Arduino based short range texting communicator (Walkie Textie)

EE 396
Design Laboratory

Ipsita Jain : 210102039

Srishti Murarka : 210108049

Under the kind guidance of

Prof. Manish Bhatt

Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati

Abstract

This project introduces the Walkie-Textie, a short-range, localized communication device that works like a pager. It uses an Arduino Mega microcontroller as the central processing unit, allowing message exchange using a radio transceiver module. The messages are displayed on an OLED screen, and user input is captured via a 4x3 keypad matrix. The Walkie-Textie utilizes an nRF24L01 radio transceiver to ensure messages are received reliably, even in areas with limited signal strength. The design of the Walkie-Textie follows the principles of digital communication and computer networking, allowing users to exchange text messages within a predefined range quickly and efficiently. The user interface is designed with an OLED display and a keypad matrix to make message creation and reception as intuitive and easy as possible. The Walkie-Textie demonstrates the effectiveness of Arduino-based systems in meeting specific communication requirements. It emphasizes dependability, efficiency, and simplicity in messaging applications within proximity, making it ideal for emergency response coordination, remote site monitoring, and group collaboration scenarios.

Contents

1 Introduction

2 Concepts Used

2.1 Serial Peripheral Interface (SPI) Communication Protocol . . .
2.1.1 MOSI and MISO Signals . . .
2.1.2 Chip Select Signal . . .
2.1.3 Clock Signal . . .
2.2 Gaussian Frequency Shift Keying (GFSK) Modulation . . .
2.3 Open System Interconnection (OSI) Model . . .
2.3.1 OSI Link Layer . . .
2.4 Security . . .

3 Components Used

4 Brief Description of Tools

4.1 Hardware . . .
4.1.1 Arduino Mega 2560 R3 . . .
4.1.2 nRF24L01+ Module . . .
4.2 Software . . .
4.2.1 Utilities: Arduino IDE . . .
4.2.2 Language: C++ . . .

5 Procedure

5.1 Alphanumeric Keyboard . . .
5.2 Integrating OLED Display . . .
5.3 Adding Transceiver - Comparison between LoRa and nRF24L01 . . .
5.3.1 Initial Transmission using LoRa Module . . .
5.3.2 Employing nRF24L01 Module . . .
5.4 Additional Features: Cryptography . . .
5.4.1 Encryption . . .
5.4.2 Decryption . . .

6 Working

6.1 Flow Diagram . . .
6.2 Circuit Schematic . . .

7 Arduino Code

7.1 Code with respect to user 1 . . .
7.2 Code to test nRF24L01 . . .

8 Results

9 Discussions

9.1 Applications . . . . .

9.2 Cost Analysis . . . . .

9.3 Limitations . . . . .

9.4 Future Scope . . . . .

10 Conclusion

1 Introduction

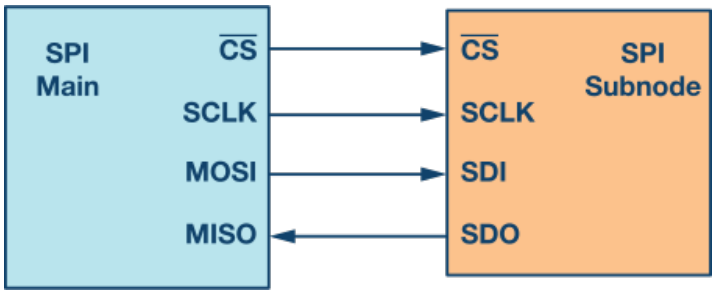
Walkie-Textie introduces a novel communication solution similar to traditional pagers, leveraging Arduino Mega, nRF24L01 radio transceiver module, OLED display, and a 4x3 keypad. In an era of ubiquitous connectivity, this device offers a localized communication alternative ideal for scenarios where conventional mobile networks prove impractical or inaccessible. This project embodies the convergence of hardware and software technologies to create a versatile near-field messaging device capable of facilitating swift and efficient communication within limited geographic areas. By harnessing the power of Arduino Mega as the central processing unit, coupled with the nRF24L01 module for wireless communication, Walkie-Textie empowers users to exchange text messages seamlessly within a short-range environment. The project incorporates an OLED display and a keypad to improve user interaction, making message composition and reception more intuitive. This innovative project combines traditional communication methods with modern principles, demonstrating how Arduino-based systems can be used to meet specific communication needs. By exploring hardware design, software implementation, and user experience considerations, this project aims to provide a detailed guide for enthusiasts and practitioners who wish to develop similar communication solutions for various practical applications.

2 Concepts Used

2.1 Serial Peripheral Interface (SPI) Communication Protocol

The Serial Peripheral Interface (SPI) is a communication interface that facilitates data transfer between multiple devices. Typically, these devices are arranged in a master-slave configuration, where the master device controls the slaves, and the slaves receive instructions from the master. In most implementations, a single device acts as the master, while all other devices in the configuration act as slaves. SPI is a synchronous, full-duplex protocol that allows for simultaneous transmission and reception of information at high data transfer rates. The communication is based on a main-subnode configuration and is synchronized on the rising or falling clock edge. It is intended for board-level communication over short distances. The SPI interface can be either 3-wire or 4-wire. The signals required to implement a 4-wire SPI device are:

1. MOSI: Main out, subnode in
2. MISO: Main in, subnode out
3. SCLK: Serial Clock
4. CS: Chip Select



2.1.1 MOSI and MISO Signals

The communication lines utilized for SPI (Serial Peripheral Interface) communication are referred to as MOSI and MISO signals. MOSI, also known as Master Out Slave In, is responsible for transmitting data from the main device to all subnodes. Conversely, MISO, or Master In Slave Out, facilitates the transmission of data from one of the subnodes to the main device. Given that SPI operates as a full-duplex interface, both the main device and subnodes have the capability to transmit data concurrently through the MOSI and MISO lines.

2.1.2 Chip Select Signal

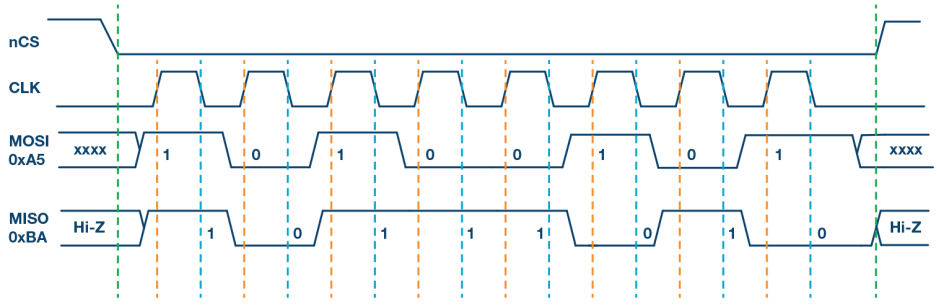
The main utilizes the chip select pin ($\overline{CS}$) to select which subnode to communicate with. This signal is usually active low. If multiple subnodes are used, each subnode will require its dedicated chip to select a signal from the main while sharing the clock and data lines. When the main wants to communicate with a specific subnode, the chip select line for that subnode should be pulled low. After the main has finished communicating with the subnode, the chip select line is pulled back high.

2.1.3 Clock Signal

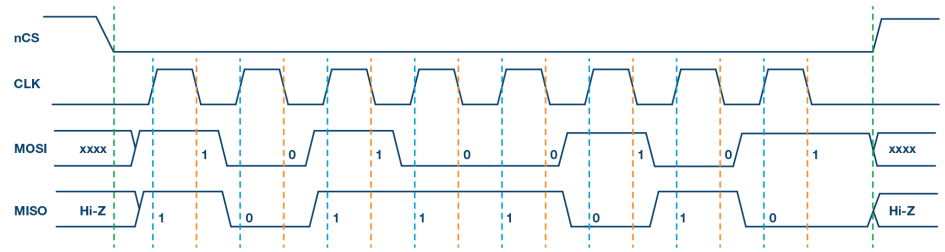
The clock signal serves as a synchronization mechanism for coordinating data transfer between devices. It is generated by the main device at a specific frequency and can be configured using two properties: clock polarity (CPOL) and clock phase (CPHA). The CPOL bit dictates the polarity of the clock signal, allowing it to idle either low (0) or high (1). A clock signal idling low exhibits a high pulse with a rising leading edge, while a clock signal idling high presents a low pulse with a falling leading edge. On the other hand, the CPHA bit determines the clock phase, specifying whether the rising or falling clock edge is utilized to sample and shift the data. The main device must select the appropriate clock polarity and clock phase based on the requirements of the subnode. Depending on the chosen CPOL and CPHA configurations, four distinct SPI modes become available.

SPI Mode	CPOL	CPHA	Clock Polarity in Idle State
0	0	0	Logic low
1	0	1	Logic low
2	1	0	Logic high
3	1	1	Logic high

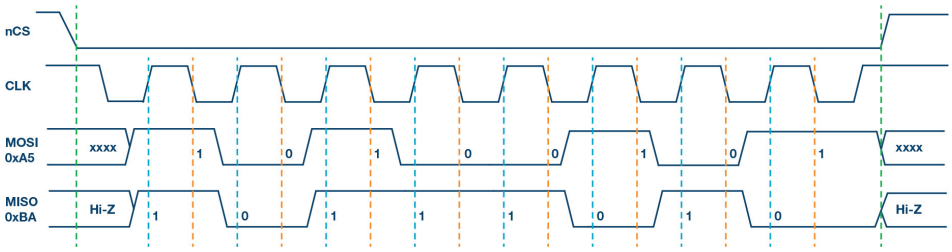
The timing diagram of communication in the four SPI modes is as follows:



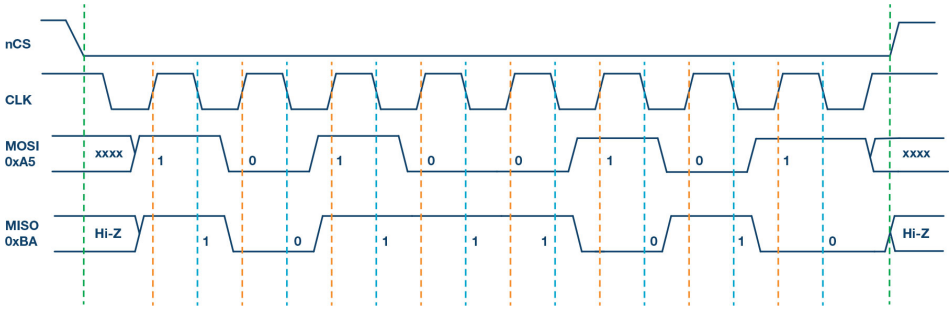
SPI Mode 0: data sampled on the rising edge and shifted on the falling edge



SPI Mode 1: data sampled on the falling edge and shifted on the rising edge



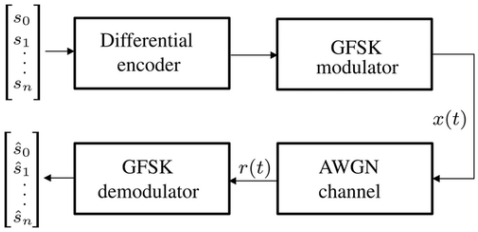
SPI Mode 2: data sampled on the falling edge and shifted on the rising edge



SPI Mode 3: data sampled on the rising edge and shifted on the falling edge

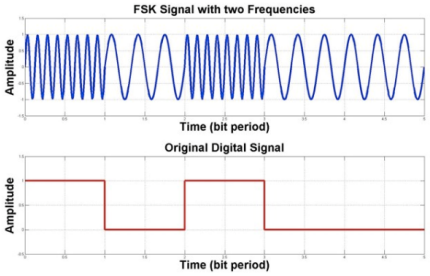
2.2 Gaussian Frequency Shift Keying (GFSK) Modulation

Gaussian Frequency Shift Keying (GFSK) serves as a modulation method employed in wireless communication setups for encoding digital data. It operates by smoothly altering the frequency of the carrier signal across different values. Prior to modulation, GFSK utilizes a Gaussian filter to mold the baseband waveform signal, effectively diminishing the sudden frequency shifts commonly observed in conventional Frequency Shift Keying (FSK). Renowned for its resilience against interference and its capability to limit emissions to a narrow spectral band, GFSK exhibits a high resistance to noise, thereby bolstering the dependability of wireless devices.

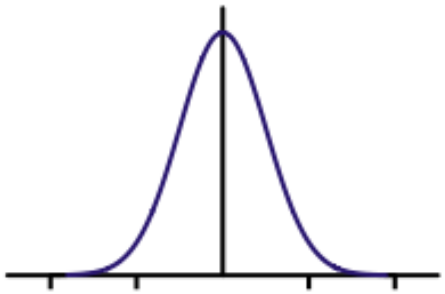


GFSK represents a modified iteration of the FSK modulation method. Unlike traditional FSK, where the modulated signal's frequency experiences abrupt changes at the onset of each symbol period of binary data, GFSK ensures a smoother transition between bit 0 and bit 1, or vice versa. This

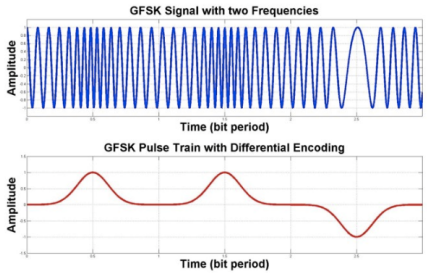
results in relatively fewer amplitude and phase variations within the modulated signal. In conventional FSK, distinct local oscillators are employed for the in-phase and quadrature components. These oscillators are switched at the commencement of each symbol period to generate the carrier frequency for modulation. However, since not all independent oscillators maintain the same amplitude and phase at the start of each symbol period, this leads to sudden and abrupt frequency changes with every bit transition in the transmitted signal. Consequently, the modulated GFSK signal exhibits considerable extension and non-negligible side lobes, as depicted in the accompanying figure.



When a signal is received by the receiver, various factors such as interference, thermal regeneration within the receiver, and other distortions caused by the wireless channel can lead to performance degradation in FSK. This degradation makes it challenging to accurately decode the signal, particularly with respect to precise amplitude and phase values after each symbol period. In the GFSK modulation scheme, a Gaussian filter is incorporated before the baseband waveform signal. This implies that the data signal, illustrated in Figure 1, undergoes modulation by the FSK modulator circuit subsequent to passing through the Gaussian filter. The integration of this Gaussian filter prior to modulation distinguishes GFSK from FSK. Typically, a Gaussian filter exhibits an impulse response characterized by a Gaussian function, as depicted in the accompanying figure.



The frequency response of a filter employing a Gaussian function in the time domain retains its Gaussian shape in the frequency domain, leading to a highly narrow frequency response. Filtering an input signal using this method results in a reduced spectral width compared to the FSK scheme, which lacks such filtering. By eliminating abrupt frequency changes that can occur in FSK, this filtering process ensures smoother transitions at the onset of each symbol period. The response of the GFSK signal is visually depicted in the figure below.



By filtering out unwanted interference from adjacent channels, this approach reduces sideband power and out-of-band spectrum interference. However, narrowing the spectral width of the FSK signal causes time domain dispersion, leading to intersymbol interference (ISI). To minimize the effects of ISI, it's essential to design a Gaussian filter with optimal cut-off frequencies. Employing other signal processing techniques and channel equalization methods can also help mitigate ISI-related issues.

The pulse shaping stage, which occurs before signal modulation, involves filtering the data pulses to produce a clean output signal with sharp rise and fall times. This enables accurate determination of the carrier frequency of the received signal.

The GFSK technique is widely used in various wireless systems and technologies, such as the Improved Layer 2 protocol, Bluetooth, IEEE 802.15.4, and Z-wave.

2.3 Open System Interconnection (OSI) Model

In 1984, the ISO created the OSI model as a reference framework outlining how computer data is transmitted. This model comprises seven layers that work together to carry out specialized network functions, enabling a more structured approach to networking. The OSI serves as a standard that allows various computer systems to communicate seamlessly with one another. It is like a universal language for computer networking, dividing a communication system into seven abstract layers that build upon each other. Here are the seven abstraction layers of the OSI model, listed from top to bottom:

1. Application Layer
2. Presentation Layer
3. Session Layer
4. Transport Layer
5. Network Layer
6. Data Link Layer
7. Physical Layer

2.3.1 OSI Link Layer

The OSI network architecture model places the data link layer as the second layer from the bottom. Its primary function is to deliver data between nodes, ensuring error-free transmission. Additionally, it manages encoding, decoding, and organizing incoming and outgoing data. This layer is considered the most intricate of the OSI model as it conceals all the underlying hardware complexities from the layers above. The data link layer is subdivided into two sub-layers, as follows:

1. **Logical Link Control (LLC)**
   This sublayer within the data link layer manages multiplexing, facilitating the data flow between applications and various services. Logical Link Control (LLC) is also critical in delivering error notifications and acknowledgments.

2. **Media Access Control (MAC)**
   The Media Access Control (MAC) sublayer is tasked with overseeing the interaction of devices. It plays a crucial role in addressing frames and governs access to physical media.

2.4 Security

Security is a crucial consideration in the domain of circuits, particularly in the field of electronics, communication and computer systems. Circuits are the building blocks of modern technology, powering a wide range of devices and systems, including computers, smartphones, IoT devices, and more. The security of these circuits is paramount to protect against potential threats and vulnerabilities that could compromise their functionality, integrity, and confidentiality. One of the key aspects of circuit security is ensuring the confidentiality of information. Circuits may process sensitive data, such as personal information, financial transactions, or intellectual property. Robust security measures, such as encryption, authentication, and access controls, are implemented to prevent unauthorized access and protect the confidentiality of this data. Authentication mechanisms are also crucial for verifying the identities of communicating parties. Techniques like digital signatures and cryptographic certificates help ensure that data is exchanged only between trusted entities. Integrity is another critical aspect of communication circuit security. Any unauthorized alterations or tampering with communication channels can lead to data corruption, message forgery, or manipulation of transmitted information. To combat vulnerabilities like man-in-the-middle attacks, secure communication protocols, and intrusion detection systems are vital for maintaining secure, trustworthy channels. In conclusion, robust security measures are critical for protecting communication circuits, and ensuring the secure exchange of information in our interconnected world.

3 Components Used

Following components were used in making of this project.

1. Arduino Mega 2560 R3

Microcontroller	ATmega2560
Clock Speed	16MHz
SRAM	8KB
Operating Voltage	5V
Input Voltage	7 - 12 V
Output Voltage	5V, 3.3V

2. NRF24L01 Module

Operating Frequency Band	2.4GHz
Communication Protocol	SPI
Operating Voltage	3.3V
Distance Range	50-200 Feet

3. OLED Display

Controlling Chip	SSH1106
Resolution	128 x 64 Pixels
Interface Type	IIC
Driving Voltage	3.3-5V

4. 4x3 Matrix Membrane Keypad
5. Breadboard
6. Jumper Wires
7. Connecting Wires

4 Brief Description of Tools

4.1 Hardware

4.1.1 Arduino Mega 2560 R3

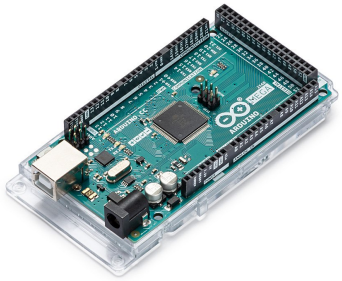
The Arduino Mega 2560 R3 is a powerful and versatile microcontroller board built around the ATmega2560 chip, an 8-bit AVR architecture microcontroller running at 16 MHz. This board has 256 KB of flash memory for program storage, 8 KB of SRAM for temporary data storage, and 4 KB of EEPROM for non-volatile data. It offers 54 digital I/O pins, with 15 supporting PWM output for controlling the brightness of LEDs, speed of motors, and other applications requiring variable output levels. Additionally, the board features 16 analog input pins with a 10-bit resolution analog-to-digital converter (ADC), allowing it to read analog voltages ranging from 0 to 5 volts.

The Arduino Mega 2560 R3 supports various protocols including UART, SPI, and I2C for communication. It has 4 hardware UARTs for serial communication, one SPI interface for high-speed communication with SPI-compatible devices, and one I2C interface for connecting to I2C-compatible devices like sensors and displays. Apart from the 256 KB flash memory, it has 8 KB of SRAM and 4 KB of EEPROM. The board can be powered via USB from a computer or an external USB power adapter, or through an external DC power supply with a voltage range of 7V to 12V. It features a reset button for restarting the program execution and a built-in LED on pin 13 for primary status indication.

The Arduino Mega 2560 R3 is programmed using the Arduino programming language based on C++. This makes it accessible to beginners and experienced programmers, as C++ is a widely used and versatile programming language. Regarding compatibility, the Arduino Mega 2560 R3 works seamlessly with the Arduino IDE, making it easy to write, upload, and debug code written in C++. It also supports most Arduino shields, extending its capabilities by adding features such as Wi-Fi, Ethernet, motor control, and more. Additional features include a barrel jack for external power input, multiple ground and power pins for easy connectivity and an ICSP header for in-circuit serial programming. Overall, the Arduino Mega 2560 R3 is a robust and feature-rich board suitable for many projects, from simple LED blinking experiments to complex robotics and automation projects. Its extensive I/O capabilities, large memory, compatibility with various sensors and modules, and programming in C++ make it a popular choice among hobbyists, educators, and professionals.

4.1.1.1 Development Board

The development board of Arduino Mega 2560 R3 is a versatile platform tailored for those requiring to expand their projects' capabilities, especially compared to the standard Arduino boards. Here are some specifics about the Arduino Mega 2560 R3 development board:



1. **Form Factor:** The Arduino Mega 2560 R3 development board is more significant than standard Arduino boards, with dimensions of approximately 10.3 cm x 5.5 cm. Its larger size accommodates more pins and features, making it suitable for more complex projects and prototyping on breadboards or standalone setups.

2. **Microcontroller:** At the heart of the Arduino Mega 2560 R3 is the ATmega2560 microcontroller, an 8-bit AVR architecture chip running at 16 MHz. This chip provides the board with 256 KB of flash memory for program storage, 8 KB of SRAM for temporary data storage, and 4 KB of EEPROM for non-volatile data.

3. **Power Supply:** The board can be powered via USB from a computer or a USB power adapter, or through an external DC power supply with a voltage range of 7V to 12V. The board includes a voltage regulator with a stable 5V output to power the microcontroller and other components.

4. **USB-to-Serial Converter:**The Arduino Mega 2560 R3 features a built-in USB-to-serial converter, facilitating programming and debugging of the board using the USB port. This allows for easy uploading of sketches (programs) and monitoring of serial output during development.

5. **Digital and Analog I/O Pins:** The board offers 54 digital I/O pins, with 15 supporting PWM output. Additionally, there are 16 analog input pins with a 10-bit ADC for reading analog signals. These pins enable interfacing options with sensors, actuators, and other electronic components.

6. **Communication Interfaces:** For communication, the board supports UART, SPI, and I2C protocols, making it versatile for connecting to a variety of devices such as GPS modules, Bluetooth modules, and other microcontrollers.

7. **LEDs and Buttons:** The Arduino Mega 2560 R3 includes a built-in LED connected to pin 13 for primary status indication and a reset button for restarting the program execution. These features aid in debugging and testing during development.

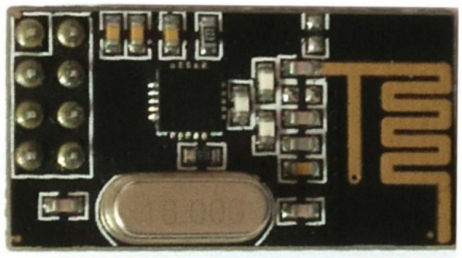
8. **Breadboard-Friendly:** Despite its larger size, the Arduino Mega 2560 R3 is designed with pin headers that allow it to be used on a standard breadboard for prototyping, offering flexibility in circuit design and component connections.

9. **Compatibility and Community Support:** The board is fully compatible with the Arduino IDE, which is based on C++. This makes it accessible to both beginners and experienced

programmers. Additionally, the Arduino Mega 2560 R3 benefits from a vast community of users, providing extensive support, documentation, tutorials, and libraries to aid in project development.

4.1.2 nRF24L01+ Module

Operating within the 2.4 GHz worldwide ISM frequency band, the nRF24L01+ module employs GFSK modulation for data transmission. This module provides users with the flexibility to configure data transfer rates at 250kbps, 1Mbps, or 2Mbps. The 2.4 GHz band falls under the Industrial, Scientific, and Medical (ISM) bands, widely utilized by numerous unlicensed low-power devices including cordless phones, Bluetooth devices, NFC devices, and WiFi networks.



In terms of power, the module's operating voltage ranges from 1.9V to 3.9V. However, it is crucial to note that powering the module with 5V can damage it. While the module operates within the 1.9V to 3.6V range, its logic pins are 5V tolerant, eliminating the need for a logic-level translator. The module's output power can be programmed to 0 dBm, -6 dBm, -12 dBm, or -18 dBm. At 0 dBm, the module consumes only 12 mA during transmission, comparable to the consumption of a single LED. It consumes merely 26 µA in standby mode and 900 nA in power-down mode, making it ideal for low-power applications.

The nRF24L01+ uses a 4-pin SPI (Serial Peripheral Interface) with a maximum data rate of 10Mbps. This SPI interface allows for configuring various parameters like frequency channel, output power, and data rate. In most setups, the Arduino acts as the master, while the nRF24L01+ serves as the slave. It is worth noting that, unlike the I2C bus, SPI supports a limited number of slaves; typically, one can connect up to two nRF24L01+ modules to a single Arduino.

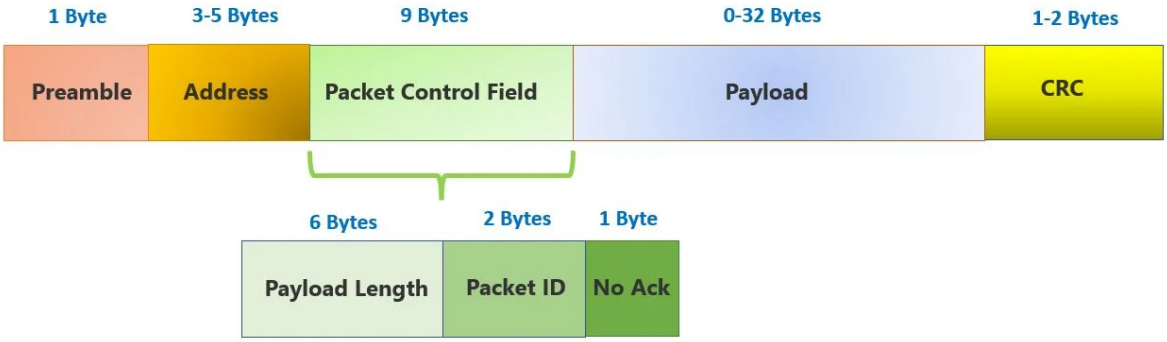
Here are the technical specifications of the nRF24L01+:

- **Frequency Range:** 2.4 GHz ISM Band
- **Maximum Air Data Rate:** 2 Mb/s
- **Modulation Format:** GFSK
- **Max. Output Power:** 0 dBm
- **Operating Supply Voltage:** 1.9 V to 3.6 V
- **Max. Operating Current:** 13.5mA
- **Min. Current (Standby Mode):** 26µA
- **Logic Inputs:** 5V Tolerant
- **Communication Range:** 800+ m (line of sight)

In summary, the nRF24L01+ module offers a robust and efficient wireless communication solution, particularly suitable for low-power IoT applications, with a decent communication range and versatile SPI interface for configuration and data exchange.

4.1.2.1 Enhanced ShockBurst (ESB) Protocol

The nRF24L01+ utilizes Enhanced ShockBurst as its packet structure. This new and improved version includes the Packet Control Field (PCF) and the previous Preamble, Address, Payload, and Cyclic Redundancy Check (CRC) fields. This addition allows for more advanced communication capabilities. Enhanced ShockBurst enables variable length payloads using a payload length specifier, allowing for payloads between 1 and 32 bytes.

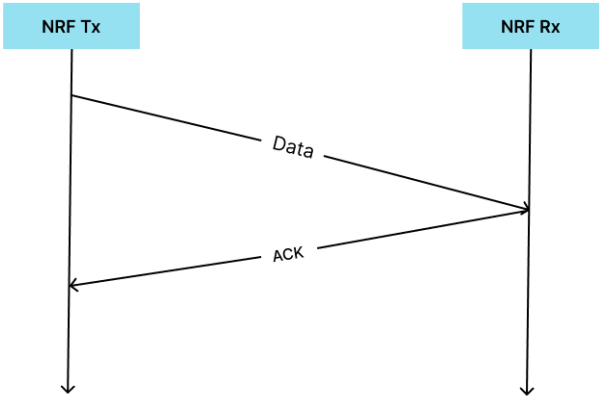


Additionally, each sent packet is assigned a unique ID to help the receiver determine whether the message is new or retransmitted. Finally, each message contains a field that requests an acknowledgment from the receiver.

4.1.2.2 nRF24L01+ Automatic Packet Handling

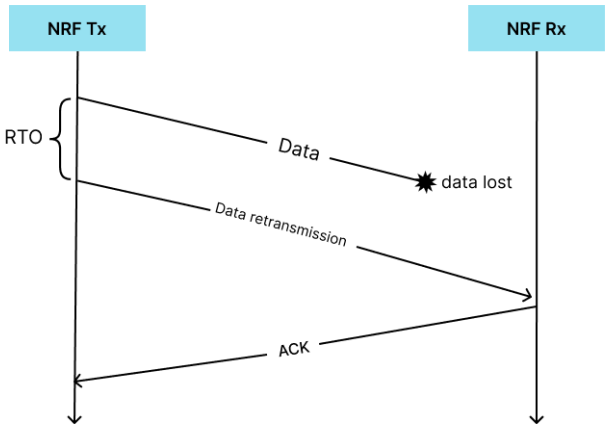
There are three scenarios to how two nRF24L01+ modules interact with one another:

- **Transaction with an acknowledgment:**



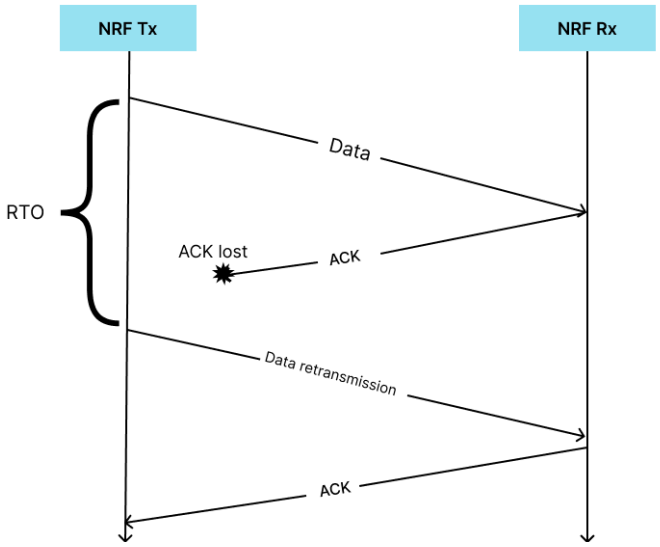
In an optimistic scenario, the communication begins when a data packet is sent by the transmitter to the receiver. After that, the transmitter waits for around 130 µs to receive an acknowledgment (ACK). Once the packet is successfully received, the receiver promptly sends the ACK. Finally, the transaction concludes upon receipt of the ACK.

- **Transaction with a lost data packet:**



In a pessimistic scenario that can be characterized by packet loss, the transmitting party patiently waits for an acknowledgment (ACK) following the packet's transmission. If the ACK is not received within the pre-set auto-retransmit-delay (ARD) time, the packet is promptly retransmitted. Upon receipt of the retransmitted packet, the receiving party sends the ACK, thus bringing the transaction to a successful conclusion.

- **Transaction with a lost acknowledgment:**



In adverse situations, retransmission is necessary due to the loss of the ACK. The transmitter might assume that the packet was lost because it didn't receive the ACK, even though the receiver successfully received it on the first attempt. As a result, the transmitter waits for the Auto-Retransmit-Delay timeout and then retransmits the packet. When the receiver receives a packet with the same ID as the previous one, it discards it and sends the ACK again. Once the transmitter receives the ACK, the transaction is complete.

The nRF24L01+ chip handles the entire packet-transaction process without the involvement of the microcontroller.

4.2 Software

4.2.1 Utilities: Arduino IDE

The Arduino Integrated Development Environment (IDE) serves as a dedicated software platform tailored for electronics applications, enabling developers to write, compile, and upload code onto Arduino boards.

Built upon the C and C++ programming languages, the Arduino IDE boasts a collection of libraries facilitating seamless interaction with sensors, actuators, and various electronic components. Its user-friendly graphical interface streamlines the programming process for Arduino boards, catering to both hobbyists and professionals.

A notable strength of the Arduino IDE lies in its broad support for an array of Arduino boards, encompassing popular models like the Arduino Uno, Arduino Mega, and Arduino Nano. This versatility empowers developers to select the board most suited to their project requirements.

Furthermore, the IDE features a serial monitor facilitating communication with the Arduino board and displaying program output, thereby simplifying troubleshooting and debugging tasks.

4.2.2 Language: C++

C++ stands out as a high-level programming language extensively utilized in electronics to craft software applications that manage and engage with electronic devices. Renowned for its robust features encompassing object-oriented programming, templates, and generic programming, C++ excels in constructing intricate software systems.

Embedded systems heavily rely on C++ for crafting device drivers, firmware, and real-time systems, leveraging its prowess in delivering the requisite performance and reliability. Additionally, C++ finds widespread adoption in domains like robotics, automation, and control systems, where its capabilities meet the demands of these applications.

A notable advantage of employing C++ in electronics lies in its provision of low-level hardware access, enabling developers to interact with hardware components at a granular level compared to other programming languages. This capability facilitates the development of applications capable of directly interfacing with sensors, actuators, and various electronic components.

Furthermore, C++ boasts an extensive array of libraries and tools tailored for electronics development, exemplified by the Arduino IDE which is founded on C++. This integration streamlines the process of prototyping and developing electronics projects, empowering developers to swiftly iterate and innovate without delving into the intricacies of low-level code.

5 Procedure

5.1 Alphanumeric Keyboard

To enable input for the texting communicator, a 4x3 Matrix membrane keypad is utilized. The keypad matrix consists of 4 rows and 3 columns, providing a total of 12 keys.

1	2	3
4	5	6
7	8	9
*	0	#

These keys are mapped to alphanumeric characters as follows:

Key	**Alphanumeric Mapping**
1	a, b, c, 1
2	d, e, f, 2
3	g, h, i, 3
4	j, k, l, 4
5	m, n, o, 5
6	p, q, r, 6
7	s, t, u, 7
8	v, w, x, 8
9	y, z, 0, 9
*	Backspace
0	( ), (.), (,), (?), (!)
#	Send

This setup allows users to input alphanumeric characters for composing messages. The Arduino code detects the number of key presses (taps) and updates the message accordingly. However, this mapping of keys requires a huge memory space which is not supported by either Arduino Nano or Arduino Uno, with an SRAM of 2KB only. Therefore, Arduino Mega 2560 R3, with an SRAM of 8KB has been used.

5.2 Integrating OLED Display

To provide visual feedback to the user, an I2C OLED (Organic Light Emitting Diode) display is integrated into the system. The display is driven using the U8g2 library with the SH1106 controller. This library provides easy-to-use functions for displaying text and graphics on OLED displays.

The OLED display is connected to the Arduino Mega using the hardware I2C interface. This requires connecting the SDA (data) and SCL (clock) pins of the OLED display to the corresponding pins on the Arduino Mega. The display shows both the outgoing and incoming messages, enhancing user interaction and feedback.

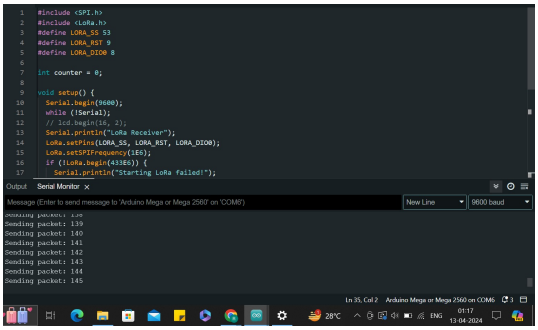
5.3 Adding Transceiver - Comparison between LoRa and nRF24L01

5.3.1 Initial Transmission using LoRa Module

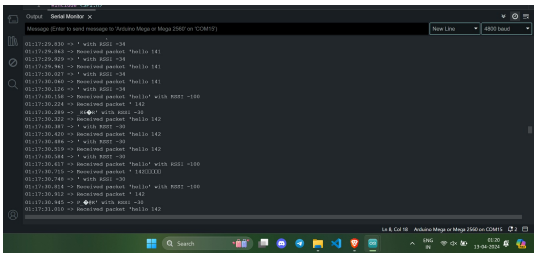
LoRa is a wireless modulation technique that utilizes chirp pulses, similar to the communication methods of dolphins and bats, to encode information on radio waves. This technology is derived

from Chirp Spread Spectrum (CSS) is known for its robustness against disturbances, allowing it to be received across great distances. LoRa is particularly well-suited for transmitting small amounts of data with low bit rates. Compared to other technologies such as WiFi, Bluetooth, or ZigBee, its more extended transmission range makes it an ideal choice for sensors and actuators operating in low power mode. LoRa can be operated on license-free sub-gigahertz bands, including 915 MHz, 868 MHz, and 433 MHz. It can also be used on 2.4 GHz to achieve higher data rates, although this comes at the cost of range. These frequencies fall under the internationally reserved ISM bands for industrial, scientific, and medical purposes.

LoRa was first chosen as the transceiver module due to its above mentioned features. Following is the simulation of the example code of LoRa:



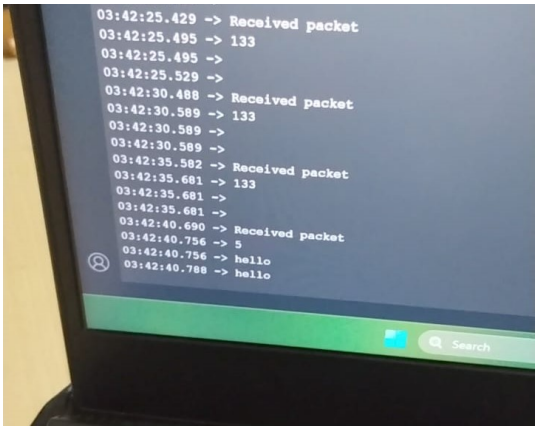
(a) Example LoRa transmitter



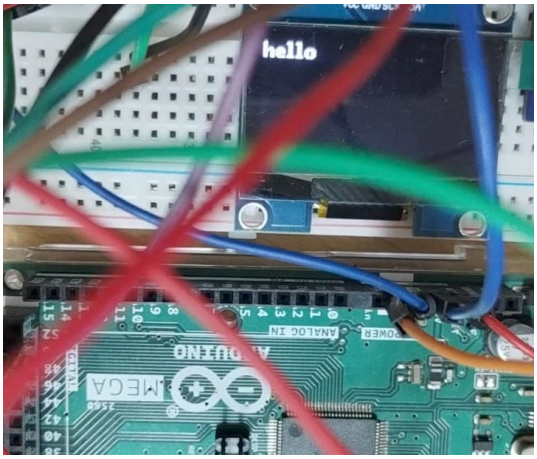
(b) Example LoRa receiver

Figure 1: Example LoRa Transmitter and Receiver Pair transmitting "hello" every 5 seconds

The following figure shows how the Receiver received a packet after missing many packets:



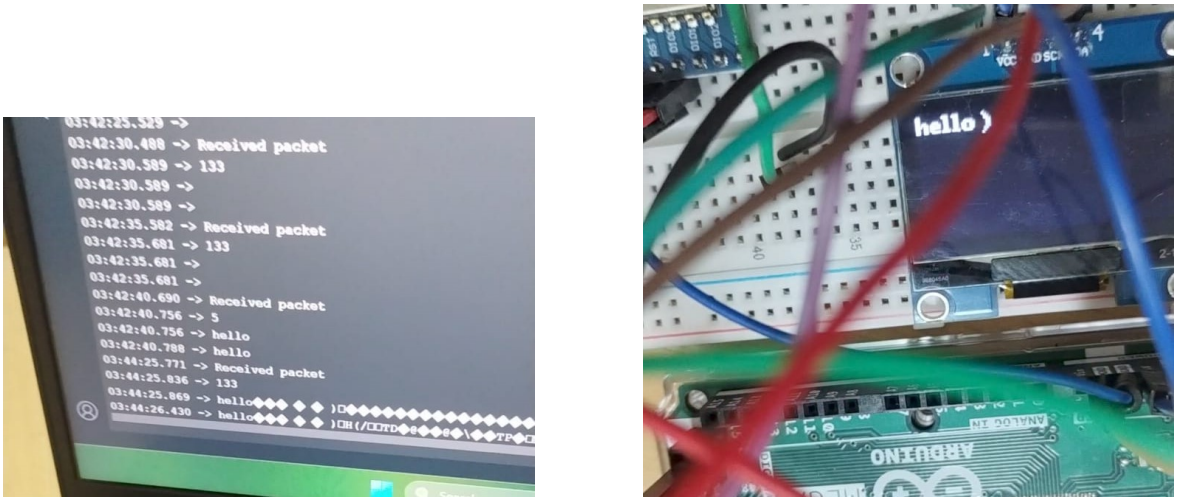
(a) LoRa receiver status



(b) Receiver Display

Figure 2: LoRa Receiver and the display showing the "hello" text

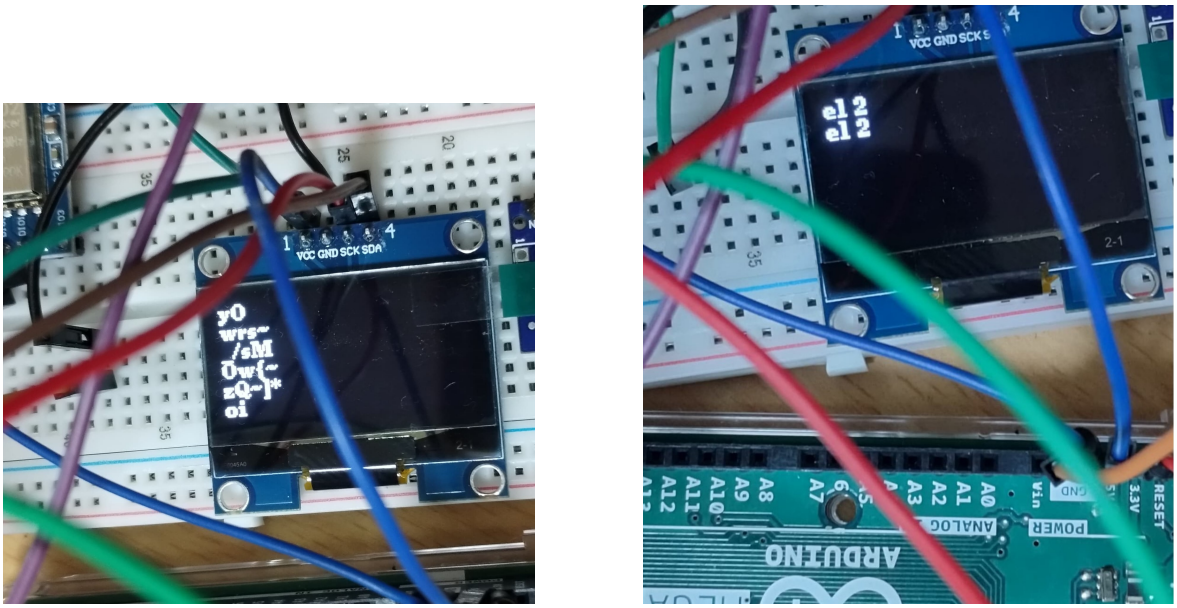
From the figures 3 (a) & (b), we can observe noisy packets that are affecting our data:



(a) LoRa receiver status **(b)** Receiver Display

Figure 3: LoRa Receiver and the display showing the "hello" text with noise

From the figures 4 (a) & (b), we can observe only noise being transmitted:



(a) LoRa receiver status **(b)** Receiver Display

Figure 4: Display showing the random noise

Due to the noise affecting our data, LoRa was dropped as the transceiver module for this project

5.3.2 Employing nRF24L01 Module

For communication between devices, an NRF24L01 radio transceiver module is employed. The NRF24L01 module operates on the 2.4 GHz frequency band and provides a range of up to 100 meters in ideal conditions.

The module is connected to the Arduino Mega using the SPI (Serial Peripheral Interface) interface. The following pin configuration is used:

NRF24L01 Radio Transceiver	Arduino Mega
MOSI	51
MISO	50
SCK	52
CE	9
CSN	8
Vcc	5V
GND	GND

The Arduino communicates with the NRF24L01 module using the RF24 library, enabling wireless transmission and reception of messages. The module is configured with appropriate addresses for transmitting and receiving messages between devices. Utilising ESB protocol for transmission, noiseless transmission of messages is noticed. Therefore, nRF24L01 is chosen over LoRa as radio transceiver module. A sample simulation can be seen in figures 5 (a) & (b) in the Result Section.

5.4 Additional Features: Cryptography

To ensure the security of transmitted messages, encryption and decryption techniques are implemented using Symmetric Cipher Technique.

5.4.1 Encryption

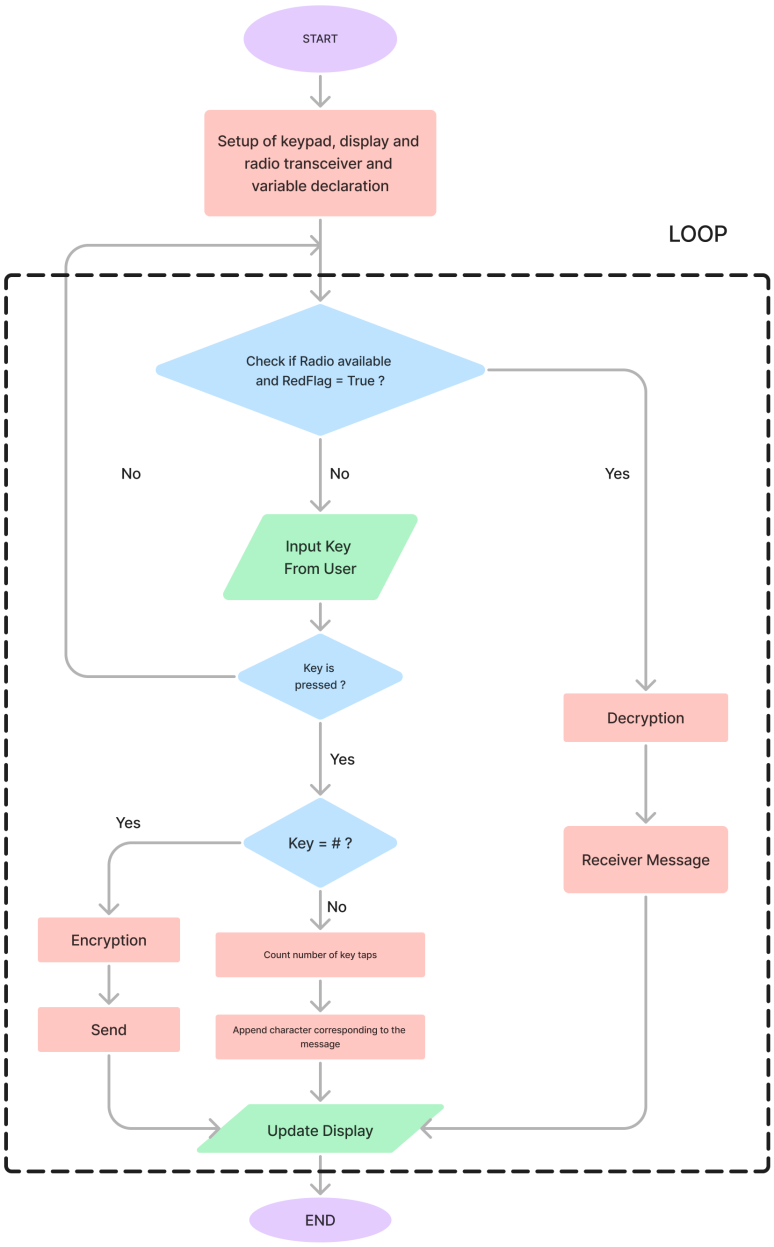
Before sending a message, it is encrypted using a randomly generated key. The message is converted into an array of characters, and each character is added or subtracted alternatively with a random key. This process involves arithmetic operations between each character of the message and the key. The encrypted message, along with the key, is then transmitted wirelessly using the NRF24L01 radio transceiver module.

5.4.2 Decryption

Upon receiving a message, the encrypted data is decrypted using the same key that was used for encryption, which is retrieved from the message itself. The Arduino then performs arithmetic operations between each character of the encrypted message and the key to recover the original message. This process reverses the encryption process, restoring the original message. The decrypted message is then displayed on the OLED display for the user to view.

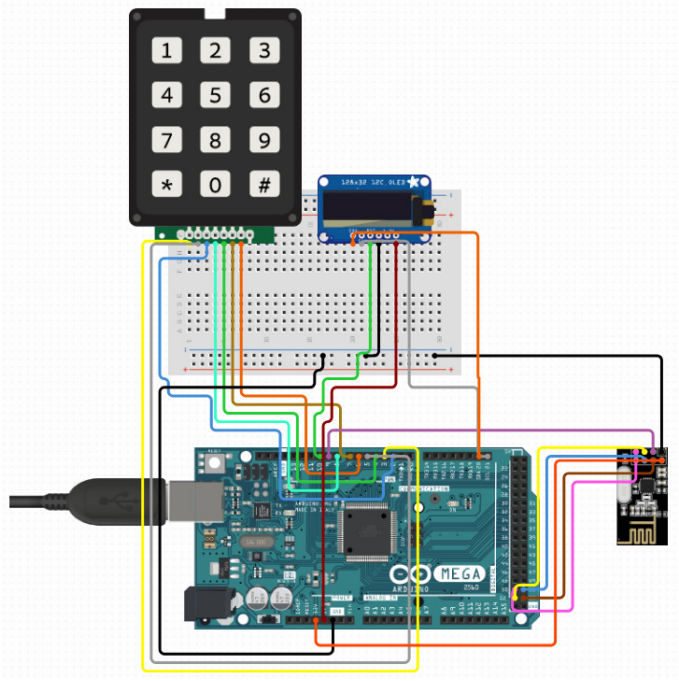
6 Working

6.1 Flow Diagram



According to the above flow diagram, the radio transceiver first checks if it has received a message from the other user. If a message is received, it is decrypted using the cipher key embedded in the message and the original message is displayed on the OLED display. If no message is received, the user can type their own message using the alphanumeric keyboard. When a key is pressed, the number of taps corresponding to it is counted, and accordingly, the associated character is appended to the message. When the send key (#) is pressed, the message so formed is first encrypted using a randomly generated key and then sent to the other user via wireless transmission. The display is updated with a confirmation message: "Message Sent".

6.2 Circuit Schematic



The above circuit diagram illustrates the complete connection diagram. The detailed connections of each component is given below.

Keypad	Arduino Mega
Row 1	A8
Row 2	A9
Row 3	A10
Row 4	A11
Column 1	A12
Column 2	A13
Column 3	A14

I2C OLED Display	Arduino Mega
3.3V	3.3V
GND	GND
SDA	SDA
SCK	SCL

NRF24L01 Radio Transceiver	Arduino Mega
MOSI	51
MISO	50
SCK	52
CE	9
CSN	8
Vcc	5V
GND	GND

7 Arduino Code

7.1 Code with respect to user 1

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Arduino.h>
#include <U8g2lib.h>
#ifdef U8X8_HAVE_HW_SPI
#endif
#ifdef U8X8_HAVE_HW_I2C
#include <Wire.h>
#endif
#include <Keypad.h>

//====================KEYPAD====================
const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'},
};

byte rowPins[ROWS] = {A8,A9,A10,A11};
byte colPins[COLS] = {A12,A13,A14};

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

char prevkey = NO_KEY;
long long int count =0;
int tap=0;
char mp[9][4]={
  {'a','b','c','1'},
  {'d','e','f','2'},
  {'g','h','i','3'},
  {'j','k','l','4'},
  {'m','n','o','5'},
  {'p','q','r','6'},
  {'s','t','u','7'},
  {'v','w','x','8'},
  {'y','z','0','9'}
  };
```

```
42
43  char mp1[5]={'-','.',',','?','!'};
44  //=====================KEYPAD=====================
45
46  //=====================DISPLAY====================
47  U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/
        U8X8_PIN_NONE);
48  String messageToSend = "S:-";
49  String messageReceived = "";
50
51  void updateDisplay(){
52
53    u8g2.clearBuffer();
54    for (int i=0; i<=messageReceived.length(); i=i+15) {
55      u8g2.drawStr(0, (int)((i*2)/3) +10,
            messageReceived.substring(i,min(i+15,messageReceived.length())).c_str());
56    }
57    for (int i=0; i<=messageToSend.length(); i=i+15) {
58      u8g2.drawStr(0, (int)((i*2)/3) +30,
            messageToSend.substring(i,min(i+15,messageToSend.length())).c_str());
59    }
60    u8g2.sendBuffer();
61  }
62  //=====================DISPLAY====================
63
64  //=====================RF_TRANSCEIVER=====================
65  RF24 radio(9, 8); // CE, CSN
66  const byte srishtiAddr[6] = "00420";
67  const byte ipsitaAddr[6] = "00069";
68  bool recFlag = true;
69
70  void sendMsg()
71  {
72    Serial.println(F("Sending..."));
73    radio.stopListening();
74    bool rslt;
75    byte key = random(10);
76    const char replyData[messageToSend.length()];
77    char encryptedData[messageToSend.length()+1];
78    messageToSend.toCharArray(replyData, messageToSend.length()+1);
79    for (int i=0; i<messageToSend.length(); i++){
80      if (i&1) encryptedData[i]=replyData[i]+char(key);
81      else encryptedData[i]=replyData[i]-char(key);
82    }
83    encryptedData[messageToSend.length()]=char(key);
```

```
84    Serial.println(replyData);
85    Serial.println(encryptedData);
86    rslt = radio.write( &encryptedData, sizeof(encryptedData) );
87    radio.startListening();
88
89    if (rslt) {
90      Serial.println(F("Acknowledge received"));
91      messageToSend = "S: Message Sent";
92      updateDisplay();
93      delay(2000);
94      messageToSend = "S:";
95      updateDisplay();
96    }
97    else {
98      Serial.println(F("Tx failed"));
99    }
100
101 }
102
103 void receiveMsg()
104 {
105   char dataReceived[21] = "";
106   radio.read( &dataReceived, sizeof(dataReceived) );
107   messageReceived = String(dataReceived);
108   Serial.println("Encrypted Message:");
109   Serial.println(messageReceived);
110   const char encryptedMessage[messageReceived.length()];
111   messageReceived.toCharArray(encryptedMessage,
          messageReceived.length());
112   char key = encryptedMessage[sizeof(encryptedMessage)-1];
113   messageReceived="";
114   for (int i=0; i<sizeof(encryptedMessage)-1; i++){
115     if (i&1) messageReceived+=encryptedMessage[i]-key;
116     else messageReceived+=encryptedMessage[i]+key;
117   }
118   Serial.println("Message Received:");
119   Serial.println(messageReceived);
120   updateDisplay();
121   if (messageReceived!="") recFlag = false;
122 }
123 //====================RF_TRANSCEIVER====================
124
125 //======================SETUP=======================
126
127 void setup()
```

```
128 {
129   //----------SerialSetup----------
130   Serial.begin(9600);
131   Serial.println("::Srishti's-Window::");
132   //----------SerialSetup----------
133
134   //----------RFSetup----------
135   radio.begin();
136   radio.setDataRate( RF24_250KBPS );
137   radio.openWritingPipe(ipsitaAddr); // NB these are swapped compared to
        the master
138   radio.openReadingPipe(1, srishtiAddr);
139   radio.setRetries(3,5); // delay, count
140   radio.startListening();
141   //----------RFSetup----------
142
143   //----------DisplaySetup----------
144   u8g2.begin();
145   u8g2.setFont(u8g2_font_ncenB08_tr);
146   u8g2.clearBuffer();
147   u8g2.drawStr(0, 30, "S:-");
148   u8g2.sendBuffer();
149   //----------DisplaySetup----------
150 }
151 //====================SETUP====================
152
153
154 //====================LOOP====================
155 void loop()
156 {
157    if ( radio.available() && recFlag) receiveMsg();
158   recFlag = true;
159
160   char key = keypad.getKey();
161   if (key!=NO_KEY)
162     Serial.println(key);
163
164   if(count>40000)
165   {
166     count =0;
167     if(prevkey!=NO_KEY)
168     {
169       if (messageToSend.length()<=20)
170         {
171           if (prevkey>= '1' && prevkey <= '9')
```

```
            messageToSend += mp[prevkey-'1'][(tap-1)%(4)];
            else if (prevkey == '0')
            messageToSend += mp1[(tap-1)%(5)];
            else if (prevkey == '*'){
              for(int i = 0;i<tap;i++) if(messageToSend.length()>3)
                  messageToSend.remove(messageToSend.length()-1);
            }
          }

        // Update display
        updateDisplay();
      }

      prevkey = NO_KEY;
      tap=1;
    }
    // If a key is pressed
    if (key != NO_KEY){
      if (key == '#') {
        sendMsg();
      }

      count=0;
      // If it's a digit or * or #
      if (key == '1' || key == '2' || key == '3' || key == '4' || key ==
          '5' || key == '6' || key == '7' || key == '8' || key == '9' ||
          key == '0' || key == '*') {
        // Add the key to the messageToSend
          if(key==prevkey)
          {
            tap++;
          }
          else
          {
            tap = 1;
          }
            prevkey =key;
        }

      }
    count++;
  }
  //====================LOOP====================
```

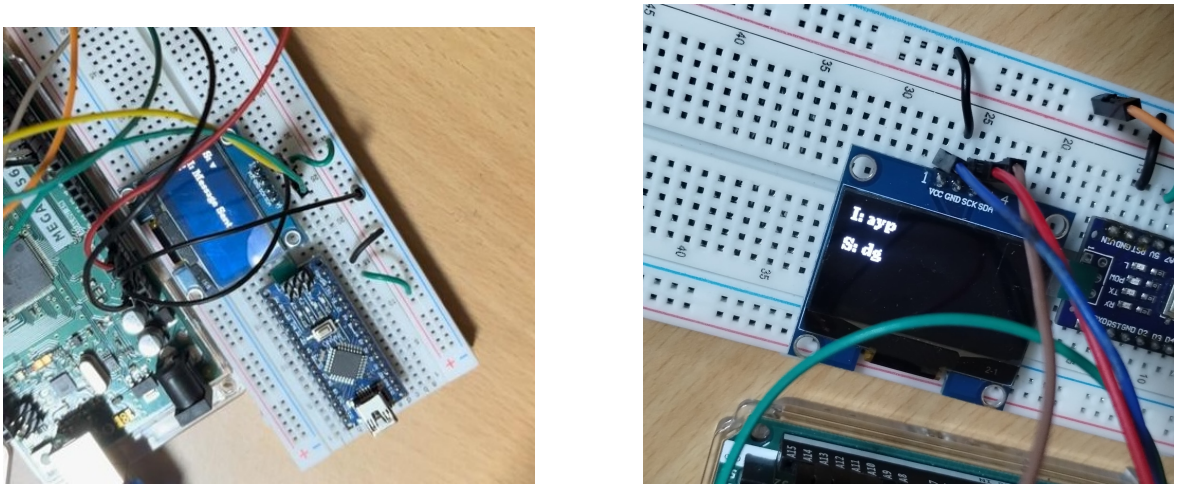
For user 2, the user addresses are swapped in the setup of reading and writing pipes.

7.2 Code to test nRF24L01

```
/*
  If your serial output has these values same then Your nrf24l01 module
     is in working condition :

  EN_AA          = 0x3f
  EN_RXADDR      = 0x02
  RF_CH          = 0x4c
  RF_SETUP       = 0x03
  CONFIG         = 0x0f

  This code is under public domain

  Last updated on 21/08/28
  https://dhirajkushwaha.com/elekkrypt
 */

#include <SPI.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(9, 8);

byte addresses[][6] = {"1Node", "2Node"};

void setup() {
  radio.begin();
  radio.setPALevel(RF24_PA_LOW);

  radio.openWritingPipe(addresses[0]);
  radio.openReadingPipe(1, addresses[1]);
  radio.startListening();

  Serial.begin(9600);
  printf_begin();

  radio.printDetails();

}
void loop() {
//  empty
}
```

8 Results

The figures 5 (a) & (b) show the successful wireless transmission of a message from one user to the other.



(a) Sender Display

(b) Receiver Display

Figure 5: Transfer of a message from one user to other

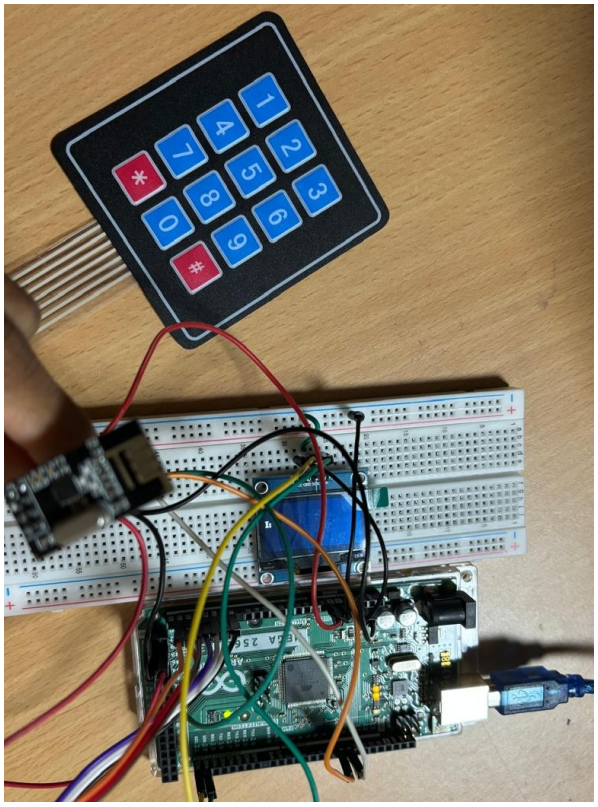


Figure 6: Full Circuit View

9 Discussions

9.1 Applications

1. **Home Automation System:** This device can be integrated into a home automation system that enables users to effortlessly manage their entire household ecosystem. The system comprises various electronic devices in a household serving as subnode devices. A remote control, acting as the main device, receives the user's input to change the status of any of the subnodes. The subnodes communicate their status back to the main device, providing a dynamic and responsive household environment. This system can be enhanced by adding a servo motor to operate mechanical switches. The system delivers a customized environment of comfort and convenience, enhancing daily life.

2. **Restaurant Ordering System:** This device streamlines the restaurant ordering process, empowering both management and customers. The main device is operated by the restaurant administrator, while subnodes are allocated to customers. Customers use an intuitive keypad interface to input their orders, which are promptly acknowledged and confirmed by the administrator. The system generates electronic receipts for transparency and efficiency. This system optimizes order processing and customer service, exemplifying the use of technology in enhancing service industry efficiency.

3. **Defence Communication:** This device is a reliable tool for short-range defense communication that requires noiseless transmissions of sensitive information. It is engineered with advanced encryption capabilities to safeguard every communication against unauthorized access and external threats. The exclusivity of its encryption protocol ensures that messages can only be decrypted by the intended recipient's device, minimizing the risk of data infiltration or interception.

Additionally, Walkie-Textie serves as a versatile communication tool, perfectly suited for environments with weak or no cellular signals, such as during outdoor activities, crowded events, and large industrial sites. Beyond offering a cost-effective messaging solution, it also plays a crucial role in supporting search and rescue operations and enhancing security patrolling measures.

9.2 Cost Analysis

Sr. No.	Component	Quantity	Price per unit (in Rs.)
1	nRF24L01 Module	2	57
2	I2C OLED Display	2	359
3	Jumper Wire pack	1	100
4	Breadboard	2	56
5	4x3 Keypad	2	56
6	Arduino Mega 2560	2	3299
Total:			7754

9.3 Limitations

1. **Limited Range:** This device is a short-range communication device suitable for outdoor activities and small-scale operations. Its effectiveness is limited over longer distances.
2. **Memory Capacity:** It has limited memory capacity, which restricts its ability to store extensive chat history. This limitation may result in the loss of vital information.
3. **Fragile Connections:** Connections within the Walkie Textie are delicate and may become unreliable due to wear and tear over time. It is crucial to handle the device carefully and perform regular maintenance to prevent connection failures and ensure prolonged usability.

9.4 Future Scope

1. **Enhanced Range:** To extend the transmission range of radio communication, a relay system consisting of multiple radio transceivers can be employed. This system works by receiving signals from the original radio and then retransmitting them, effectively increasing the range of communication. By strategically placing the relay stations along the transmission path, the signal can be relayed over longer distances and obstacles.
2. **Authentication:** To enhance security, various authentication mechanisms can be implemented, such as passwords, RFID (Radio Frequency Identification) verification, or biometric verification. These mechanisms can be used alone or in combination with one another to create a multi-factor authentication system, to protect against unauthorized access to the device.

These future scopes have the potential to significantly enhance the Walkie-Textie project, increasing its security and efficiency.

10 Conclusion

The Walkie-Textie is a localized communication device. It offers a versatile solution for exchanging text messages over short distances. The Walkie-Textie is simple, reliable, and adaptable and can be useful in various domains such as outdoor activities, events, industrial settings, and educational environments. However, it has certain limitations, such as range constraints, memory capacity, and fragile connections.

These limitations can be addressed through continuous improvement efforts and innovation. In the future, the Walkie-Textie has promising opportunities for enhancing range, security, integration with emerging technologies, and market diversification. As technology continues to evolve, the Walkie-Textie can prove to be a valuable tool for facilitating communication and collaboration in diverse scenarios. It contributes to greater efficiency, connectivity, and convenience in everyday life.

References

[1] https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf.

[2] https://www.everythingrf.com/community/what-is-gfsk-modulation

[3] https://www.elprocus.com/nrf24l01/