14th IMEKO TC10 Workshop on Technical Diagnostics
New Perspectives in Measurements, Tools and Techniques
for system's reliability, maintainability and safety
Milan, Italy, June 27-28, 2016

# Interfacing External Sensors with Android Smartphones Through Near Field Communication

Tore Leikanger[1], Juha Häkkinen[1], Christian Schuss[1]

[1]*Circuits and Systems (CAS), University of Oulu, Erkki Koiso-Kanttilan katu 3, 90570 Oulu, Finland, firstname.lastname@ee.oulu.fi*

***Abstract* – In this paper, a new approach to communicate with inter-integrated circuit (I2C) enabled circuits such as sensors, over Near Field Communication (NFC) is presented and evaluated. The NFC-to-I2C interface was designed using a non-standard NFC command to control the I2C bus directly from a smartphone, which was controlling both read and write operations on the I2C bus. The NFC-to-I2C interface was reporting back the data bytes on the bus to the smartphone when the transaction was done. This system was tested in action both with write and read requests to a commercial microcontroller featuring a hardware I2C port, as well as reading a commercial I2C enabled humidity and temperature sensor. The results of the experimental tests of the system show that this approach gives an easy interface between the smartphone and external sensors, which is useful for the smartphone user.**

## I. INTRODUCTION

Near Field Communication (NFC) is an increasingly popular technology, which lately has been implemented in most medium to high-end Android smartphones. In smartphones, the technology is most used for bluetooth pairing to share data between the devices. Recently, NFC payment systems have been developed, embedded in a credit cards or by using a smartphone applications. Also, NFC tags are used for e.g. smart poster applications, and for ticketing systems. [1, 2]

Recent research has been exploring the possibility of using NFC as a measurement system. The NFC enabled sensors are usually dual-interface devices, embedding an NFC proximity inductive coupling card (PICC) core at one end and an inter-integrated circuit (I2C) slave core at the other [3, 4]. Another approach is to emulate the NFC PICC with an NFC proximity coupling device (PCD) circuit, and control this through a microcontroller at the sensor side [5]. Both of these approached have the draw-back of needing an active master device on the sensor side controlling both the sensor and the NFC circuit.

This paper addresses a new approach of using the NFC circuit as a bridge between a I2C slave and a smartphone acting as the I2C master. This removes the need of an active master device between the NFC circuit and I2C slave circuits, such as, sensors. This approach gives a more flexible solution, where "firmware" updates are done purely in the smartphone software.

This approach was experimentally verified as results are verified in this paper. The verification is done both by data transaction with a microcontroller over the I2C bus, and by reading a sensor both through an FPGA development platform implementing the NFC-to-I2C interface as well as verifying the readings with a commercial microcontroller featuring an hardware I2C port. The results of both tests are then compared to see that the transmission of the data throug the NFC-to-I2C interface was successful.

An example application of this NFC-to-I2C interface can be used is a feedback platform for the indoor environment. In this example scenario, a smartphone application would be programmed to upload the measurement data to a cloud service. If the humidity and temperature levels in the room are outside the comfort area more people would read the sensor system, and the temperature and/or humidity in the room can be adjusted.

This paper is divided into four sections. First, the similarities and differences between the NFC protocol and the I2C protocol are addressed, as well as how the I2C bus can be controlled over NFC. Secondly, the NFC core and the I2C core used in the experimental set-up is described. Then, the experimental set-up is described, and experimental results are presented. The paper is concluded by a discussion of the system and the results.

## II. NFC AND I2C PROTOCOLS

NFC and I2C are in many aspects similar protocols. Both NFC and I2C protocols have a very strong master and slave relationship, where the master always controls the bus and the slave only responds to the master commands. NFC slaves (PICCs) only responds when they are in their active state, while I2C slaves only responds if the command includes the correct address, thus enables multiple slave busses for both technologies. However, NFC is a single master multiple slave technology, while I2C is a multiple master multiple slave technology. Also, I2C devices can switch between being a master and a slave on the I2C bus, also not possible for NFC devices exept in peer-to-peer mode (a dual master no slave system). [6, 7]

### A. The used NFC protocol

The PICC core used in the system described in this paper is a NFC Forum type 2 tag, which is a subgroup of the NFC-A standard [8]. The PICC core features 7 unique identification (UID) bytes and 48 user memory bytes, and all standard NFC Forum commands for type 2 tags including read and write capability.

NFC-A, as other NFC technologies, communicates over a amplitude modulated 13.56 MHz electromagnetic field. PCD to PICC (reader to tag) communication is done with modified miller encoding, and PICC to PCD communication is done with manchester encoding on a 848 kHz subcarrier. The bit rate for NFC Forum type 2 tags is 106 kHz. [9]

### B. The I2C protocol

The I2C protocol is a two wire communication protocol in which the data transfer is synchronized on the clock line (SCL), and the data is transfered on the data line (SDA). For standard low-speed I2C systems (up to 400 kHz) the SCL and SDA lines are pulled up using 4.7 kΩ resistors, and clocking of the bus as well as data transfer is done by actively driving the lines to ground. [7]

I2C anticollision is solved by sending first a start contition to the I2C bus, the first sending a start contition controls the bus until a stop condition is sent. If another master is trying to use the bus, this master has to first wait for the bus to be free. [7]

When the master is sending data over the I2C bus, the master has to first send the I2C slave address (usually 7 bits) and rw flag indicating a read or a write request. Reading over the I2C bus is done by first writing a command to the I2C slave, and then reading. [7]

### C. Controlling the I2C bus over NFC

A non-standard command has been implemented in the NFC Forum type 2 core described above, exclusively to control the I2C bus. This command includes the I2C address, the rw flag, as well as the number of byte to read or write over the I2C bus, before the actual data bytes are transmitted. The PICC then responds with the actual bytes on the I2C bus, used to verify the data written or to return the data read over the I2C bus. The non-standard NFC I2C request is shown in Fig. 1, the NFC I2C responce is shown in Fig. 2, while the timing is shown in Fig. 3. Also, description of each byte in the NFC I2C request and responce are described in Table 1.

## III. THE DIGITAL DESIGN OF THE NFC-I2C INTERFACE

### A. The NFC core

The NFC core is based on the NFC Forum type 2 PICC described in the NFC Forum standard documents [8, 9].

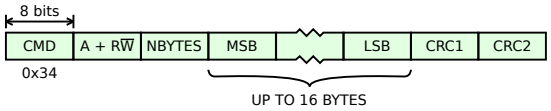


*Fig. 1. The NFC I2C request used to initiate the I2C serial communication. CMD is the command code, A+$R\overline{W}$ is the address and rw flag, NBYTES is the number of bytes, MSB is most significant byte, LSB is the least significant byte, and CRC1 and 2 are the cyclic redundancy check bytes*

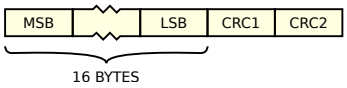


*Fig. 2. Response to the NFC I2C request. MSB is most significant byte, LSB is the least significant byte, and CRC1 and 2 are the cyclic redundancy check bytes*

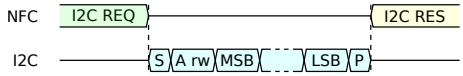


*Fig. 3. Timing of the NFC I2C request and responce, and the I2C transmission. S is the start condition, MSB is most significant byte, LSB is the least significant byte, and P is the stop condition*

*Table 1. Description of the NFC I2C request and response*

| Byte | I2C request | I2C response |
|---|---|---|
| CMD | Command code | |
| A + R$\overline{W}$ | 7-bit address and rw flag | |
| NBYTES | Number of bytes | |
| MSB | MSB of data | MSB of data |
| LSB | LSB of data | LSB of data |
| CRC1&2 | CRC bytes | CRC bytes |

For this tag to work, several sub-modules are needed, including a modified miller demodulator, CRC generator, request and responnce logic, as well as a manchester code modulator [9]. The submodules are connected as shown in Fig. 4. Further, as this is a type 2 PICC with 7 UID bytes, the core is designed around a finite state machine with the states IDLE, READY 1, READY 2, ACTIVE and HALT. The IDLE state is the initial state, i.e. the state the NFC core enters during a reset or a power on. The READY 1 and 2 states are the handshake/anti-collision states used by the PCD to initialize the PICC. The ACTIVE state is the main state of the core, where the actual data transfer between the PCD and PICC happens, such as reading from and writing to the memory. The HALT state is for putting the PICC to an inactive state while the PCD is communicating with other PICCs, if other PICCs are present in the field. [8]

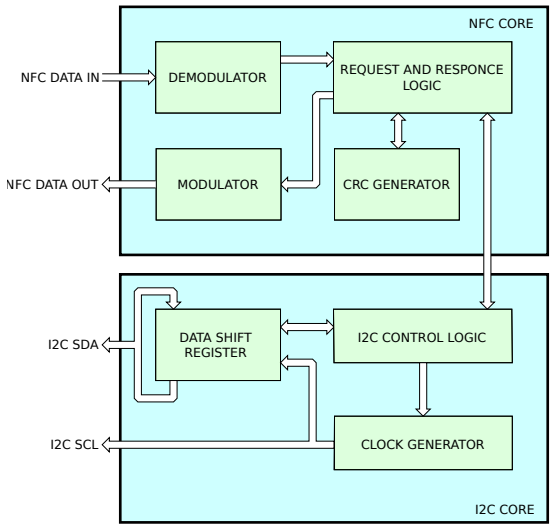


*Fig. 4. The NFC core and the I2C core, showing the connection between the cores.*

The demodulator is regenerating the data bits and shifting them into a data byte, befiore forwarding the byte to the request and response logic. When the request is completely sent, i.e. all request bytes including the CRC bytes are received, they are shifted into the CRC generator to confirm that the request is free of errors. The request and response logic continues then to interpret the request. Now, if the request is valid, the request and response logic must decide whether the PICC should respond or not depending on the state of the PICC. Which requests a PICC responds to and in which states are discussed in detail in the NFC Forum standard documents [8, 9]. In addition to the functionality described in the NFC Forum documents, the I2C request is added in this NFC core, as described above. If a response is to be sent to the PCD, the response is shifted into the Machester code modulator.

The NFC I2C request is available when the PICC is in the ACTIVE state. When this request is received, the NFC core initiates an I2C transfer (read or write) in the I2C core, and responds to the NFC I2C request with the data bytes in this transfer as discussed above. The connection between the NFC core and the I2C core is shown in Fig. 4.

### *B. The I2C core*

The I2C core features a shift register for shifting in and out data through the SDA line. Also a clock generator controlling the timing and pace of the data shifting is embedded in the core, generating the clock signal at the SCL line. This shift register is slightly modified to send a start signal at the beginning of a data transfer and a stop signal at the end of the data transfer, as well as sending ACK and NAK signals between data bytes when the I2C core is reading over I2C. Both SDA and SCL line are dual direction line, with internal pull-down transistors and external pull-up resistors. This makes data transfer in both directions possible, as well as enables the I2C slave to delay the data transfer by pulling the SCL line down. The connections between the different parts of the I2C core is shown in Fig. 4.

The data shift register as shown in Fig. 4 latches data from the I2C logic when requested, and shift out this data on every falling edge on the SCL line. Futher, the data on the SDA line is shifted into this shift register on every rising edge on the SCL line. Note that data in the shift register is stationary when the SCL line is pulled low by a I2C slave device. Between each byte, i.e. during the ACK/NAK time, the data in the shift register is made available to the I2C control logic while the next byte is latched in from the I2C control logic.

### *C. Waveforms at the inputs and outputs of the NFC and I2C cores*

The waveforms at the NFC and I2C pins during an NFC I2C request and response as measured by a logic analyser is shown in Fig. 5. In this example waveform, the NFC-to-I2C interface is requested to write a data byte with value 0xE3 to an I2C slave with address 0x40 by the Samsung Galaxy A3 smart phone. This is the same command as is used when a temperature measurement by the SHT21 humidity and temperature sensor is initiated [12].

As can be seen in Fig. 5, the I2C core initiates the I2C communication at once when the NFC I2C request has been received, and initiates the NFC I2C response immidiately after the I2C communication is done. Because the maximum number of bytes in the I2C communication has been set to 16 in this NFC-to-I2C interface, the time between the NFC I2C request and response is within the limits in the NFC Forum standard [9].

## IV. EXPERIMENTAL SET-UP AND RESULTS

The device implementing the NFC Forum type 2 core is an FPGA development platform specifically targeting NFC development, featuring an NFC analog front end (influenced by the Chameleon mini project [10]), power circuitry, electrically erasable programmable read only memory (EEPROM), and an Altera Max 10 field-programmable gate array (FPGA). This device also has 16 general purpose in-out pins broken out on the edge of the circuit board, where the I2C bus is included. A block diagram and a picture of the device can be seen in Figures 6 and 7.

The experimental set-up can be seen in Fig. 8. The multiple master feature on the I2C bus makes it possible to read the same sensor using the NFC-to-I2C interface and the microcontroller at the same time, in order to minimize the error due to delay of the measurements. The microcontroller is programmed to work as both an I2C master and a

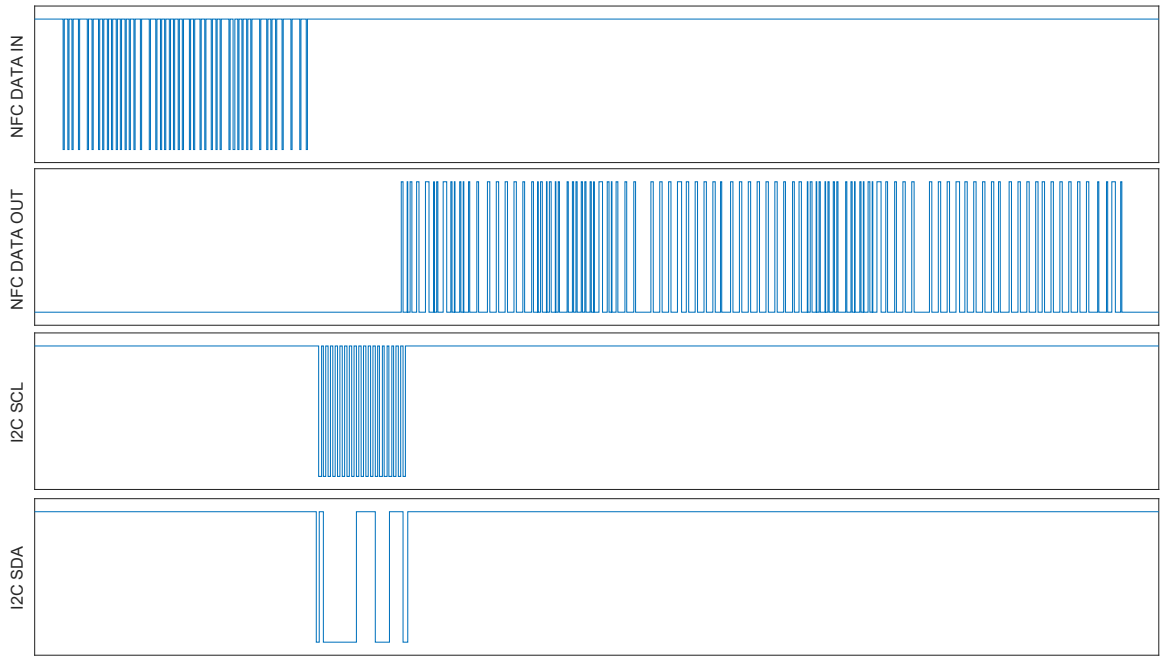


*Fig. 5. The waveforms at the NFC and I2C pins during an NFC I2C request and response as measured by a logic analyser.*

slave. This enables two separate tests to be done. The first is the echo test, where the NFC-to-I2C interface transmits a data byte to the microcontroller before the NFC-to-I2C interface reads the same data byte from the microcontroller. This test is then successfull if the data byte remaines the same during the whole test. The second test is then to use the microcontroller to read the sensor each time the sensor has been read by the NFC-to-I2C interface. The microcontroller also reports the data to a host computer through an USB connection. The microcontroller chosen for this application is the Atmel Atmega328P used in the Arduino platform [11], and the sensor is the Sensirion SHT21 humidity and temperature sensor [12]. The smartphone used to perform the tests is the Samsung Galaxy A3.

The Android programming code needed to implement the NFC I2C request to write and read is done with the code below. This example code initiates a temperature measurement on the Sensirion SHT21 humidity and temperature sensor, and then requests the temperature to be read.

```
byte[] init_temp = {(byte)0x34,
     (byte)0x80,(byte)0x01,(byte)0xE3};
byte[] read_data = {(byte)0x34,
     (byte)0x81,(byte)0x03};

tag.transceive(init_temp);
byte[] data = tag.transceive(read_data);
```

The raw data received is then converted to degrees celsius


*Fig. 6. Picture of the FPGA based NFC development device.*

as instructed in the datasheet [12] by the commands

```
double raw_temp = (data[0]&0xFF) / 256.
             + (data[1]&0xFC) / 65536.;

double temp = -46.85+(175.72*raw_temp);
```

To read the humidity, init_humid = {0x34, 0x80, 0x01, 0xE5} was used [12].

In the echo test, one data byte is written to the the microcontroller before reading the same data byte, both operations through the NFC-to-I2C interface. The data byte is randomly generated by the Android application controlling the NFC-to-I2C interface. The android application then reports the data sent and the data received as hexadecimal

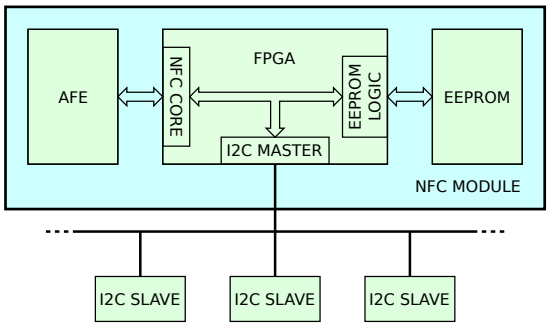


*Fig. 7. The different blocks on the NFC tag module. AFE is the analog front end.*

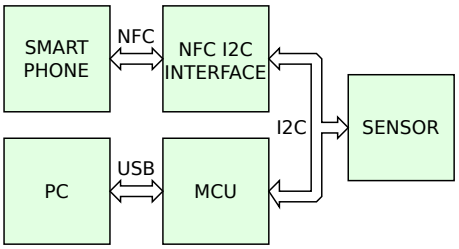


*Fig. 8. Experimental set-up for verifiyng the functionality of the circuit. MCU is the microcontroller.*

*Table 2. I2C echo test. A random data byte is sent to the microcontroller, and then read by the phone. The phone reports the value of both the sent and the received data byte, and the microcontroller reports the received data byte.*

| Sent (phone) | Received (phone) | Reported (MCU) | Pass |
|---|---|---|---|
| 0x3A | 0x3A | 0x3A | Yes |
| 0xE6 | 0xE6 | 0xE6 | Yes |
| 0xCD | 0xCD | 0xCD | Yes |
| 0x84 | 0x84 | 0x84 | Yes |
| 0xC2 | 0xC2 | 0xC2 | Yes |
| 0x20 | 0x20 | 0x20 | Yes |
| 0x49 | 0x49 | 0x49 | Yes |
| 0x5A | 0x5A | 0x5A | Yes |
| 0xCC | 0xCC | 0xCC | Yes |
| 0xD8 | 0xD8 | 0xD8 | Yes |

values. Also, the microcontroller reports the value of the data byte to the host computer. The test was done 10 times, and the results of each test was evaluated. The results of the data reported by the android application and the microcontroller, as well as the pass or fail indicator is shown in Table 2. As can be seen from the table, the system passed all tests, meaning that the datatransmission was successfull in both directions.

*Table 3. Temperature readings by the smartphone through the NFC-to-I2C-interface and by the microcontroller. The raw data in hexadecimal are listed in paranthesis.*

| Temperature readings | |
|---|---|
| NFC-I2C interface | Microcontroller |
| 21.59 °C (0x63B4) | 21.60 °C (0x63B8) |
| 21.59 °C (0x63B4) | 21.59 °C (0x63B4) |
| 21.59 °C (0x63B4) | 21.59 °C (0x63B4) |
| 21.58 °C (0x63B0) | 21.58 °C (0x63B0) |
| 21.58 °C (0x63B0) | 21.58 °C (0x63B0) |
| 21.58 °C (0x63B0) | 21.59 °C (0x63B4) |
| 21.58 °C (0x63B0) | 21.59 °C (0x63B4) |
| 21.58 °C (0x63B0) | 21.58 °C (0x63B0) |
| 21.57 °C (0x63AC) | 21.57 °C (0x63AC) |
| 21.57 °C (0x63AC) | 21.58 °C (0x63B0) |

The second test was to read the sensor using the NFC-to-I2C-interface and the microcontroller, and to compare the results. This was done by programming the microcontroller to read the sensor 1.5 s after a specific request was sent from the smartphone (through the NFC-to-I2C interface). 1.5 s was chosen because Sensirion recommends a delay of at least one second between each time the sensor is read. In the test scenario, the smartphone reads the sensor, and then requests the microcontroller to read the sensor after a delay of 1.5 s. This test was also done 10 times, with random delays of much more than 10 s. The sensor data as read through the NFC-to-I2C interface as well as the sensor data as read by the microcontroller can be seen in tables 3 and 4 for the temperature and relative humidity respectively. The results have small differences due to noise and minute enviromnental fluctuations. The measured values are $21.58\,^{\circ}\mathrm{C} \pm 0.01^{\circ}\mathrm{C}$ and $27.6\,\%RH \pm 0.4\%RH$ by the smartphone through the NFC-to-I2C interface, and $21.58\,^{\circ}\mathrm{C} \pm 0.02^{\circ}\mathrm{C}$ and $27.6\,\%RH \pm 0.4\%RH$ by the microcontroller. The difference in the measurements through the NFC-to-I2C interface and by the microcontroller are simmiliar enough to say that the error is due to the accuracy tolerance in the sensor reading rather than the interface. The uncertainty in the sensor are significantly larger than the error in these measurements, reported in the datasheet as $\pm 0.3\,^{\circ}\mathrm{C}$ and $\pm 2\,\%$RH [12].

## V. CONCLUSION

In this paper, a new approach to communicate with sensors or other I2C devices with NFC enabled smartphones has been presented and evaluated. This approach eliminates the need of programming of microcontrollers on the sensor side of the system, by implementing an I2C master core in the NFC circuit, controlled by a simple non-

*Table 4. Relative humidity readings by the smartphone through the NFC-to-I2C-interface and by the microcontroller. The raw data in hexadecimal are listed in paranthesis.*

| Relatice humidity readings | |
|---|---|
| NFC-I2C interface | Microcontroller |
| 27.47 %RH (0x448E) | 27.47 %RH (0x448E) |
| 27.71 %RH (0x450A) | 27.94 %RH (0x4586) |
| 27.31 %RH (0x443A) | 27.27 %RH (0x4426) |
| 27.51 %RH (0x44A2) | 27.51 %RH (0x44A2) |
| 27.47 %RH (0x448E) | 27.51 %RH (0x44A2) |
| 27.51 %RH (0x44A2) | 27.67 %RH (0x44F6) |
| 27.87 %RH (0x455E) | 27.82 %RH (0x4546) |
| 27.74 %RH (0x451E) | 27.74 %RH (0x451E) |
| 27.47 %RH (0x448E) | 27.67 %RH (0x44F6) |
| 27.47 %RH (0x448E) | 27.67 %RH (0x44F6) |

standard NFC command. The programming is thus moved completely to the programming of the Android application, making the NFC circuit more flexible and easier to work with. It has also been mentioned that this approach reduces the delay in data transfer from the smartphone to an I2C chip, because the NFC circuit no longer needs to be polled from both sides for the data to be transmitted. The digital design of the system has been described. Further this system has then been experimentally evaluated and tested, and it has been shown that the data sent and received through the NFC-to-I2C interface indeed is valid. These tests has been done with a commercially avalable humidity and temperature sensor as well as a microcontroller featuring a hardware I2C port.

REFERENCES


[1] “About Near Field Communications", http://www.nearfieldcommunication.org/about-nfc.html. Accessed 04.03.2016.

[2] “Ways to Use Near Field Communications", http://www.nearfieldcommunication.org/using-nfc.html. Accessed 04.03.2016.

[3] J. Maxa, T. Krachenfels and H. Beikirich, “Near Field Communication Interface for a Packet-Based Serial Data Transmission Using a Dual Interface EEPROM", Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on, 2015, pp. 1-4

[4] “NTAG I2C - Energy harvesting NFC Forum Type 2 Tag with field detection pin and I2C interface datasheet NT3H1101/NT3H1201", technical report, 2015.

[5] M. Hillukkala, M. Heiskanen and A. Ylisaukkooja, “Practical implementations of passive and semipassive NFC enabled sensors", Near Field Communication, 2009 First International Workshop on, 2009

[6] S. Ahson and M. Ilyas, “RFID handbook : applications, technology, security, and privacy”, Boca Raton: CRC Press, 2008.

[7] P. Corcoran, “Two Wires and 30 Years : A Tribute and Introductory Tutorial to the I2C Two-Wire Bus",Consumer Electronics Magazine, IEEE, 2013, vol. 1, issue 3, pp. 30-36

[8] “Type 2 Tag Operation", NFC Forum Technical Specification, Rev. 1.2, 2014

[9] “NFC Digital Protocol", NFC Forum Technical Specification, Rev. 1.1, 2014

[10] “The Chameleon project", https://github.com/emsec/ChameleonMini/wiki. Accessed 26.02.2016.

[11] “Arduino/Genuino UNO", https://www.arduino.cc/en/Main/ArduinoBoardUno. Accessed 26.02.2016.

[12] “Datasheet SHT21. Humidity and Temperature Sensor IC", technical report, 2014.