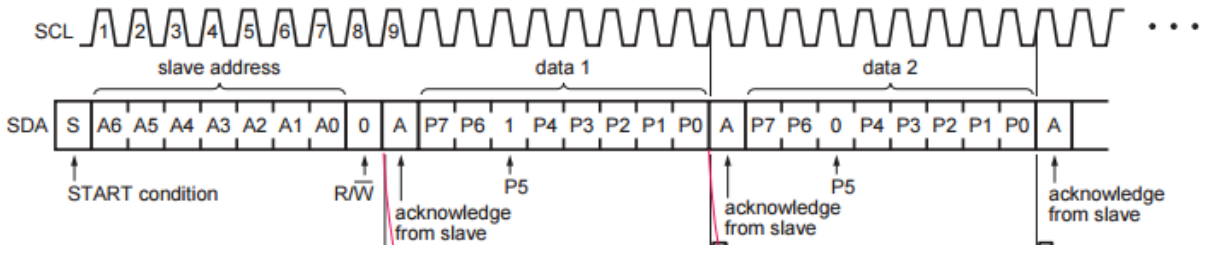
Like you can see, the first four digits of the address are already fixed. This means that our first address in coding (for the PCF8574) has to start with 56 (0111000). I set the address of the IC/module using dipswitches. Hereby I can set the address of this IC from 56 (0111000) up to address 63 (0111111). The additional digit in the header is to set if you want to read (set digit high) or write (set digit low) to the 8-bit I/O expander. Like you can see on the flow below one data bit is transferred for each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse in order to be valid.



A HIGH-to-LOW transition of the data line (SDA) while the clock signal (SCL) is HIGH indicates a START condition, and a LOW-to-HIGH transition of the SDA while SCL is HIGH defines a STOP condition.



If we want to send data to the slave (e.g. change the indication color of the product) A START commando is send on the serial line followed by the desired address of the reader and the additional read/write digit. If the slave readers responds with an ack. data can be transmitted until the master sends the STOP command (LOW-to-HIGH transition on SDA while SCL is HIGH).



Sometimes the length of the header including the address in the code is only 7 bits. The additional $R/\overline{W}$ digit can be dropped off because some libraries can already detect/command if the master want's to write or read to/from the slave module.

I never had such detailed experience with this protocol before and I can say that it took me a while to master it completely, finding the correct IC's and set up a running and stable system.

The beautiful thing about this is that you can now add any device you want as a slave on the bus system. This makes it easy to implement other extensions in the future. Instead of NFC readers you can also add a temperature module, calorimeters, product alarms, locks etc. you name it, and this all on the same stable serial connection.

9.3.Tristate serial switch

One of the components on the slave module that I actuate from the master is a tristate serial switch or Single bus buffer gate/line driver (74HC2G125). This little five legged component is smaller than 3mm and is located in the centre of the slave boards. The tri state means that it's able to switch serial communication on and off without losing any values. (+ 0 -).

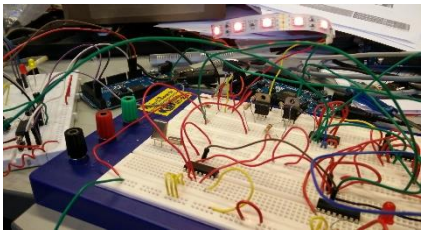
This switch is responsible to op and close the data line of the NFC readers on demand. It's important that only one of the switches on the bus is open at the same time.

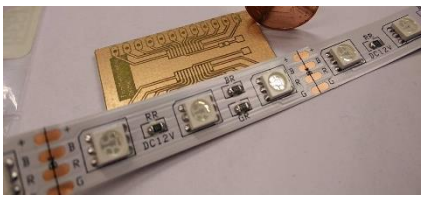
9.4.LED indication

The other components that the master can control are LED drivers. By colour indication the user has a view over his/her fridge in an eye blink. Which product is healthy or not by a red or green colour, which products may be reserved by someone else by a purple or customized colour (forget 'Post-its'), even visualizing the read process at a demonstration.

Actually you can do anything with the indication of these LEDS, and it has a very fancy look too. The master can allocate seven different colours towards the slave modules which can have a million of meanings and purposes (red, green, yellow, blue, bright blue, purple and white).

To bring the prototype to the max I designed a custom component for the LED strips so that they are powered on board instead of connected with wires that are able to have bad connections. I also did custom designs for a.o. I2C Bus interfaces and temperature sensors. It may look quite straightforward to colour RGB's over a serial communication, but I can ensure you it is not always that easy.





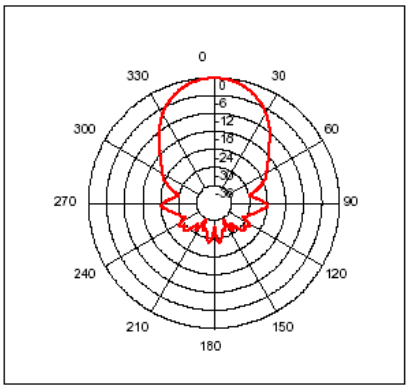
9.5. External sensors

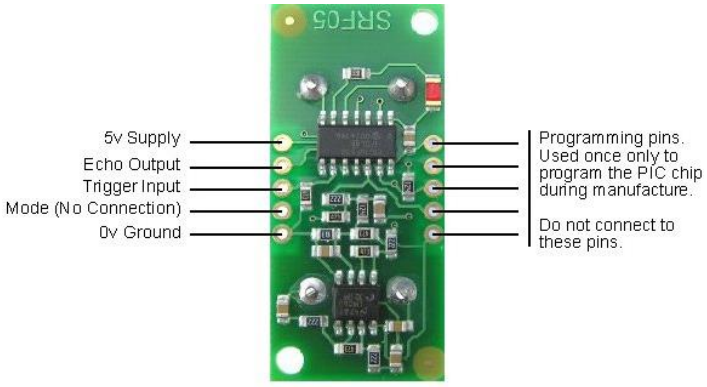
To add extra features to the Smart Fridge a few external components communicate with the Master module.

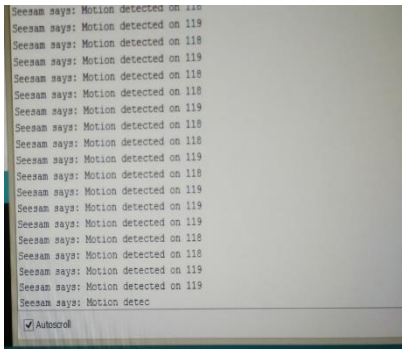
9.5.1. Ultrasonic User-detection

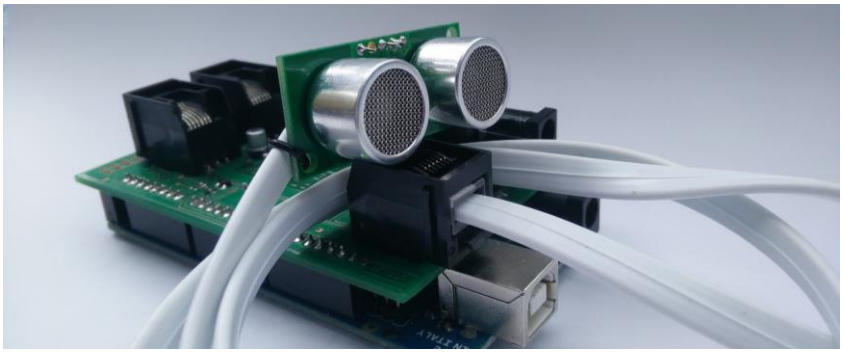
Wouldn't it be nice if your fridge detects if there is a user in front of it, and handle different functions based on that data? For example the fridge can start talking to the person or put the system out or in sleep whether a user is detected or not. This is better for the durability of the system and energy saving for the customer.

All the data is send to the master module so anything is possible as a reaction on the detection of a user in the kitchen/in front of the fridge.
To detect the user I use ultrasonic waves send and received by the SRF05 module:









9.5.2. Temperature sensors

A pretty obvious sensor is the temperature sensor. On the master board I use the LM335 to do this measurements. Like mentioned before it is now perfectly possible to add temperature sensors on the I2C bus too. You can for example put three temperature sensors inside your fridge (main, small freezer, white wines) and one outside the fridge.

I recommend to use one of the following because they have an integrated I2C interface.

- LM75A
- SE95 → SE97
- NE1617A
- SA56004

10. The Slave modules

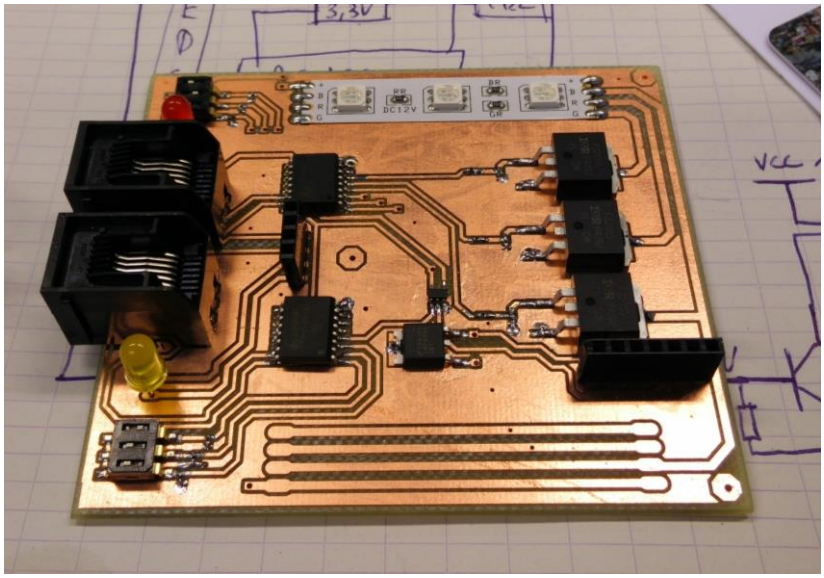
First of all, the primary challenge was to accomplish the slave boards. These will detect and indicate the products with RGB-strips as well as read and send them to the master.

The slave boards are capable to:

- Communicate with the master module by I2C-protocol
- Adopt the data of NFC-tag and send it over serial line
- Give visual indication when reading
- Close the sending port when not reading
- Drive RGB LED by demand of master module
- Approachable by an address for NFC-reading
- Approachable by an address for LED-indication
- Be accurate and failproof

10.1. The testing board

After countless hours of coding and experiencing a few ups and downs I made the first testing board. This board was the first step in the direction of a working prototype.

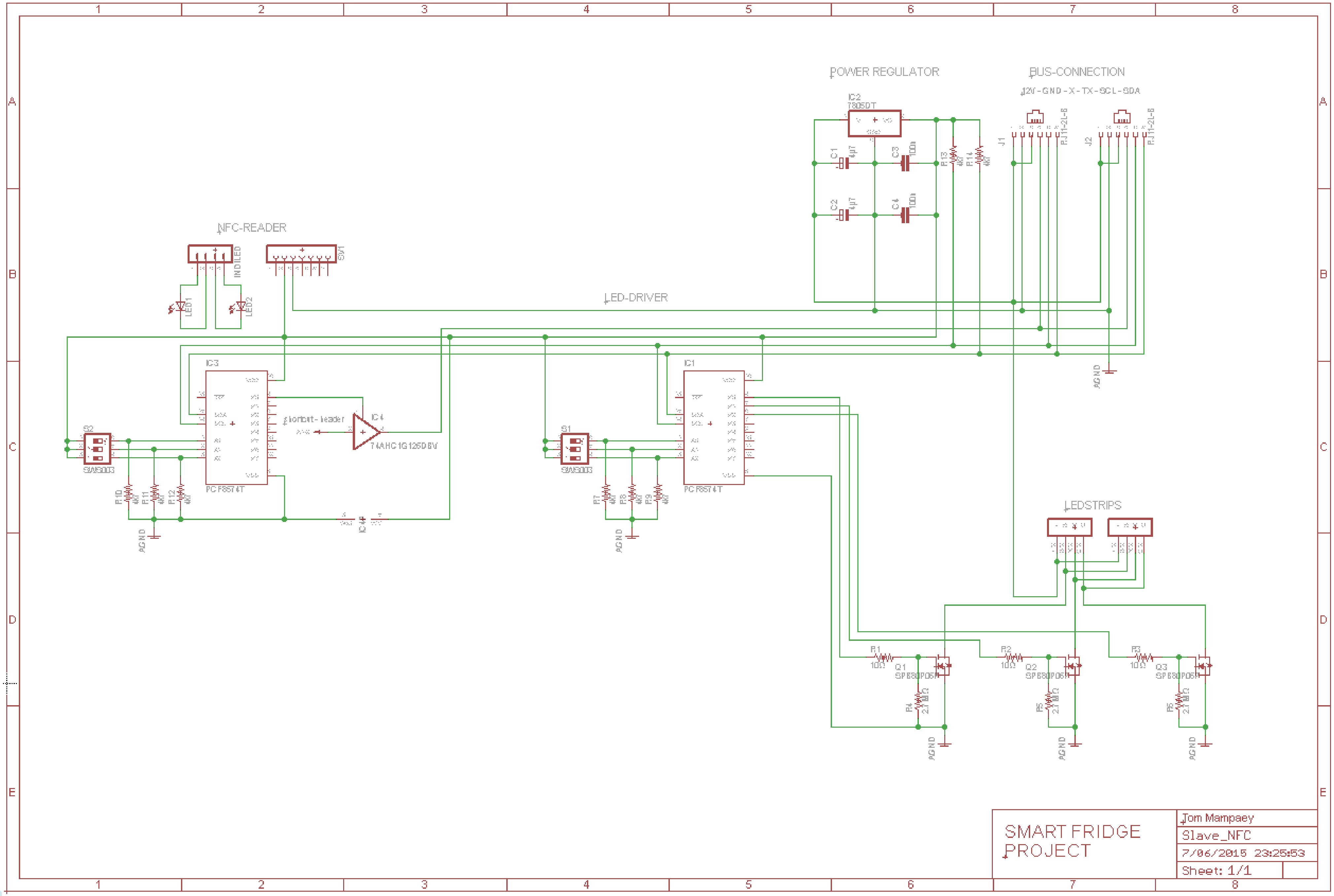


It's always a good idea to make test board like this one, i.e. your components are already in the right place, faults can be measured, you suffer no interference or bad connections resulting from breadboards etc. Thanks to this board I caught a lot of initial problems and interferences that I hadn't thought of before.

When this board was fully assembled and finally without interferences and burning components I could start optimizing the board and schematic to start coding again.

10.2. Final schematic

➔ Unfold next page (A3-size)


POWER REGULATOR
IC2
7805DT
C1
C2
C3
C4
R13
R14
BUS-CONNECTION
12V-GND-X-TX-SCL-SDA
J1
J2
RJ11-2L-6
NFC-READER
LED1
LED2
LED-DRIVER
IC3
IC1
IC4
74AHC1G125DBV
PCF8574T
S1
S2
R10
R11
R12
R7
R8
R9
AGND
LEDSTRIPS
R1
R2
R3
R4
R5
R6
Q1
Q2
Q3
SMART FRIDGE PROJECT
Tom Mampaey
Slave_NFC
7/06/2015 23:25:53
Sheet: 1/1

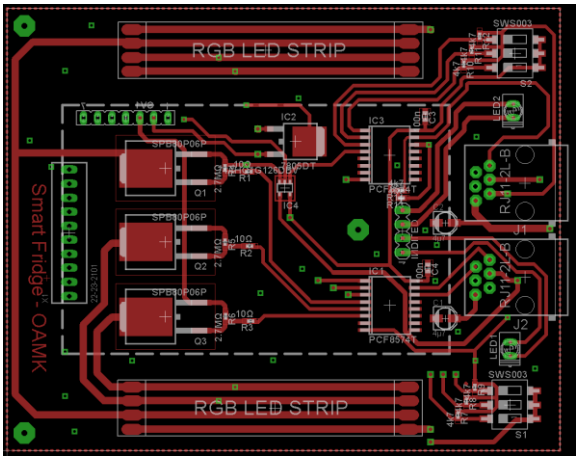
10.3. PCB design

When I had triple checked the schematic, components and the connections between them, it was time to design the final PCB board. To put the performance to the maximum and interferences to the minimum it's important to:

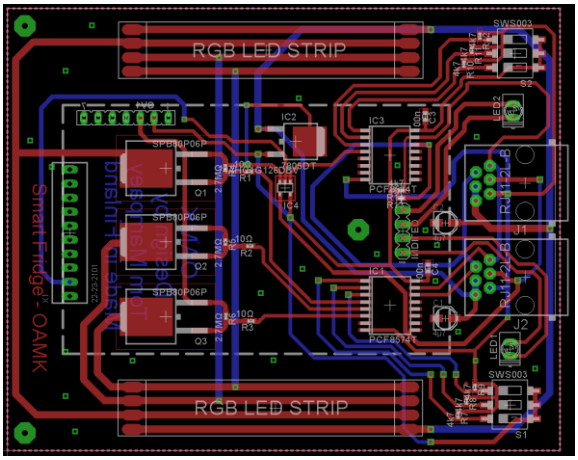
- Place the components on very strategic places
- Make the ground plane as large as possible
- Put the filter capacitors as near as possible to the supply lines
- Use large pads and diagonal wire, bent for optimum connection
- Never put radiating/electrostatic/sensitive/high current components together

10.3.1. The layers

The striped line is where the NFC board will be connected.

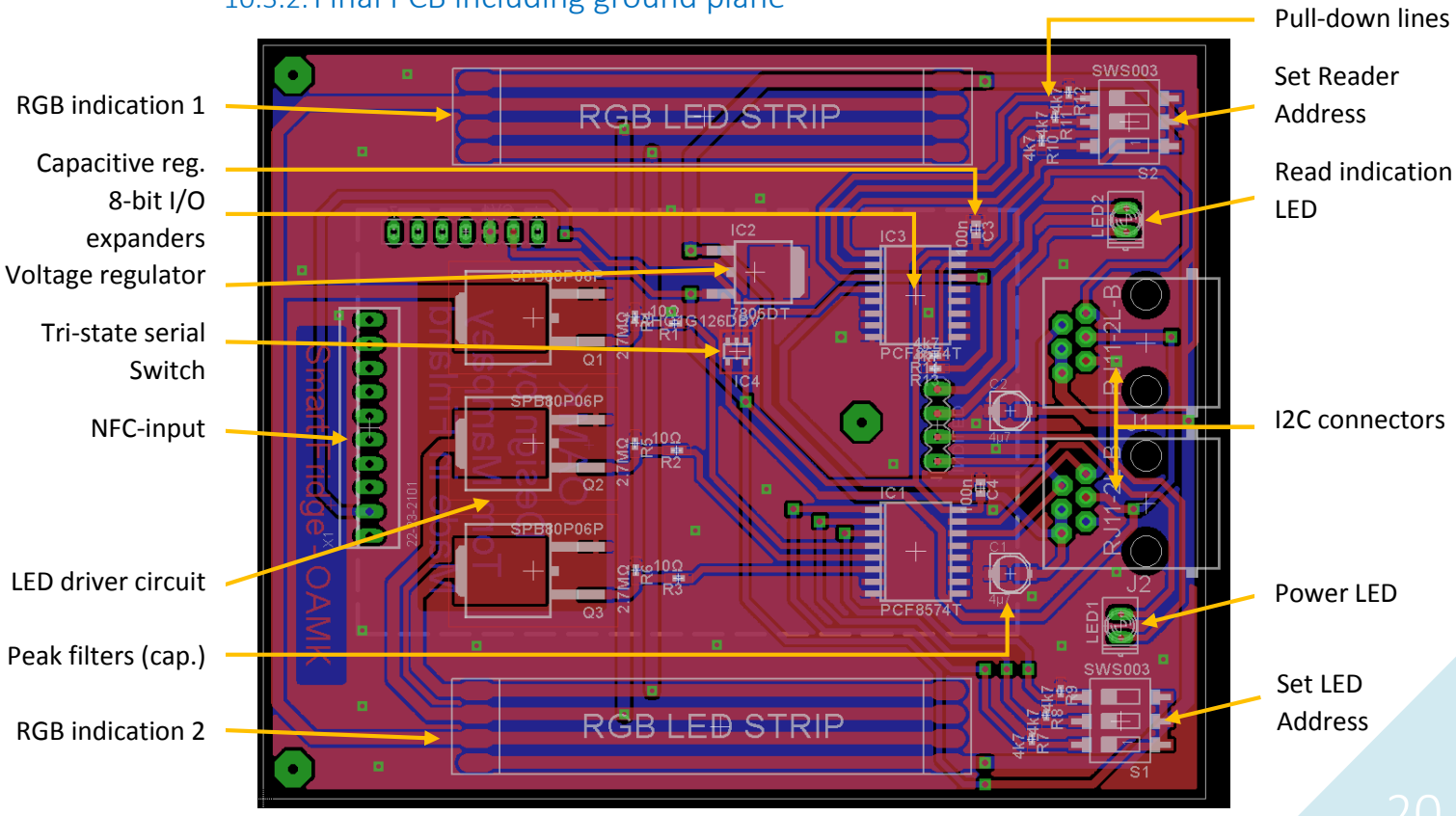


Main (/top) layer



Added second (/bottom) layer

10.3.2. Final PCB including ground plane



11. The Master module

The master module is actually the brain of the smart fridge and controls the whole system. It communicates with all the connected devices, NFC readers, sensors, etc. After the necessary calculations/translations it processes the data and sends it to the output port. The master board is based on an Arduino Mega/Due and operates as a shield.

The master board can:

- Open the serial TX-line by commanding the slave module by (serial switch) address
- Read and translate the incoming date of the slave modules on the RX line
- Control the seven LED colors of every individual module by (LED driver) address
- Control up to 127 devices connected to the I2C bus
- Translate and process the data and send it in a header to the output port
- Convert the voltage level on the serial line from 5 to 3.3 when in use with DUE
- Filter interferences and power the whole (control) system.
- Give you the option to switch the serial lines of the output port by a jumper
- Read the temperature outside of the fridge
- Read the distance of a user in front of the fridge and interact by it.

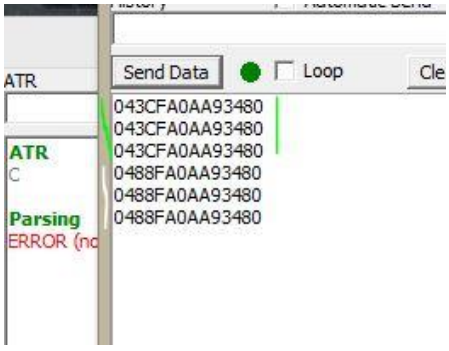
11.1. Algorithms

Because the datasheet of the reader module is very limited I had to figure out by myself how to get values out of the tag and had to find an appropriate and stable algorithm that would convert the data coming from the NFC reader to an ID. This was definitely one of the most time consuming challenges.

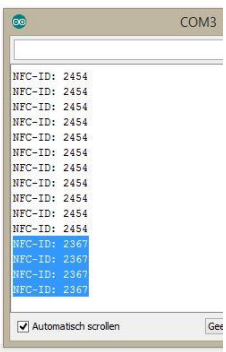
11.1.1. Older versions

Before finally establishing a sufficient algorithm, I had already tried countless other algorithms. I had tested all of the attempts from filtering on the brand of the tag (i.e. the first chars) to reading the first 40 chars on tag detection.

At times the incoming data was completely random, other times the data only encompassed two chars. In the spirit of trial & error, every time I believed I had found a successful algorithm, I caught a flaw that forced me to rethink and start over. Common problems were for example baud rates, serial level, output-pin, etc. Many commits on GIT can confirm this.

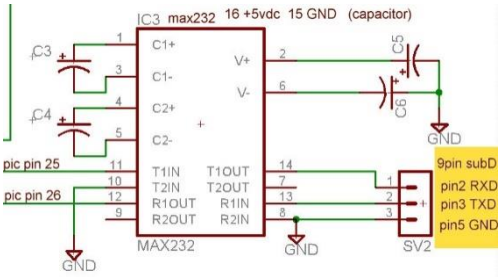


```
int incomingByte = 0;   // for incoming serial data
int tempValue = 0;

void setup() {
        Serial.begin(9600);

}

void loop() {

        if (Serial.available() > 0) {
                incomingByte = Serial.read();
                if (incomingByte == 0)
                {
                Serial.print("NFC-ID: ");
                Serial.println(tempValue);
                tempValue = 0;
                }
                else
                {
                  tempValue = incomingByte + tempValue;
                }
        }
```



11.1.1. Serial level conversion

The foremost reason I had to change the algorithms so often was the unstable output on the RS232 pin of the NFC-reader. It became clear that I had to convert the serial level in order to not fry the fragile components with uart voltage peaks of (+~-) 10V.
The solution would be to put in a MAX231 circuit to convert the level:



When I was making this circuit on a breadboard I started to study, disassemble and measure the NFC board on component level. It became clear that there was already a Chinese replica of the famous MAX232 on the reader itself. After finding the correct pins I could drain the data to the perfect level.

11.1.2. Final algorithm

All in all the previous algorithms were very unstable. Then I started thinking outside of the box. I was always looking for something I could filter the data on, while I actually had to find a way to select the repetitive string and length independent. This can be different by brand.
The (**very simplified**) idea goes as follows:
e.g. NFC-ID = '123456'
Incoming data on RX = 123456123456123456123456123456123456123456123456

The main idea was to compare box A and box B until they were the same. If they weren't, an extra character would be added to the boxes. In the Smart Fridge I started the boxes at a length of 6 to compare to each other, because the ID of an NFC-tag will never be shorter than six.
I can compare the places by putting them in a buffer array of chars.

IN: *123456123456123456123456123456123456123456123456*

Box[2]: *12 34 56 12 34 56 12 34 56 12 34 56 12 34 56 12 34 56 12 34 56*

Box[3]: *123 456 123 456 123 456 123 456 123 456 123 456 123 456*

Box[4]: *1234 5612 3456 1234 5612 3456 1234 5612 3456 1234 5612 3456*

Box[5]: *12345 61234 56123 45612 34561 23456 12345 61234 56123 456*

Box[6]: *123456 123456 123456 123456 123456 123456 123456 123456*

Box[…] → Until Box A == Box B
After converting the idea into code (see 11.2.) this algorithm seemed to work very well and stable. All the ID's were processed and saved correctly. This really was a welcome breakthrough.

11.2. Code: Mater module

```
/*

Master Module of Smart Fridge
-----------------------------
Made by Tom Mampaey
at 'OAMK University of Applied sciences'
Finland - May 2015

The mastermodule controls all the connected slavemodules by I2C-
protocol on the two RJ-11 connectors (on right).
It also has a motionsensor port (on left), a temraturesensor and a output p
ort to Wi-Fi (underneath).
The mastermodule is powerd by an 12V 1,5A DC adapter

This code controls all the serial switches on the NFC-
readers and converts this data to ID's.
Seven diffrent colors can be dedicated to the RGB-
strips of the selected Slave Module.
The converted data will be set in a header before sending over the serial l
ine of the output-port.
*/

// Module adresses:
#define  MOD1  56
#define  MOD2  57
#define  MOD3  58
#define  MOD4  59
#define  MOD5  60
#define  MOD1_R  61
#define  MOD2_R  62
#define  MOD3_R  63
#define  MOD4_R  64
#define  MOD5_R  65

// Colors:
#define Off B00000000
#define Read B00000001
#define Green B00000001
#define Red B00000010
#define Yellow B00000011
#define Blue B00000100
#define BrightBlue B00000101
#define Purple B11111110
#define White B11111111

#include <Wire.h>

// NFC
int incomingByte = 0;
char charBuf[100];
char tempcharBuf[100];
String incomingString = "";
String tempString = "";
boolean ReadNFC = true;
int count = 100;
String productID = "0000";
int ReadDelay = 1000;

int Temprature = 20; // test value
boolean Available = true;
```

```
// UserDetection
int duration;                 // Stores duration of pulse in
int distance;                 // Stores distance
int sensorpin = 7;
int UserDistance = 30;        // in cm

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  ModuleWrite(Off, MOD1);
  ModuleWrite(Off, MOD2);
  ModuleWrite(Off, MOD3);
  ModuleWrite(Off, MOD4);
  ModuleWrite(Off, MOD5);

  ModuleWrite(Off, MOD1_R);
  ModuleWrite(Off, MOD2_R);
  ModuleWrite(Off, MOD3_R);
  ModuleWrite(Off, MOD4_R);
  ModuleWrite(Off, MOD5_R);
}


void loop()
{
  //VisualRGBCheck();
// Visual knight rider of 7 colors for visual check of connection

  // if (DetectUser()) {}
// Can be interactive, from chaning colors to sending to server (works)

  for (int i = MOD1_R ; i <= MOD3_R ; i++)// Read all the readers.
  {
    SetMod(i);                            // Select NFC-reader
    readID();
    if (productID != "0000")
    {
      ModuleWrite(Green, i - 5);          // Product detected
      Available = true;
    }
    else
    {
      ModuleWrite(Red, i - 5);            // No product
      Available = false;
    }
    SendToWifi(productID, i, Available, Temprature);

    productID = "0000";
    delay(ReadDelay);
  }
}


// Sends assigned data (color or state of serial switch) to specific Module
.
void ModuleWrite(byte txData, int Module_Address)
{
  Wire.beginTransmission(Module_Address);
  Wire.write(txData);
  Wire.endTransmission();
}
```

```
// Reads the data and translates it to an ID
void readID()
{
  // ----- NFC-Tag Detection -----
  int LengthCharArr = sizeof(charBuf);

  for (int i = 0; i <= LengthCharArr; i++)
  // Put incomingbytes in a string of 100 chars
  {
    if (Serial.available() > 0) {
      incomingByte = Serial.read();
      tempString = String(incomingByte);
      incomingString += tempString;
    }
  }

  incomingString.toCharArray(tempcharBuf, LengthCharArr);
  // Put incomingString (size 100) in char array (100)
  incomingString = "";

  boolean StartShifting = false;
  int index = 0;
  for (int i = 0; i <= LengthCharArr; i++)
  {
    if (tempcharBuf[i] == '4' && tempcharBuf[i + 1] == '8' && tempcharBuf[i
       + 2] == '5' && tempcharBuf[i + 3] == '2' && (i + 3) < LengthCharArr)
     // Shifting array to brand, can be removed to read other tags too
    {
      StartShifting = true;
    }
    if (StartShifting)
    {
      charBuf[index] = tempcharBuf[i];
      index++;
    }
  }

  for (int i = 6; i <= LengthCharArr; i++)
  // Divide char array in 'boxes' (Starting at 6 because ID never shorter)
  {
    String boxA = "";
    String boxB = "";

    // Box A
    for (int j = 0; j < i; j++)
// Make first box bigger every round by adding first chars starting from
char 0 -> 6
    {
      boxA += charBuf[j];
    // Add the char at location J (Start: 0 -> 6 and increments each round)
    }
    // Box B
    for (int j = 0; j < i; j++)
 // Make second box bigger every round by adding the chars starting from
char 6 -> 12
    {
      if (j + i < LengthCharArr)
    // As long as the locations are smaller than the buffer size
      {
        boxB += charBuf[j + i];
// Add the char at location i + j (starting after chars from box A) 6 -> 12
      }
    }
```

```
    // Comparing the boxes
    if (boxA == boxB && boxA != "")
    {
      productID = boxA;    // Final productID
      for ( int i = 0; i < LengthCharArr;  ++i )  // Clear buffers of chars
      {
        charBuf[i] = (char)0;
        tempcharBuf[i] = (char)0;
      }
    }
  }
}


// Puts variables in headers on serial line to send out the output port
void SendToWifi(String iD, int location, boolean Available, int temprature)
{
  char charArray[50];
  String tempStr = "XD" + iD + "DA" + (String)location + "AB" +
(String)Available + "BT" + (String)temprature + "TX";

  tempStr.toCharArray(charArray, 50);
  Serial.write(charArray);
}


// Visual knight rider of 7 colors for visual check of connection
void VisualRGBCheck()
{
  int timer = 200;
  for (int count = 0; count < 3; count++) {
    ModuleWrite(Green, MOD1 + count);
    delay(timer);
    ModuleWrite(Red, MOD1 + count);
    delay(timer);
    ModuleWrite(Yellow, MOD1 + count);
    delay(timer);
    ModuleWrite(Blue, MOD1 + count);
    delay(timer);
    ModuleWrite(BrightBlue, MOD1 + count);
    delay(timer);
    ModuleWrite(Purple, MOD1 + count);
    delay(timer);
    ModuleWrite(Green, MOD1 + count);
    delay(timer);
  }
  for (int i = 0; i < 3; i++) {
    for (int count = 0; count < 3; count++) {
      ModuleWrite(Off, 56 + count);
    }
    delay(timer);
    for (int count = 0; count < 3; count++) {
      ModuleWrite(White, 56 + count);
    }
    delay(timer);
  }
}
```

```
// Detects users in front of fridge, gives countless options to implement
boolean DetectUser()
{
  pinMode(sensorpin, OUTPUT);
  digitalWrite(sensorpin, LOW);                          // Make pin low
before sending a short high to trigger ranging
  delayMicroseconds(2);
  digitalWrite(sensorpin, HIGH);                         // Sends short 10
microsecond high on pin to start ranging
  delayMicroseconds(10);
  digitalWrite(sensorpin, LOW);                          // Set pin low
again before waiting for pulse back in
  pinMode(sensorpin, INPUT);
  duration = pulseIn(sensorpin, HIGH);                   // Reads echo
pulse in from SRF05 in microseconds
  distance = duration / 58;                              // /58 is distance
in cm
  Serial.print("Seesam says: ");
  if ( distance < UserDistance)
  {
    Serial.println("USER DETECTED IN FRONT OF FRIDGE!");
    return true;
  }
  else
  {
    Serial.print("Motion detected on ");
    Serial.println(distance);
    return false;
  }
  delay(500);
}

// Configures modules correctly to make sure only that module is reading!
void SetMod(int SelectedMod)
{
  switch (SelectedMod) {
    case 61:
      ModuleWrite(Purple, MOD1);
      ModuleWrite(White, MOD2);
      ModuleWrite(White, MOD3);
      ModuleWrite(Off, MOD1_R);
      ModuleWrite(Read, MOD2_R);
      ModuleWrite(Off, MOD3_R);
      break;
    case MOD2_R:
      ModuleWrite(White, MOD1);
      ModuleWrite(Purple, MOD2);
      ModuleWrite(White, MOD3);
      ModuleWrite(Off, MOD1_R);
      ModuleWrite(Off, MOD2_R);
      ModuleWrite(Read, MOD3_R);
      break;

    case MOD3_R:
      ModuleWrite(White, MOD1);
      ModuleWrite(White, MOD2);
      ModuleWrite(Purple, MOD3);
      ModuleWrite(Read, MOD1_R);
      ModuleWrite(Off, MOD2_R);
      ModuleWrite(Off, MOD3_R);
      break;
  }
}
```

11.3. Compose header to send

When all the required data was collected by the master it was ready to send to the output port. In order to make sure that the data is received correctly by the Wi-Fi slave I inserted the data in a sort of a header.

I did this in the following function with declared parameters. Respectively: The NFC-ID of the product, the location of the product in the fridge, the availability of the product and the in- or outside the refrigerator.

```
SendToWifi(String iD, int location, boolean Available, int temprature)
```

The headers will be sent continuously to the output port so that a change in the header is immediately detected. I sent the headers of the different slave modules in succession to the Wi-Fi slave, with the following alignment:

XD4875242556256255935332DA56AB1BT3TX*XD4875242556256255948567DA57AB0BT2TX...*

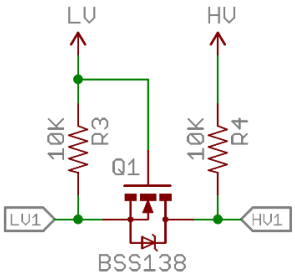
Every header is mutually parted by two **X**'s. Between these two the content of the header is constructed. Between the:
D = NFC-ID of product
A = The address/location of the product in the refrigerator
B = The availability of the product (if it is still there)
T = The temperature of the fridge

11.4. Serial level conversion

Because the master module has to deal with heavy calculations /algorithm and has to be able to control up to 127 devices on the serial I2C protocol with an acceptable delay, the use of an Arduino DUE instead of the MEGA recommended.

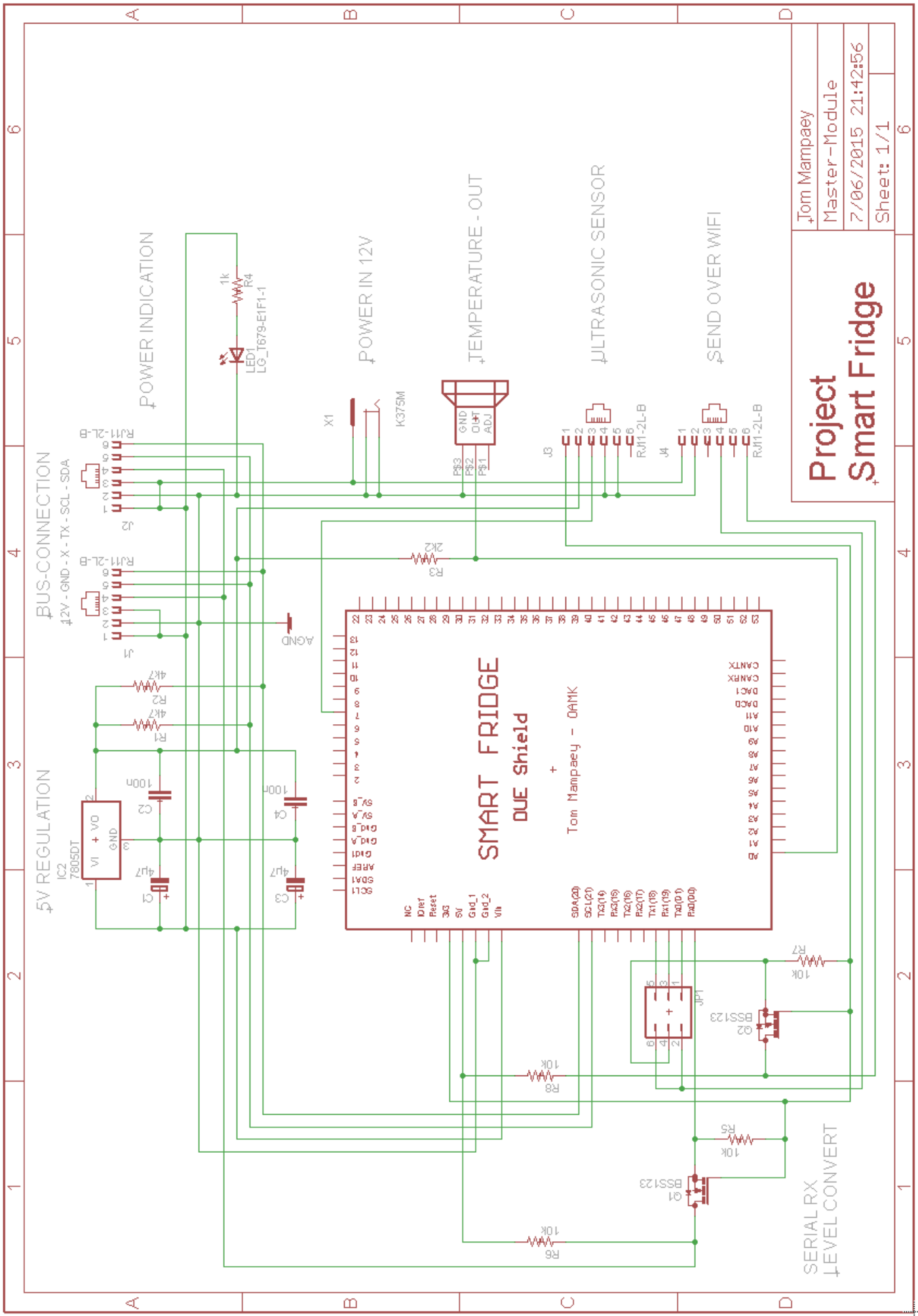
	Arduino MEGA	Arduino DUE
System voltage	5V	3.3V
Clock speed	16MHz	84MHz
Programming interface	USB (ATMega16US)	USB Native

To convert the serial level of the data from 5V level to 3V3 I constructed the following circuit. These are used on the serial RX lines towards the DUE and coming from the Wi-Fi slave. (D1;D2)



11.5. Schematic

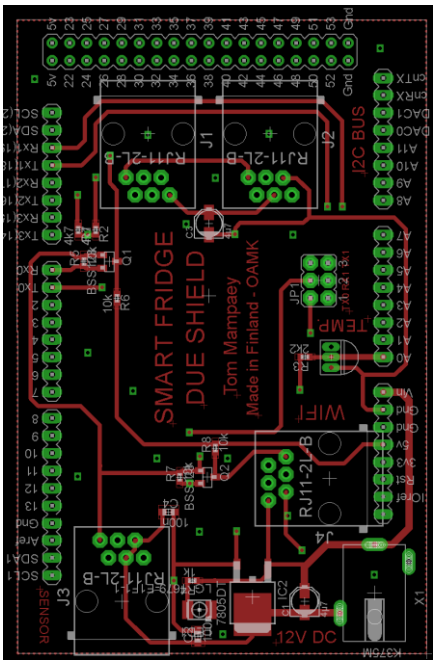
➔ See next page (full page view)


Project
Smart Fridge
Tom Mampaey
Master-Module
7/06/2015 21:42:56
Sheet: 1/1
POWER INDICATION
POWER IN 12V
TEMPERATURE - OUT
ULTRASONIC SENSOR
SEND OVER WIFI
BUS-CONNECTION
12V - GND - X - TX - SCL - SDA
5V REGULATION
SERIAL RX
LEVEL CONVERT
SMART FRIDGE
DUE Shield
Tom Mampaey - OAMK
LED1
LG_T679-E1F1-1
R4
1k
X1
K375M
GND
OUT
ADJ
J3
J4
RJ11-2L-B
J2
J1
R3
2k2
AGND
R2
4k7
R1
4k7
IC2
7805DT
VI
VO
GND
C2
100n
C1
4µ7
C4
100n
C3
4µ7
R7
10k
R8
10k
R5
10k
R6
10k
Q2
BSS123
Q1
BSS123
JP1

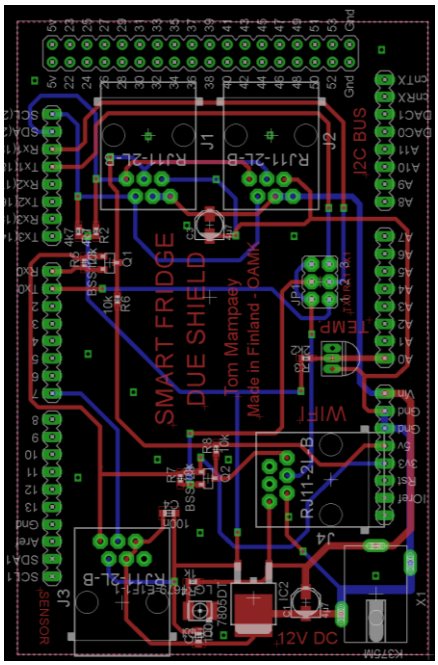
11.6. PCB design

Similar to the slave modules I gave a lot of attention to placing the components and also added enough labels at every in- or output. In this way it's very clear what to connect where, so no mistakes are made. Up to 127 devices (slaves) can be connected to the two RJ-11 connectors on the right.

11.6.1. Layers

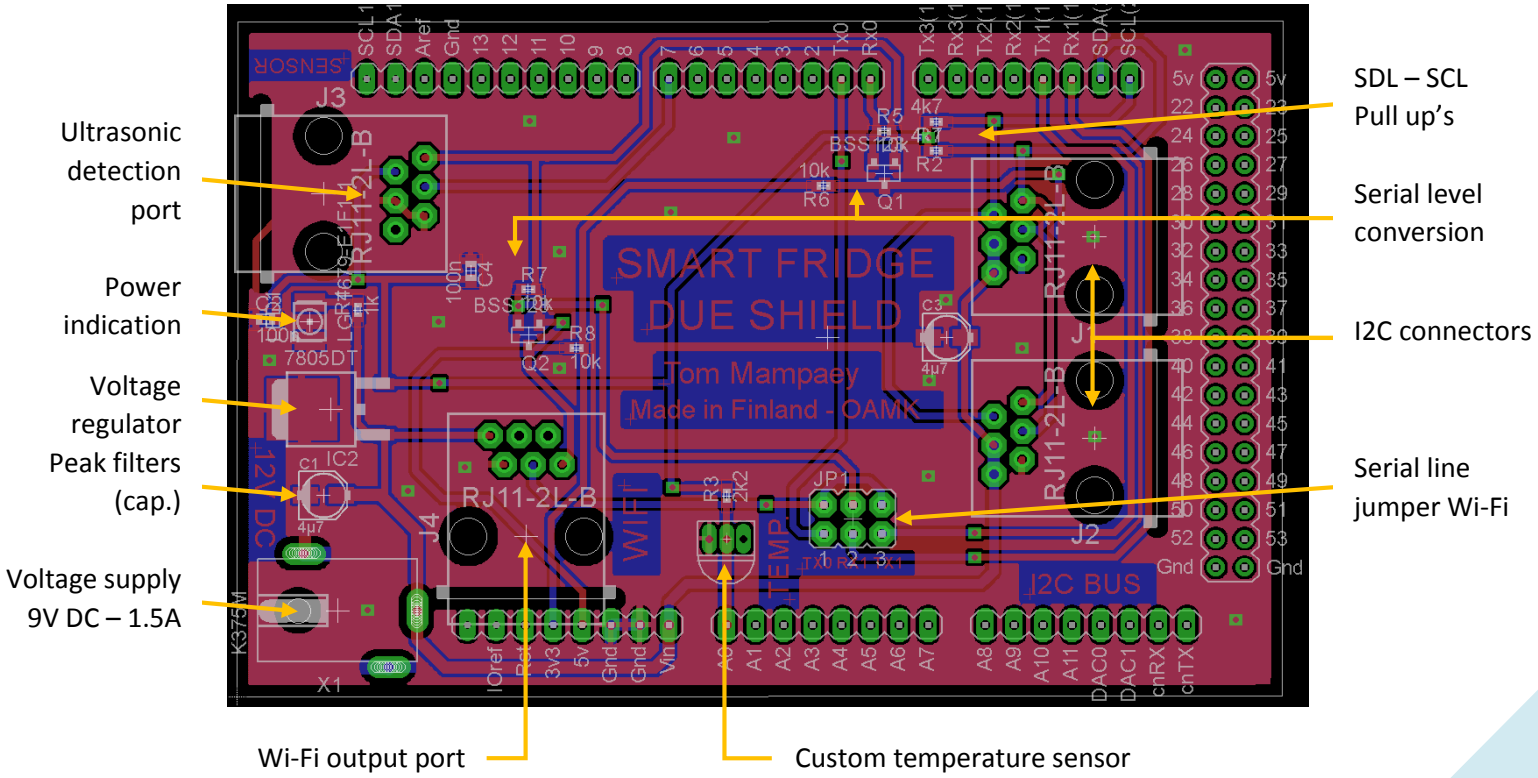


Main (/top) layer



Added second (/bottom) layer

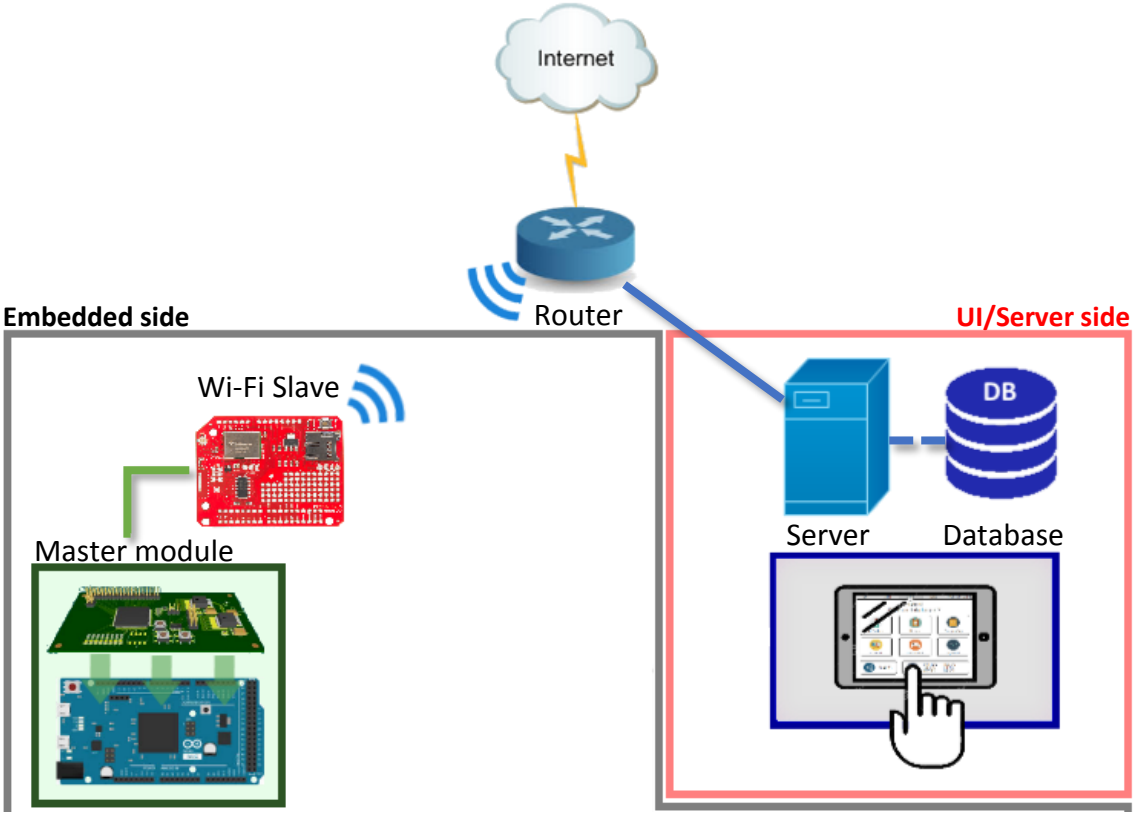
11.6.2. Final PCB including ground plane



12. Extension: Wi-Fi Module

The original idea was to connect the master module to a computer through an USB connection and to send the data to the server with an external application. The more I was working towards that idea the more I thought it was actually unnecessarily complicated, let alone the overkill it would be.
So I did everything in the planning to make this possible in a performant and enhanced way.
Most 'internet of things' (IoT) nowadays are mostly connected through Wi-Fi, which led me to replace the computer by a Wi-Fi module that would send the data straight to the server. I also did research and testing to potentially use an Ethernet shield but the Wi-Fi technology suited me more.
It soon became clear that it was a good idea to replace the computer by a Wi-Fi slave.

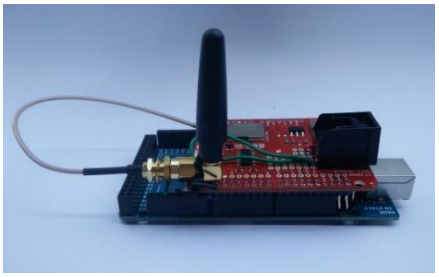
The modification/improvement to the original High-level schematic (see 7.):



12.1. CC3000 module

The module I chose for this extension is the CC3000 from Texas Instruments. The CC3000 utilizes the SPI (16MHz) interface instead of the more standard UART communication method. In this way it is possible to control the flow of data. The module uses the IEEE 802.11 b/g standard and can be configured in the different security modes WEP, WPA/WPA2 (AES and TKIP - Personal).
The CC3000 has an onboard Wimax antenna and an optional connection for an external one.

I used an Arduino Mega to control this module (DUE not supported). The Arduino will receive the headers coming from the master module and convert them into a JSON string. I programmed the module so that it will automatically connect with the 'Smart Kitchen' router.
I had to customize the shield specific to the project.

12.2. Code

```
/*
SEND PRODUCT INFORMATION TO SERVER
----------------------------------
Made by Tom Mampaey
at 'OAMK University of Applied sciences'
Finland - May 2015

Gets product information from MASTER-board SmartFridge on Serial 1 line.
Sends JSON data (PUT) to the server using Wi-Fi

Code based on the GET-request example of WebClient SFE
*/
#include <SPI.h>
#include <SFE_CC3000.h>
#include <SFE_CC3000_Client.h>

// Pins
#define CC3000_INT 2   // Needs to be an interrupt pin
#define CC3000_EN  7   // Digital pin
#define CC3000_CS  10  // cs pin (best 10 on if you use uno)

// Connection info data lengths
#define IP_ADDR_LEN 4   // Length of IP address in bytes

// Internet settings (constants)
char ap_ssid[] = "SmartKitchen";           // SSID of network
char ap_password[] = "12345678";           // Password of network
unsigned int ap_security = WLAN_SEC_WPA2; // Security of network
unsigned int timeout = 30000;              // Milliseconds
char server[] = "api.verhofstadt.eu";      // Remote host site
String data = " ";
int led = 50;

// Reading from Master Module
char incommingByte;
String incommingString;
char charArray[80];
boolean initComplete = false;
boolean dataReceived = false;
String NFC_ID = "";
String Address = "";
String Availability = "";
String Temprature = "";
String inData = "";

// Global Variables
SFE_CC3000 Wi-Fi = SFE_CC3000(CC3000_INT, CC3000_EN, CC3000_CS);
SFE_CC3000_Client client = SFE_CC3000_Client(Wi-Fi);

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  pinMode(led, OUTPUT);
  digitalWrite(led, LOW);

  ConnectionInfo connection_info;
  int i;
  // Initialize Serial port
  Serial.println("----------------------------------");
  Serial.println("SmartFridge CC3000 - PUT to Server");
  Serial.println("----------------------------------");
```

```
  // Initialize CC3000 (configure SPI communications)
  if ( Wi-Fi.init() ) {
    Serial.println("CC3000 initialization complete");
  } else {
    Serial.println("Error: CC3000 initialization incomplete");
  }

  // Connect using DHCP
  Serial.print("Connecting to SSID: ");
  Serial.println(ap_ssid);
  if (!Wi-Fi.connect(ap_ssid, ap_security, ap_password, timeout)) {
    Serial.println("Error: Could not connect to AP");
  }

  // Connection details and print IP address
  if ( !Wi-Fi.getConnectionInfo(connection_info) ) {
    Serial.println("Error: Could not obtain connection details");
  } else {
    Serial.print("IP Address: ");
    for (i = 0; i < IP_ADDR_LEN; i++) {
      Serial.print(connection_info.ip_address[i]);
      if ( i < IP_ADDR_LEN - 1 ) {
        Serial.print(".");
      }
    }
    Serial.println();
  }

  // Make TCP connection to remote host
  Serial.print("Performing HTTP PUT to: ");
  Serial.println(server);

  client.connect(server, 80);
  if (client.connected()) { //initiate connection
    Serial.print("Connected to: ");
    digitalWrite(led, HIGH);
  }
  else {
    Serial.print("Failed to connect with: ");
    digitalWrite(led, LOW);
  }
  Serial.println(server);

  /*
    // Make a HTTP GET request (works!)
    client.println("GET /service/exercises HTTP/1.1");
    client.print("Host: ");
    client.println(server);
    client.println("Connection: close");
    client.println();
    Serial.println();
  */

}
```

```
void loop()
{
  inData = "";

  if (Serial1.available())
  {
    delay(100);           //allows all serial sent to be received together
    while (Serial1.available())
    {
      char recieved = Serial1.read();
      inData += recieved;
    }
  }

  if (!inData.equals(""))
  {
    inData.toCharArray(charArray, 100);

    //Constructing Json to PUT
    String ID = GetId();
    String Address = GetAddress();
    String Available = GetAvailability();
    String temprature = GetTemprature();
    data = "{\"IdNFC\":\"" + ID  + "\",\"Address\":\"" + Address +
"\",\"Available\":\"" + Available + "\",\"TemperatureFridge\":\"" +
temprature + "\"}";
    SendToServer(data);
  }
}

String GetContentByFilterOn(char filter)
{
  bool readd = false;
  bool stopread = false;
  String content = "";

  for (int i = 0; i < sizeof(charArray); i++)
  {
    if (stopread)
    {
      return content;               //return string with filtered data
    }

    if (readd)
    {
      if (charArray[i] != filter)
      {
        content.concat(charArray[i]);//connect chars in buff to form string
      }
    }

    if (charArray[i] == filter) // Keep reading until filter char detected
    {
      readd = !readd;
      if (readd == false)
      {
        stopread = true;
      }
    }
  }
}
```

```
void SendToServer(String data)
{
  if (client.connect(server, 80))
  {
    if (client.connected())
    {
      client.println("PUT /service/Hardware/Product HTTP/1.1");
      delay(5);                  // added short delays to avoid overload
      client.print("Host: ");
      client.println(server);
      delay(5);
      client.print("Content-Length: ");
      client.println(data.length()); // without quotes
      delay(5);
      client.println("Cache-Control: no-cache");
      delay(5);
      //client.println("Connection: keep-alive");
      //client.println("Content-Type: text/plain; charset=utf-8");
      //client.println("Content-Transfer-Encoding: base64");
      //client.println("Accept-Encoding: gzip, deflate, sdch");
      client.println("Content-Type: application/json");
      delay(5);
      client.println();
      delay(5);
      client.println(data);
      Serial.println(data);
            for (int i = 0; i < 50; i++){
          char c = client.read();
          Serial.print(c);
      }
      client.stop();
    }
  }
  else
  {
    Serial.println("Time-out: Could not make TCP connection");
    client.connect(server, 80);
  }
}


String GetHeader()
{
  return GetContentByFilterOn('X');   // Filter NFC-ID by data between 'D'
}
String GetId()
{
  return GetContentByFilterOn('D');   // Filter NFC-ID by data between 'D'
}
String GetAddress()
{
  return GetContentByFilterOn('A');   // Filter Address by data between 'A'
}
String GetAvailability()
{
  return GetContentByFilterOn('B');   // Filer Av. by data between 'B'
}
String GetTemprature()
{
  return GetContentByFilterOn('T');   // Filter Temp. by data between 'T'
}
```

12.2.1. GetContentByFilterOn

One of the smart functions I made in this code is the GetContentByFilterOn method.

```
String GetContentByFilterOn(char filter)
```

This function will read and decode the data of the headers sent by the master module.
The beautiful thing about this function is that I can use it to filter on every single different char.

If (header) data is detected on the Serial 1 line, a buffer will be filed with these values. This method will then be used to filter out the data on the specific char.

E.g. XD4875242556256255935332DA56AB1BT3TX
→ `GetContentByFilterOn('D');`
→ Result: 4875242556256255935332

To make the code as readable as possible I added extra methods for every char.
This makes it very easy to expand extra values into the header. (see last 5 methods)

12.2.1. SendToServer

When all the data is successfully declared out of the header I transform the values into a JSON string.
The connection with the server is already made during the setup of the code.

```
data = "{\"IdNFC\":\"" + ID  + "\",\"Address\":\"" + Address +
"\",\"Available\":\"" + Available + "\",\"TemperatureFridge\":\"" +
temprature + "\"}";
```

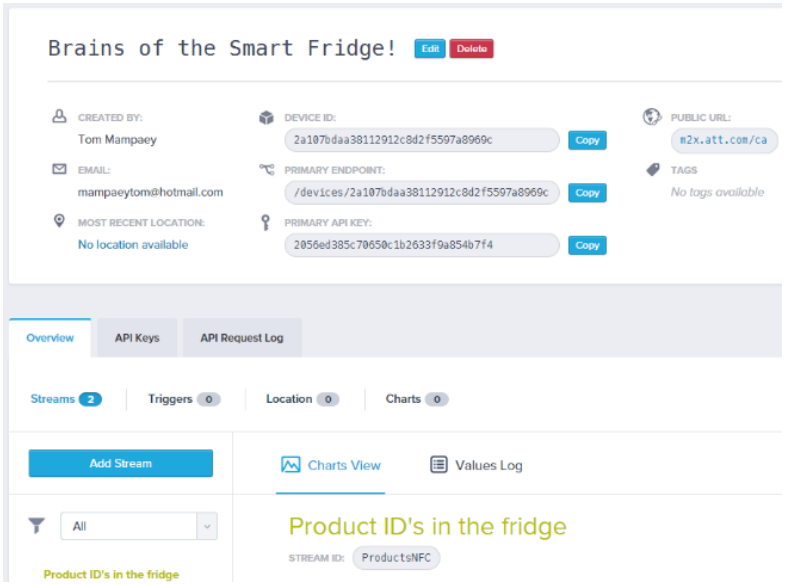
If the JSON string is ready: e.g.
{"IdNFC":"4875242556256255935332","Address":"56","Available":"1","TemperatureFridge":"3"}

I request to do a PUT on *(http://api.verhofstadt.eu)/service/Hardware/Product HTTP/1.1*
This is where the databank is located on the net (Host: api.verhofstadt.eu).

To make it error proof it's important to specify the content type (application/json), the cache control (no-cache) and the specific length of the JSON string.

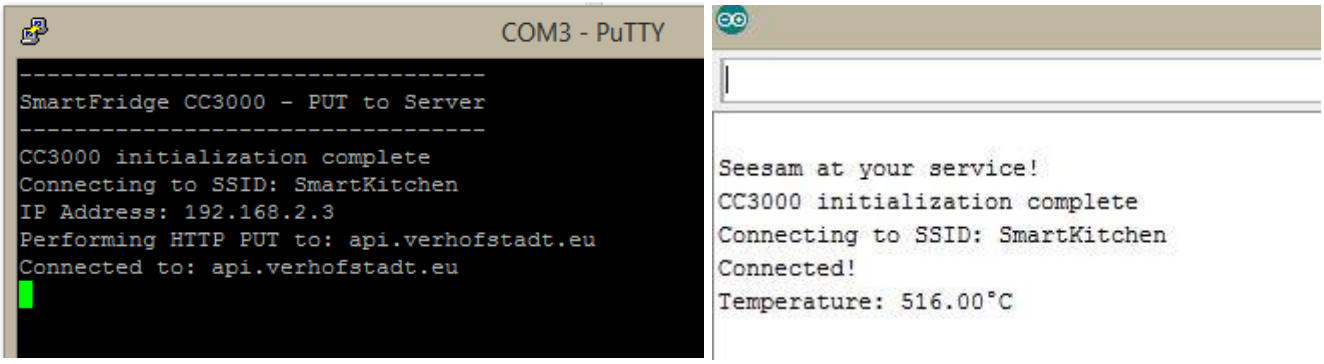
Extra parameters can be added in the header here but isn't necessary for this application.

Note: For testing purposes I configured a small M2X-databank before testing it on the real one.
(https://m2x.att.com AT&T IoT Services: easy to implement RESTful API's and native SDK's)



12.3. Response of server and ping

To make things easier for myself and to deliver a transparent system I print everything that happens to the serial monitor. When you connect your computer's USB to the Wi-Fi slave module you can monitor every step and get error messages, should errors occur.



12.3.1. Ping to server

I installed, configured and completely tested all the connections my slave board made the first ping to the server.

```
Connecting to: SmartKitchen
Connected!
IP Address: 192.168.1.196
Looking up IP address of: api.verhofstadt.eu
IP address found: 81.169.135.90
Pinging 81.169.135.90 3 times...
Pong!

Packets sent: 6
Packets received: 3
Min round time (ms): 61
Max round time (ms): 71
Avg round time (ms): 64

Disconnected
Finished ping test
```

When this finally worked I started to test a PUT-request to the server via 'Postman - REST Client' with JSON string that the Wi-Fi slave generated. If I receive a HTTP status code of 200 as response of the server the PUT has been successfully. Of course I had these problems too:

```
HTTP/1.1 409 Conflic{"IdNFC":"0000","Address":"62","Available":"0","TemperatureFridge":"20"}
HTTP/1.1 409 Conflic{"IdNFC":"0000","Address":"63","Available":"0","TemperatureFridge":"20"}
HTTP/1.1 409 Conflic{"IdNFC":"0000","Address":"61","Available":"0","TemperatureFridge":"20"}
HTTP/1.1 409 Conflic{"IdNFC":"0000","Address":"62","Available":"0","TemperatureFridge":"20"}
HTTP/1.1 409 Conflic{"IdNFC":"0000","Address":"63","Available":"0","TemperatureFridge":"20"}
```

HTTPS response 409 = conflict with the current state of the resource

Most common HTTPS status codes:

200	400	404	500	550
OK / Success	Bad request	Not found	Server error	Permission denied

13. Custom Printed Circuit Boards

When prototyping a project, one of the main subjects is to design a performant printed circuit board for the application. This is a part that really can't be underestimated. For every little component you have to check the datasheets, sepcs, operating temperatures (fridge), testing on a breadboard, placing them strategic and solder them with a microscope to make sure the connection is secure.

13.1. Test boards

It's obvious that you can't make the perfect functional print of the first time. You have to make different versions in order to detect and discover all the unforeseen errors to optimize the board to the fullest. I did this for every board I made, including the breadboard and simulation.

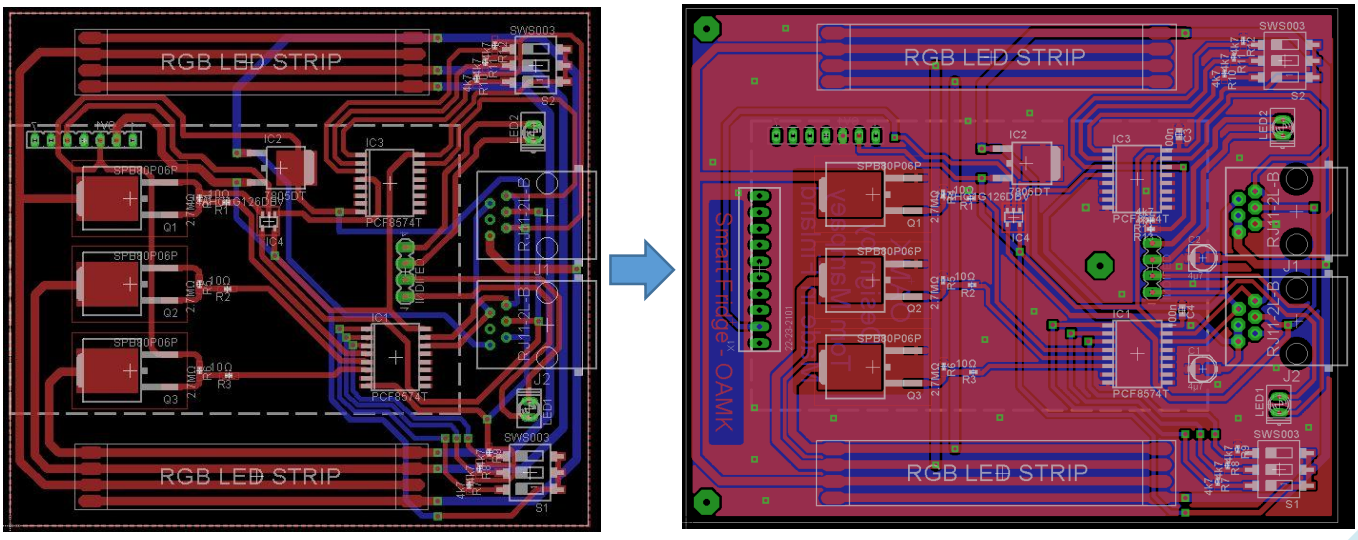
13.1.1. Problem solving

As stated previously the main reason for making a test board is to optimize the board to the maximum and to eliminate even the smallest flaws. This can go from adding 'pull up/down' resistors to complete filters and rewiring the connections. But also practical things, like making some pads a little bit bigger to enable to solder it solid.

A good example of why I'm happy that I made those test boards is a gigantic disturbance that appeared on the LED drivers when a tag was detected.

For driving and controlling the different colors of the RGB LEDS I used Field Effect Transistors (bka FET's). One of their properties is that they are very sensitive to static electricity.

If you look closely to the construction of the Slave boards you can see that these FET's are very close to the NFC reader part. This reader has of course an antenna to detect the NFC tags in his range. With the antenna so close to the sensitive FET's lots of interference by radiation occurred. When I noticed this problem, I was able eliminate it with an extra ground and a diffusedly ground plane. These were faults I hadn't thought of beforehand. This proves that it's useful to make some test boards first!

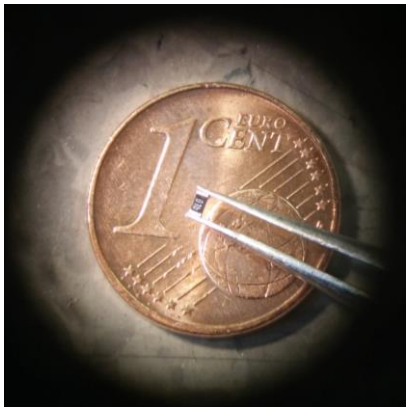


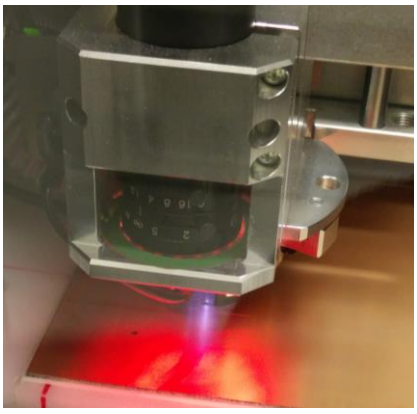
Test board version 1.0 *Final PCB design*

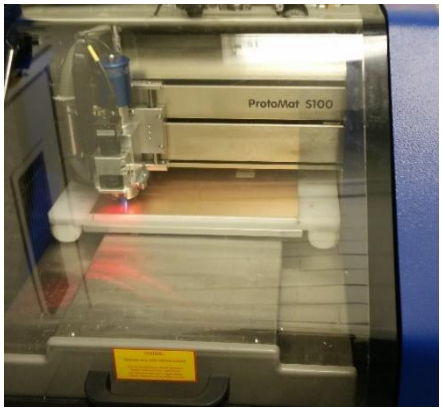
13.2. Surface Mounted Device (SMD)

I was very excited when I saw that I had the option to work with SMD's for this project. I had never done this before but I have certainly been interested in this technology for a long time. OAMK was very experienced in this technology and provided professional facilities.

SMD is the group of components that are the smallest solderable components in the world. These components are soldered on top of a copper plate. The connections between them are milled by a very accurate machine. Solder components smaller than 2mm aren't

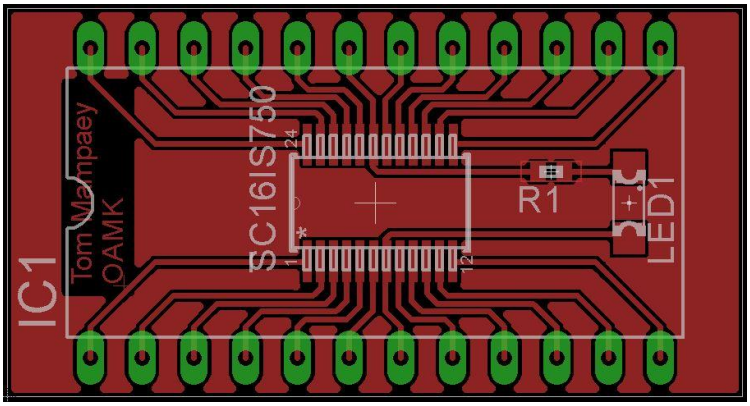


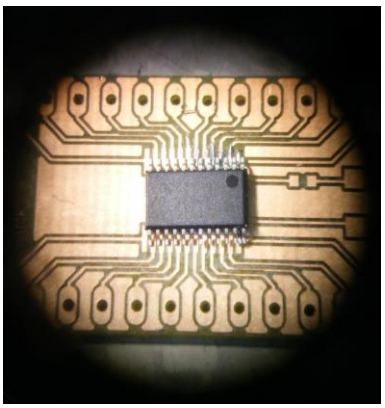


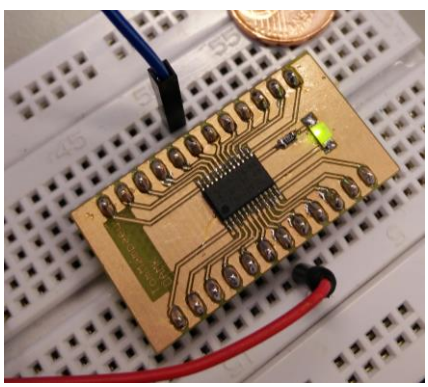


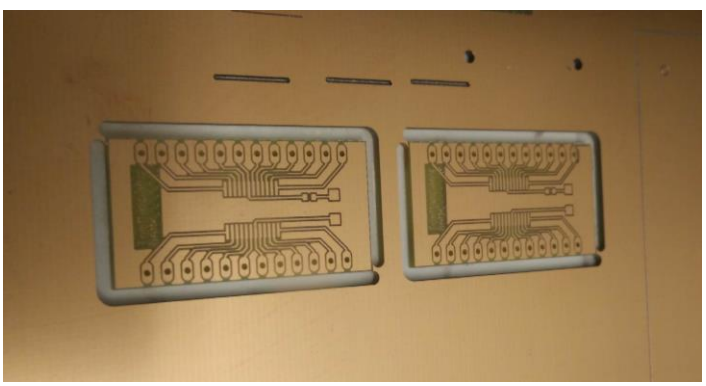
Considering the fact that I never had any experience with SMD designing, soldering or doing some data research about it, was a good small start. I designed a custom component in Eagle 6.5.0 for the SC16IS750, which is a Single UART with I2C-bus/SPI interface that I was planning to use in the beginning.

When my design was checked by the experienced prototype manager I was ready to mill my very first little SMD board. After spending almost half a day trying to solder it, my first SMD was ready.





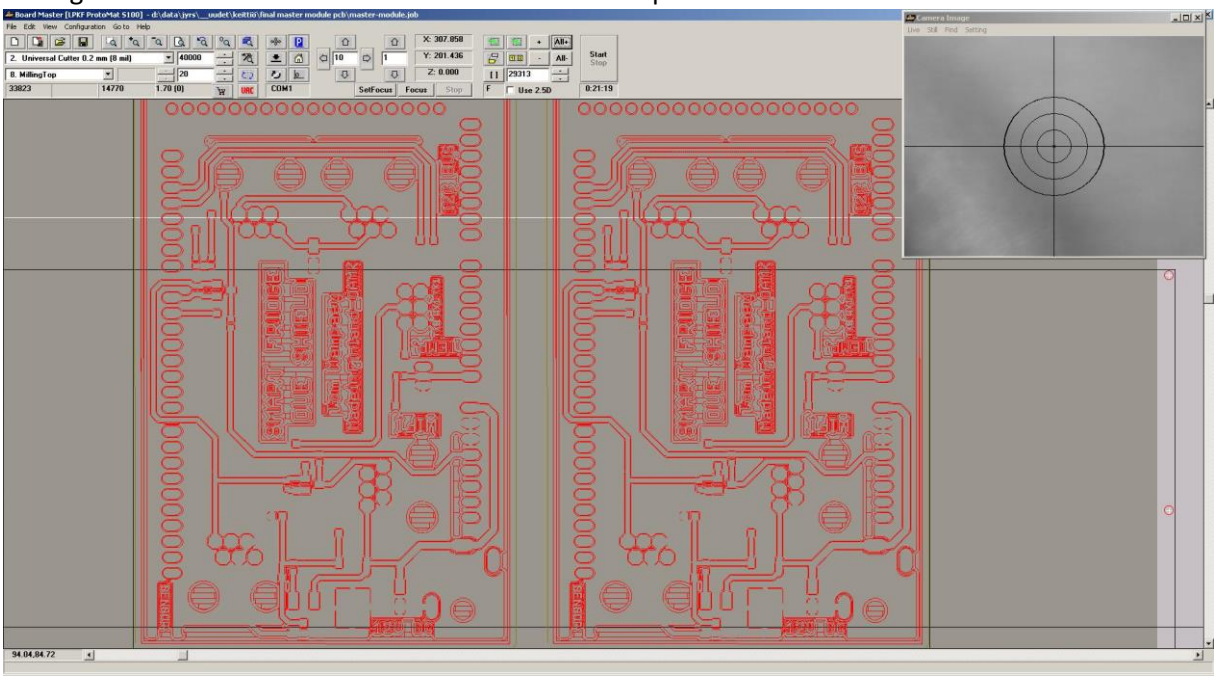




13.3. Production of final PCB

13.3.1. Design and conversion

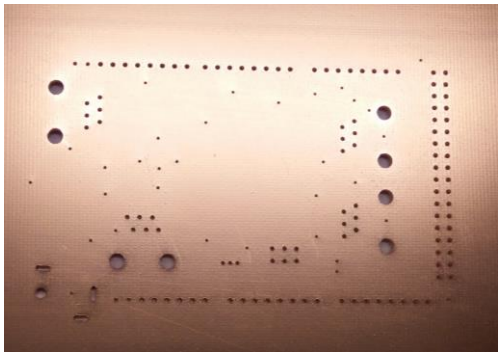
When the design has been triple checked the required files are generated for the milling machine (.lmd, .drill, .mill , .lpkf,…). The commands (in universal G-code) send from the computer to the milling machine are based on these files. For the example I will take the master and slave board.

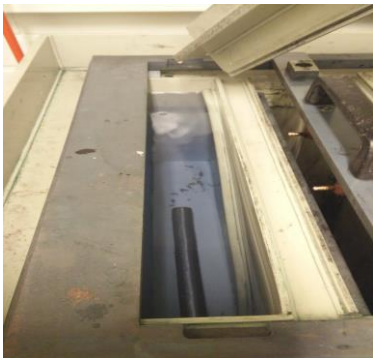


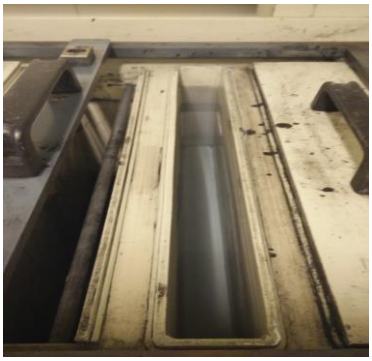
Software that controls milling machine. Top layer - Master board. Milling camera at right-hand corner

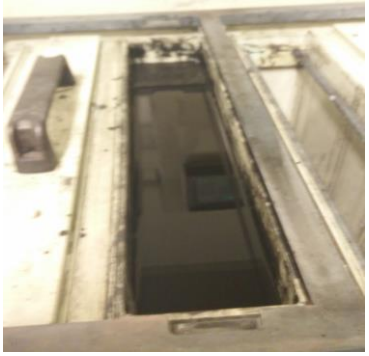
13.3.2. Drilling and through hole plating

The next item on the agenda, after the calibration and setting the correct tools, was to drill the holes very accurately into the copper plate. When the drilling is done, the copper plate goes into a series of chemical cleaning substances: three different kinds of cleaning baths, each approximately for 2 to 10 minutes.

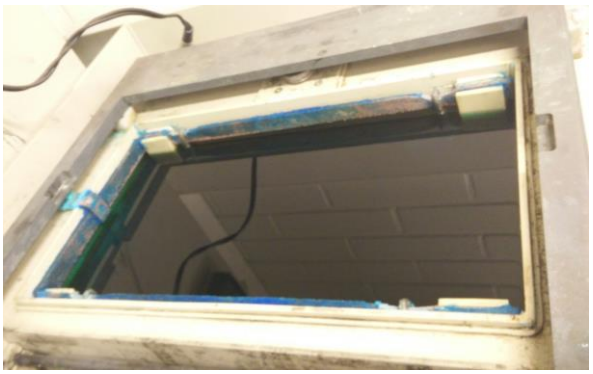








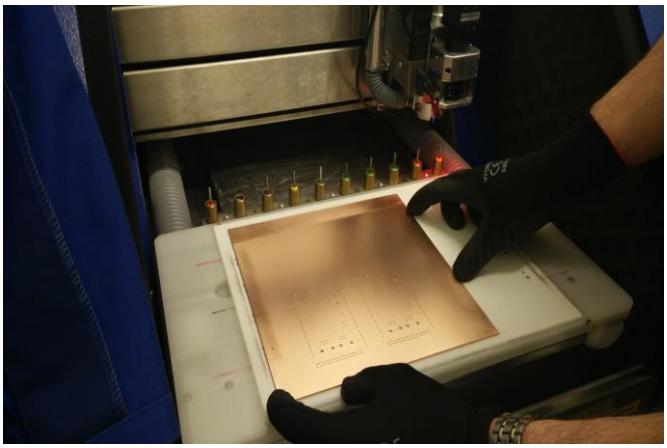
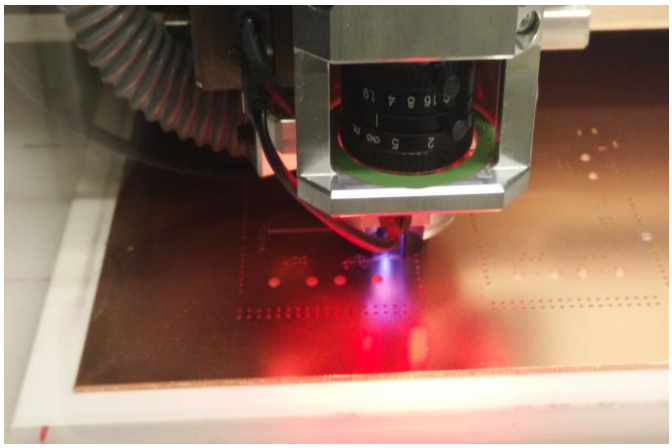
If the copper is clean it's ready for through hole plating. The board was immersed into chemicals and connected to a power source. The anode and cathode between the bath and the board will have the chemical effect of attracting and saturate copper particles at the inside of the holes.



This is a process of approximately two hours. The best results are achieved with a low voltage (~ 0,5V) and a high current (~5A). The inside of the holes are now solid copper. This ensures that the connections are secure and the different layers and planes are connected to each other.

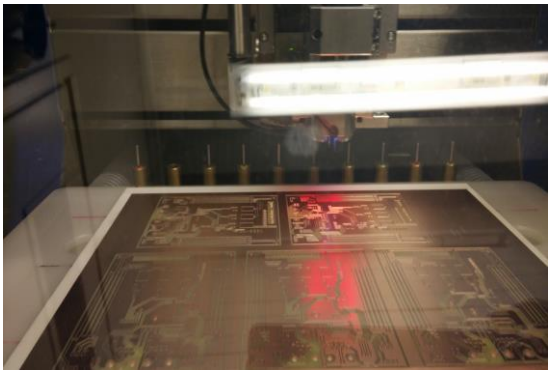
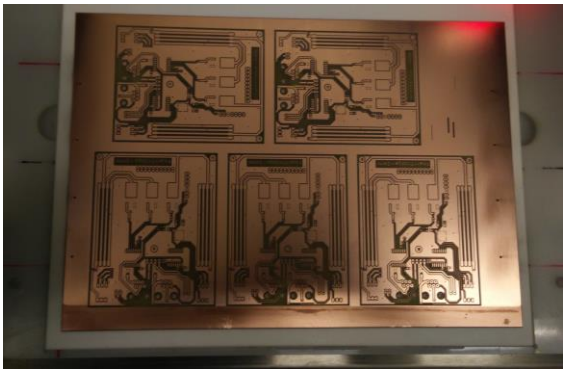
13.3.3. Milling

Now it's time to start milling. The plate is put back in the machine and after calibration the mill will do its work. The PCB is milled layer by layer. It took around two and a half hours, for each layer, to mill the slave board (two layers). OAMK can construct PCB with up to 6 parted layers on one PCB. Looking at the milling process is like watching paint dry, but a little more entertaining.

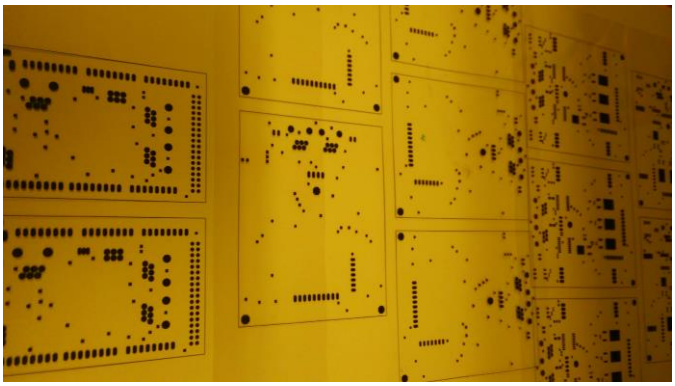



13.3.4. Coating

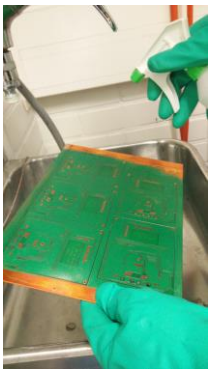
The milled plate already looked really good. As the last step towards the final PCB a protective coating is applied. A few steps must be taken:

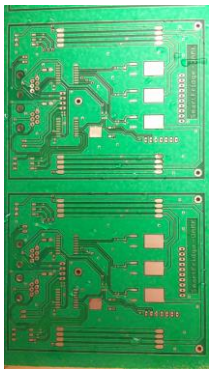



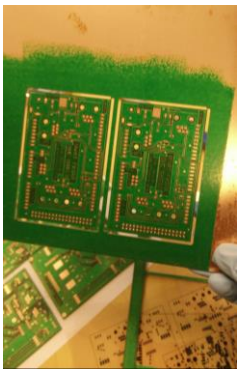
First the places/pads that should not be coated must be laser printed on a transparent foil. These foils must now be aligned with the milled PCB board and put into the image exposure unit. This unit will illuminate the copper that isn't covered by the foil with UV-radiations. (Of course you have to watch out with sunlight.)

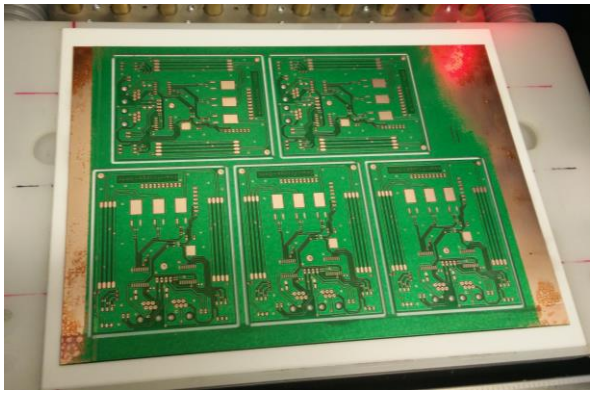


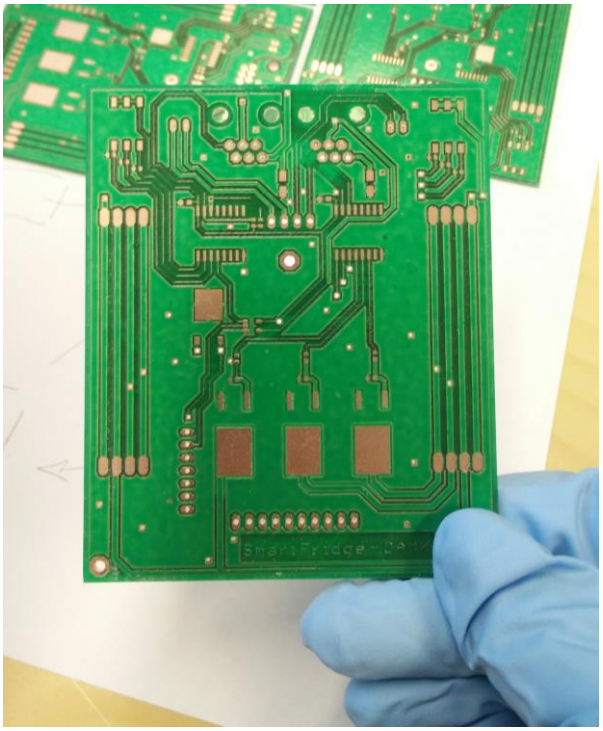
After the first chemicals and pre-drying in a hot air oven, the coating is spread evenly over the PCB. When the PCB is dry the coating on the 'non-exposed' places can be removed with a brush and acid. Finally let it harden in the hot air over for about 45 minutes. Clean with acid and water and it's ready. Now the board has to go one last time to the milling machine to cut them out. I was very euphoric about the end result.





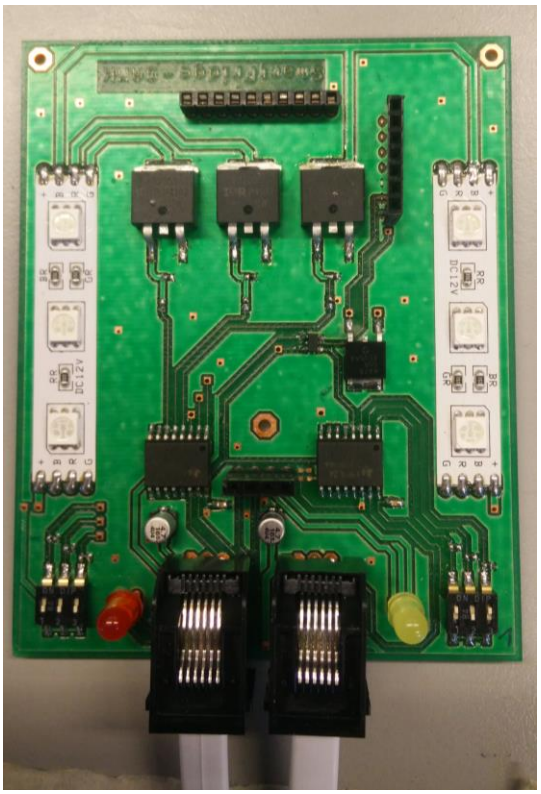


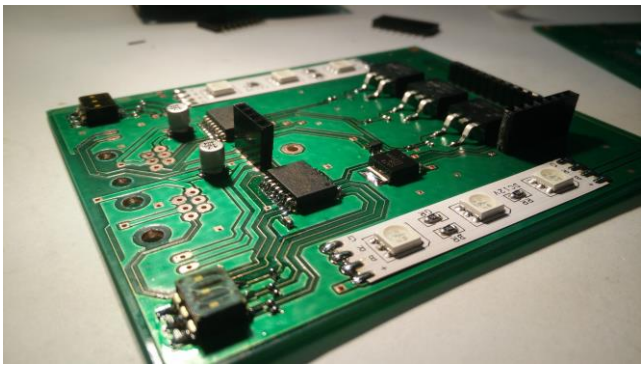


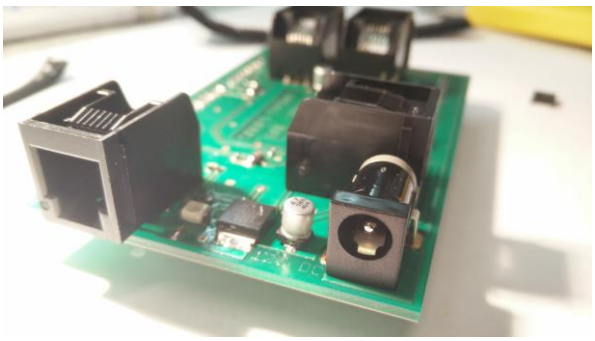


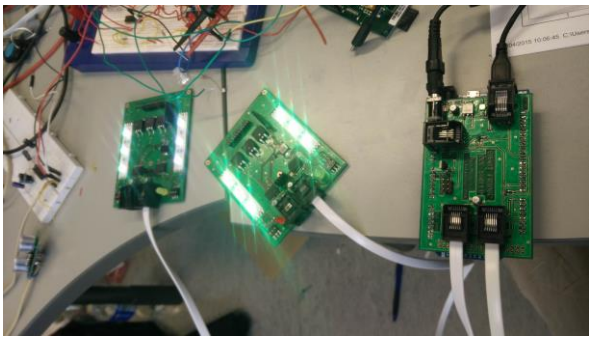
13.3.1. SMD soldering

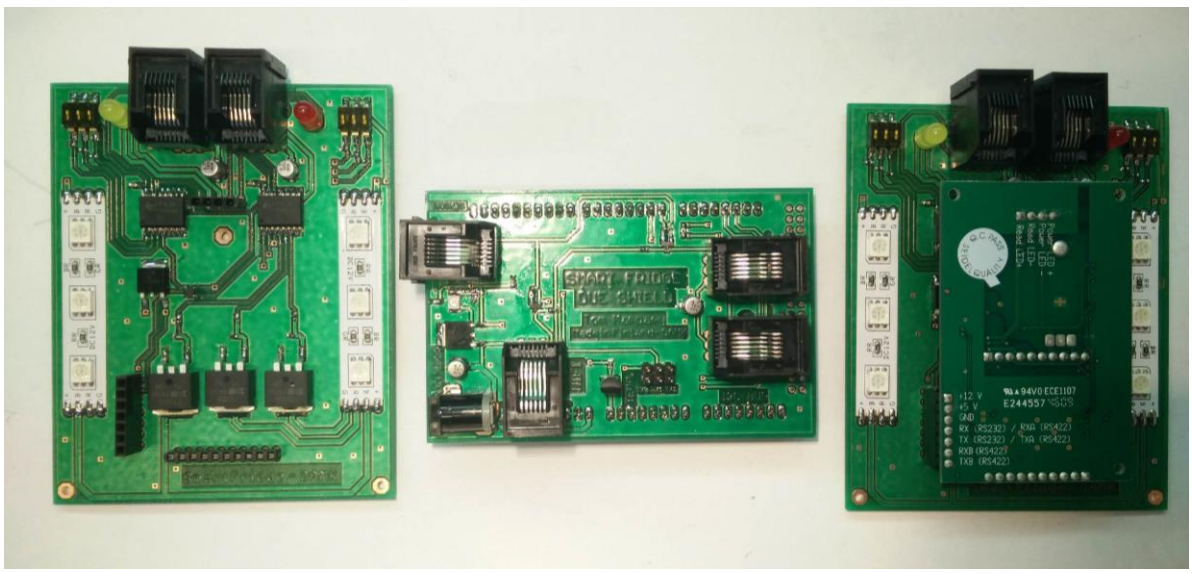
When the PCB was ready and cooled off, I took them to the solder lab for high precision soldering. It took a while before I had mastered the skill of SMD soldering through a microscope. It required my undivided attention, but I can be proud of the boards I delivered.











14. 3D modulating & printing

As a real '*Finnishing'* touch I had the idea to put this prototype and my acquired experience to the maximum by designing and 3D printing a box for the slave boards at the university.

This was even after the final presentation and when I was the last member of the project still in Finland. We had a guided tour around the printers around March and I just couldn't resist to ask them when I had the chance.

14.1. The design and software

When I contacted the staff responsible for the machines I asked them about what software I could use and which software they would recommend. I valued their professional opinion very much, as I didn't have any 3D-printing experience .

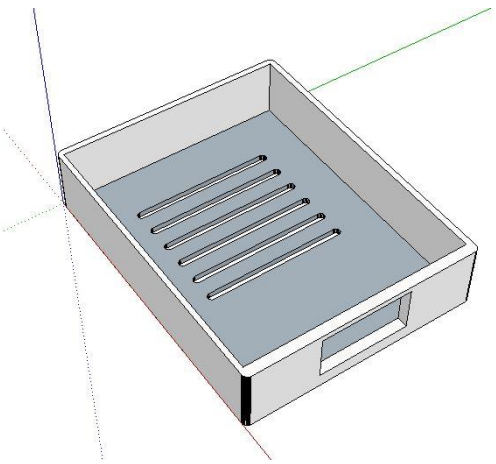
Recommendation: 3D CAD SolidWorks

Other software:

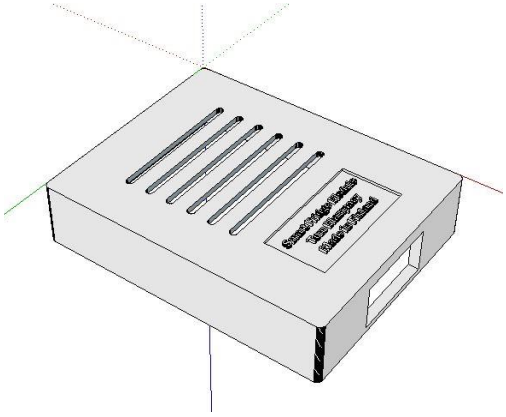
AC3D	Moi3D
Blender	Netfabb
OpenScad 3D	TinkerCad

All 3D-designing software that can export .stl files …

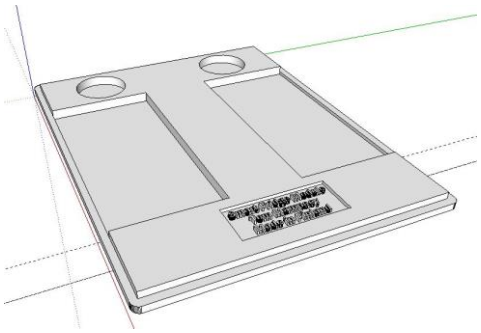
After some sketches and tests I came up with this final design for the Slave module:



Top view



Bottom view



Lid (bottom view)

I made dilutions in the lid of the box so the bright light of the RGB-LED can shine through. This gives a beautiful and clear color effect around the product. Also the indication LEDS, power and reading, become visible for the user/developer by the round dilutions in the lid.

14.2. The Printers

OAMK has three amazing industrial 3D printers. They even have one that they were able to purchase from Nokia for a very generous price (laser 3D printer, Nylon material). They make grateful use of the printers for all kinds of purposes. From prototyping student projects to printing scanned human bones and skulls to prepare surgeons for medical operations. These printers are very professional with a high accuracy.

EOSINT P380

14.2.1. Stratasys Fortus eV

The machine that was the best fit for my prototype was the *Stratasys Fortus eV.* This is a very high quality printer that has different possibilities in colors and (support) materials. The material I used is white ABS M30 which is up to 25-70 % stronger than standard Stratasys ABS. It is an ideal material for conceptual modeling, functional prototyping, manufacturing tools, and end-use parts.

14.2.1. Details

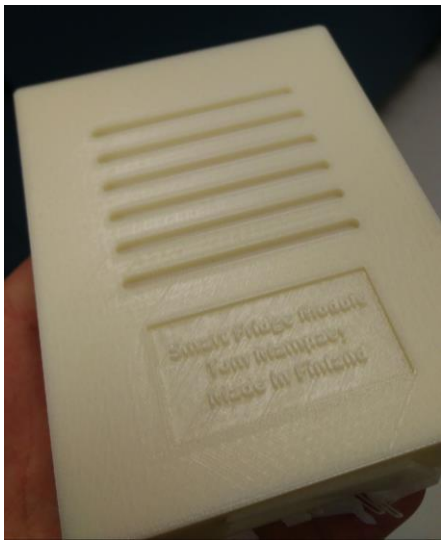
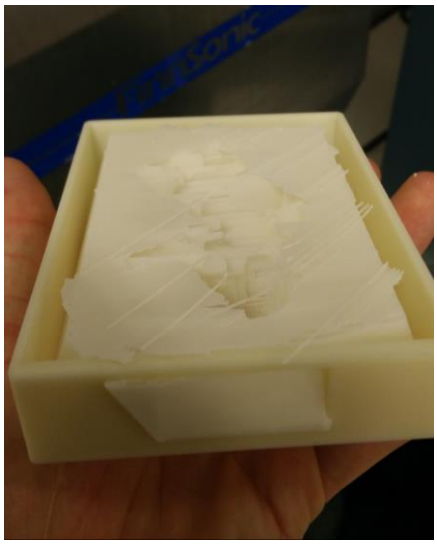
As you can see in the pictures below, the boxes aren't completely stuffed with the support material. We configured the machine with Smart Print. This saved us support material, which was added on the inside of the box to allow printing the bottom of the box, faced upwards, layer by layer.

The support material can be removed by hand or by soaking it in water (or acid for a faster result).

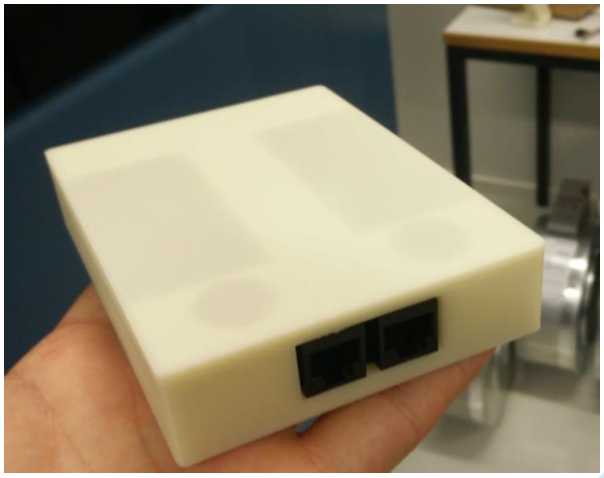
Total volume for three boxes:	174,5 cm^3
Price of materials: (Customers)	€0.30/cm^3 (University) €0.50/cm^3
Estimated price for companies:	~ €400
Total duration of printing:	36 hours (day and night)

14.3. Result, production, assembly

After the first day of printing I came back for observation and I was already very excited about how it looked. When it was finally ready and I removed all the support material, my Slave board really fit great, as if they were made for it!






15.1. What I learned

Something very important about a project is evaluating yourself and your work, to reflect on what you have learned. When I think back to the beginning of the project I can say that I have learnt a lot thanks to this project, the University and the overall experience.

In the table below you can see a scale on what I have learned and which skills I improved.

Note: Only applies to this project and independent of foreknowledge

Subject	Beginner		Experienced		Advanced		Expert	
SMD - design multiple layers								
Component desing								
Prototyping								
SMD component techniques								
Arduino/coding								
Reasearch/Datasheets								
Github								
PCB design and placements								
Sensors								
Schematics								
Microscope comp. Placing								
Shield construction								
Measurement/test equipment								
Hardware/Software debug!								
I2C protocol								
Near Field Communication								
WiFi, HTTP methods, ...								
PCB manufacturing								
Handeling unexpected failures								
ASUS Xtion PRO								
Raspberry Pi								
3D-printing								
M2X Development								
iPi Mocap Studio								
RF-ID								
Microsoft Kinect								
Wheatstone bridge weightscale								
Working independently								
Planning skills								
Cooperate in English								
Tutorial								
Time management								
Teamwork								
Scrum								
Preparing meetings								

16. Optimizations and further extensions

Projects like this are unstoppable. You can always find something to innovate upon and there will always come new technologies to optimize every project. With the future in mind and for the successors of this project I analyzed it, searching for further optimizations and extensions.

16.1. One PCF8574/address for each module

In the current system every slave has a separate address for NFC read control and LED driver control. In the future this can be improved by having only one PCF8574 for each board.
The PCF could use three output pins for controlling the LED drivers and one for controlling the flow of the NFC data. In this way there don't have to be two PCF's on board and every slave module could only have one address, which is much more performant.
This optimization also makes the slave boards much smaller, which made me think about integrating multiple NFC readers/ slaves in the shelves of the fridge. Or maybe even all the readers on one plate. This would be much more compact and closer to the end product.

16.2. Reading and powering NFC readers

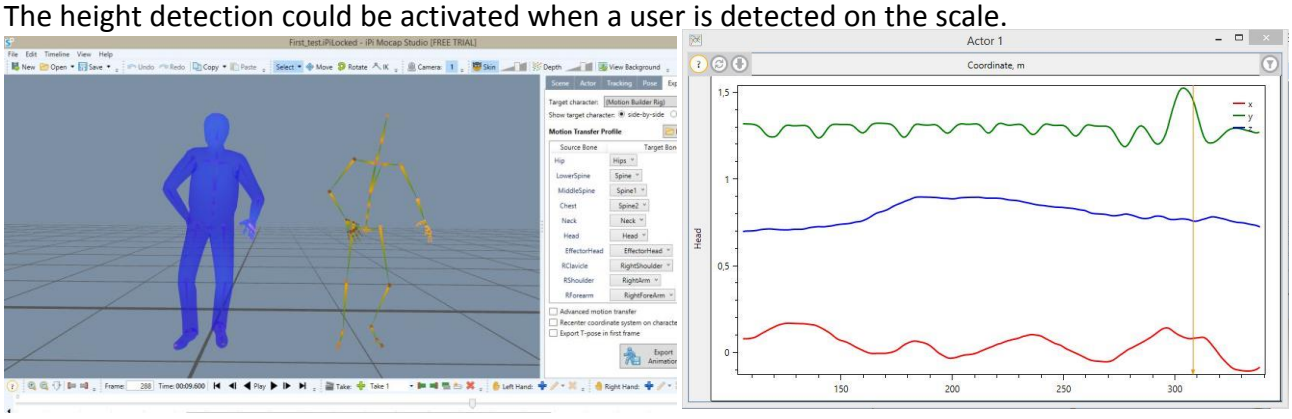
When the complete system was operating I noticed that it was a bit unnecessary to constantly read all the tags that are in the fridge. In the next version it would be better to only power the readers when the master gives the command to get the ID of the reader on that location. This would also be good for energy saving.

16.3. Scale – Weight detection

Since the user now has to enter in his/her weight manually it is unnecessarily complicated. It would be better to have a (thin) scale in front of the fridge that would declare this value automatically. A sensor like this could also detect if there is a user in front of the fridge.

16.4. Camera - Height detection

Just like the weight, the height also has to be entered by the user him/herself. I did some research and testing with the Kinect, as recommend by OMAK, mostly with the Asus Xtion Pro, to also detect the height of the user. I tested this with iPi Mocap Studio and I managed to detect the height of the moving user. The average can be calculated from this movement, in combination with the BMI-scale.
The height detection could be activated when a user is detected on the scale.



Note: Values of jumping around are a little bit low because of the calibration.

17. Conclusion and evaluation

It took a while before we finally found the project we were looking for. In our first month on Erasmus in Finland we got thrown from project to project, until I met one of the project managers at the game development lab.

He contacted me about a new international project about making a Smart Kitchen.
Those two words were music to my ears and I was immediately sold to work on this.
We had the honor of participating Sesam, with one of the most used devices in your kitchen: the Smart Fridge.

It was very nice to work on such a future-oriented project around the Internet of things and smart objects. I'm really happy I was one of the first students that was able to participate in this project.

The extraordinary thing about this is that we really had to start from a blank paper. This way we had to figure out by ourselves which intelligence and technologies the customer would like to have built in his refrigerator of the future. It was a great experience to start a project like this, and it really gives another view of how it development like this could be like in the business world. Even though it took a lot of extra time to do some research, brainstorming and testing.

When I look back at this project, I see lots of ups but also a few firm downs. At some moments I was thinking that it wasn't possible to have this much problems, errors and setbacks in one single project. Even the laws of nature weren't agreeing sometimes.
But in the end all these problems make the satisfaction of the end result even greater.

Working on this project in an international teamwork environment was a very nice experience for me. The language barrier and different culture were a challenge yet it was also very much an opportunity for learning, a valuable opportunity to challenge myself further. For example, I would think 'Oh, the Koreans of the Smart Table may do this and that… I will show them and add an extra 'that''. I made it my own little competition, not only to the external project, but also for myself. This was a huge incentive for me. Also using scrum and doing regular meetings was a very valuable experience for me.

In the end I can confidently say that I am proud of what I was able to establish. I always wanted to improve, optimize and extend the project to keep making it better, even after the final presentation and when the scores were already given my enthusiasm was unstoppable.

I can say that my successors in this project have a great and extensive base to start from. Certainly with the embedded systems side and server side working together now almost anything is possible as the demand of the client.

What started in an ordinary kitchen in a university in Finland is now reality.
I would like to use this opportunity to thank everybody that made this possible.

Kiitos! It has been an honor.

Annex:

Content:

NFC:

- https://developer.android.com/guide/topics/connectivity/nfc/index.html
- http://en.wikipedia.org/wiki/Near_field_communication
- http://www.digitaltrends.com/mobile/nfc-explained
- http://mashable.com/category/nfc
- http://www.identivenfc.com/en/what-is-nfc
- http://www.postcrescent.com/article/20120130/APC0101/201300409/Micro-security-chips-may-help-deter-property-theft?odyssey=mod%7Ctopnews%7Ctext%7CAPC-News
- http://www.rfidjournal.com/article/view/392/1/2
- http://www.contactlessnews.com/2006/11/30/contactless-inlays-from-smartrac-ordered-for-us-epassport-project
- http://electronics.howstuffworks.com/difference-between-rfid-and-nfc.htm
- http://arstechnica.com/gadgets/guides/2011/02/near-field-communications-a-technology-primer.ars
- http://gizmodo.com/5707321/what-is-near+field-communication
- http://www.forbes.com/sites/andygreenberg/2012/01/30/hackers-demo-shows-how-easily-credit-cards-can-be-read-through-clothes-and-wallets/
- http://www.usatoday.com/tech/news/story/2012-01-08/cnbc-near-field-communication-mobile/52443756/1
- http://www.pcadvisor.co.uk/how-to/mobile-phone/what-is-nfc-how-nfc-works-what-it-does-3472879

I2C:

- http://www.nxp.com/documents/data_sheet/PCF8574_PCF8574A.pdf
- https://en.wikipedia.org/wiki/I%C2%B2C
- http://nl.wikipedia.org/wiki/I%C2%B2C-bus
- http://nl.wikipedia.org/wiki/Serial_Peripheral_Interface
- http://www.totalphase.com/support/articles/200349156-I2C-Background
- http://www.nxp.com/documents/leaflet/75015676.pdf
- http://elger.org/wiki/projects/i2c_pcf8574_8bit_port_expander

SRF05:

- http://www.robot-electronics.co.uk/images/srf05p2.jpg
- http://www.f15ijp.com/2012/09/arduino-ultrasonic-sensor-hc-sr04-or-hy-srf05

Wi-Fi:

- https://learn.sparkfun.com/tutorials/cc3000-hookup-guide/all
- https://github.com/adafruit/Adafruit_CC3000_Library

PCB:

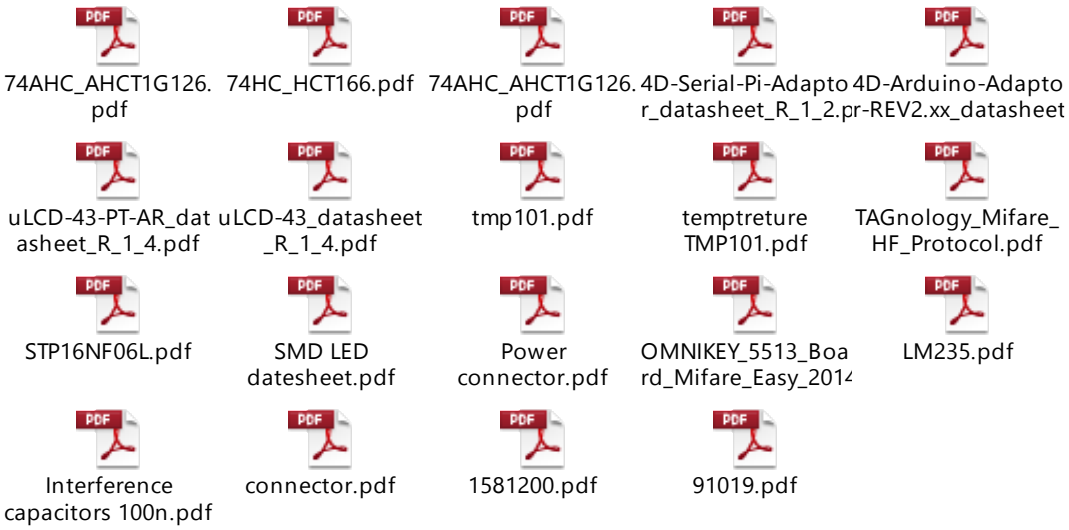
- http://www.eurocircuits.com/clientmedia/ecImage/pages/electroless-copper-deposition/10%20electroless-copper-deposition_big
- http://www.logismarket.es/ip/tecnologia-tecom-trazador-protomat-catalogo-de-productos-protomat-604190.pdf

3D Printing:

- OAMK University
- http://en.wikipedia.org/wiki/List_of_3D_modeling_software
- http://www.redeyeondemand.com/abs-m30
- http://formlabs.com/company/press/formlabs-announces-draft-mode-and-smart-supports

---

Datasheets:



Link: https://github.com/OAMK-Smart-Kitchen/Smart-Fridge-OAMK/tree/Embedded-development/SmartKitchen-Emdebbed%20System/Datasheets

Images:

- Front-page image:
  http://www.revistaexclusiva.com/?p=25851
- All the pictures can be found on:
  https://www.dropbox.com/sh/ei0k6ph26ccd29w/AACgIXhduUCAZlzj1vqn8d1Qa?dl=0

Supplementary component list:

ID:	Subject/Component:	Use/Part:	Quantity	Shop/Example/Info:	≈Price pp:
1	IC, I2C BUS EXPANDER 16DIP	To test on breadboads	3	http://be.farnell.com/texas	€ 1,17
2	I/O EXPANDER, 8BIT, I2C, TSSOP-20	For Final (SMD)-purposes	6	http://be.farnell.com/texas	€ 1,24
3	LM335AZ/NOPB IC, TEMP SENSOR, 5V, 3-TO-92	To measure temp in & outside	3	http://be.farnell.com/texas	€ 1,20
4	IC, TEMP SENSOR, 5V, SMD	For Final (SMD)-purposes	2	http://be.farnell.com/texas	€ 0,72
5	RGB-ledstrips or RGB-SMD-Power-LEDS	To indicate foods	7	http://www.dx.com/p/12v-	€ 45,00
6	MOSFET, N, 55V, 110A, TO-220	To control RGB-LEDS (3/shield) T	4	http://be.farnell.com/intern	€ 5,39
7	IRF3205SPBF MOSFET SMD	For Final (SMD)-purposes	17	http://be.farnell.com/intern	€ 1,74
8	Flatcable or Ethernet (board)plugs (4-W)	To connect the BUS-system (2 for each)	12	OAMK	€ -
9	HC-SR04 (I have one from Belgium)	To detect distance user	1	http://www.dx.com/nl/p/h	€ 3,00
10	Temprature-sensor	checking (mabe controlling)	2	http://www.dx.com/p/ardu	€ 5,00
11	Motion detector	To detect users for energy saving	1	http://www.dx.com/p/robo	€ 3,00
12	LM3940 1-A Regulator for 5-V to 3.3-V	Vcc for SC16IS750	10	http://be.farnell.com/texas	€ 1,20
13	Short female-female flatcable 4W	Connecting leds	8	http://be.farnell.com/texas	€ 5,60
14	3 way dipswitches	adressing for leds	6	http://nl.farnell.com/omron	€ 3,00
15	Ledstrip connector	to connect strip to board	8	http://fi.farnell.com/phoen	€ -
16	Flatcable 6 wire / meter	To connect devices/modules	8	http://fi.farnell.com/pro-po	€ 18,00
17	Capacitors 4µ7 16V	Current peaks	14	http://fi.farnell.com/panas	€ 0,36
18	PCF8574A	I2C addressing	10	http://fi.farnell.com/texas-i	€ 1,68
19	Lower header (7mm instead of 8,5mm) 10-pin	To connect NFC-reader	12	OAMK	€ -
20	SMD dipswitches	for addressing	10	http://fi.farnell.com/omron	€ 1,74
21	5mm Red LED	Power indication	10	http://fi.farnell.com/kingbr	€ 0,26
22	5mm Yellow LED	Reader indicaton	10	http://fi.farnell.com/kingbr	€ 0,26
23	Sockets 6wire	To connect devices/modules	5	http://fi.farnell.com/molex	€ 0,72
24	Power connector	To give the system power	2	http://fi.farnell.com/lumbe	€ 1,53
25	Interference capacitors 100nF	interference over 5V-lines	14	http://fi.farnell.com/kemet	€ 0,01
26	Screen or Tablet	GUI on fridge	1	OAMK (Web-application)	€ -
27	Arduino Mega	Controlling fridge	2	OAMK	€ -
28	RGB-ledstrips	To indicate needed foods	12	http://www.dx.com/p/12v-	€ 20,00
29	Compennts custom-Arduino-shield	RGB driver,TIP31, expierence,...	5	https://www.dfrobot.com/i	€ -
30	HC-SR04	To detect distance user	1	http://www.dx.com/nl/p/h	€ 3,00
31	Temprature-sensor	checking (mabe controlling)	3	http://www.dx.com/p/ardu	€ 5,00
32	NFC-readers	To read products	5	OAMK	€ 80,00
33	NFC-stickers	To identify ingredients	100	http://www.dx.com/p/0001	€ 15,00
		Total:	314	Total:	€ 224,82

Github:

All information, data and steps can be found here:

https://github.com/OAMK-Smart-Kitchen/Smart-Fridge-OAMK