Unlocking the Secrets of ID Cards: Exploring Wireless Magic of Near Field Communication with Software-Defined Radios

Dickson Akuoko Sarpong and Hongzhi Guo *
School of Computing
University of Nebraska-Lincoln

October 9, 2024

Contents

1 Introduction

In this material, you will explore the world of Near Field Communication (NFC) technology and its underlying principles. NFC is a versatile and widely used technology that allows devices to exchange data wirelessly

*This project is supported by National Science Foundation under grant no. 2341846 and 2310856.

over very short distances [1]. You have likely encountered NFC in various aspects of your daily life, such as making payments with your smartphone, entering buildings with a smart card, or connecting Bluetooth devices with a simple tap. But do you know how NFC functions? What kind of signals are involved? How can you capture and visualize them? When you make payment using cell phone, what are the devices talking about and how do they talk?

This guide aims to answer these questions and more, providing you with a comprehensive understanding of NFC technology. We will begin by introducing key concepts related to digital signals and wave properties, essential for grasping how NFC operates. Additionally, we will delve into the realm of Software Defined Radio (SDR), a powerful tool that can tune to any frequency and process signals using software on a computer. SDR technology plays a crucial role in our exploration of NFC signals. You will learn how to connect an SDR device and an antenna to your computer, use SDR software to tune into the 13.56 MHz frequency utilized by NFC, and capture and visualize signals from NFC devices. By integrating SDR with NFC, you will gain insights into the radio spectrum and the specific signals involved in NFC communications. The practical component of this material involves setting up a PN5180 NFC reader interfaced with an Arduino Uno [2] to read the UID of an ISO15693 card. After successfully detecting the UID, you will use an RTL-SDR dongle [3] to capture the NFC signal between the PN5180 and the card using SDR Sharp software. The captured .wav file will then be processed in Python, enabling a detailed analysis of the NFC communication.

By the end of this material, you will have a deeper understanding of NFC technology, the principles of digital signals and wave properties, and the capabilities of SDR in exploring the radio spectrum. This knowledge will equip you with the skills to capture, analyze, and understand the signals that power NFC technology. Let us get started on this exciting journey into the world of wireless communication!

2 Learning Objectives

By the end of this lab, students are expected to:

1. Gain a solid understanding of how NFC operates, including its basic principles, typical use cases, and technological component.
2. Learn about the types of signals involved in NFC and their characteristics.
3. Develop practical skills in setting up and interfacing an NFC reader (PN5180) with an Arduino Uno.
4. Learn to set up and use Software Defined Radio (SDR) tools to capture and analyze NFC signals.
5. Process and analyze captured signals in Programming environments to understand the data flow and coding techniques used in NFC.

3 Safety Precautions

1. Ensure all electronic components are properly connected to avoid short circuits.
2. Avoid touching the RTL-SDR dongle during operation to prevent static discharge.
3. Ensure the workspace is free from any conductive materials that could interfere with the NFC signals.

4 Materials and Software

Materials	Software
PN5180 NFC Reader	
Arduino Uno	Arduino IDE
ISO15693 NFC card	Programming Tools (Jupiter Lab is preferred)
RTL-SDR Dongle	SDR Sharp Software
Donut SW Antenna	
Jumper wires	
Breadboard(optional)	

Table 1: The setup includes a PN5180 NFC Reader, Arduino Uno, and an ISO15693 NFC card for capturing NFC signals. Additionally, it utilizes an RTL-SDR Dongle and SDR Sharp software for signal analysis, with connections made via jumper wires on a breadboard. Programming is done in Arduino IDE and Jupyter Lab.

5 Wireless Communication

Wireless communication is a pivotal technology in our interconnected world, enabling the transmission of data over distances without the need for physical connections like wires or cables. This technology forms the backbone of modern communication systems, facilitating everything from mobile phone calls and internet browsing to satellite transmissions and IoT (Internet of Things) devices. At its core, wireless communication relies on electromagnetic waves to transfer information. These waves travel through the air and can be modulated to carry data. The primary types of electromagnetic waves used in wireless communication include radio waves, microwaves, and infrared waves. Each type has specific properties and applications, such as long-range communication for radio waves and short-range communication for infrared waves. Key components of wireless communication systems include transmitters, receivers, and antennas. The transmitter encodes and modulates the data into a signal, which is then transmitted through an antenna. The receiver captures the signal via its antenna and demodulates it to retrieve the original data. This process is governed by various protocols and standards, ensuring reliable and efficient communication.

One of the significant advancardes of wireless communication is its flexibility and mobility. It eliminates the need for extensive wiring, making it ideal for mobile devices and remote locations. This mobility is particularly crucial for technologies like smartphones, Wi-Fi, and Bluetooth, which rely on seamless and constant connectivity.

Wireless communication also underpins several advanced technologies. For instance, in cellular networks, different generations (from 1G to the current 5G) have progressively improved data rates, connectivity, and efficiency. Additionally, wireless communication is essential for the operation of satellite systems, enabling global positioning systems (GPS), weather monitoring, and international broadcasting.

5.1 Near-Field Communication

Near-Field Communication (NFC) is a short-range wireless communication technology that allows devices to exchange data when they are in close proximity, typically within a few centimeters. NFC operates at a frequency of 13.56 MHz and is widely used for secure, contactless transactions and data transfer. NFC technology is based on inductive coupling between two loop antennas [4], one in the transmitting device and one in the receiving device. When the antennas are brought close together, a magnetic field is created, allowing data to be transferred wirelessly. The magnetic induction-based communication has been adopted in many complex environments, such as underground and underwater [5, 6, 7].

The NFC technology supports various modes of operation, including reader/writer mode, peer-to-peer mode,

and card emulation mode [1]. One of the most common applications of NFC is in contactless payment systems, where users can make secure payments by simply tapping their NFC-enabled devices, such as smartphones or smart cards, on a payment terminal. Additionally, NFC is used for access control in buildings, data exchange between electronic devices, and pairing Bluetooth devices quickly and easily. NFC's simplicity and ease of use make it a popular choice for many applications, providing a seamless and secure way to connect and transfer data between devices without the need for physical contact.

5.2 Software Defined Radio (SDR)

Software Defined Radio (SDR) is a transceiver capable of generating and receiving various wireless signals through software programming. This flexibility underscores SDR's versatility and adaptability, as it can perform multiple functions such as filtering, error correction, synchronization, modulation, and demodulation. SDRs function similarly to desktop computers, where a single hardware platform can execute various tasks based on the software applications installed. They utilize fast reprogrammable devices like Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs), or General-Purpose Processors (GPPs) to handle tasks traditionally managed by hardware in conventional radio systems. This design allows SDRs to easily receive and transmit different radio protocols through software updates, enhancing their ability to utilize the radio frequency spectrum effectively without the need for hardware changes.

SDR offers a flexible and cost-effective solution, benefiting service providers, product developers, and end-users by managing equipment expenses more efficiently. Moreover, Software Defined Radio (SDR) represents a significant advancement in radio technology, providing unparalleled flexibility, upgradability, and security. By replacing traditional hardware components with software, SDRs enable multifunctionality and adaptability, leading to improved performance across various applications. Moreover, SDRs empower individuals and organizations to monitor and secure their radio communications proactively, mitigating potential vulnerabilities and ensuring data privacy. The widespread adoption of SDRs marks a transformative shift towards more efficient, versatile, and secure radio solutions in the digital age.

5.3 Applications of Near Field Communication

Near Field Communication (NFC) technology is widely used in various applications, offering convenience and enhanced functionality in everyday activities. One of the most common uses of NFC is in contactless payment systems, such as those enabled by smartphones and smartwatches, where HQ-antennas are employed to enhance this communication [8]. Users can make secure transactions by simply tapping their device on a compatible payment terminal, eliminating the need for cash or physical cards.

In addition to payments, NFC is extensively used for access control and identity verification. Smart cards and mobile devices equipped with NFC can serve as keys for secure access to buildings, offices, and vehicles. This technology is also employed in public transportation systems, where passengers can use NFC-enabled cards or devices to quickly and efficiently pay for fares and access transit services, In [9], a new approach for ticketing systems using NFC-enabled smartphones is proposed, which includes a method for validating tickets.

NFC technology enhances connectivity and data sharing between devices. For example, NFC can be used to quickly pair Bluetooth devices, such as headphones and speakers, by tapping them together. It also facilitates the exchange of information between smartphones, enabling users to share contacts, photos, and other files with a simple tap [10].

Moreover, NFC is used in the retail industry to improve customer experiences. NFC cards embedded in products or advertisements allow customers to access additional information, such as product details, reviews, and promotional offers, by tapping their smartphones. This interactive capability helps retailers engage with customers and provide personalized services.

In the healthcare sector, NFC technology is utilized for patient identification and medical record management. NFC-enabled wristbands or cards can store patient information, ensuring accurate and efficient access to medical histories and treatment plans. This reduces errors and enhances the overall quality of care [11].

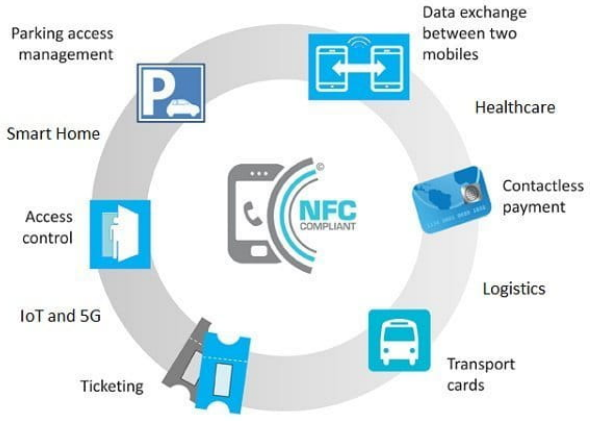


Figure 1: Applications and Future of Near Field Communication [12].

6 NFC System: Theory and Signal Recording

6.1 Signals

In communication systems, signals are typically generated by transmitters, converted into electromagnetic waves, and transmitted through the air as part of the electromagnetic spectrum. These signals can represent various types of information, such as sound and visual data, and are received by receivers where they are converted back into their original form for interpretation or processing. Thus, in the context of communications, a signal serves as the carrier of information, enabling communication over distances.

Common types of signals include sine waves (smooth periodic oscillations), square/rectangular waves (abrupt transitions between volcarde levels, with square waves having equal duration segments), triangular/sawtooth waves (linear volcarde variations over time, with sawtooth waves featuring immediate drop), and pulses (sudden disturbances representing glitches/errors or individual pieces of digital information). Understanding these signals is crucial for signal analysis and interpretation.

Signals are characterized by many parameters, among which the frequency and phase are two important ones. Frequency represents the rate at which a signal repeats itself and is measured in Hertz (Hz). It indicates the number of cycles the signal completes in one second, with higher frequencies corresponding to more rapid oscillations. The period of a signal, on the other hand, is the time it takes for the signal to complete one full cycle. Frequency and period are reciprocals of each other, with 1/period equaling the frequency.

Figure 2 shows different frequencies with the respective time and amplitude.

6.2 Communication between NFC Reader and Card

In every NFC transaction, the reader initiates communication, and only one participant—either the reader or the card—transmits information at a time. Each data packet is framed by a start-of-frame (SOF) signal at the beginning and an end-of-frame (EOF) signal at the end. This enclosed data packet is referred to as a frame.

Communications between the reader and the card utilize Amplitude Shift Keying (ASK) modulation, i.e, different strength of the signal represent different data. Two modulation indexes, 10 percent and 100 percent,

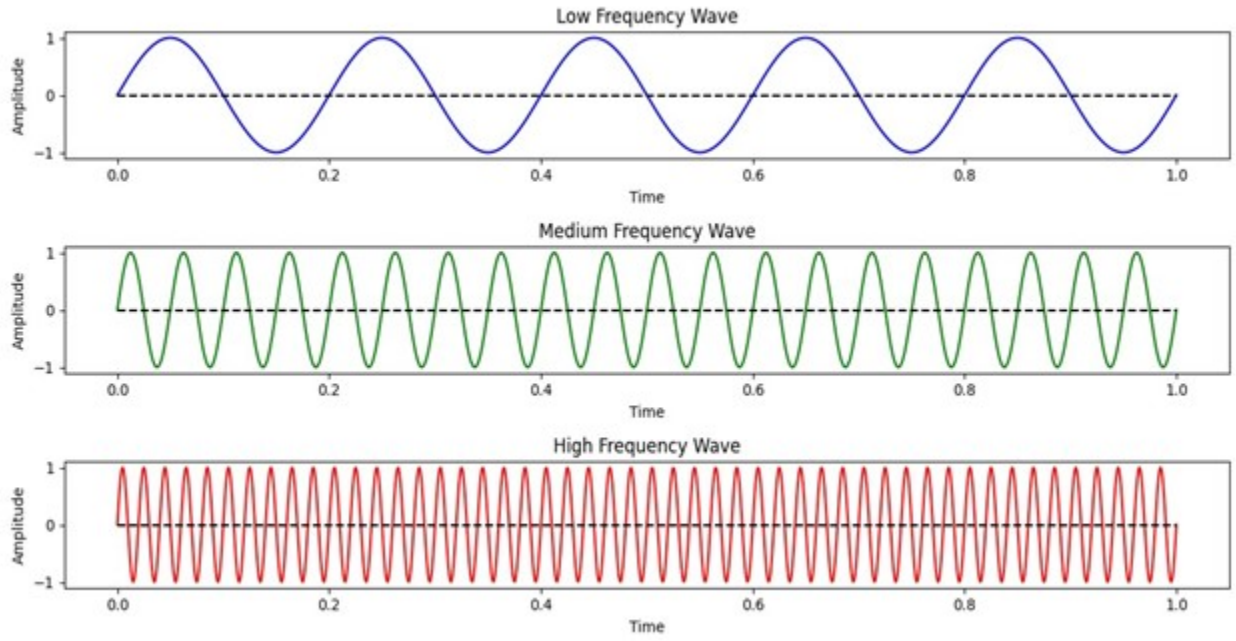


Figure 2: Signals with different frequencies with respective amplitude and time.

are employed, and the card must be capable of decoding both. The reader determines which modulation index is used for each communication.

Data coding is implemented using pulse position modulation. The card must support two data coding modes. The reader will select the coding mode and indicate it to the card within the start of frame (SOF). Framing is designed for easy synchronization and protocol independence. Each frame is marked by a start of frame (SOF) and an end of frame (EOF). Unused options are reserved for future ISO/IEC use. The card must be ready to receive a frame from the reader within 300 µs after sending one, and within 1 ms of activation by the powering field.

6.3 Reader to Card Communication

6.3.1 Start of Frame

The start-of-frame (SOF) pattern consists of two modulation pauses. The position of the second pause indicates whether the frame will use the "1 out of 256" or "1 out of 4" data coding mode.

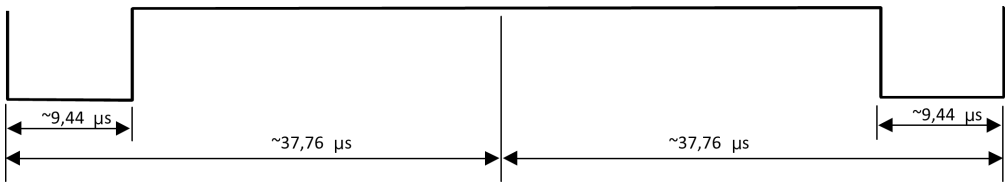


Figure 3: Start of frame of the 1 out of 256 coding mode.

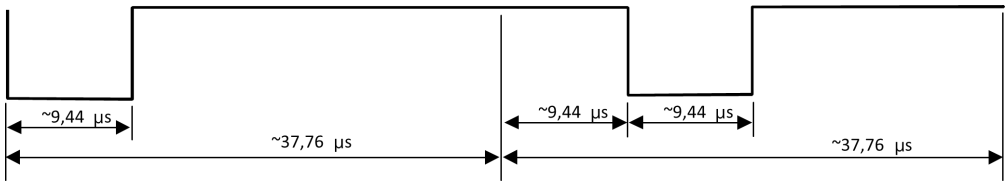


Figure 4: Start of frame of the 1 out of 4 coding mode.

6.3.2 End of Frame

The transmission of the end-of-frame (EOF) pattern lasts for 37.76 microseconds. The EOF is consistent for both coding modes and consists of one modulation pause.

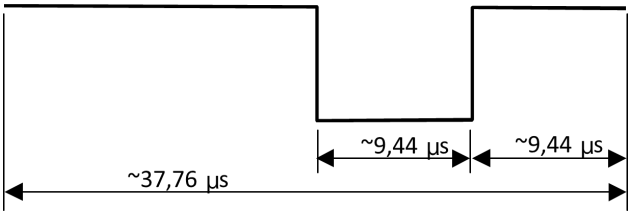


Figure 5: Reader to card EOF (identical for both coding modes).

6.3.3 Data Coding (Reader to Card)

Pulse Position Modulation (PPM) is a data encoding method where the position of a pulse within a fixed time slot represents specific bits of data. In this lab, the "1 out of 4" mode is used for reader to card transmission(termed as **Inventory request**), where each pulse position encodes two bits, such as 00, 01, 10, or 11. A byte, consisting of 8 bits, is formed by combining four successive pairs of these bits, with the least significant pair transmitted first. This modulation method allows for a data transmission rate of 26.48 kilobits per second (kbits/s). Data coding and bit representation of inventory request is described by Figure 6.

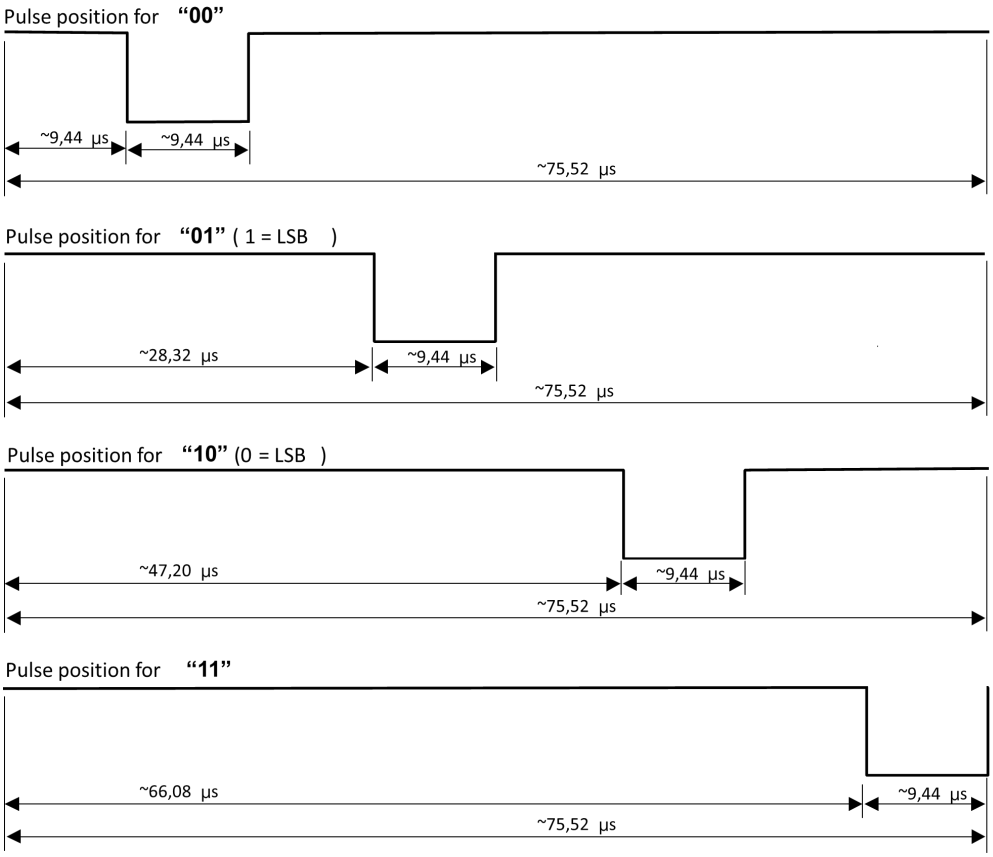


Figure 6: Reader to card pulse position modulation for 1 out of 4.

6.3.4 Inventory Request Command

The Inventory request command in NFC is used to detect tag(s) within a reader's range by initiating an anticollision sequence. This request includes several components: flags to control options and indicate the presence of fields like the Application Family Identifier (AFI), the command code(also known as inventory), optional AFI for identifying specific tag families, mask length and value for selecting tags, and a CRC for error detection. The message structure begins with a Start of Frame (SOF) and includes these components before concluding with a CRC16 and End of Frame (EOF).

SOF	Flags	Inventory	AFI	Mask Length	Mask Value	CRC16	EOF
	8 bits	8 bits	8 bits	8 bits	0-64 bits	16 bits	

Table 2: Inventory request format

6.4 Card to Reader Communication

6.4.1 Single Subcarrier

The start-of-frame (SOF) sequence includes an unmodulated period of 56.64 μs, followed by 24 pulses at a frequency of 423.75 kHz, and a logic 1 signal that begins with an unmodulated period of 18.88 μs and is followed by 8 pulses at a frequency of 423.75 kHz.

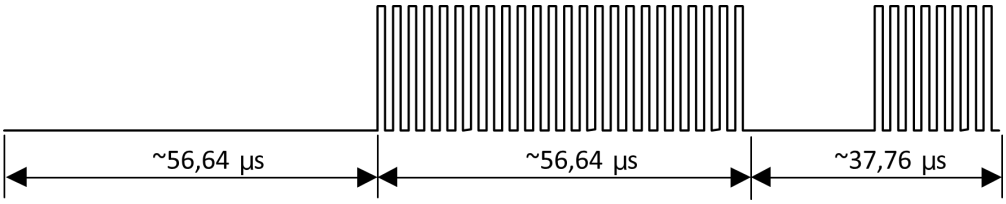


Figure 7: card to Reader SOF, single subcarrier.

6.4.2 Two Subcarriers

The start-of-frame (SOF) sequence includes three components: initial pulses at a frequency of 484.28 kHz, followed by 24 pulses at a frequency of 423.75 kHz, and a logic 1 signal that starts with 9 pulses at 484.28 kHz and ends with 8 pulses at 423.75 kHz.

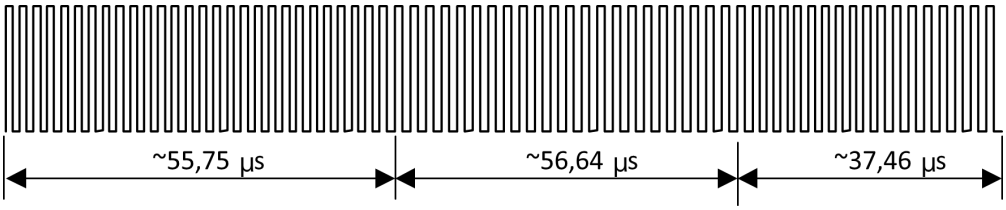


Figure 8: Card to Reader SOF, single subcarrier.

6.4.3 EOF When Using One Subcarrier

The end-of-frame (EOF) consists of three parts: it starts with a logic 0 signal, which includes 8 pulses at a frequency of 423.75 kHz followed by an unmodulated period of 18.88 μs. This is followed by 24 pulses at a frequency of 423.75 kHz and concludes with an unmodulated period of 56.64 μs.

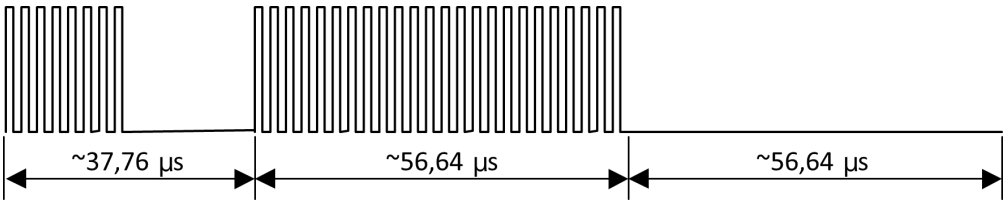


Figure 9: End of frame, one subcarrier.

6.4.4 EOF When Using Two Subcarriers

The end-of-frame (EOF) consists of three parts: a logic 0 signal, starting with 8 pulses at a frequency of 423.75 kHz followed by 9 pulses at a frequency of 484.28 kHz; then 24 pulses at a frequency of 423.75 kHz; and finally, 27 pulses at a frequency of 484.28 kHz.

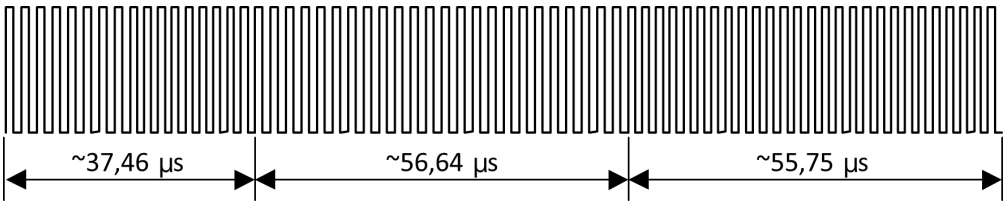


Figure 10: End of Frame When Using Two Subcarriers.

6.4.5 Data Coding (Card to Reader)

Manchester coding is a method used to maintain synchronization between transmitting and receiving devices by encoding data bits with a specific transition within each bit period. In a data transmission from a card to a reader(also termed as **Inventory response**), a logic 0 is encoded by transmitting 8 pulses at 423.75 kHz followed by an unmodulated period of 18.88 µs. Conversely, a logic 1 is encoded by starting with an unmodulated period of 18.88 µs, followed by 8 pulses at the same frequency. Data shall be encoded using Manchester coding, according to the schemes described in Figure 11 and Figure 12.

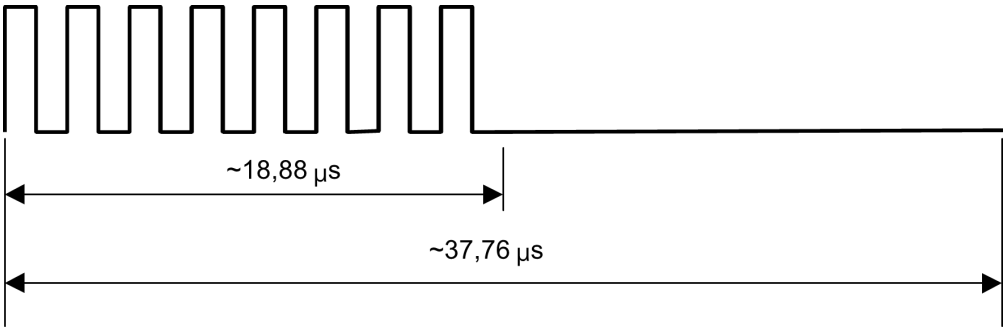


Figure 11: Logic 0.

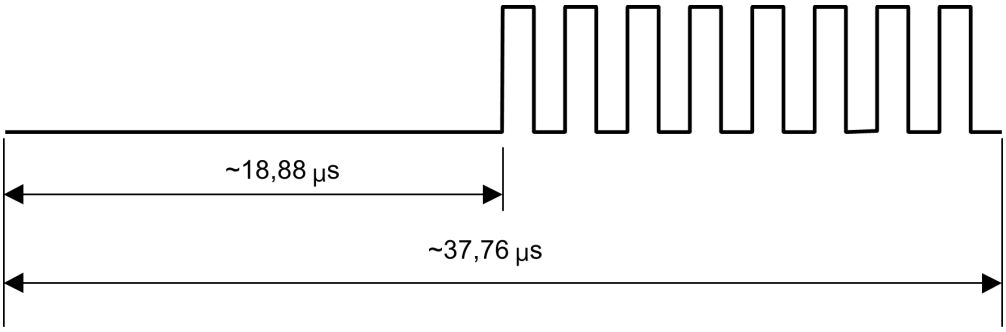


Figure 12: Logic 1.

6.4.6 Inventory Response Command

The inventory response in an NFC system is a structured data packet sent by a card in response to an inventory request from the reader. It includes an 8-bit DSFID (Data Storage Format Identifier) that indicates the card's data format, and a 64-bit unique ID (UID) that serves as a globally unique identifier for the card. The response is framed by a Start of Frame (SOF) and End of Frame (EOF), with an 8-bit Flags field

providing additional control information, followed by the DSFID, UID, and a 16-bit CRC for error detection, ensuring the integrity of the communication.

SOF	Flags	DSFID	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Table 3: Inventory response format

6.5 Lab Materials

In this study, we used electronic devices such as the reader, Arduino, the card, and the SDR dongle, cables to connect them, antenna attached to the dongle for receiving the signal and some software applications to receive and process the signals. Therefore, the system is basically divided into three major parts: initiating the reader—card signals, receiving and saving the signal data samples in a file using SDR software and then processing the samples collected with a bunch of supportive libraries in python.

6.5.1 Arduino Uno and PN5180

Hardware Requirements:

- Arduino Uno
- PN5180 NFC Reader
- Jumper wires
- Breadboard (optional)

Software Requirements:

- Arduino IDE
- PN5180 library for Arduino

6.5.2 Hardware Connections

The connection between the PN5180 NFC Reader and the Arduino UNO are shown in Fig. 13 and Table 4.

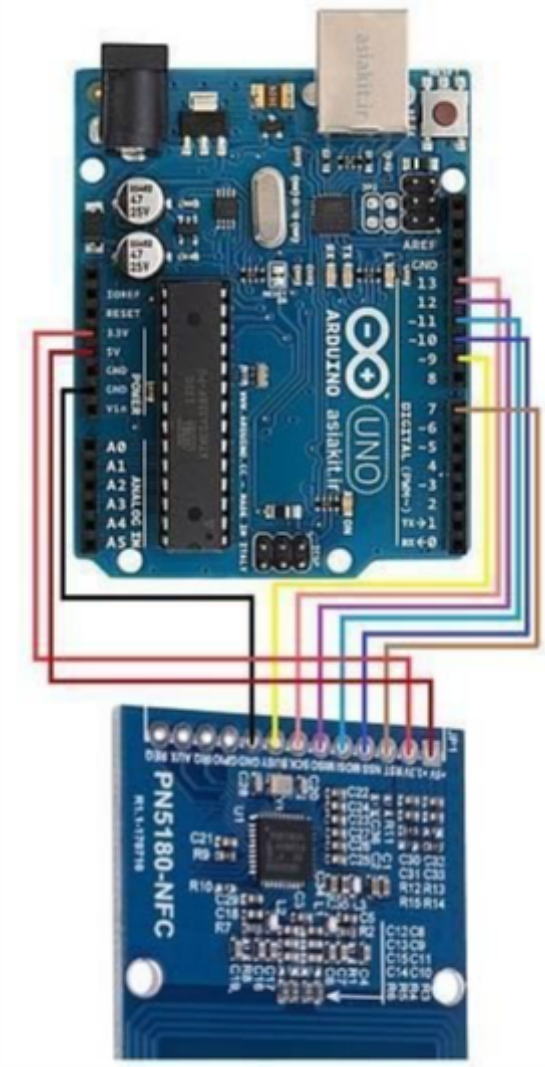


Figure 13: Wiring Diagram for PN5180 NFC Reader Interfaced with Arduino Uno.

PN5180 PIN	**ARDUINO UNO PIN**
VCC(+5V)	5V
VCC(+3.3V)	3.3V
RST	PIN 7
NSS	PIN 10
MOSI	PIN 11
MISO	PIN 12
SCK	PIN 13
BUSY	PIN 9
GND	GND

Table 4: Connection between the PN5180 and the Arduino Uno.

6.6 Software Setup:

1. **Install Arduino IDE:**
   - Download and install the Arduino IDE from the Arduino website https://www.arduino.cc/en/software [13].
2. **Install PN5180 Library:**
   - Open the Arduino IDE
   - Go to Sketch > Include Library > Manage Libraries
   - In the Library Manager, search for "PN5180" (To Install the "PN5180" library by Elec house, GitHub or other available libraries)
3. **Writing the Code**
   - Open the Arduino IDE
   - Create a new sketch by going to File > New.
   - Copy and paste the "Arduino code" attached to the manual into the new sketch
4. **Uploading the Code**
   - Connect your Arduino Uno to your computer using a USB cable.
   - Select the correct board and port:
     - Go to Tools > Board and select Arduino Uno.
     - Go to Tools > Port and select the port to which your Arduino Uno is connected.
   - Upload the code to the Arduino by clicking the **Upload button** (right arrow icon) in the Arduino IDE.

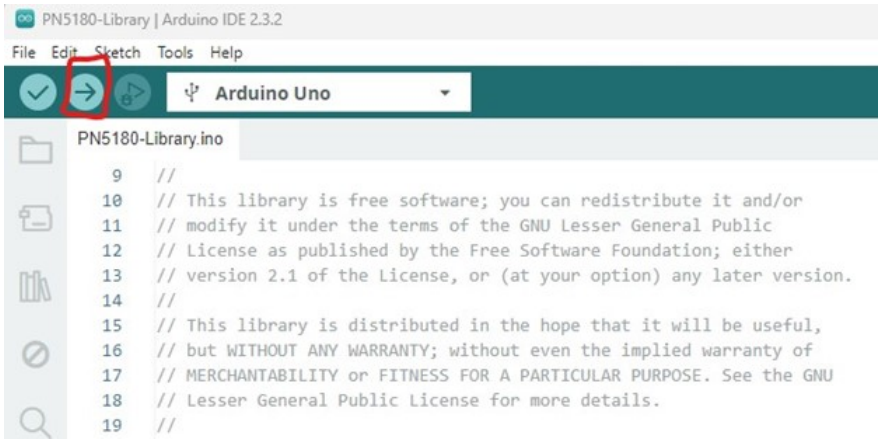


Figure 14: Indication of Upload Button.

5. **Testing the setup**
   - Open the Serial Monitor from the top right corner

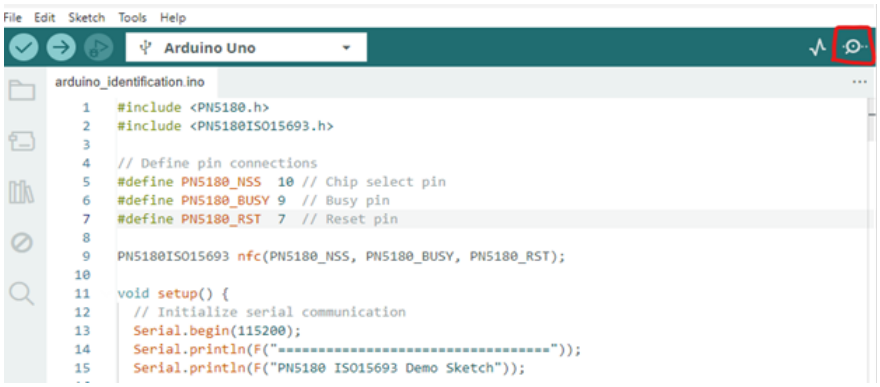


Figure 15: Indication of Serial Monitor Button.

- Set the baud rate to 115200 baud in the Serial monitor's window

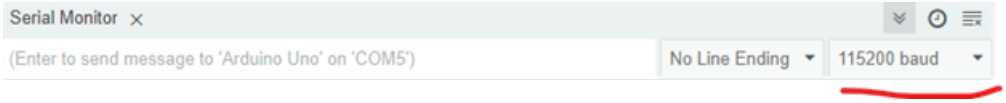


Figure 16: Setting baud rate.

- Place an NFC card near the PN5180 reader

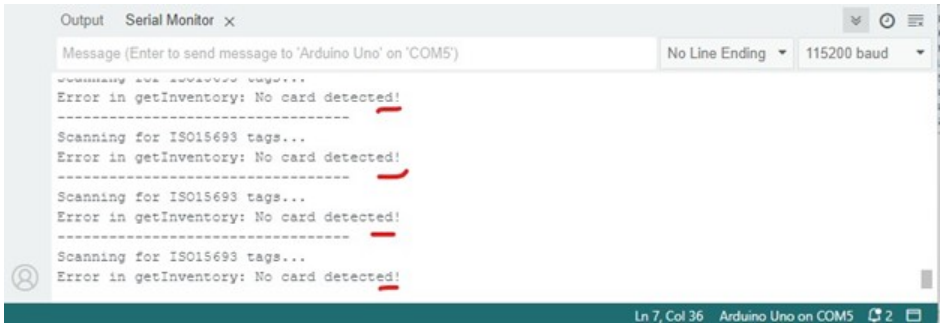


Figure 17: If no card is found, "No card detected" displayed by serial monitor.

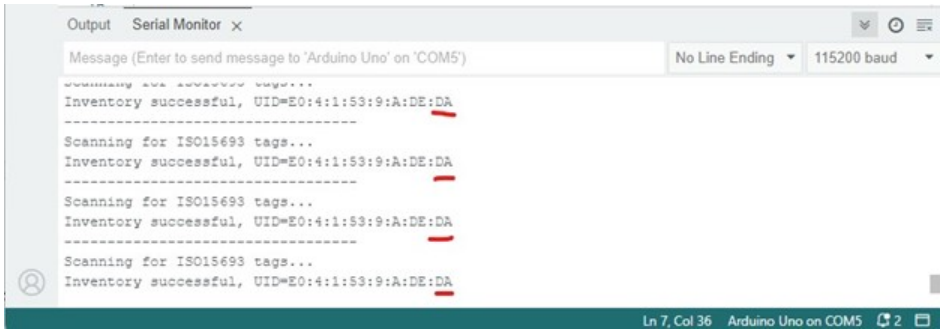


Figure 18: If card is found, "Inventory successful and UID of card" displayed by serial monitor.

This completes the process of setting up the PN5180 NFC reader with an Arduino Uno and running code to identify the UID of NFC cards.
6. **SDR# Software** The SDR software application serves as the interface for controlling the SDR dongle, capturing the raw In-phase and Quadrature (IQ) samples of the NFC signal, and saving them to a file for further analysis.

- RTL-SDR: It captures signals across a wide variety of frequency spectrum and converts them into digital data for further processing.
- SMA male to SMA male cable: From the SDR dongle to the antenna.
- Donut SW Antenna: Capturing the signals.

Installing the **SDR# software** is straightforward following the steps below:

- Go to the link: `https://airspy.com/download/` [14], and click on download from the top menu.
- Click on the download button next to "Software Defined Radio Package"

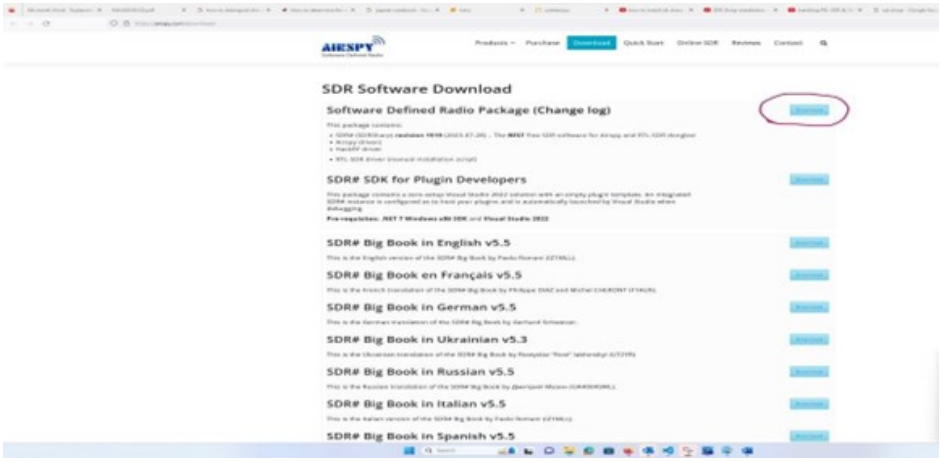


Figure 19: Downloading SDR# Software Package.

- After the file has been downloaded, extract the zipped folder to the desktop.
- Click on the file install-rtlsdr in the unzipped folder in Figure 20.

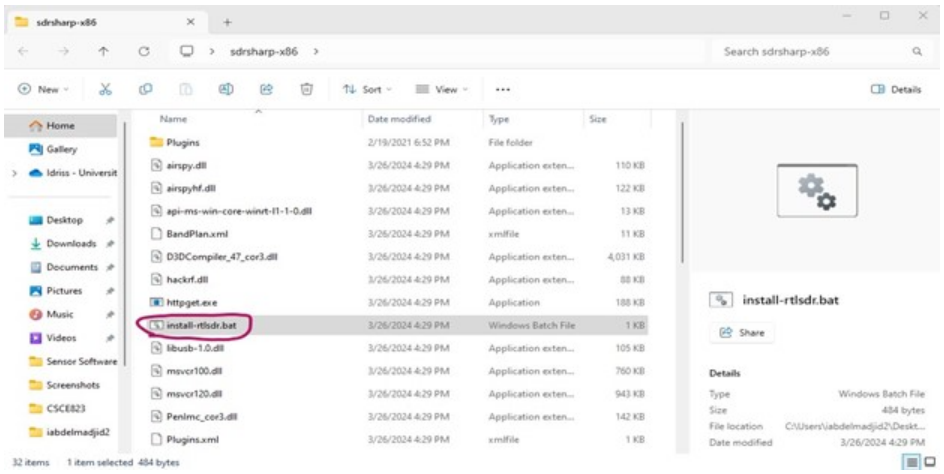


Figure 20: Unzip and run rtl-sdr file.

- When the command line window shows up as in Figure 21, press any key to continue. It is going to take a few seconds to finish the installation.

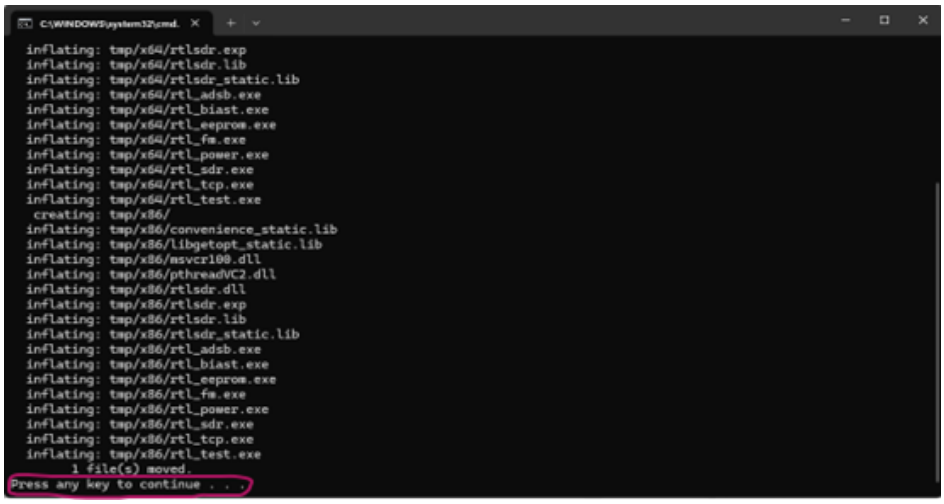


Figure 21: Command window

- Go back to the unzipped folder and click on the Zadig file
- When the window opens, select Options, then List All Devices
- Select Bulk – In, Interface (Interface 1) from the drop downlist then press Reinstall Driver as shown in Figure 22.

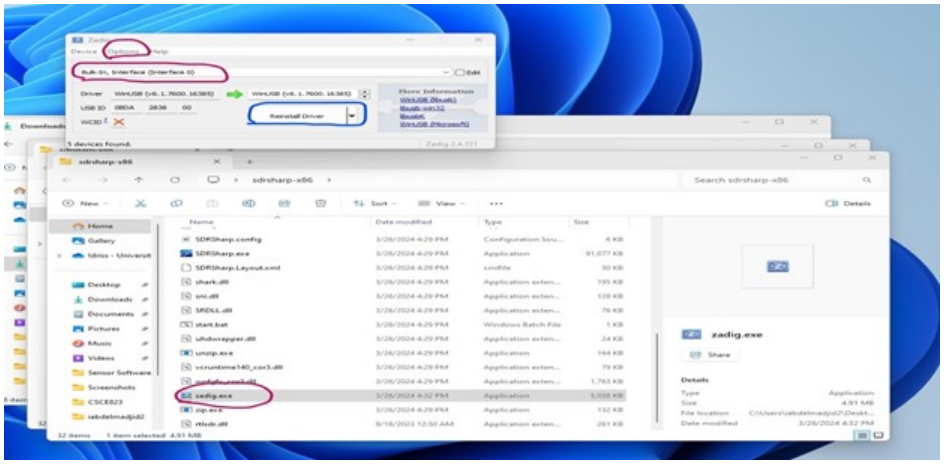


Figure 22: Installing drivers.

- Figure 23 will show up when it is done, then you are all set with the installation.

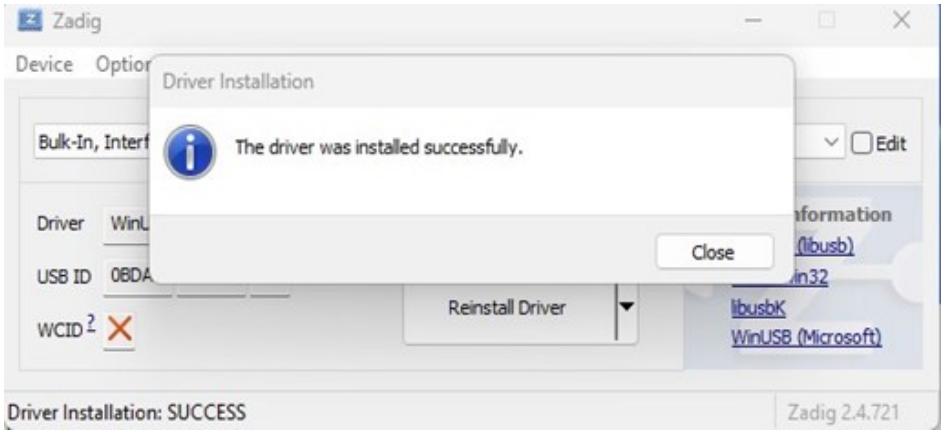


Figure 23: Drivers installed successfully.

6.7 Steps to Capture and Save IQ Samples in a File

- Run SDR# application clicking on the icon as shown in Figure 24.

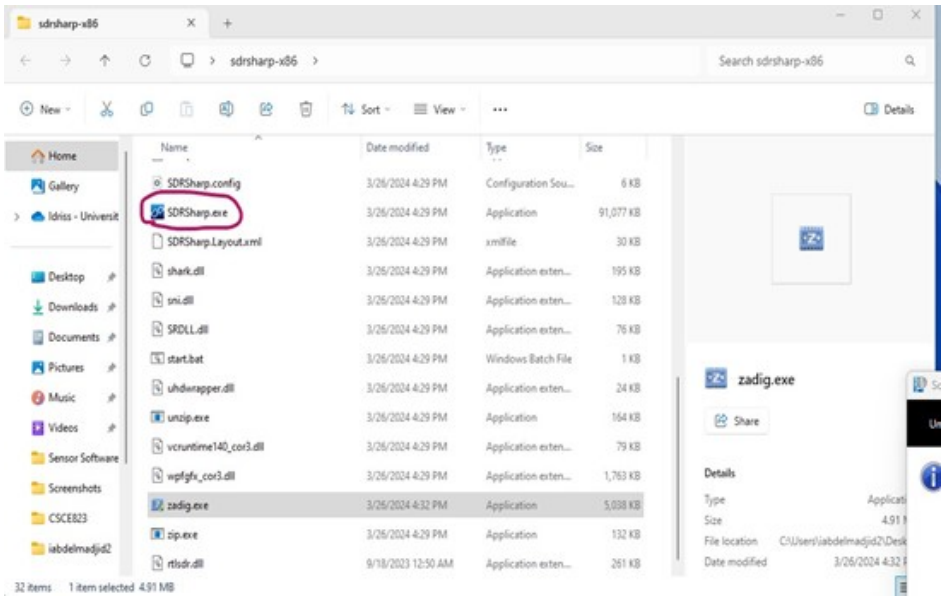


Figure 24: SDR# application.

- Click from the main menu and select from the source list "RTL-SDR(USB)" shown in Figure 25.

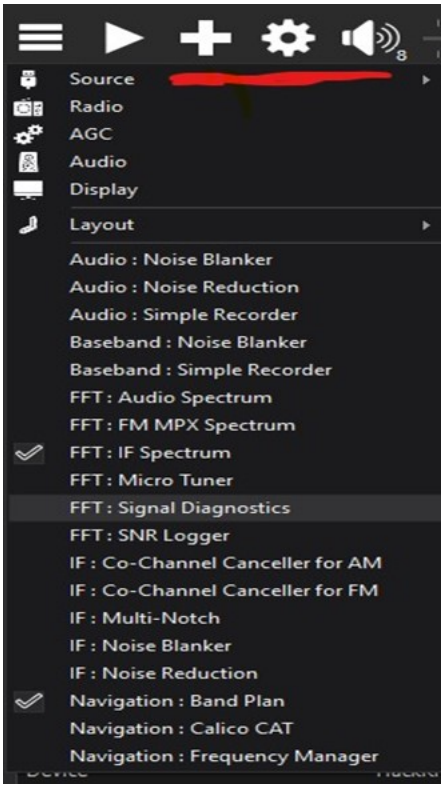


Figure 25: Source option.

- The following GUI in Figure 26 shows up.

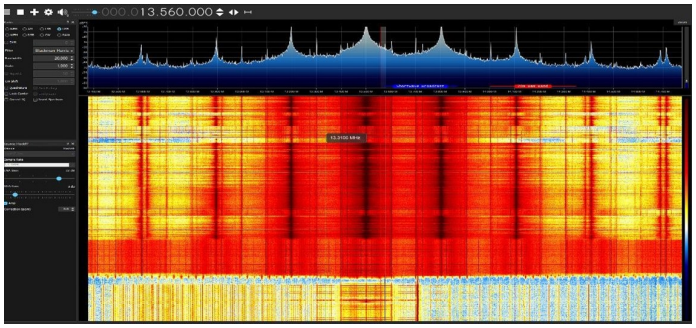


Figure 26: SDR# GUI.

- Connect your SDR dongle – attached with the antenna - to your computer if it is not yet connected.
- Set the frequency to 13.56 MHz as you see in Figure 27.

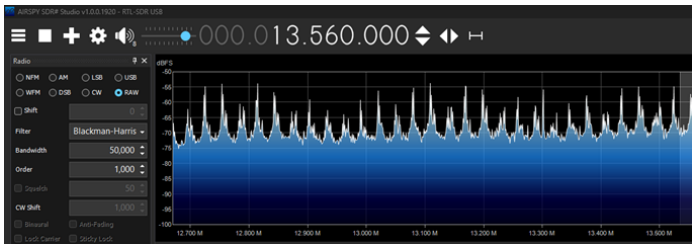


Figure 27: Frequency setting.

- Adjust the gain, sampling rate and filter settings. You might need to experiment to find the optimal settings for capturing the NFC signal as in Figure 28.

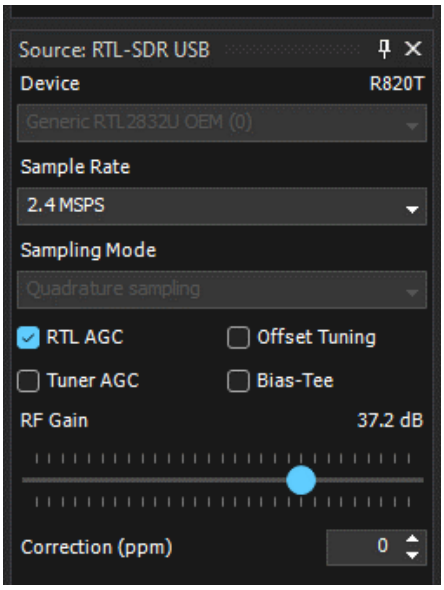


Figure 28: Gain, sampling rate and filter setting.

- Position your NFC reader close to the SDR antenna.

- Initiate communication with the NFC reader/card by running the uid identification code on Arduino ide. This will generate the signal you want to capture.
- In SDR#, activate the "Baseband Recorder" plugin and start recording the signal. As in Figure 29.

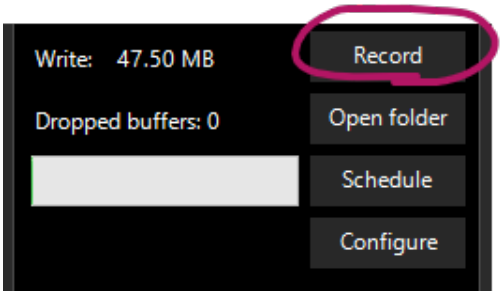


Figure 29: Baseband recorder.

- Stop recording once you have captured the desired signal length as in Figure 30.

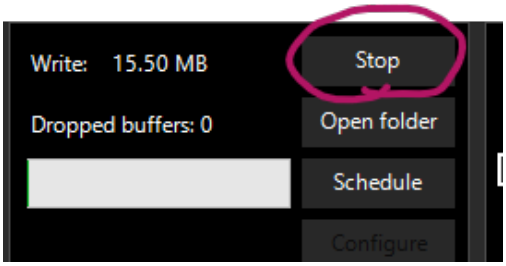


Figure 30: Captured signal length in Baseband recorder.

- **The file will be saved upon clicking stop** and its file format will depend on the plugin used, but **WAV** is common.

6.7.1 Experiment Setup

In Figure 31, the setup demonstrates an NFC (Near Field Communication) testing environment, where a PN5180 NFC Reader interacts with an ISO15366 NFC Card. The communication signals between the reader and tag are captured and displayed in real-time on a monitor using an RTL-SDR Dongle, which is connected to a computer. The setup includes a Donut SW Antenna for signal capturing, an Arduino Uno R3 for controlling the reader, and a breadboard for circuit prototyping. During this experiment, several precautions were taken to ensure safety and data accuracy. Electrical safety measures, such as proper insulation and grounding of components like the Arduino Uno R3 and PN5180 NFC Reader, were prioritized to prevent short circuits and electrical noise. The setup was kept stable and free from movement, especially the antenna and NFC card, to ensure consistent signal readings. Additionally, the experiment was conducted in a controlled environment, away from potential electromagnetic interference from other electronic devices.

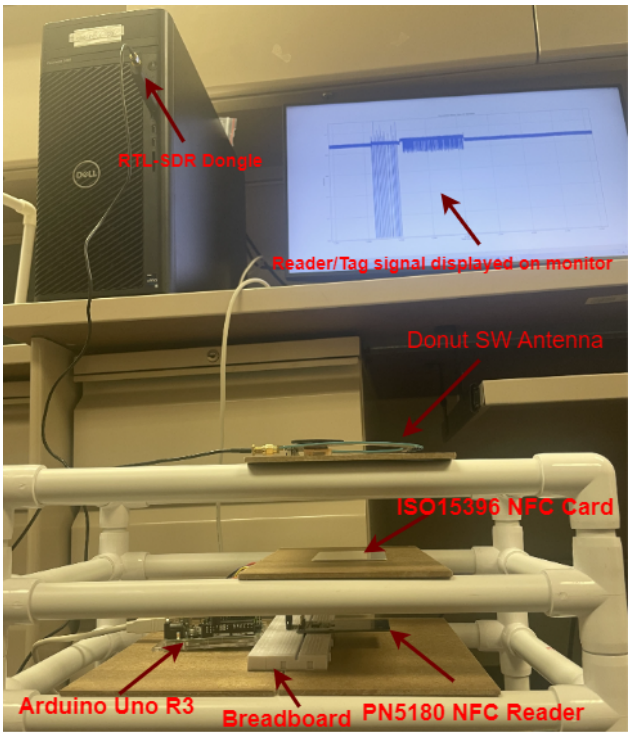


Figure 31: Experiment setup.

Additional Notes:

- The captured signal will be raw data and will require further processing or analysis to extract meaningful information.
- Dedicated NFC analysis tools might be needed to decode the captured signal depending on the specific NFC protocol used.

7 Analyzing Captured Signal

7.1 Python for Plotting the Signals

Having installed python and with the aid of additional extensions and libraries, we could read and print the IQ samples from the captured NFC signals. The IDE used to run Python script is Visual Studio Code.

1. Copy the python code **(IQsample.py)** attached to the manual into the Visual studio code and specify your recorded WAV file path. This prints the IQ samples from the recorded NFC (reader's) signals for visualization. This is shown in Figure 32 below:

```
(0.130767822265625-0.356048583984375j)
(0.00515747070703125-0.37957763671875j)
(-0.128265380859375-0.355987548828125j)
(-0.230316162109375-0.293121337890625j)
(-0.316650390625-0.206695556640625j)
(-0.371612548828125-0.088836669921875j)
(-0.37945556640625+0.044677734375j)
(-0.34014892578125+0.162506103515625j)
(-0.26947021484375+0.264617919921875j)
(-0.1673583984375+0.3431396484375j)
(-0.041717529296875+0.38238525390625j)
(0.0838623046875+0.366607666015625j)
(0.193756103515625+0.3272705078125j)
(0.2957763671875+0.240814208984375j)
(0.358551025390625+0.122955322265625j)
(0.37420654296875+0.005126953125j)
(0.35845947265625-0.12048339843751j)
```

Figure 32: Printed IQ samples.

2. Create a new python file and copy the code **(IQwave.py)** attached to the manual into your python file. This plots the reader's NFC request signal for visualization. This is shown in Figure 33 below:

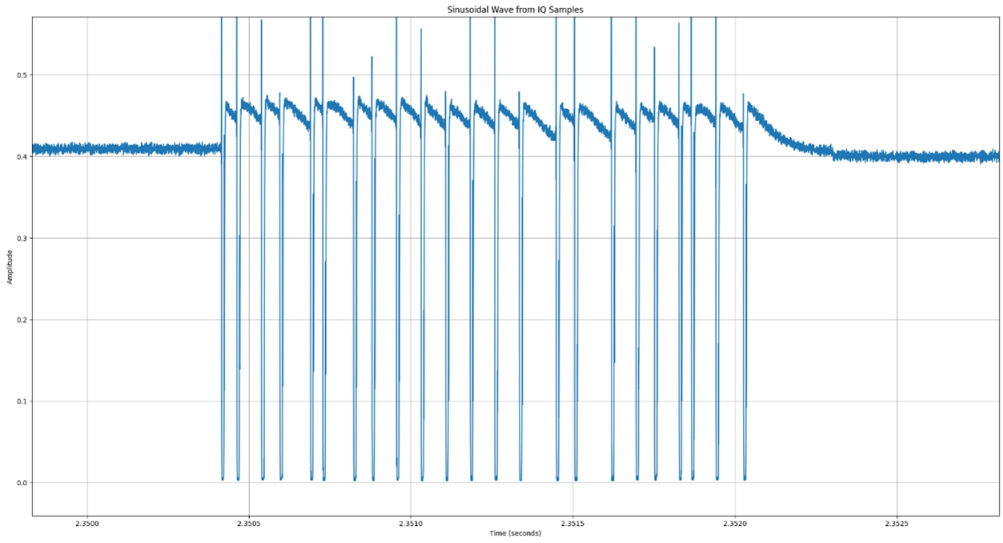
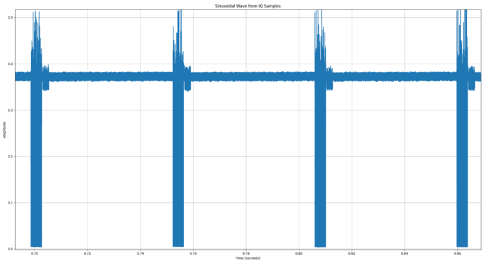
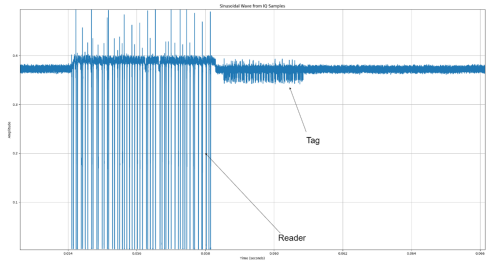


Figure 33: Reader's NFC request signal.

3. Initiate NFC communication by running the code on Arduino IDE. Now Position your NFC reader and card close to the SDR antenna. Create a new python file and copy the code **(IQwave.py)** attached to the manual into your python file. This plots the reader's NFC request and card's response signal for visualization. This is shown in Figure 34a and 34b below:



(a) Reader/card signals.



(b) Single reader/card request/response signal.

Figure 34: A plot of both reader and card signal for captured NFC signal length.

7.2 Bit Representation and Coding

Data shall be encoded using Manchester coding, according to the schemes described in section 1.12. This represents the data sent to the reader as a response to a particular reader's request.

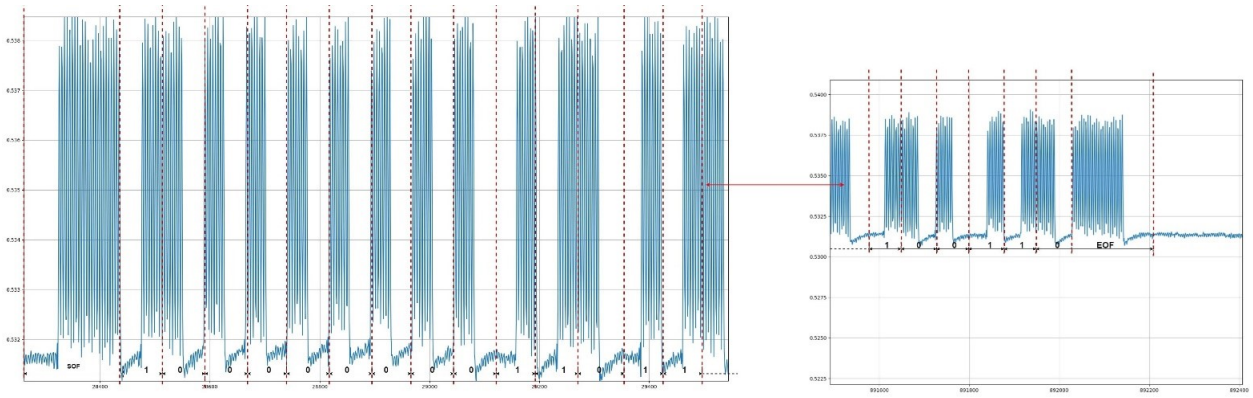


Figure 35: Data coding of card's response signals.

8 Conclusion

This lab provides a comprehensive guide to understanding and implementing Near Field Communication (NFC) technology using Software Defined Radio (SDR). It covers key concepts, equipment setup, and step-by-step procedures for capturing and analyzing NFC signals. Practical exercises include interfacing an Arduino Uno with a PN5180 NFC reader and using SDR # software and Python for signal processing. This lab equips students with essential skills for exploring NFC technology and its applications.

References

[1] NFC Forum, "NFC Forum: Shaping the Future of Contactless Technology." https://nfc-forum.org/, 2024. Accessed: 2024-08-09.

[2] Arduino Uno , "Arduino uno rev3." https://store.arduino.cc/products/arduino-uno-rev3, 2024. Accessed: 2024-08-12.

[3] RTL-SDR, "Radioworld magazine article about software defined radios for shortwave listening." https://www.rtl-sdr.com/, 2024. Accessed: 2024-08-12.

[4] Guo, Hongzhi and A. A. Ofori, "The internet of things in extreme environments using low-power long-range near field communication," *IEEE Internet of Things Magazine*, vol. 4, no. 1, pp. 34–38, 2021.

[5] Guo, Hongzhi, Z. Sun, and P. Wang, "Joint design of communication, wireless energy transfer, and control for swarm autonomous underwater vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1821–1835, 2021.

[6] A. A. Ofori and Guo, Hongzhi, "Magnetic blind beamforming for battery-free wireless sensor networks," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1819–1832, 2022.

[7] Guo, Hongzhi, M. Prince, J. Ramsey, J. Turner, M. Allen, C. Samuels, and J. A. Nuako, "A low-cost through-metal communication system for sensors in metallic pipes," *IEEE Sensors Journal*, vol. 23, no. 8, pp. 8952–8960, 2023.

[8] T. W. C. Brown and T. Diakos, "On the design of nfc antennas for contactless payment applications," in *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, pp. 44–47, 2011.

[9] C. Saminger, S. Grünberger, and J. Langer, "An nfc ticketing system with a new approach of an inverse reader mode," in *2013 5th International Workshop on Near Field Communication (NFC)*, pp. 1–5, 2013.

[10] M. H. Alharbi and O. H. Alhazmi, "Prototype: User authentication scheme for iot using nfc," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–5, 2019.

[11] Y.-C. Weng, "Nfc convenient application: A case of personal health record," in *2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pp. 246–249, 2020.

[12] R. Baby, "Applications and future of near field communication," tech. rep., Rajiv Baby, 2024. Accessed: 2024-07-26.

[13] Arduino IDE, "Arduino ide software." https://www.arduino.cc/en/software, 2024. Accessed: 2024-08-12.

[14] Airspy, "Airspy software defined radio." https://airspy.com/download/, 2024. Accessed: 2024-08-12.