Sajid Hameed

IOT SENSOR NODE DESIGN WITH NFC DATA COMMUNICATION USING OFF-THE-SHELF COMPONENTS

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Bachelor of Engineering, Information Technology
October 2023

ABSTRACT

Centria University of Applied Sciences	Date	Author
	October 2023	Sajid Hameed

Degree program Bachelor of Engineering, Information Technology	
Name of thesis IOT SENSOR NODE DESIGN WITH NFC DATA COMMUNICATION USING OFF-THE-SHELF COMPONENTS	
Centria supervisor Jari Isohanni	**Pages** 40 + 13
Instructor representing commissioning institution or company. Kari Halonen, Aalto University	

Near-field communication (NFC) technology was introduced and specified for short-range wireless connectivity during the first decade of the 21st century. It allows the exchange of information wirelessly between two radio frequency identification (RFID) compatible devices. In addition to its widespread use in mobile payments, data transfer, secure access control, digital authorization, and authentication, the NFC protocol has expanded its applicability to wearable medical devices. Recent advances in integrated electronics have enabled battery-free operations by optimizing the power efficiency of medical devices and implants. These gadgets absorb energy from their surroundings and store data on-chip memory for later retrieval.

This project describes the development of an internet-of-things (IoT) sensor node utilizing an Arduino-based NFC system to transmit pre-recorded data using the NFC handshake protocol. A sensor node hardware platform is conceived with battery operated power management block and built to sense and store environmental data with timestamps in the local memory. The validation of the platform's functionality enables the formulation of specifications to design an on-chip NFC communication block. The work further contributes to the development of self-powered medical sensor nodes based on the IoT employing printed integrated circuit technology. The project is managed by Professor Kari Halonen's research group at the research and development (R&D) facilities of Aalto University.

Key words
Biomedical sensor node, data recording and organization, environmental monitoring, internet of things (IoT), near field communication (NFC), radio-frequency-identification (RFID), wearable medical devices, wireless communication.

CONCEPT DEFINITIONS

ADC	Analog to Digital Converter
ASIP	Application Specific Instruction Set Processor
ASIC	Application Specific Integrated Circuit
CS	Chip Select
EEPROM	Electrically Erasable Programmable Read Only Memory
IDE	Integrated Development Environment
I2C	Inter-integrated Circuit
IEC	International Electrotechnical Commission
IoT	Internet of Things
ISP	In System Programming
LED	Light Emitting Diode
LiPo	Lithium-Ion Polymer
LSB	Least Significant Bit
LDO	Low Drop Out
MISO	Master In Slave Out
MOSI	Master Out Slave In
MSB	Most Significant Bit
NFC	Near Field Communication
PCB	Printed Circuit Board
PC	Personal Computer
RAM	Random Access Memory
RFID	Radio Frequency Identification
RTC	Real Time Clock
RD	Research and Development
RF	Radio Frequency
SD	Secure Disk
SCK	Serial Clock
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SS	System Select
SISO	Serial In Serial Out

TWI	Two Wire Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VNA	Vector Network Analyzer
ROM	Read Only Memory

ABSTRACT
CONCEPT DEFINITIONS
CONTENTS

FIGURES

TABLES

CODES

1 INTRODUCTION

Due to security reasons, the widespread use of near field communication (NFC) in mobile payments, secure access control, digital authorization, and authentication has confined its readable range protocols. The aim of this thesis research was to demonstrate how wearable medical devices leverage NFC technology. The use of NFC on the internet-of-things (IoT) based sensor nodes has gained attention after its integration in smartphone devices. There have been efforts to extend the communication range of NFC technology for data exchange and wireless power transmission. The applications the NFC based devised can considerably be extended by increasing the usable distance. For example, the extended use of NFC technology in wearable biomedical devices. An important benefit of NFC technology is that it makes it possible to monitor physiological parameters in real-time when users are engaged in daily activities. NFC technology does not require complicated procedures like pairing, transmission control protocols and high-power consumption as in the case of Bluetooth and Wi-Fi.

In this work the challenges to adopt the NFC in the medical field of wearable medical devices due to its limited communication distances are accessed and analyzed. An Arduino based IoT sensor node was designed and developed with NFC data communication using off-the-shelf components. The sensor node is capable of sensing, recording, and transmitting important environmental parameters such as temperature and humidity. A power management block with lithium-ion polymer (LiPo) battery was developed to ensure wireless applications of the proposed IoT sensor node. The validation of the platform's functionality enables the formulation of specifications to design an on-chip NFC communication block. Recent technological advancements have made it possible to optimize the power consumption of electronic devices to reduce the device size and improve the operation time. The work further contributes to the future developments of self-powered medical sensor nodes based on the IoT employing printed integrated circuit technology.

The background information and literature review of this research is presented in Chapter 2. Chapter 3 elaborates the materials and techniques carried out to conduct this study. The results are explained in Chapter 4. Finally, Chapter 5 concludes the study and provides the future prospects.

2 TECHNICAL BACKGROUND

This chapter provides the background and literature review of radio-frequency-identification (RFID) and NFC and discusses their operation modes and tag types. It also provides a review of the hardware and software components used to develop the proposed IoT sensor node.

2.1 Near field communication (NFC)

RFID was initially introduced during World War II as part of the identification system known as "friend or foe,". It was developed to authenticate an allied aircraft and it has continued to serve the same purpose afterwards. RFID employs both read-and-write functions to accomplish wireless data communication. The reader can modify and update the information stored in a tag at any time of a tracking cycle (Lehpamer 2012, 1.)

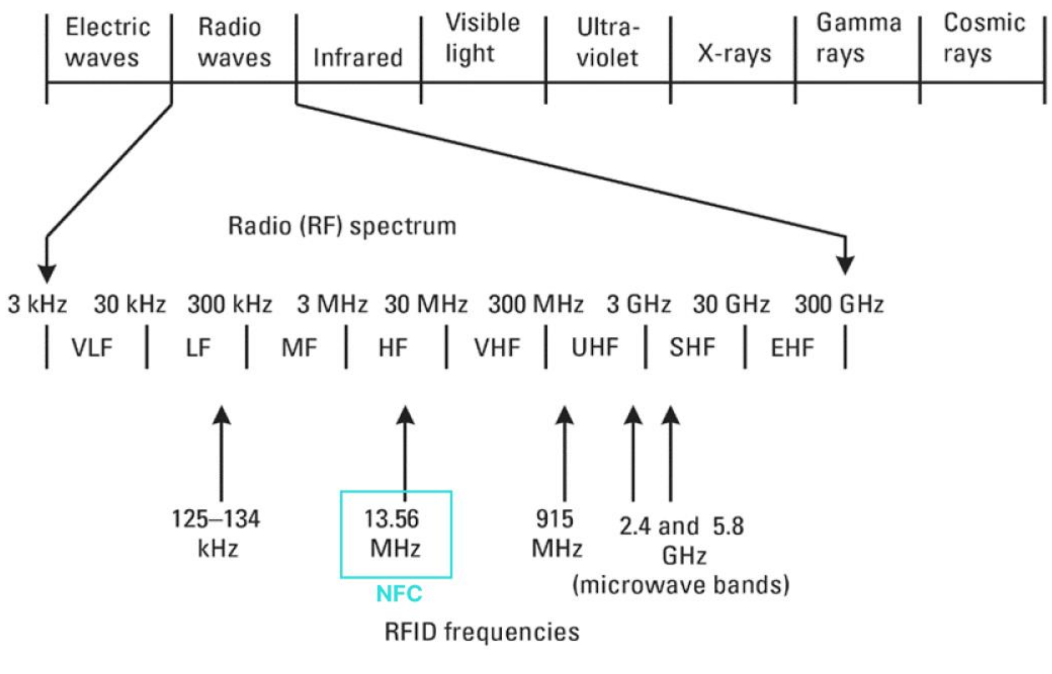


FIGURE 1. Electromagnetic Spectrum (Lehpamer 2012, 5.)

In FIGURE 1 The electromagnetic spectrum encompasses all the distinct manifestations of electromagnetic energy, ranging from very low frequency electric waves to very high frequency cosmic rays. Long wavelengths are associated with low frequency while short wavelengths are associated with high frequency. Radio waves have higher frequency as compared to electric waves, followed by radio waves, microwaves, visible light, and ultraviolet radiation on electromagnetic frequency spectrum. The radio frequency (RF) part of the electromagnetic spectrum has frequencies that range from about 3 kHz to 300 GHz (Lehpamer 2012, 4-5.) In the last couple of decades, the RFID systems have witnessed a significant expansion in the range of applications. There are several different tag families available in the market, and each family fulfills a different group of requirements. An RFID tag consists of an antenna, a memory to record information and an on-chip support circuitry. Some of the tags contain batteries and are called active tags. Whereas passive tags do not contain batteries. (Perret and Etienne 2014, 3.)

In the late 1800s, when Thomas Edison was conducting research on radio frequencies, he had the idea of NFC, which stands for near-field communication. The NFC was first developed in 1983 by Charles Walton. In 2002 marked, Sony and NXP Semiconductors had a collaboration to develop an innovative form of NFC technology. The official NFC forum was founded in 2004, when Sony, Philips, and Nokia centered their efforts on incorporating NFC into their production processes. In 2006, Nokia unveiled the world's first phone that was equipped with NFC. This event marked the beginning of the use of NFC as a medium for information exchange, in addition to its traditional use as a payment method. (Kang SG, Song MS, Kim JW, Lee JW, Kim J 2021, 2.)

2.1.1 NFC Forum

The NFC Forum is an association that was established for the goal of defining the NFC standards built on the standards established by international standard organization (ISO) and international electrotechnical commission (IEC). The NFC Forum was established with the intention of easing the process of implementing NFC technology and fostering its spread around the world. To make sure that different products and services can communicate with one another, the NFC Forum promotes the development of new NFC technology as well as the standardization of existing NFC products. (Vedat Coskun, Kerem Ok, Busra and Ozdenizci 2012, 9.)

The NFC Forum selected the 13.56 MHz frequency because it offers a good balance between range and performance for short-range wireless communication. The NFC Forum was pioneer to develop the peer-to-peer (P2P) standards in 2009. These standards make it possible to send and receive data using NFC equipped. From 2010 to 2016, several mobile applications were developed for safe financial transactions using NFC equipped Android phones, smart tags, and Felica cards. by 2020, the NFC has already been used in a wide range of sensors, Internet of Things (IoT) nodes, and smart home applications. (Kang SG, Song MS, Kim JW, Lee JW & Kim J 2021, 2.)

2.1.2 NFC Modes

There are three distinct NFC modes; reader/writer mode, peer-to-peer mode, and card emulation mode. These modes are further described below.

The reader and writer modes of the tag are described in FIGURE 2, when both the NFC smartphone and the NFC tag are working in reader/writer mode, they can communicate with one another and share information. This is because both devices can read and write information. This mode may be divided into two separate sub modes. Each sub mode has its own unique characteristics. If a mobile phone is set to read mode, then it will be able to access the information stored on an NFC tag. Whereas in a write mode, the mobile phone can write data to the NFC tag. (Coskun, Ok, Busra & Ozdenizci 2012, 9.)

Tag Reader/Writer
NFC

FIGURE 2. Reader/Writer Mode.

FIGURE 3 depicts the peer-to-peer communication between two NFC enabled devices. In this mode, two mobile devices can communicate with one another and share data using NFC communication protocol. Both NFC devices are powered by their own power supply to complete the communication process. It is necessary for the second device to maintain its listening position to receive the data transmitted by the first device. Once the first device has finished its transmission, only then will the second device start send data. (Coskun, Ok, Busra & Ozdenizci 2012, 9.)

FIGURE 3. Peer-to-peer Mode.

FIGURE 4 demonstrates the operation of an NFC device in card emulation mode. A mobile device with NFC capability may be able to behave as a contactless smart card in this mode. Most of the contactless smart cards that have been successfully emulated are credit cards and debit cards. (Coskun, Ok, Busra & Ozdenizci 2012, 9.)

FIGURE 4. Card Emulation Mode.

2.1.3 NFC tag types

The NFC Forum has standardized five distinct types of tags, all are governed by the defined control protocols. The ISO-14443A standard serves as the basis for types 1, 2, and 4, whereas the ISO-18092 standard provides the foundation for type 3. All types of tags can be purchased from a broad variety of vendors. (Igoe, Coleman & Jepson 2014, 17.)

Type 1 tag adheres to the ISO-14443A specification. It can either have read-only access or read/ write access. The range is supported from 96 bytes to 2 kilobytes of memory. It has a baud rate of up to 106 KB. However, data collisions are not protected in this data type. Examples include the "Broadcom BCM20203" and the "Innovision Topaz". (Igoe, Coleman & Jepson 2014, 17.)

Type 2 tags are the same as type 1 tags and are built on the ISO-14443A-based NXP/Philips Mifare Ultralight Tag ISO-14443A standard. It has the option of either being read-only or having read/write access. It supports memory range from 96 bytes up to 2 data transfer rates up to 106 KB per. The tags are featured with anti-collision support. An example of this tag type is the NXP Mifare Ultralight. (Igoe, Coleman & Jepson 2014, 17.)

Type 3 tags are based on the Sony FeliCa tag standards (ISO-18092 and JIS-X-6319-4). However, they lack FeliCa's capabilities for encryption and authentication. These are factory configured to operate either in read-only or in read/write modes. It is commonly used for peer-to-peer communication. It has variable memory with a 1MB exchange limit. Data communication speed is either 212 or 424 Kbps. It also supports protection against data collisions. An example of this tag is Sony FeliCa. (Igoe, Coleman & Jepson 2014, 17-18.)

Type 4 tags are based on the NXP DESFire tag (ISO-14443A) specification similar to type 1 tags. These are also factory-configured either in read-only or in read/write mode. They have memory capacity of 2, 4, or 8KB. These support variable memory up to 32KB for each exchange. Three communication speeds of 106Kbps, 212Kbps, and 424 Kbps are supported. These tags are protected against collisions. NXP DESFire and SmartMX-JCOP are examples of tag 4 types. (Igoe, Coleman & Jepson 2014, 17-18.)

Type 5 The fifth well-known tag type is supported by most of the devices. This type is exclusive to NXP Semiconductors and is the most prevalent in NFC use as present Tag Mifare Classic (ISO–14443A). It has various memory choices like 192, 768, and 3,584 bytes. It supports the transmission data rate of 106 Kbps and provides anti-collision support. NXP Mifare Classic 1K, the Mifare Classic 4K, and the Mifare Classic Mini are examples of these tag types. (Igoe, Coleman & Jepson 2014, 18.)

2.2 Microcontroller

A microcontroller is an application-specific instruction-set processor (ASIP) that can act as a link between a general-purpose central processing unit (CPU) and an application-specific integrated circuit (ASIC). An application-specific integrated circuit, or ASIP, is a programmable processor made for a group of related applications, like embedded control, digital signal processing, or telecommunications. The person who makes this kind of machine can change the data path to fit the needs of the application class. This could mean adding certain functional units for processes that happen frequently and getting rid of other units that are not used very frequently. The microcontroller is a central processing unit made for embedded control purposes. It is used in different applications like PC Keyboard, Mobiles phones, satellite phones and in different instruments. (Dawoud & Shenouda 2022,134.)

Microcontroller is a full computer system that has been optimized for hardware control and has the CPU, memory, and all I/O peripherals on a single chip. Because of this, the chip can work as a single unit. Serial clock and power, these are two requirements for a microcontroller to start work. Compared to parts that are linked to the outside of the chip, parts that are on the same chip as other parts make processing go faster. This is because internal I/O ports on the same chip as other parts take less time to read or write data. Input and output devices, as well as memory, are often all put on a single chip for most of the different kinds of microcontrollers. (Dawoud & Shenouda 2022,135.)

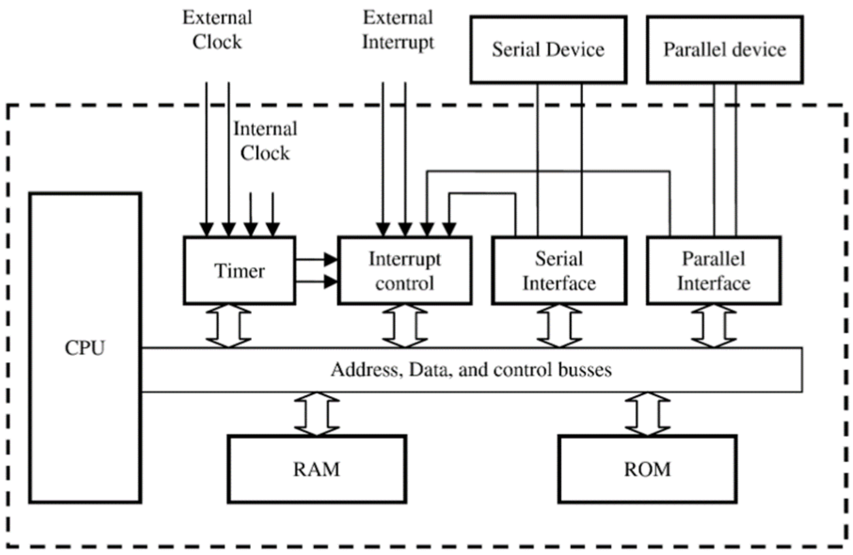


FIGURE 5. Architecture of Microcontroller (Dawoud, Shenouda 2022,135.)

FIGURE 5 shows the architecture of microcontroller. With a timer module, the microprocessor can do things in predetermined amounts of time. Data moves (synchronously or asynchronously) between the microcontroller and other devices via Serial I/O, like a PC or another microcontroller. analogue input and output, which can be used to get information from sensors. This can be used to control things and find out things. It has interrupt capability. Connections between the ROM or RAM inside the computer and the external bus. It has tools for solving bugs and a built-in monitor. It can be used to get data from different devices like sensors and motors. It processes the collected data for required result and for implementation of required tasks, it uses the output devices. (Dawoud, Shenouda 2022,135.)

2.2.1 Internal structure of microcontroller

All microcontroller chips must contain a certain set of components (resources) for chips to work properly. The CPU, input/output units (ports), memory units, serial communication, timers, watchdog timers, reset and brownout detectors, oscillators, and, in many cases, analogue-to-digital converters are among these resources. The input/output unit (ports) and memory unit consist of the computer's working memory, while the central processing unit (CPU) is its brain. In addition to these hardware resources, every microcontroller also needs software resources, commonly known as programs. (Dawoud & Shenouda 2022,135.)

2.2.2 Central processing unit (CPU)

The CPU is the most important part of any microcontroller. It runs code correctly based on what the user wants. There are memories and registers. The designer's code can run correctly because of these registers and memory. A microprocessor is the core of every microcontroller, and every company that makes them uses it as a key part. Intel's 8051 microprocessor, for example, has a CPU core that is an 8085. The core of Motorola's M68HC11 microcontroller, which is like this, is a 6800 microprocessor. (Dawoud & Shenouda 2022,137.)

2.2.3 Memory unit

All microcontrollers have internal memory as a basic element of their structure, which is located on the chip. This memory is used by a microcontroller to program and store data. In cases where more memory is required by some applications, external memory units are used by microcontrollers due to the low capacity of internal on-chip memory. These external memories are in the form of semiconductor memory chips. These memories are known as RAM and ROM. There are two types of memories: one is program memory, which is used for program instructions. The second type of memory is data memory, which is used for data storage. Static Random Access Memory (SRAM) offers high-speed, volatile data storage, enabling rapid access for processors in devices. Electrically Erasable Programmable Read-Only Memory (EEPROM) is non-volatile, retaining data without power, vital for microcontrollers to store configuration and critical information, ensuring reliability and functionality across power cycles. (Dawoud & Shenouda 2022,137-138.)

2.2.4 Input-output unit

A machine may perform tasks when its CPU, internal memory, and lines are all integrated onto a single chip. The device can do some tasks if it can communicate with objects outside of itself. The computer makes use of buses. A module with numerous memory slots connects the CPU to both the internal data bus and devices outside of it. To be more precise, one end of these memory locations is attached to the chip's pins, and the other end is connected to the chip's internal data bus. The locations where these connections take place are referred to as ‘ports,’ and the objects that link to the outside world are referred to as ‘I/O devices’ or ‘computer peripherals.’ Each port has a specific position for both reading and writing. When reading the port address, the microcontroller receives data. Data is transmitted to the port address when it is written. The three levels of a microcontroller's connection to its surroundings are ports, interfaces, and accessories. (Dawoud & Shenouda 2022,138.)

2.2.5 Ports

The terminals of the microcontroller are identical to the microprocessor's external lines. Each port can only communicate with one device, whereas each bus can connect to multiple devices. In other words, the port is like a bus in that only one device (either an input or output device) can be connected to it simultaneously. A microcontroller's port is a collection of receptacles, like a bus is a collection of lines. The microprocessor registers are where the buses begin, and the port is a data register within the microcontroller. The lines connect the central processing unit to the remainder of the computer. Similarly, the registers reveal how the CPU communicates with the rest of the world. Ports can either receive or transmit data, or they can do both. (Dawoud & Shenouda 2022,139.)

2.2.6 Serial communication

The ability to send and receive data in the form of words or bytes between the computer system and the outside world is made possible by I/O devices. This approach, known as "parallel communication”. It has several drawbacks, such as the requirement of numerous lines (or channels) to send data. The number of bits in a word determines how many lines are required. Serial communication allows sending and receiving serial data from external devices through a port. One bit at a time, the serial connection delivers data words from the computer to the output. This process is known as "converting from parallel to serial." By taking in external data one bit at a time, joining those bits to form words, and then giving those words to the computer system, it similarly converts serial data to parallel data. (Dawoud & Shenouda 2022,140.)

2.2.7 Clock oscillator

The processor executes the program stored in memory at a set rate. The frequency at which the clock oscillator operates determines how quickly instructions are executed. The time is controlled by the clock. The clock is responsible for ensuring that the system timer, CPU machine cycle, and central processing unit (CPU) each have the time they require. An internal RC oscillator, an oscillator that incorporates a quartz crystal, an LC resonant circuit, or an RC circuit as a time component are all possible choices for the clock oscillator. (Dawoud & Shenouda 2022,140.)

2.2.8 Timer

When dealing with a microprocessor, timers are typically the component that is responsible for generating a consistent delay. Timers are used in microcontrollers for a great many different purposes. The microcontroller system makes use of the timer for a variety of purposes, including timing events, counting events (both external and internal), and several other tasks. (Dawoud, Shenouda 2010,140.) The watchdog timer monitors the error-free operation of the microcontroller. A free-running counter functions as a timer. The watchdog timer waits for a tick generated by the software, which indicates that the program is operating accurately, as opposed to generating one itself. If a pulse does not reach the watchdog within a predetermined amount of time, the watchdog timer will cause the microcontroller to reset. (Dawoud & Shenouda 2022,141.)

2.2.9 ADC converters

Most control systems use analogue signals, which are very different from the 0 and 1 signals that a microprocessor can understand. The information must be changed into a digital form so that a microcontroller can understand. This is done with an analogue-to-digital converter (ADC). Under software control, the ADC block changes analogue data into binary numbers and sends them to a CPU block so that the CPU block can do more work with them. Most controls have pulse-width modulators that can be used with an external RC filter to get an analogue signal. (Dawoud & Shenouda 2022,141.)

2.2.10 Reset and brownout detector

The controller's reset circuit makes sure that all its parts and control circuits start up in a predetermined state and that all the necessary registers are set up correctly. Resets can be started by using the watchdog timer, the brownout detector, or the microcontroller's power-on procedure. The controller can stop working if the data in the registers and memories gets corrupt. The brownout detector is a circuit that monitors the voltage coming from the power source and resets the processor if there is a brief drop in voltage. If a failure causes the clock to slow down below a predetermined frequency, a clock monitor detector may also cause a reset. (Dawoud & Shenouda 2022,141-142.)

2.3 Communication protocols

There are two main types of communication protocols, called SERIAL and PARALLEL. Each has its own specifications. Serial communication is the process of sending data over a communication route one bit at a time. This is different from parallel communication, which is when multiple bits are sent at the same time over a link with several parallel lines. When compared to parallel interfacing, serial communication is easy and doesn't require a lot of extra hardware. Serial communication is used a lot due to its less hardware and less complexity. It is a way for two or more devices to talk to each other. Usually, one device is a computer or a microcontroller, and the other can be a modem, a printer, another computer, another microcontroller. One of the most important ways that microcontrollers are used is for serial communication. Many microcontrollers can communicate to each other in a series using one or more methods I2C, SPI, single-bus interface and UART (Dawoud & Shenouda 2020,1-2.)

2.3.1 I2C communication interface

Inter-integrated circuits, also called two wire interfaces (TWI), are used for interconnections between two ICs or two PCBs. It is being used widely due to its low cost, simple, internal clock, and the fact that most microcontrollers have an I2C interface. Its data rate is 100kb/s, 400kb/s and 1.34Mb/s on different data speed modes. Its working range depends on the number of nodes connected to it and the data rate. In a PCB, its normal range is less than a foot, and at the lowest data rate, its range is 10m. More than one master can be handled, but only one master can use the bus at one time. 10m. More than one master can be handled, but only one master can use bus at one time. Multiple nodes can be connected, and the number of nodes depends on the medium capacitance. This interface uses external pull-up resistors. (Louis E. &Frenzel 2015,65-66.)

2.3.2 Serial peripheral interface (SPI)

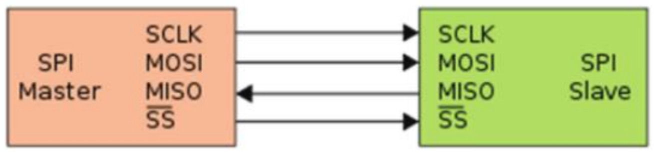


FIGURE 6. SPI Interface (Dawoud, Shenouda 2020,193.)

FIGURE 6 defines a communication interface for synchronous serial communication over short distances. It is made up of three communication wires. One wire is for master-slave communication, while the other is for clock oscillations. SS (slave select) is a fourth wire that is typically used for sending and receiving data between many integrated circuits. The number of slaves that can be linked to a bus is theoretically unlimited, but the number of slaves that may be attached is limited by the bus's load capacitance. The SPI protocol (serial peripheral interface) sends and receives data in a continuous stream. This protocol is recommended for fast data transport. The maximum speed it can deliver is 10 Mbps. It has four wires: MISO (Master in Slave Out), MOSI (Master Out Slave In), SS (Slave Select), and Clock. (Dawoud & Shenouda 2020,38.)

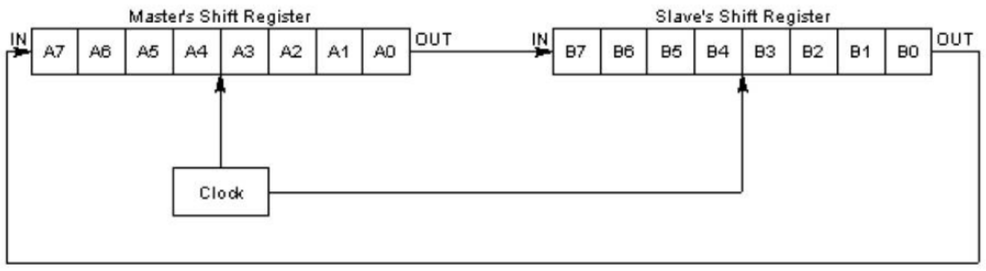


FIGURE 7. Initial State (Dawoud, Shenouda 2020, 195.)

In FIGURE 7 Master and Slave devices each contain an 8-bit shift register. The Serial-In/Serial-Out (SISO) mode is used by these shift registers. The Slave's shift register's output, MISO, is connected to the input of the master's shift register, and the master's shift register's output, MOSI, is connected to the Slave's shift register's input. As this is a synchronous serial data transfer protocol, a clock is required to synchronize the data flow between the master and slave. The master device is responsible for opening and managing the connection; therefore, the clock source, SCK, of the master device must be used to synchronize the data flow. Let the master's shift register hold data from A7 to A0 (MSB to LSB) and let the slave's shift register hold data from B7 to B0 (MSB to LSB). By lowering the voltage on the SS/CS pin, the master turns on the slave, which then makes 8 clock pulses. A single bit of information is sent from the master to the slave after every cycle of the clock, and the same thing happens in reverse. (Dawoud & Shenouda 2020,194.)

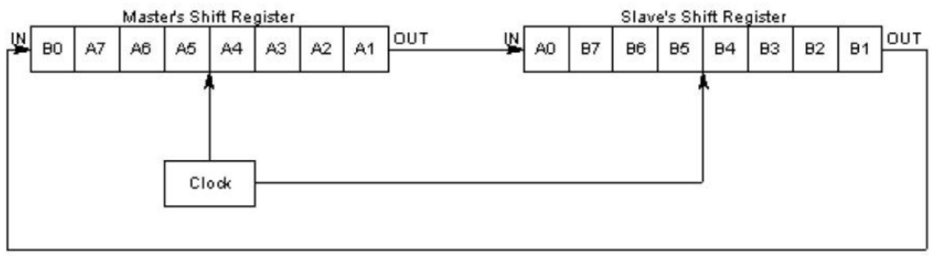


FIGURE 8. First Clock Cycle (Dawoud, Shenouda 2020,195.)

In FIGURE 8 the first clock signal is applied. By applying the first clock the master shift registers transfer one bit to the slave. The bit A0 transfers from master's shift register to slave shift register and slave shift register's one-bit B7 enters to the master shift register. One clock signal uses to process one bit. FIGURE 9 stage when eighth clock signal generates and process the data bits. It shows that all data bits have been transferred from Master register to slave register (Left to right) at the end of eighth clock cycle. Master register has received complete data of slave and slave has received complete data of master. (Dawoud, Shenouda 2020,196.)

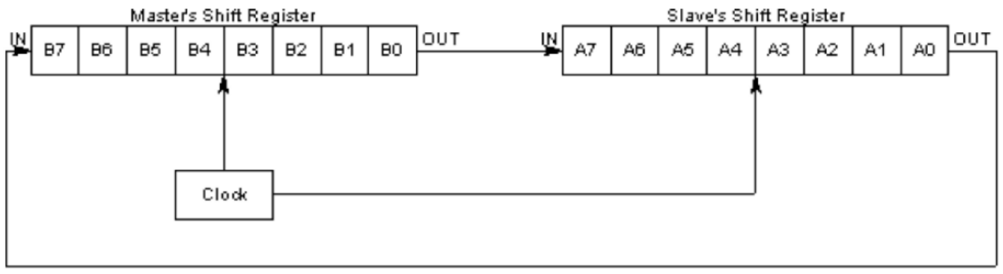


Figure 9. Eighth Clock Cycle (Dawoud, Shenouda 2020,196.)

2.3.3 Single-bus communication interface

The 1-wire communication protocol facilitates data transfer between a controller device and a peripheral device using a single cable. A single controller device establishes communication with one or more 1-Wire peripheral devices through a single data line, which can also serve as a power source for the peripheral devices. The 1-Wire temperature sensors have gained significant popularity due to their affordability and user-friendly nature, as they offer calibrated digital temperature readings directly. Long wire lengths connecting the sensor and Arduino can be tolerated without any measurement errors. (1-Wire Protocol, 2023.)

2.3.4 UART communication interface

Universal Asynchronous Receiver Transmitter is a serial interface widely used in embedded electronics. that does not use clock signal to synchronize the sender and receiver. Pre-defined clock signal can synchronize serial communication. Microcontroller has two PINs for UART interface, but a single microcontroller can be attached with multiple UARTs by defining the microcontroller PINs. UART has two independent wires. Rx is connected to the Tx of opposite side and Tx is connected to the Rx of opposite side. For smooth communication it is necessary to known about the frame rate of data. (Lacamera D. 2018, 16-17.)

3 MATERIALS AND METHODS

This chapter discusses the hardware and software resources as well as the techniques used to accomplish the desired results. The details of hardware and software are discussed here with their requirements. The hardware section discusses the details of used components such as microcontroller, memory, power management, sensors, NFC controller, antenna design, and communication interfaces. Whereas the software section talks about the used code libraries, firmware bootloader, NFC controller configuration, and program code. Figure 10 depicts the block diagram of IoT sensor node and the programming interface including debugging and charging interface.

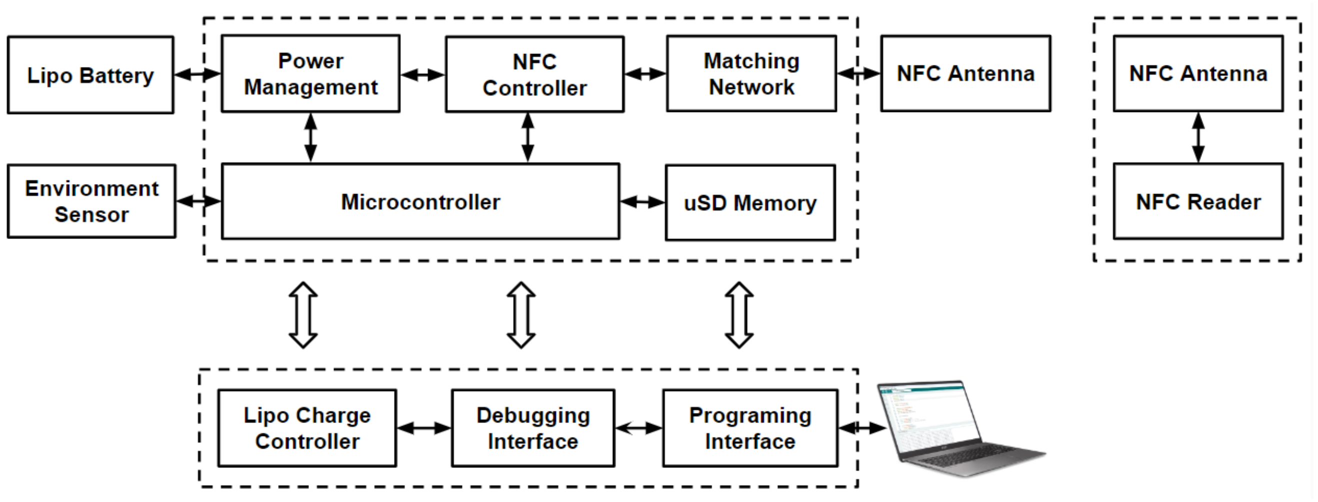


FIGURE 10. Block diagram of IoT sensor node.

3.1 Sensor node hardware

Off the shelf components are used to develop the hardware structure of the IoT sensor node. It includes power management, microcontroller, NFC controller, memory, environmental sensors, communication interfaces and antenna matching network. The microcontroller receives environmental data from the sensors, processes this data and sends it to the NFC controller for wireless transmission. The data is also stored in an on-board memory. Altium Designer (AD) is used to create schematics and develop a two-layer printed circuit board (PCB) for the IoT sensor node. The components are deployed on both sides of the PCB to reduce the hardware size of IoT node. The design is sent to a commercial manufacturer for fabrication of designed PCB. The PCB is assembled and soldered in-house using soldering station (CD-2BQF, JBC) at Aalto University labs. FIGURE 11 (a) shows the front side and (b) depicts the backside of assembled PCB of IoT sensor node. The schematic of the IoT sensor node is given in Appendix 1. Each block of the IoT sensor node is explained in the following subsections.

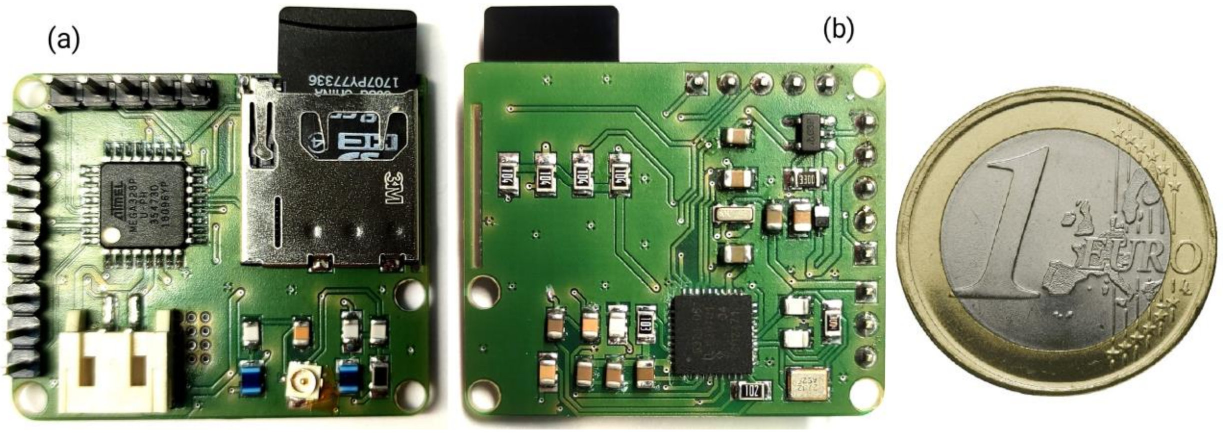


FIGURE 11. (a) Front-side and (b) back-side of IoT sensor node.

3.1.1 Power management

The power management ensures the constant and stable power supply to all the components of IoT sensor node. Power management is designed to obtain power either from wired connection or from a single cell lithium polymer (LiPo) battery. The wired input power source is utilized during the development and debugging phase of the project. While the LiPo battery is employed to power the IoT sensor node during operations. FIGURE 12 shows the schematic of the regulator circuitry.

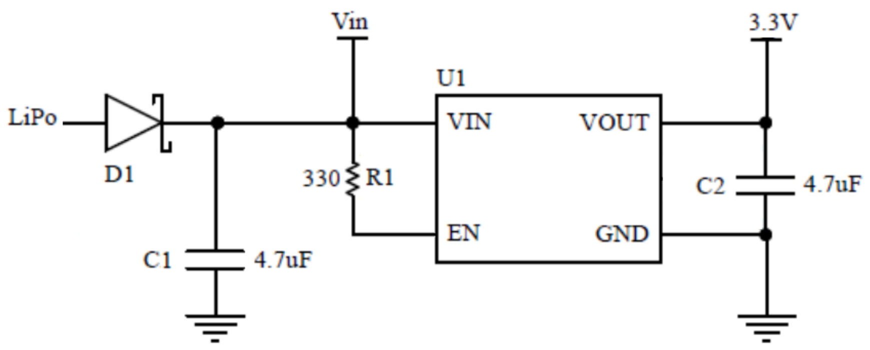


FIGURE 12. Voltage regulator circuit with two inputs.

The power management comprised of an off-the-shelf low dropout (LDO) voltage regulator AP2112 (Diodes Incorporated) to provide constant 3.3 volts to all the blocks of hardware. The LDO regulator can regulate from input supply from 3.4 to 6.5 V to cover both wired voltages from the programing laptop and single cell LiPo voltages. It has a low dropout voltage, which means that it can keep its maximum output voltage even if the voltage goes a bit lower. Quiescent current, which is also called "standby current," is low, which is important for uses that need to save energy. It has the highest output current of about 600 mA, which makes it good for many low-power to medium-power uses. The output voltage is well controlled, with only a small amount of variation. The device has a safety feature that turns it off if it gets too hot to avoid harm. The SOT-23 package is small and easy to solder as well, making it feasible to have area optimized designs. A Schottky diode (BAT60A by Infineon) with low forward voltage drop of 120 mV is used to isolate the input power from LiPo battery. The enable pin of regulator is pulled up to supply voltages to configure it for normal operation.

3.1.2 Microcontroller

A high performance and low power 8-bit microcontroller (ATmega328P, Microchip) is selected for the IoT sensor node. It is very popular in Arduino based hardware because of its convenient configuration and variety of peripherals. The operating voltage of ATmega328P is 1.8V to 5V. Atmega328P can operate at maximum clock of 16 MHz with an external clock generator when supplied with 5 volts. However, it is configured at 8 MHz clock frequency for this to operate on lower supply of 3.3 V for the IoT sensor node. It is equipped with 32 kB flash memory to store programs, 2 kB SRAM memory for the storage of data, and 1 kB of EEPROM memory for the storage of non-volatile data. The microcontroller has a flexible set of 23 general-purpose input and output lines to provide an easy interface for external devices.

ATmega328P microcontrollers are available in a wide range of packages. The thin quad flat pack (TQFP) package of microcontroller is used here to optimize the area and bring convenience in assembly. FIGURE 13 shows the schematic diagram of a microcontroller circuitry used for IoT sensor node. An off-chip crystal oscillator is connected to the microcontroller for stable operation. Single wire communication bus interface is employed for the environmental sensor and SPI communication interface is established with NFC controller and on-board micro secure digital (microSD) memory card.

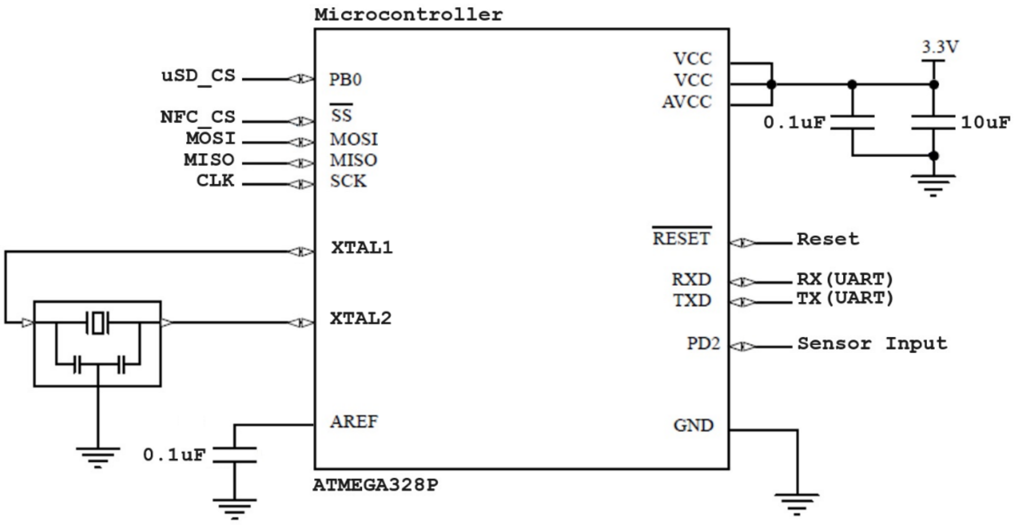


FIGURE 13. Schematic circuitry of microcontroller for IoT sensor node.

3.1.3 NFC controller

NFC controller is a key component to establish the wireless communication for IoT sensor node. PN532 is a flexible NFC controller integrated circuit that supports a variety of NFC Forum protocols including MIFARE reader/writer (ISO/IEC 14443A), FeliCa reader/writer, reader/writer (ISO/IEC 14443B), MIFARE classic card emulation mode (ISO/IEC 14443A), FeliCa card emulation and ECMA 340 (ISO/IEC 18092) peer-to-peer mode. These protocols are required to communicate with various types of NFC tags. The NFC controller PN532 is selected for IoT sensor node because of its compatibility with peer-to-peer communication mode, supply voltage range, ease to configure and the availability of SPI interface. ECMA 340 (ISO/IEC 18092) peer-to-peer configuration mode enables two NFC devices to communicate securely with each other.

The ECMA 340 protocol is employed to establish peer-to-peer communication mode for NFC Interface which used inductively coupled antennas designed at the center frequency of 13,56 MHz. The card emulation mode and read-write mode are employed to debug and improve the functionality of NFC communication during the firmware development process of the IoT sensor node. FIGURE 14 depicts the required minimum circuitry to function with matching network, antenna, and interfaces with host processor. An external crystal of 27.12 MHz is used to clock the NFC controller. Decoupling capacitors of 4.7 uF and 100 nF are used to counter big dips or spikes in supply voltage. The controller has the capability to operate between 2.7 to 5.5 volts. We, as explained in the power management section, have used 3.3 volts for this project to ensure reliable communication with the host processor.

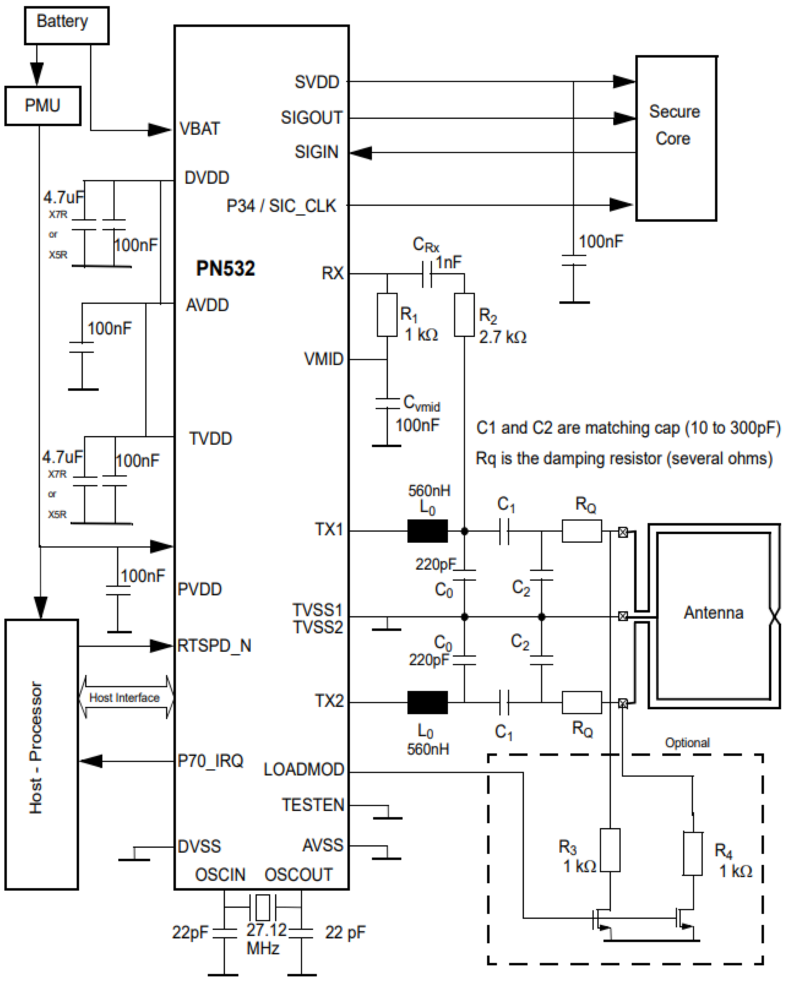


FIGURE 14. NFC Controller PN532 with its interfaces, matching network, and antenna (NFC Integrated solutions 2023.)

3.1.4 NFC antenna design

The construction of an antenna for NFC communication is based upon positioning a loop on the application PCB. Its impedance is matched with the internal tuning capacitance value (C_T) to resonate the circuit at 13.56 MHz. The fundamental equation governing the tuning frequency is shown in Equation 1 where inductance of antenna is denoted with L_A and the tuning frequency is denoted with F_T (NFC antenna, 2019).

$$F_T = \frac{1}{\sqrt{L_T \times C_T}} \quad (1)$$

An antenna operating at 13.56 MHz can be designed in various shapes. The equivalent inductance (L_A) of antenna is the key factor to have it resonating at the frequency of 13.56 MHz. An NFC module like the PN532 and its antenna need to be connected to a circuit that matches the antenna to facilitate an effective transfer of power and an excellent level of communication performance. The purpose of this circuit is to ensure that the impedance of the antenna is compatible with the impedance of the transmission and reception circuits on NFC controller.

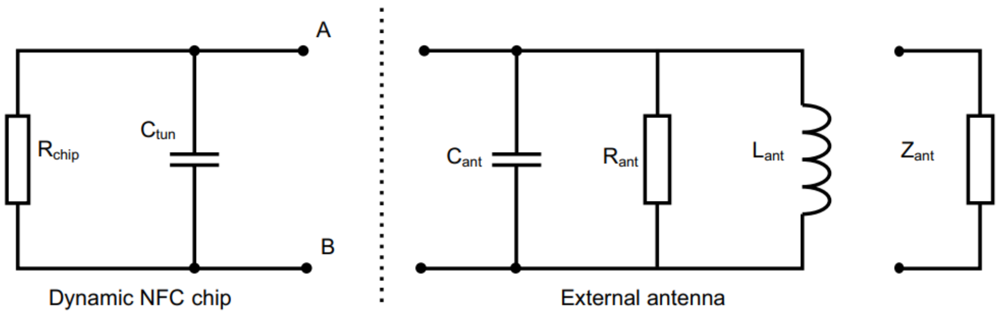


FIGURE 15. Equivalent circuit for the NFC controller and antenna (NFC antenna, 2019).

An antenna impedance matching circuit works for impedance matching to transfer the antenna signal from PN532 to the antenna with maximum signal strengthen. Matching network is a circuit including inductors, capacitors, and resistors. This matching circuit is designed according to antenna’s specifications. A stable communication system for peer-to-peer communication are the results of matched impedance to maximize the power transfer and enhance the signal strength. FIGURE 15 shows and equivalent circuit of the matching network for the NFC antenna design. Although C_{ant}, R_{ant}, and L_{ant} are constants, the resulting impedance (their parallel combination) varies with frequency. The imaginary portion of the antenna impedance Z_{ant} is null at self-resonance frequency, and Z_{ant} is simply resistive. The imaginary part of the antenna impedance is positive below the self-resonance frequency, and the antenna behavior is inductive. (NFC antenna, 2019.)

TABLE 1. Antenna Specification

Name	**Specifications**				
	Turns (N)	**Length x Width (mm)**	**Conductor Width (mm)**	**Conductor Spacing (mm)**	**Inductance (μH)**
Antenna (a)	8	20 x 20	0.2	0.49	1.47
Antenna (b)	4	42 x 32	0.2	0.60	1.48
Antenna (c)	3	80 x 40	0.6	0.60	1.47
Antenna (d)	6	30 x 30	0.5	0.60	1.51
Antenna (e)	7	40 x 40	1.12	1.00	1.55
Antenna (f)	2	100 x 100	0.6	1.00	1.47
Antenna (g)	4	80 x 40	1.6	1.00	1.50
Antenna (h)	4	104 x 52	3	1.50	1.55
Antenna (i)	4	120 x 80	6	2.00	1.50

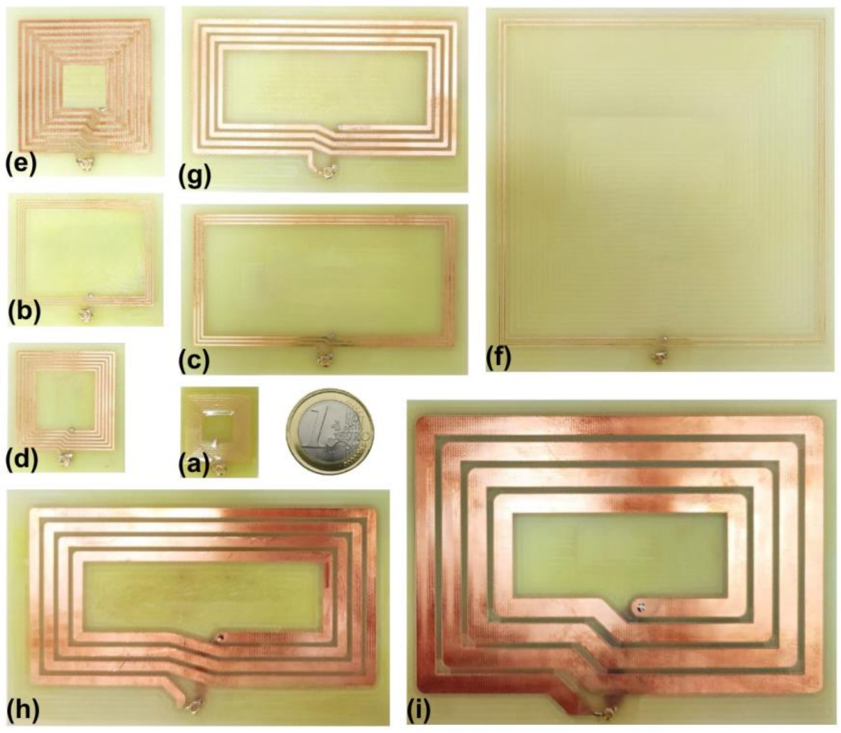


FIGURE 16. The NFC antennas designed and fabricated in Aalto University.

A matching network is designed for the NFC controller with loop antenna having inductance of 1.47 µH to resonate at frequency of 13.56 MHz. Nine different antennas are designed with a focus on working with the same matching network. The designed inductance is kept similar for all antennas. The number of turns (N), antenna dimensions, conductor width, and spacing between conductors are varied for all the proposed designs to vary the impedance of antennas. The loop antennas are designed using NFC inductance design tool of ST Electronics for the NFC controller to assess the effect of variation on the communication distance achieved in peer-to-peer mode. The antennas are designed in Altium PCB design tool and fabricated using PCB milling machine (S63, LPKF ProtoMat) at Aalto University labs. FIGURE 16 shows the designed antennas. Whereas TABLE 1. presents the sizes, number of turns, track widths, designed inductance and inter-track distance.

3.1.5 Environmental sensor

A project that requires the continuous monitoring and management of environmental variables in real time, the sensor is an essential component because of its long-term reliability and high level of precision. DHT22 (DFRobot) provides temperature measurement precision of 0.5°C and humidity measurement accuracy of about 2-5%. Temperature and humidity are typically operatable within a range between -40°C and 80°C and 0–100%, respectively. If a sensor needs to be positioned in a location remote from the microcontroller, DHT22 can function with extended wire lengths without the measurement’s errors. DHT22 is employed for the IoT sensor node as environmental sensor Because of its low power requirements and its convenient single-bus digital interface. FIGURE 17 shows DHT22 sensor by Robots.

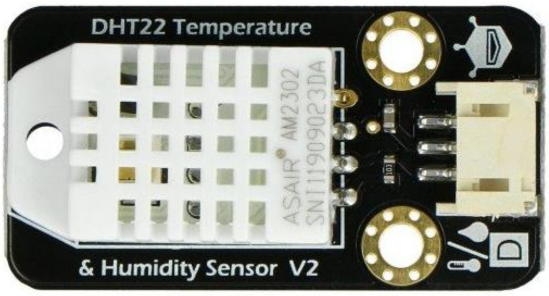


FIGURE 17. Temperature and humidity sensor (Gravity, 2022.)

3.1.6 Data recording

An on-board memory is established by interfacing a microSD card on the IoT sensor node to store the recorded data when the NFC reader is not available. Real-time data points of environmental sensors are stored in the microSD. Memory capacity can be changed as per data size and data sample rate requirements. An SPI interface is employe to establish communication between memory and the microcontroller.

3.1.7 On-board communication interfaces

The SPI interface is employed for on-board communication for IoT sensor node due to its key feature of high data transfer with low latency. The SPI is deployed interface NFC controller and microSD card with microcontroller. The SPI interface is also utilized to burn Arduino bootloader in the microcontroller of IoT sensor node from the in-circuit Serial Programmer (ISP). A standard clock speed of 100 kHz is used to establish the SPI communication. Single-bus digital communication interface is established between microcontroller and environmental sensor to collect the environmental parameters from the sensor. The UART interface is employed to establish the communication between microcontroller and Arduino integrated development environment (IDE).

3.1.8 Charge controller and programming interface board

A portable programming interface circuitry is designed and developed separately to burn codes and debug the firmware during the software development process. The separate development of this board serves several purposes including area reduction of IoT sensor node, programming and debugging interface for the firmware development and charging the on-board LiPo battery. FIGURE 18 (a) and (b) respectively depict the front and back side of the charge controller and programming interface board. The board is designed in a way to connect and disconnect to IoT sensor node with the help of pin header conveniently.

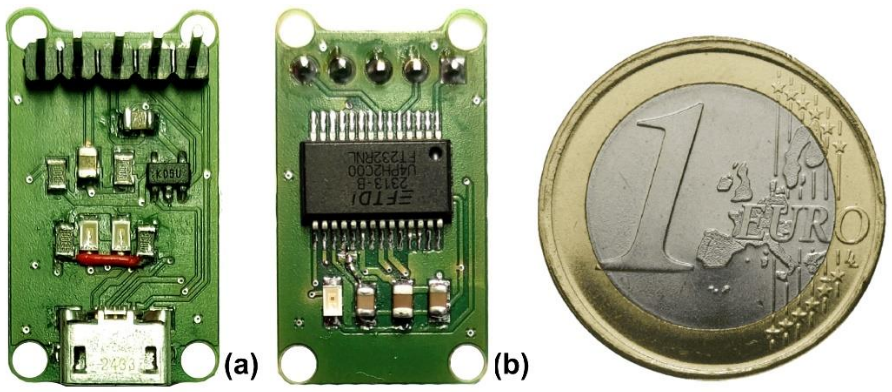


FIGURE 18. (a) Front-side and (b) back-side of charge controller and programming interface board.

3.1.9 LiPo charging circuit

A single cell fully integrated LiPo charge management controller MCP73831 (Microchip) is employed on the programming interface board to charge the LiPo battery. The input power is supplied with the USB interface. FIGURE 19 shows the schematic diagram of single cell Lipo charger controller. Decoupling capacitors are used to stabilize the input and output voltages. The voltage regulator ensures are regulated 3.3 V output for the IoT sensor node. Single cell LiPo batteries are widely used in low power Arduino projects due to their portability and durability.

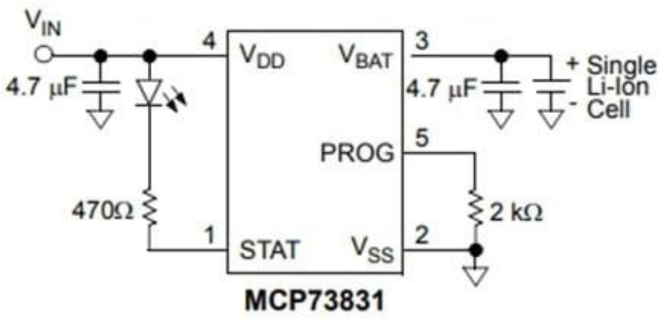


FIGURE 19. LiPo charging circuit (MCP73831 2023.)

3.1.10 USB to serial communication

A universal serial bus (USB) to serial UART interface chip FTDI232 (FTDI Chip) is used to establish the communication between IoT sensor node and Arduino IDE firmware development and debugging environment. The FTDI232 is also equipped with an on-chip level shifting circuitry which is used here which shifts the levels between 5 and 3.3 V to ensure reliable communication. FIGURE 20 shows the schematic diagram of USB to serial UART communication circuitry for communication between computer and microcontroller. Light emitting diode (LED) indicators are used for visual debugging. The decoupling capacitors are used to stabilize the voltages of input power supply.

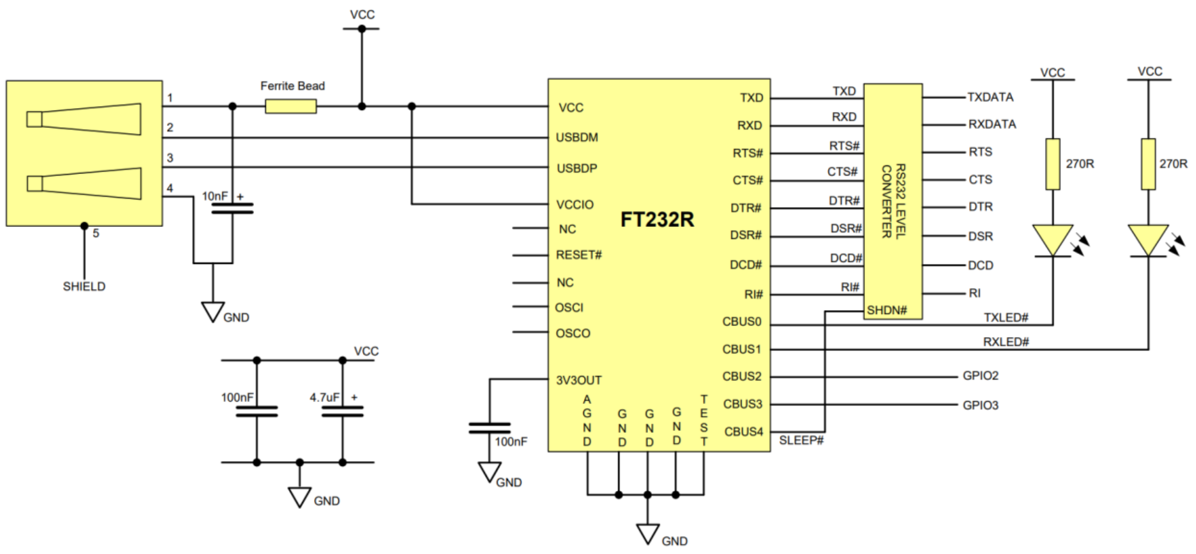


FIGURE 20. USB to serial communication circuit (FT232R, 2023.)

3.2 Firmware design

Firmware design, a crucial aspect of embedded systems development, involves a systematic process to create software that resides on hardware devices. The process involves several essential steps, each playing a vital role in ensuring the efficiency, reliability, and functionality of the final product. The requirements of the IoT sensor node operation are first defined which includes reading data from environmental sensor, adding appropriate timestamps with the acquired data, storing it to the on-board microSD card in organized files, configuring the peer-to-peer communication protocol and looking for the NFC reader to transfer the organized data.

Once the requirements are defined, the algorithm is designed to outline the logical flow of operations and the flow chart is created to better visualize the logic of algorithm.

The actual firmware code is written based on the algorithm and flowchart. A set of rigorous tests is executed to identify bugs and ensure the firmware behaves as expected. Serial communication port of Arduino IDE is used to debug the code. Simple form of codes is first written to establish the basic functionality of file creation in memory, opening and adding data to the files in microSD memory. Similarly, the NFC tag detection, basic read and write operations are established. On the sensor end the data of temperature sensor and humidity sensor is read, processed, and visualized on serial monitor before moving towards the development of actual firmware. A series of code optimizations is conducted after the working firmware is conceived. The detailed algorithm, flow chart and final version of the firmware code is further discussed in the sub sections of this chapter.

3.2.1 Algorithm development

The algorithm development includes four important parts to ensure the proper operation of the IoT sensor node. Firstly, all the relevant Arduino based libraries are identified and included to establish the crucial functionality of the firmware. Then the constants and global variables are identified and declared for the functionality of IoT sensor node. In the setup function, the initialization and configuration of all components is executed. Finally, the functions of loop section are identified and established to perform the functions of IoT sensor node in continues loop. The algorithm to outline the functionality of IoT sensor node is shown in Appendix 2.

3.2.2 Flowchart

The flowchart is the structural presentation to the algorithm of the system firmware which provides ease in understanding about the data flow and conditions while writing the code. FIGURE 21 represents the flowchart of the firmware according to the algorithm provided in previous subsection. In the start of program basic parts like SPI for serial communication, NFC controller PN532 and environmental sensor DHT22 are initialized. After initialization, in the setup function, the firmware version of NFC controller (PN532) is checked, and the NFC communication mode is configured. If the firmware version of NFC controller is missing, then an error message is generated. The data storage function in microSD card is also initialized and the error message is generated if the initialization fails.

In the loop function, the microcontroller reads the temperature and humidity data from the environmental sensor. Timestamp for the sensed data is attached and data is organized. Microcontroller performs a check whether the datafile is already created inside microSD memory. It creates the data file, names it in a structural way and opens the file if it was not found in microSD card. If datafile is found in microSD card, then it opens it and writes organized date to the file and closes it. The NFC controller is configured in peer-to-peer communication mode and the availability of NFC reader is checked. If the NFC reader is available, then the data size of the first data file is assessed in the organized file structure. The data on a file is transferred to the NFC reader if the file size specifications are met and the transferred file is deleted. The file size is configured according to the transfer packet size of peer-to-peer communication mode. The function loop is repeated if the NFC reader is not found, or the file is successfully transferred to NFC reader.

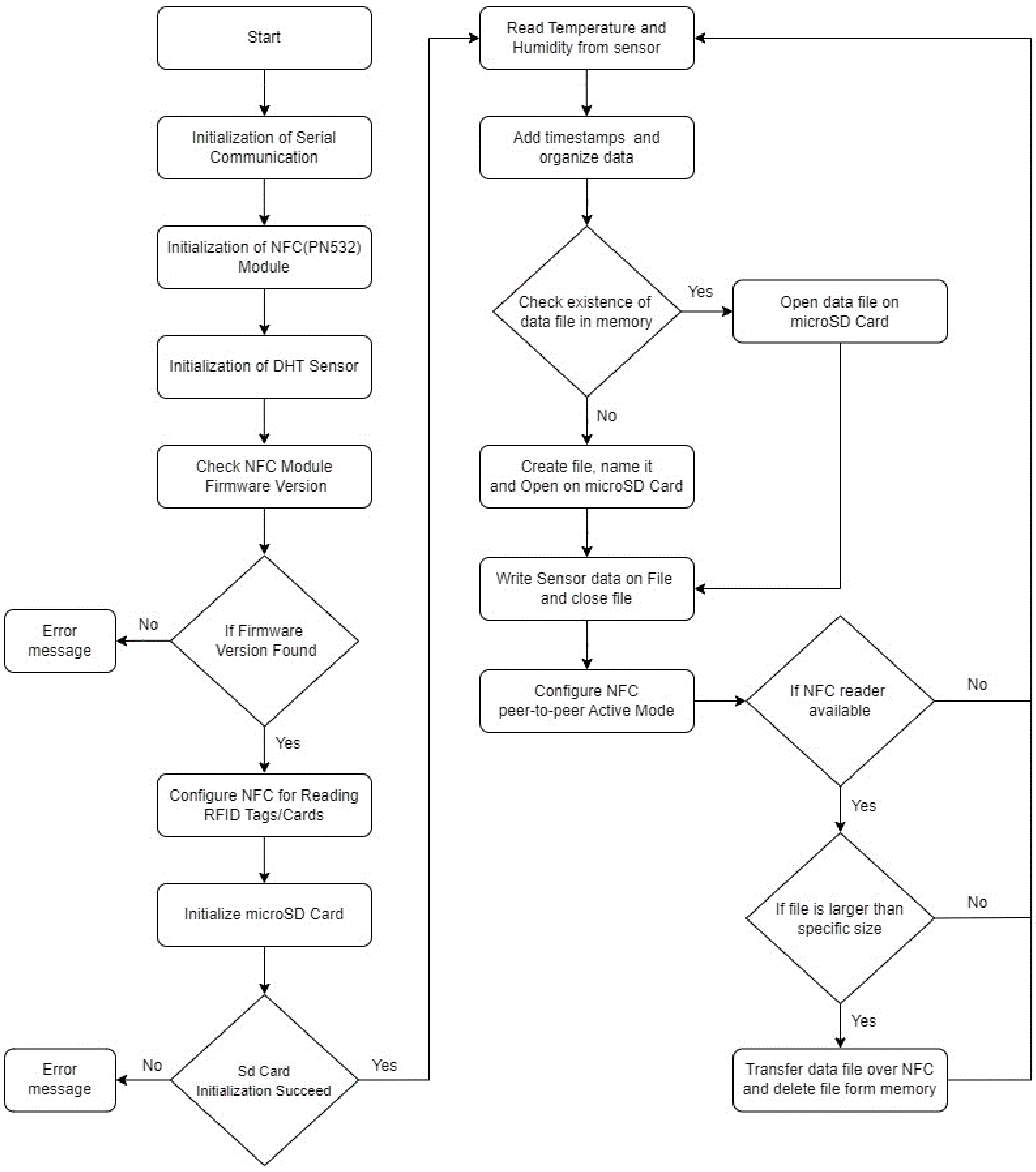

Start
Initialization of Serial Communication
Initialization of NFC(PN532) Module
Initialization of DHT Sensor
Check NFC Module Firmware Version
If Firmware Version Found
No
Error message
Yes
Configure NFC for Reading RFID Tags/Cards
Initialize microSD Card
Sd Card Initialization Succeed
No
Error message
Yes
Read Temperature and Humidity from sensor
Add timestamps and organize data
Check existence of data file in memory
Yes
Open data file on microSD Card
No
Create file, name it and Open on microSD Card
Write Sensor data on File and close file
Configure NFC peer-to-peer Active Mode
If NFC reader available
No
Yes
If file is larger than specific size
No
Yes
Transfer data file over NFC and delete file form memory

FIGURE 21. Flowchart diagram of firmware for IoT sensor node.

3.2.3 Programming code

A simplified variant of C/C++ programming language is employed to write the firmware of IoT sensor node. Arduino IDE is used to write, complied, and burn the code to the microcontroller. The Arduino bootloader is burned in microcontroller vis SPI interface before the development and testing the code. The serial monitor of Arduino IDE is used to visualize and debug the functionality of firmware. The code libraries play a crucial role in firmware coding because of their reusability, efficiency, standardization, and community support. We have employed various libraries for the firmware development of IoT sensor node. The peer-to-peer mode configuration library by Ladyada is used for NFC controller PN532 (Adafruit-PN532, 2013). microSD card configuration library, the DHT22 sensor 1-wire communication library for environmental sensor, real time clock (RTC) library for timestamps and SPI for serial communication with Arduino IDE is used under GNU Lesser General Public License v2.1.

Code 1 shows the used libraries and the global variable declarations in the header section of firmware. The digital PIN 10 of Arduino microcontroller is configured as chip select (CS) for the NFC controller PN532 to establish the SPI communication protocol. The digital PIN 2 is configured as input data pin for environmental sensor to establish the single-bus communication protocol.

```
#include <PN532.h>
#include <SPI.h>
#include <DHT.h>
#include <SD.h>
#include <RTClib.h>
#define PN532_CS 10
#define DHT_PIN 2
char DataOut[16];
char DataIn[16];
```

Code 1. Libraries initialization and global variable declaration.

```
void setup(void)
{
  nfc.begin();
  dht.begin();
  rtc.begin()
  uint32_t versiondata = nfc.getFirmwareVersion();
  if(!versiondata) {
     #ifdef NFC_DEMO_DEBUG
     Serial.print("Didn't find PN53x board");
     #endif
     while (1); // Halt
     }
nfc.SAMConfig();
if (!SD.begin(8)) {
    Serial.println("initialization failed!");
    while (1);
    }
}
```

Code 2. Setup section of firmware to initialize various communication protocols.

The Arduino based firmware code has a special setup function to initialize the various functions and protocols. Code 2 shows the setup part of the firmware code used for the IoT sensor node. The SPI is initialized here to establish the serial communication protocol. The single-bus wire communication protocol is initialized to collect temperature and humidity date form the environmental sensor. The SPI communication protocol is initialized to communicate with on-board NFC controller PN532. The firmware version of NFC controller is fetched, and the process is terminated with an error message in case of failure. The While() loop is used together with if condition statement to achieve this functionality. The microSD card is initialized and tested using if statement. The program is terminated with error message in case of failure. The RTC library function is initialized to incorporate the timestamp with data points.

```
void loop(void){
float temperature_S = dht.readTemperature();
float humidity_S = dht.readHumidity();
snprintf(dataout, sizeof(dataout), temperature_S, humidity_S);
if (SD.exists(("datafile.txt",)){
dataFile = SD.open("datafile.txt", FILE_WRITE); // Open data fi
} else  {
File dataFile = SD.open(fileName, FILE_WRITE); // create data f
}
dataFile.println(TimeStamp);
dataFile.println(temperature_S);
dataFile.println(humidity_S);
dataFile.close();
if (nfc.configurePeerAsInitiator(PN532_BAUDRATE_424K)) {
if (fileSize > 1024) {
if (nfc.initiatorTxRx(DataOut, DataIn)){
if (SD.remove("datafile.txt")) {
      Serial.println("File deleted successfully.");
    }
 delay(300000);
}}}
```

Code 3. The IoT sensor firmware code for the main loop function.

The `loop()` is the main function which repeats the programmed task according to the code inside `loop()` function. Code 3 shows the important content of the repeating codes. First, the temperature and humidity sensor data are concatenated in an array in an organized structure. The data file in the microSD card id is checked and created if it does not already exist in the memory. The data file opened, and the data is written together with time stamps and the file is closed afterwards. In the following step, the availability of NFC reader is checked, and the size of the data file is assessed. The data is then transferred to the NFC controller to send it further to the NFC reader if all conditions met. The file is removed from the microSD memory after a successful transfer and the loop is repeated indefinitely.

4 MEASUREMENTS AND RESULTS

This chapter describes the measurements setup and the measurement scenarios employed to validate the functionality of IoT sensor node. The IoT sensor node is first powered up with minimum operations case scenario to perform the power on test. The Arduino bootloader is burned via SPI interface using Arduino IDE environment. A set of simple Arduino-based programs are loaded to test the functionality of an on-board microcontroller and its communication interfaces in a controlled lab environment. The measurement setup as depicted in FIGURE 22 is established for the NFC communication of the IoT sensor node. An NXP PN532 controller based commercial NFC reader (VMA211, Velleman Group) is employed to receive the sensor data from IoT sensor node. Both the IoT sensor node and the NFC receiver are configured in peer-to-peer communication mode.

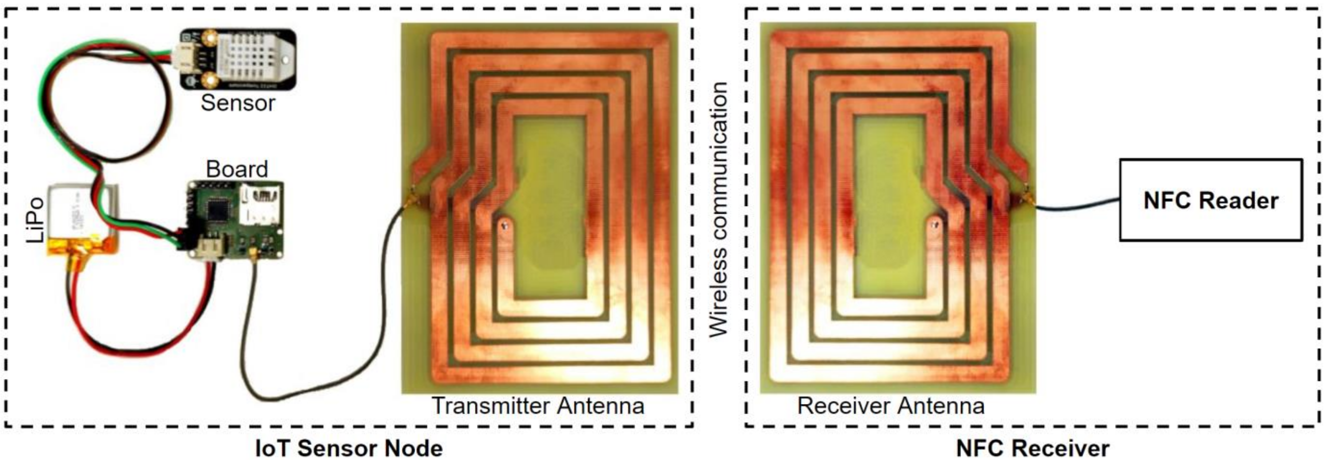


FIGURE 22. Measurement set-up to validate the functionality of IoT sensor node.

In this study, following measurement scenarios are employed:

1. NFC antennas are tested using vector network analyzer.
2. The reading distance for the designed using centimeter scale.
3. Temperature and humidity measurements using environmental sensor.

In the first measurement scenario, the characteristics of all the designed NFC antennas are measured at 13.56 MHz frequency on smith chart on vector network analyzer (VNA ZNB8, Rohde & Schwarz) at radio frequency (RF) lab of Aalto University. The inductance and the impedance of all the designed antennas are measured in a controlled lab environment. The results of the measured inductances and impedances are tabulated in TABLE 2 for all the antennas.

TABLE 2. Measurements of the designed NFC antennas.

Name	**Specifications**			**Measurements**	
	Turns (N)	**Length x Width (mm)**	**Inductance (μH)**	**Inductance (μH)**	**Impedance (Ω)**
Antenna (a)	8	20 x 20	1.47	1.46	2.92
Antenna (b)	4	42 x 32	1.48	1.54	4.03
Antenna (c)	3	80 x 40	1.47	1.51	1.69
Antenna (d)	6	30 x 30	1.51	1.53	155
Antenna (e)	7	40 x 40	1.55	1.65	1.19
Antenna (f)	2	100 x 100	1.47	1.55	2.93
Antenna (g)	4	80 x 40	1.50	1.59	1.03
Antenna (h)	4	104 x 52	1.55	1.65	0.73
Antenna (i)	4	120 x 80	1.50	1.62	0.64

The designed inductance is kept around 1.5 μH for all antennas. The number of turns (N), antenna dimensions, conductor width, and spacing between conductors are varied for all the proposed designs to vary the impedance of antennas. The impedance of every antenna mainly depends on the track width and space between the conductive tracks. An antenna smaller in size with a minimum track width has more impedance as compared to other antennas with a wider conductive track. The antenna (a), (f) and (b) measured greater impedance of around 3 and 4 ohms respectively because of the reduced track width. However, the antennas (i), (h) and (g) measured least impedance of 0.64, 0.73 and 1.03 ohms respectively at 13.56 MHz (the center frequency of NFC communication).

In the second measurement scenario, the measurement setup is established as shown in FIGURE 22. The maximum reading distance measurements are performed for each antenna with combination of all other designed antennas. A centimeter scale is used on the laboratory test bench to measure and record the maximum reading ranges. The IoT node and the NFC receiver were configured in peer-to-peer communication mode. The measurement results of maximum reading range for all antennas are tabulated in TABLE 3, the reading range of commercial NFC modules is about 4 to 6 cm normally.

TABLE 3. Communication distance of each antenna with all other designed antennas.

Antennas	**Measured Distance (cm)**	**Antennas**	**Measured Distance (cm)**	**Antennas**	**Measured Distance (cm)**
Antenna (a) vs (a)	4.0	Antenna (b) vs (h)	7.5	Antenna (e) vs (e)	5.5
Antenna (a) vs (b)	4.0	Antenna (b) vs (i)	7.5	Antenna (e) vs (f)	9.0
Antenna (a) vs (c)	3.5	Antenna (c) vs (c)	8.0	Antenna (e) vs (g)	8.5
Antenna (a) vs (d)	3.5	Antenna (c) vs (d)	5.5	Antenna (e) vs (h)	10.0
Antenna (a) vs (e)	3.0	Antenna (c) vs (e)	8.5	Antenna (e) vs (i)	10.5
Antenna (a) vs (f)	2.0	Antenna (c) vs (f)	8.0	Antenna (f) vs (f)	11.0
Antenna (a) vs (g)	4.0	Antenna (c) vs (g)	8.0	Antenna (f) vs (g)	8.0
Antenna (a) vs (h)	4.0	Antenna (c) vs (h)	9.5	Antenna (f) vs (h)	9.0
Antenna (a) vs (i)	6.0	Antenna (c) vs (i)	10.0	Antenna (f) vs (i)	17.0
Antenna (b) vs (b)	5.5	Antenna (d) vs (d)	4.5	Antenna (g) vs (g)	7.6
Antenna (b) vs (c)	6.0	Antenna (d) vs (e)	4.5	Antenna (g) vs (h)	8.0
Antenna (b) vs (d)	4.5	Antenna (d) vs (f)	4.5	Antenna (g) vs (i)	13.5
Antenna (b) vs (e)	4.5	Antenna (d) vs (g)	5.5	Antenna (h) vs (h)	14.5
Antenna (b) vs (f)	6.5	Antenna (d) vs (h)	6.0	Antenna (h) vs (i)	17.0
Antenna (b) vs (g)	7.0	Antenna (d) vs (i)	6.0	Antenna (i) vs (i)	19.5

The reading range for the designed antennas varied from 2.0 cm to 19.5 cm. The reading range of each antenna varies according to its impedance which is dependent on the antenna size, conductive track width, and inter-track spacing. For example, antenna (f) has size of 100 x 100 mm with only two turns of 0.6 mm track size leading to a distributed electromagnetic field making it very difficult to be read by small antenna as (a) having size of only 20x20 mm. The maximum reading range for these two antennas has been limited to only 2 cm. Whereas the antenna with larger size and smaller impedance like antenna (g), (h) and (i) has produced enhance reading ranges in combination of each other. The antenna (i) has the largest size of 120x80 cm, with the largest track width of 6 mm and the least impedance of 0.64 Ω among other antennas. It has produced longest reading distances with itself and with other similar antenna like (h), (g), (f) and (c). It is observed that by varying the antenna size can enhance the reading range of NFC devices with the same matching networks.

In the third measurement scenario, the environmental sensor is placed inside climate test chamber (LabEvent, Weiss Technik) at Aalto University labs to validate its functionality for variations in temperature and humidity levels. The temperature of the climate test chamber is varied from -20 C to 100 C in timespan of 30 minutes. FIGURE 23 depicts the set temperature of the climate test chamber and the measured temperature by IoT sensor node. It is observed that the IoT sensor node has almost same sensitive level as climate test chamber, and it follows the set temperature of the measurement range. The small lack in temperature measurements can be compensated for with calibration using the on-board microcontroller.

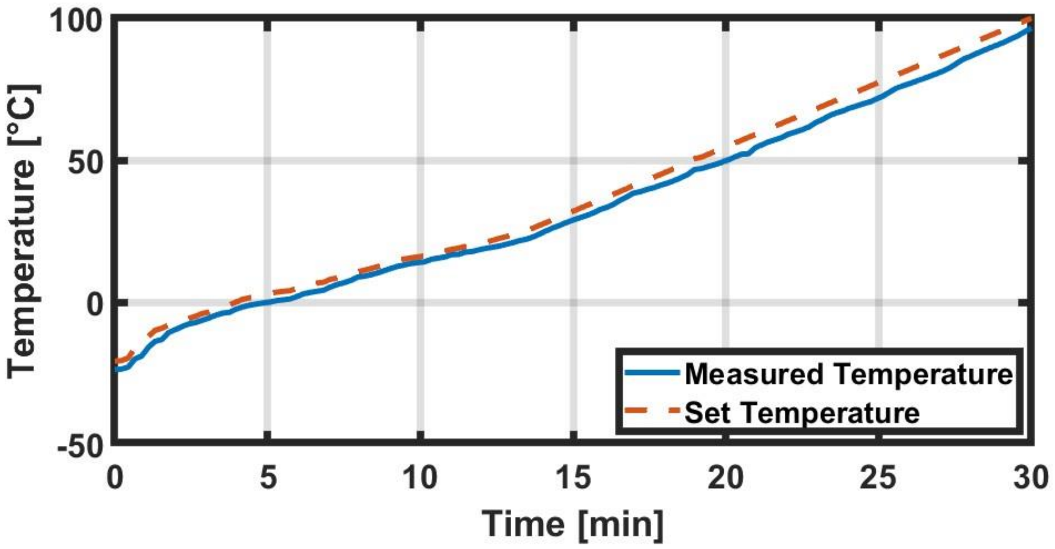


FIGURE 23. The set temperature of the climate test chamber and measured temperature by IoT sensor node.

Similarly, the humidity level is also set in the second set of measurements inside the climate test chamber. The humidity level from 5 % to 90 % is varied inside the climate test chamber over a time span of 30 minutes. The environmental sensor of the IoT sensor node is deployed inside the chamber to execute the planned measurement of humidity levels. FIGURE 24 shows the set humidity level of the climate test chamber and the measured humidity level by the IoT sensor node. It is observed that the measured humidity levels on the graph have the same trend of measurement. However, a lower slope for the measured humidity level is observed as compared to the set humidity level over the measurement span of 5 % to 90 %.

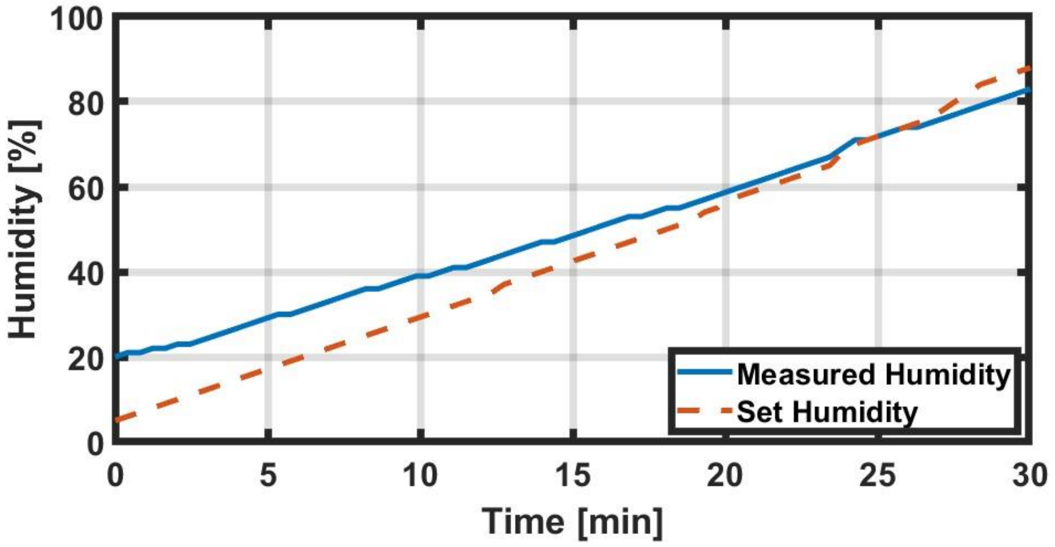


FIGURE 24. The set humidity level of climate test chamber and measured humidity level by IoT sensor node.

It is evident that the designed IoT sensor node is capable of working independently on single cell LiPo battery to obtain, process, store and transfer the environmental parameters using NFC communication technology. The measurements on the custom designed NFC antennas show that the reading range for NFC communication protocol can be enhanced 4 times than the range of the commercial antenna by changing the antenna design parameters and by keeping the same antenna matching network.

5 CONCLUSION

In this study, the design, development, and implementation of an Arduino-based IoT sensor node with NFC communication protocols was explored. The measurement results show that with the same inductance and the same antenna matching network, several antennas can be designed, and the communication range can be varied as per requirement. Commercial NFC devices have a read range of 4 to 6 cm. However, in this study, a set of custom NFC antennas were designed and fabricated by keeping the same inductance and antenna matching network and using various design parameters to achieve the remarkable reading range up to 19.5 cm. An environmental sensor was integrated with the designed IoT sensor node to showcase an application using peer-to-peer NFC communication protocol. This thesis has demonstrated the compatibility and applications of NFC based IoT sensor node which can be extended to be used in various fields of life, especially in wearable medical devices.

The project makes a significant contribution to the expanding domain of IoT sensor nodes with NFC communication protocol and acts as a great asset for various applications. Due to its open-source nature and high level of accessibility, it is anticipated that the design of IoT sensor node will serve as a building block for the internet of everything. Use of IoT sensor nodes in wearable medical devices is becoming increasingly important because of its flexibility. The integration of NFC communication protocol leads towards the possibility of wireless power transmission for low power IoT sensor node. Hence, paving way towards the development of self-powered battery less low power data collecting IoT sensor nodes. The on-chip implementation of the proposed design might be considered as future work to have power optimized system with the power-harvesting capability from the NFC reader antenna to have battery-less operation.

REFERENCES

1-Wire Protocol | Arduino Documentation. Available at: https://docs.arduino.cc/learn/communication/one-wire (Accessed: 14 August 2023).

Adafruit-PN532. Available at: https://github.com/adafruit/Adafruit-PN532 (Accessed 5 October 2023).

Dawoud, D.S. and Dawoud, P. (2020) Serial communication protocols and standards. Gistrup: River Publishers.

Dawoud, S. and Peplow, R. (2022) Digital system design: Use of Microcontroller. Gistrup: River Publishers.

Frenzel, L.E. (2015) Handbook of Serial Communications interfaces.

FT232R. Available at: https://www.ftdichip.com/old2020/Products/ICs/FT232R.html (Accessed: 14 July 2023).

Gravity - DHT22 digital temperature and humidity (2022) BOTLAND. Available at: https://botland.store/gravity-temperature-sensors/15760-gravity-dht22-digital-temperature-and-humidity-sensor-dfrobot-sen0137-5904422378004.html (Accessed: 07 September 2023).

How to design an antenna for dynamic NFC tags - application note. Available at: https://www.st.com/resource/en/application_note/an2972-how-to-design-an-antenna-for-dynamic-nfc-tags-stmicroelectronics.pdf (Accessed: 10 July 2023).

Igoe, T., Coleman, D. and Jepson, B. (2014). Beginning NFC. ‘O’Reilly Media, Inc.’
Kang, S.-G. et al. (2021) Near-field communication in biomedical applications, MDPI. Available at: https://doi.org/10.3390/s21030703 (Accessed: 28 June 2023).

Lacamera, D. (2018) Embedded Systems Architecture: Explore Architectural Concepts, pragmatic design patterns, and best practices to produce robust systems. Birmingham: Packt Publishing.

Lehpamer, H. (2012) RFID design principles. Boston (Mass.): Artech House.

MCP73831 - Microchip Technology. Available at: https://www.microchip.com/en-us/product/MCP73831 (Accessed: 2 October 2023).

NFC inductance: RF design: Edesignsuite: STMicroelectronics (no date) NFC Inductance | RF Design | eDesignSuite | STMicroelectronics. Available at: https://eds.st.com/antenna/#/ (Accessed: 7 August 2023).

NFC Integrated Solution (no date) NXP Semiconductors. Available at: https://www.nxp.com/products/rfid-nfc/nfc-hf/nfc-readers/nfc-integrated-solution:PN5321A3HN (Accessed: 20 September 2023).

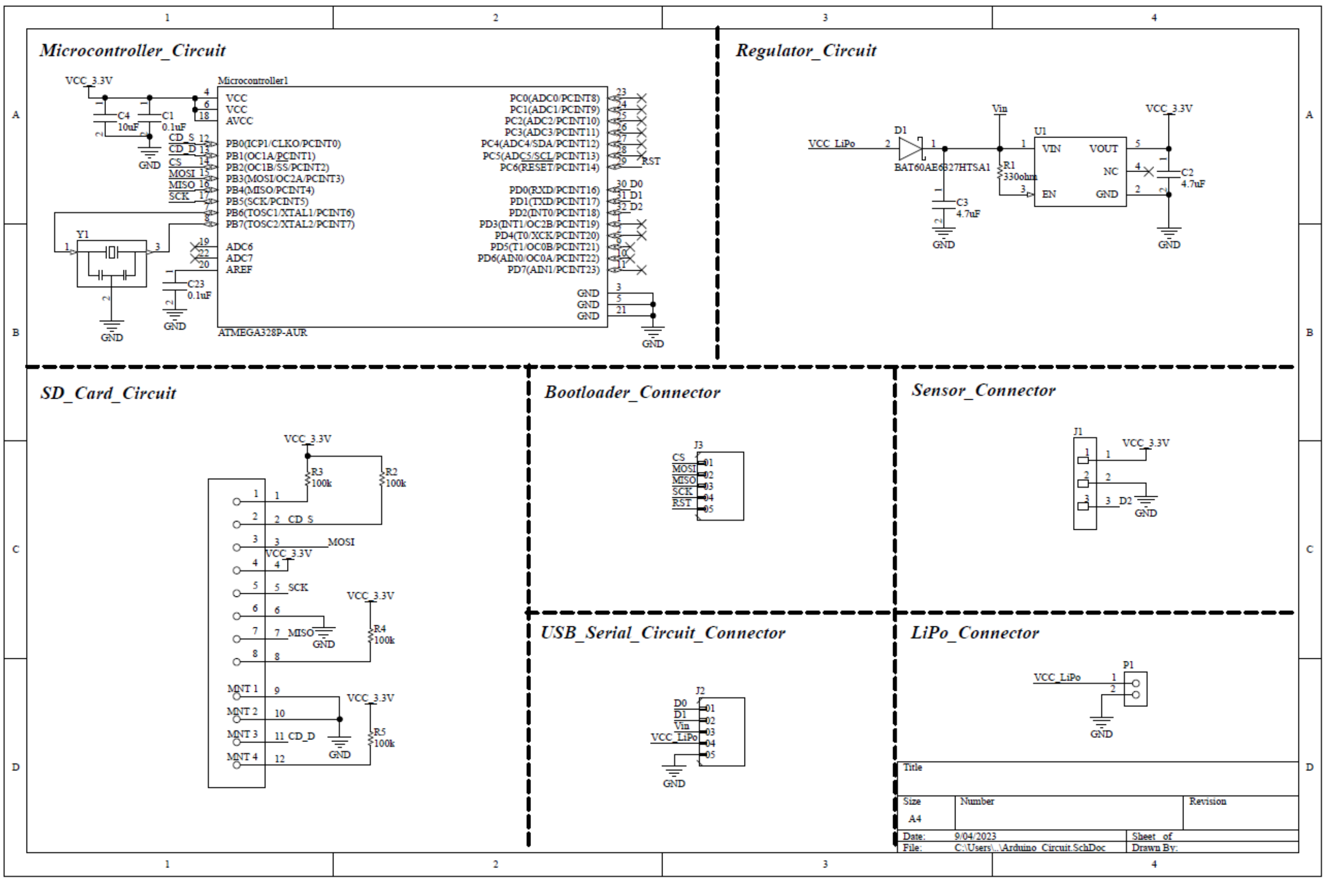
Ozdenizci, B. et al. (2012) Near Field Communication (NFC) - from theory to practice. John Wiley & Sons Incorporated.

Perret, E. (2014) Radio Frequency Identification and sensors. John Wiley & Sons.

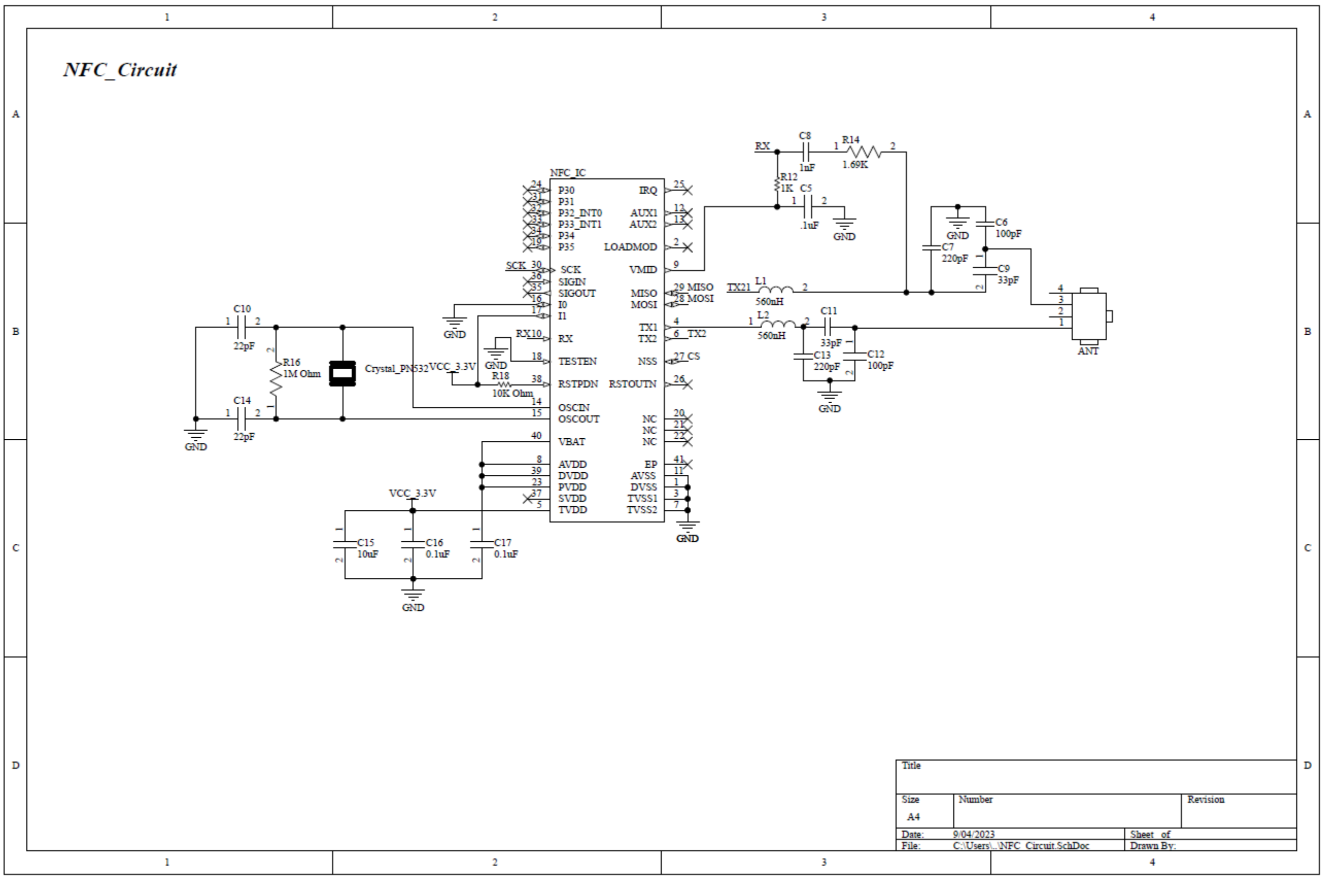
APPENDICES

Appendix 1

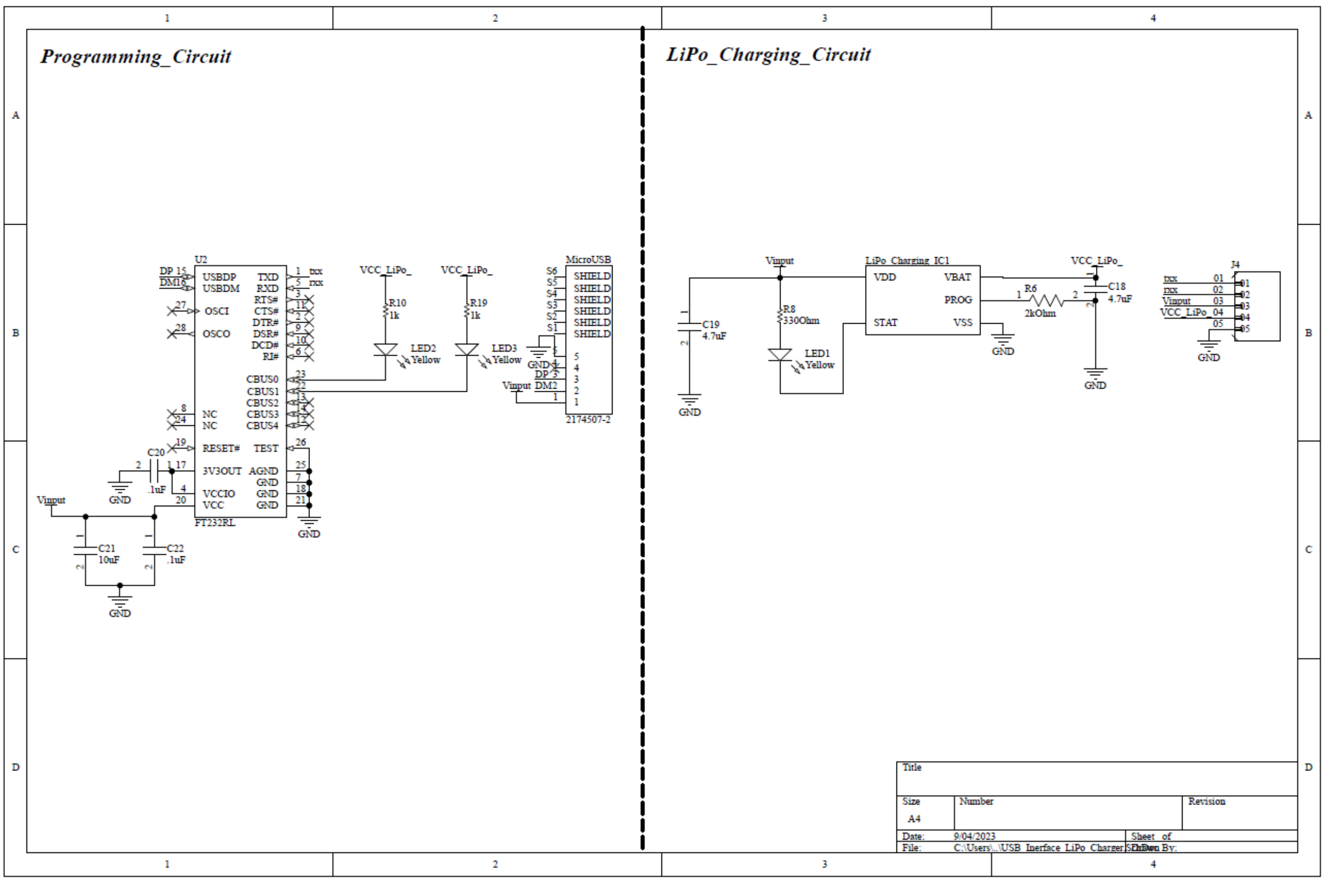
1. Schematics of microcontroller, regulator, and microSD memory.



2. Schematic for the NFC controller.



3. Schematics for Programming interface and battery charging circuity.



Appendix 2

Include necessary libraries:

- PN532.h: Library for the PN532 NFC module.
- SPI.h: Library for the Serial Peripheral Interface (SPI) communication.
- DHT.h: Library for the DHT22 temperature and humidity sensor.
- SD.h: Library for reading and writing data to an SD card.

Definition of constants and global variables:

- Pin assignment to select NFC controller on SPI port.
- Instances for the NFC controller.
- Pin assignment for the DHT22 sensor.
- Type declaration for DHT22 sensor.
- Instances for the DHT22 sensor operation.
- Pin assignment to select the microSD card.
- Create a file object to write data to an SD card.
- Various character arrays to store data.

Setup function:

- Initialize the serial communication for debugging purposes.
- Initialize the NFC module and check for its firmware version.
- Configure the NFC module to detect / read / write NFC tags.
- Initialize the DHT22 sensor.
- Initialize the microSD card and check for successful initialization.

Loop function:

- Read temperature and humidity data from the DHT22 sensor.
- Process temperature and humidity data.
- Add time stamps to the acquired data.
- Organize temperature and humidity data.
- Open or create text file with organized name structure if not created already on the microSD card to write organized data.
- If the file is opened successfully, write the temperature and humidity data to the file.
- Close the data file on microSD card.
- Configure the NFC module for peer-to-peer configuration.
- Check the availability of NFC reader.
- Transfer the text file over NFC if reader is available.
- Delete the data file after successful transmission.
- Repeat loop.

Appendix 3

Firmware code of IoT sensor node.

```
#include <PN532.h>
#include <SPI.h>
#include <DHT.h>
#include <SD.h>

#define PN532_CS 10
PN532 nfc(PN532_CS);

#define NFC_DEMO_DEBUG 1
#define DHT_PIN 2
#define DHT_TYPE DHT22
DHT dht(DHT_PIN, DHT_TYPE);

File dataFile;

char charArray[16];
char DataOut[16]; // 16 bytes
char DataIn[16];  // Should be 16 bytes

void setup(void) {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port
only
  }

#ifdef NFC_DEMO_DEBUG
  Serial.begin(9600);
  Serial.println("Hello!");
#endif
  nfc.begin();
  dht.begin();

  uint32_t versiondata = nfc.getFirmwareVersion();
  if (!versiondata) {
#ifdef NFC_DEMO_DEBUG
    Serial.print("Didn't find PN53x board");
#endif
    while (1); // Halt
  }
#ifdef NFC_DEMO_DEBUG
  // Got ok data, print it out!
  Serial.print("Found chip PN5");
  Serial.println((versiondata >> 24) & 0xFF, HEX);
  Serial.print("Firmware ver. ");
  Serial.print((versiondata >> 16) & 0xFF, DEC);
  Serial.print('.');
  Serial.println((versiondata >> 8) & 0xFF, DEC);
  Serial.print("Supports ");
  Serial.println(versiondata & 0xFF, HEX);
#endif
  // Configure board to read RFID tags and cards
  nfc.SAMConfig();
  Serial.print("Initializing SD card...");

  if (!SD.begin(8)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");
}

void loop(void) {

  float temperature_S = dht.readTemperature();
  float humidity_S = dht.readHumidity();

  Serial.print("Temperature: ");
  Serial.print(temperature_S);
  Serial.print(" Humidity: ");
  Serial.println(humidity_S);

  // Convert temperature float to char array
  String temperatureString = String(temperature_S);
  temperatureString.toCharArray(charArray, sizeof(charArray));
  strncpy(DataOut, charArray, sizeof(DataOut));

  // Append humidity float to the existing char array
  String humidityString = String(humidity_S);
  strncat(DataOut, " ", sizeof(DataOut) - strlen(DataOut) - 1);
    strncat(DataOut,   humidityString.c_str(),   sizeof(DataOut)   -
strlen(DataOut) - 1);
  // Configure PN532 as Peer to Peer Initiator in active mode
   if  (nfc.configurePeerAsInitiator(PN532_BAUDRATE_424K))  {  //  If
connection is error-free
    // Trans-receive data
    if (nfc.initiatorTxRx(DataOut, DataIn))
      delay(3000);
  }
   dataFile = SD.open("text.txt", FILE_WRITE);

  // if the file opened okay, write to it:
```

```
  if (dataFile) {
    Serial.print("Writing sensor data to text.txt...");
    dataFile.println(temperature_S);
    dataFile.println(humidity_S);
    // close the file:
    dataFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening text.txt");
  }

}
```