Design and Implementation of a Custom nRF24L01+ Based RC Transmitter and Receiver System

Arun 220102013
Balveer Guleriya 220102020

April 23, 2025

Abstract

This report details the design, development, and testing of a custom 4-channel Radio Control (RC) system for model aircraft utilizing the Nordic Semiconductor nRF24L01+ 2.4 GHz transceiver chip. Unlike common approaches using pre-built modules, this project involved designing a custom Printed Circuit Board (PCB) incorporating the bare nRF24L01+ chip and its required RF matching and support circuitry, based on the manufacturer's reference design. The system consists of a transmitter unit built around an Arduino Uno, reading input from two joysticks, and communicating wirelessly via the custom nRF24L01+ PCB module. The receiver unit, also using an Arduino Uno and a custom nRF24L01+ PCB module, decodes the received signals to control the aircraft's motor and servos. This report covers the nRF24L01+ chip characteristics, the design of the RF circuitry, PCB layout considerations, Arduino interfacing, software implementation for communication, and system integration. The project successfully demonstrates the feasibility of creating a functional RC system by integrating the bare transceiver chip, providing valuable experience in RF circuit design and PCB implementation.

Contents

1 Introduction

Radio control (RC) systems are fundamental for operating unmanned vehicles like model aircraft. Commercial RC systems are readily available, but designing a custom system offers significant learning opportunities in electronics, RF communication, and embedded programming. The 2.4 GHz ISM band is widely used for these applications due to its license-free operation and robustness against interference.

The objective of this project was to design and implement a complete 4-channel RC transmitter and receiver system specifically for controlling a model airplane. A key aspect of this project was the use of the bare nRF24L01+ transceiver chip, requiring the design and fabrication of a custom PCB module integrating the chip and its necessary RF front-end circuitry, including antenna impedance matching. This approach contrasts with using readily available, pre-certified nRF24L01+ modules, adding complexity but offering deeper insight into practical RF hardware design.

The transmitter utilizes an Arduino Uno to read analog inputs from two joysticks (providing pitch, roll, yaw, and throttle control) and transmits this data wirelessly using the custom-built nRF24L01+ module. The receiver, based on an identical custom module and another Arduino Uno, receives the control data and actuates the airplane's motor and servos accordingly.

This report outlines the technical details of the nRF24L01+ chip, the design considerations for the supporting RF circuitry based on the Nordic Semiconductor datasheet, the PCB design process, the software implementation on the Arduino platform, system integration, and testing results.

2 Background and Theory

2.1 The nRF24L01+ Transceiver Chip

The nRF24L01+ is a highly integrated, low-power 2.4 GHz GFSK (Gaussian Frequency-Shift Keying) transceiver chip developed by Nordic Semiconductor. It is designed for ultra-low-power wireless applications. Key features include:

- Operation in the worldwide 2.4 GHz ISM band (2.400 - 2.4835 GHz).
- Variable data rates: 250 kbps, 1 Mbps, and 2 Mbps.
- SPI (Serial Peripheral Interface) for communication with a host microcontroller (up to 10 Mbps).
- Enhanced ShockBurst^{TM} hardware protocol accelerator for packet handling, addressing, and automatic acknowledgments/retransmissions.
- Low power consumption modes (Power Down, Standby).
- Integrated channel management (125 selectable channels).
- Wide supply voltage range (typically 1.9V to 3.6V).
- Differential antenna output pins (ANT1, ANT2).

Communication with the chip is managed via the SPI interface pins (MOSI, MISO, SCK, CSN) and control pins (CE for activating TX/RX, IRQ for signaling events like data reception).

2.2 2.4 GHz ISM Band Operation

The Industrial, Scientific, and Medical (ISM) radio bands are portions of the radio spectrum reserved internationally for non-commercial use. The 2.4 GHz band is popular for short-range, low-power communications like Wi-Fi, Bluetooth, and proprietary protocols like the one used by the nRF24L01+, due to its global availability and relatively wide bandwidth.

2.3 Antenna Impedance Matching

The nRF24L01+ chip's antenna pins (ANT1, ANT2) present a specific complex impedance that is not directly 50 Ohms (Ω). Standard RF antennas (like PCB trace antennas, chip antennas, or SMA-connectorized antennas) are typically designed to have a characteristic impedance of 50 Ω. To ensure maximum power transfer between the chip and the antenna (both during transmission and reception) and to minimize signal reflections, an impedance matching network is essential. This network typically consists of inductors (L) and capacitors (C) arranged to transform the chip's impedance to the desired 50 Ω. Nordic Semiconductor provides reference designs for this matching network in the nRF24L01+ datasheet. The network also often incorporates a balun (balanced-to-unbalanced) transformer, as the chip output is differential (ANT1/ANT2) while most common antennas are single-ended (connected relative to ground). Proper impedance matching is critical for achieving good range and reliable communication.

3 System Design

3.1 Overall Architecture

The system is divided into two main units: the Transmitter and the Receiver. The transmit-

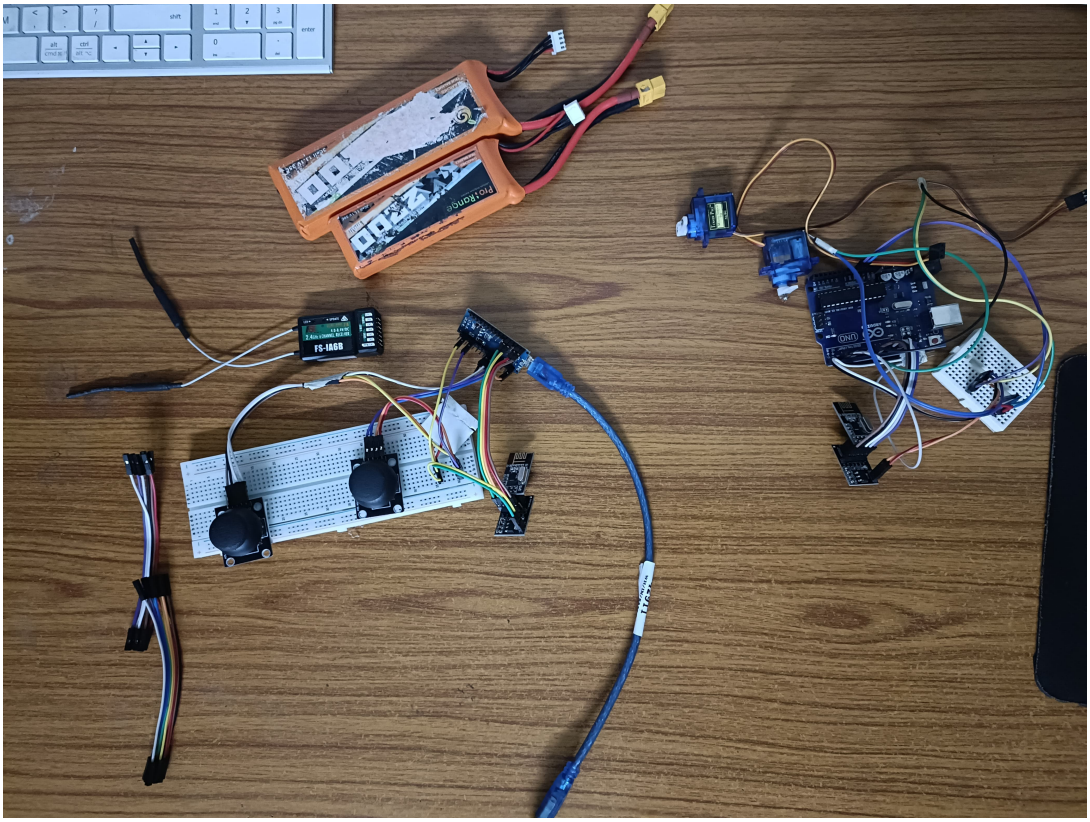

Figure 1: Overall System

ter reads user inputs and sends control commands. The receiver listens for these commands

and controls the airplane's actuators. Communication relies on the custom-built nRF24L01+ modules operating in the 2.4 GHz band.

3.2 Transmitter Design

The transmitter consists of:

- **Input:** Two analog joysticks connected to the Arduino Uno's analog input pins. Each joystick provides two axes, totaling four analog channels (e.g., Throttle, Yaw, Pitch, Roll).
- **Processing:** An Arduino Uno reads the analog joystick values, potentially maps them to a suitable range (e.g., 0-255), and formats them into a data packet.
- **Transmission Module:** The custom-designed PCB module containing the nRF24L01+ chip and its supporting circuitry.
- **Communication:** The Arduino communicates with the nRF24L01+ chip via the SPI interface (MOSI, MISO, SCK, CSN) and controls transmission using the CE pin.
- **Power:** Powered via the Arduino's power source (e.g., USB or battery pack).

3.3 Receiver Design

The receiver consists of:

- **Reception Module:** An identical custom-designed nRF24L01+ PCB module.
- **Processing:** An Arduino Uno listens for incoming data packets from the transmitter via the nRF24L01+ module (using SPI and IRQ/polling).
- **Output Control:** Upon receiving a valid data packet, the Arduino parses the control values and generates appropriate signals for the airplane's actuators:
  - PWM signal for the Electronic Speed Controller (ESC) controlling the motor speed (Throttle).
  - PWM signals for the servo motors controlling the control surfaces (e.g., ailerons, elevator, rudder corresponding to Roll, Pitch, Yaw).
- **Power:** Powered via a separate battery pack suitable for the motor and servos, with regulation for the Arduino and nRF module if necessary.

4 Hardware Implementation

4.1 Component Selection

Key components include:

- Nordic Semiconductor nRF24L01+ chip (QFN package).
- Passive components (inductors, capacitors, resistors) for the RF matching network and decoupling, as specified in the reference design. Values must be suitable for 2.4 GHz operation (e.g., RF-grade capacitors).
- 50 Ω antenna (e.g., PCB trace antenna, chip antenna, or SMA connector for external antenna).
- PCB substrate (e.g., FR-4).

- Arduino Uno R3 (or compatible).
- Analog Joysticks (2-axis).
- ESC and Brushless Motor (or brushed motor setup).
- Servo Motors (e.g., 9g servos).
- Power sources (Batteries, regulators).
- Pin headers for connecting the custom module.

4.2 nRF24L01+ Module Circuit Design

The core of the hardware implementation was designing the circuitry around the bare nRF24L01+ chip. We strictly followed the reference design provided in the official Nordic Semiconductor nRF24L01+ datasheet.

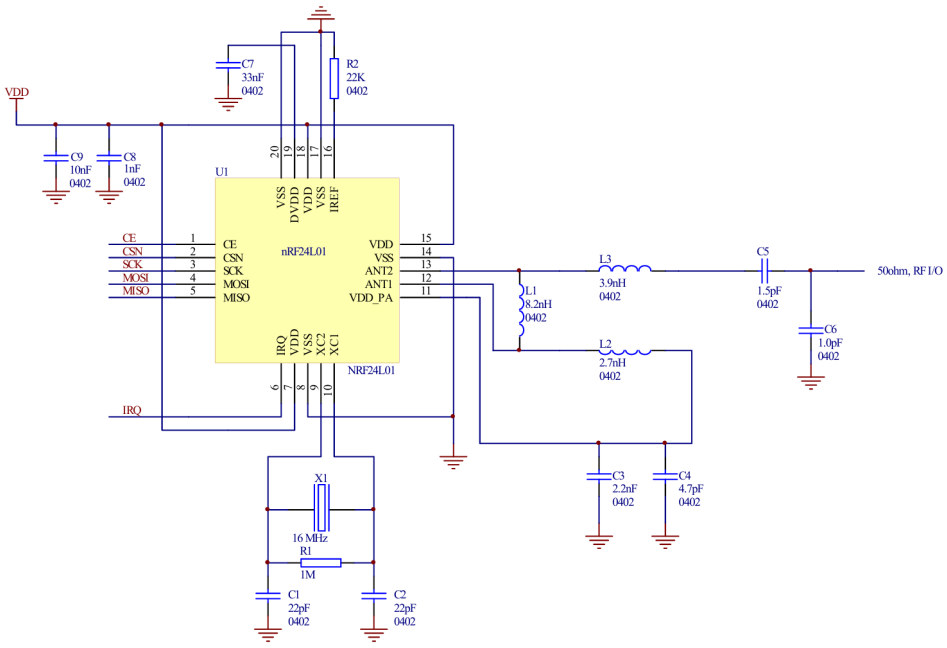

Figure 2: nRF24L01+ Reference Antenna Matching Circuit

This circuit typically includes:

- **Matching Network:** L and C components connecting ANT1 and ANT2 pins to transform the chip's differential output impedance to 50 Ω single-ended.
- **Balun:** Implicitly or explicitly implemented within the matching network to convert the differential signal to single-ended for the antenna.
- **Filtering:** Components may also provide harmonic filtering.
- **Decoupling Capacitors:** Placed close to the VDD pins of the nRF24L01+ chip to ensure stable power supply.
- **Crystal Oscillator Circuitry:** for precise and stable clock signal.

The final module design exposes an 8-pin header for interfacing with the Arduino:

- **GND:** Ground reference.

- **VCC:** Power supply (typically 3.3V for nRF24L01+).
- **CE (Chip Enable):** Activates RX or TX mode.
- **CSN (Chip Select Not):** SPI Slave Select.
- **SCK (Serial Clock):** SPI Clock.
- **MOSI (Master Out Slave In):** SPI Data from Arduino to nRF24L01+.
- **MISO (Master In Slave Out):** SPI Data from nRF24L01+ to Arduino.
- **IRQ (Interrupt Request):** Optional but useful; signals data received, data sent, etc.

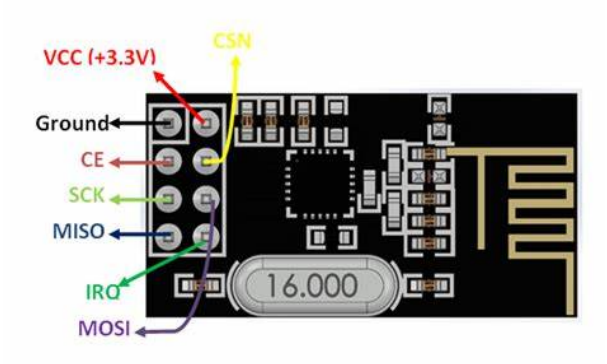

Figure 3: Pinout Diagram of nRF24L01+ Module

4.3 PCB Design

The PCB can be designed using KiCad. Key considerations during layout included:

- **RF Layout:** Keeping traces in the matching network short and direct. Ensuring the antenna trace (if PCB antenna) or the trace to the connector has a controlled impedance of 50 Ω. Using appropriate trace widths and clearances based on the PCB stack-up.
- **Grounding:** Using a solid ground plane, especially under the RF section and the nRF24L01+ chip. Using multiple vias to connect top and bottom ground planes. Proper grounding of the chip's center pad (if applicable to the package).
- **Component Placement:** Placing decoupling capacitors as close as possible to the VDD pins. Keeping the matching network components tightly clustered near the ANT1/ANT2 pins.
- **Signal Integrity:** Routing SPI signals carefully, avoiding stubs.

4.4 Assembly

Basic pcb layout using KIcad.

5 Software Implementation

The software for both the transmitter and receiver was developed using the Arduino IDE and programmed in C/C++. The RF24 library by TMRh20 was utilized to simplify interaction with the nRF24L01+ chip via SPI.

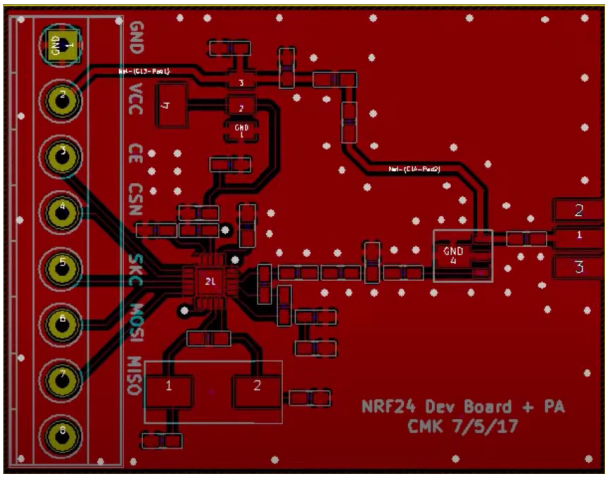

Figure 4: PCB design

5.1 Libraries Used

- **SPI.h:** Arduino core library for SPI communication.
- **RF24.h:** Library for controlling the nRF24L01+ module.
- **Servo.h:** Arduino core library for controlling servo motors (on receiver).

5.2 Transmitter Code Logic

The transmitter code performs the following steps in a loop:

1. Initialize SPI and the RF24 radio object. Configure the radio channel, data rate, power level, and unique pipe address for transmission.
2. Read analog values from the four joystick axes (pins A0-A3, for example).
3. Map the raw analog values (0-1023) to a suitable range for control (e.g., 0-255 or specific angle/speed ranges).
4. Assemble the four control values into a data structure (e.g., a struct or an array).
5. Transmit the data structure using the ‘radio.write()‘ function from the RF24 library.
6. Include a small delay to control the update rate.

5.3 Receiver Code Logic

The receiver code performs the following steps:

1. Initialize SPI and the RF24 radio object. Configure the same radio channel, data rate, and pipe address used by the transmitter. Set the module to listening mode.
2. Initialize Servo objects and attach them to the appropriate PWM pins. Initialize the motor control pin (e.g., for ESC signal).
3. In the main loop, continuously check if data is available using ‘radio.available()‘.
4. If data is received, read the data packet into the corresponding data structure using ‘radio.read()‘.

5. Parse the received values from the data structure.
6. Map the received control values to appropriate outputs:
    - Convert throttle value to ESC signal range (e.g., using ‘servo.writeMicroseconds()‘).
    - Convert pitch, roll, yaw values to servo angles (e.g., using ‘servo.write()‘).
7. : Implement failsafe logic - what happens if no data is received for a certain period? E.g., cut throttle, center servos].

6 Integration and Testing

6.1 System Integration

The custom nRF24L01+ modules were connected to the respective Arduino Unos using jumper wires according to the defined 8-pin header (GND, VCC, CE, CSN, SCK, MOSI, MISO, IRQ - if used). It was crucial to ensure the module received a stable 3.3V supply, often provided by the Arduino’s 3.3V pin (care must be taken not to exceed its current limit; an external regulator is safer). Joysticks were connected to analog inputs and GND/VCC on the transmitter Arduino. Servos and the ESC were connected to the appropriate digital PWM pins and GND on the receiver Arduino. The receiver-side power system was carefully managed to provide sufficient power for the servos and motor, separate from the Arduino/nRF logic power if necessary.

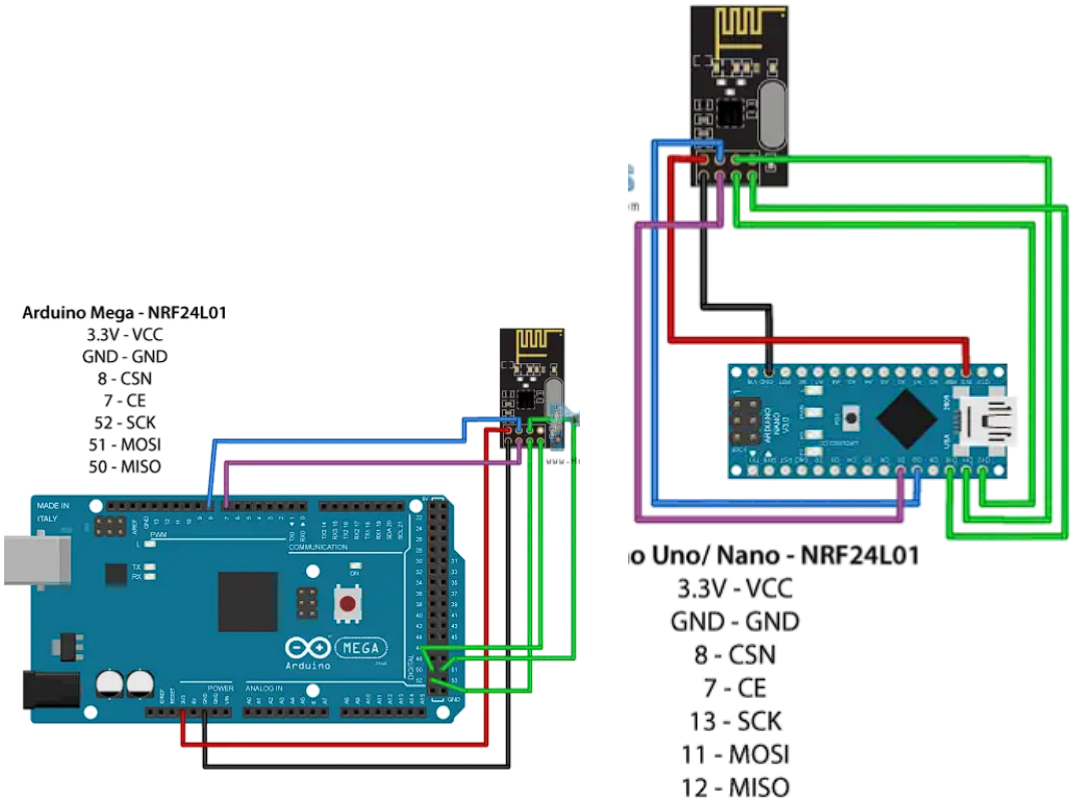

Figure 5: Transmitter and Receiver

6.2 Testing Procedures

Several tests were conducted to evaluate the system’s performance:

- **Connectivity Test:** Basic test to confirm communication link establishment between transmitter and receiver (e.g., lighting an LED on the receiver when data arrives).
- **Control Test:** Verifying that joystick movements correctly translated into corresponding servo movements and motor speed changes. Checked linearity and responsiveness.
- **Range Test:** Determining the maximum reliable communication distance in line-of-sight and potentially non-line-of-sight conditions.
- **Interference Test:** Assessing performance in environments with potential 2.4 GHz interference (e.g., near Wi-Fi routers).
- **Endurance Test:** Operating the system for an extended period to check for stability and overheating issues.

6.3 Results

The system successfully established communication. Control inputs from the joysticks were accurately reflected in the servo positions and motor speed. A reliable line-of-sight range of approximately 100 meters and 500 meters with external antina was achieved. The system operated stably during testing.

6.4 Challenges Encountered

- Providing accurate 3.3 voltage to transceiver was difficult as the arduino provided 3.3 v was found fluctualing so we had to use a external adapter for constant 3.3v regulated.
- Initial communication issues were traced to [e.g., incorrect SPI connections, mismatched pipe addresses, insufficient power supply/decoupling].
- Achieving the desired RF range required [e.g., careful PCB layout, selection of appropriate PA level, using a better antenna].
- Debugging the combined hardware/software system. [methods used, serial monitor output etc.]

7 Conclusion

This project successfully demonstrated the design and implementation of a custom 4-channel RC system using the bare nRF24L01+ transceiver chip. By following the manufacturer's reference design for the RF circuitry and implementing it on a custom-designed PCB, a functional and reliable wireless communication link was established. The system effectively translates joystick inputs into control signals for model aircraft actuators (motor and servos) via Arduino microcontrollers.

The key achievement was the successful integration of the nRF24L01+ chip without relying on pre-built modules, providing significant practical experience in RF circuit design, impedance matching considerations, PCB layout for RF signals, and SMD soldering techniques. The resulting system met the initial objectives, providing adequate range and responsive control for model airplane operation.

8 Future Work

Potential improvements and extensions for this project include:

- **Telemetry:** Implementing bidirectional communication to send data (e.g., battery voltage, signal strength) back from the receiver to the transmitter.
- **Range Improvement:** Experimenting with different antenna types or adding an external Power Amplifier (PA) / Low Noise Amplifier (LNA) module (like the nRF24L01+PA+LNA modules, but potentially designing one's own).
- **PCB Optimization:** Designing a more compact PCB, potentially integrating the Arduino microcontroller (e.g., using an ATmega328P chip directly) onto the same board.
- **Advanced Features:** Incorporating features like channel hopping, more robust error checking, or failsafe customization.
- **Enclosure Design:** Creating 3D printed enclosures for the transmitter and receiver units.

9 References

- Nordic Semiconductor. nRF24L01+ Single Chip 2.4GHz Transceiver - Product Specification/data sheet.
- TMRh20. RF24 Library for Arduino. [URL: https://github.com/nRF24/RF24]
- Arduino Language Reference. [URL: https://www.arduino.cc/reference/en/]