Implementation Of A Remote-Controlled Model Car Using Bluetooth Low Energy

Dominic Jacobo

Department of Electrical and Computer Engineering
California State University Fullerton
800 N State College Blvd, Fullerton, CA 92831

DominicJ@csu.fullerton.edu

***Abstract* – A remote-controlled (RC) car can be designed and implemented in various ways, with design choices significantly influencing key features such as cost, weight, and power consumption. This paper presents the development of an affordable RC car using fundamental microprocessor and embedded systems concepts, cost-effective hardware components, and an accessible software development kit. The implementation was divided into two main parts: the controller and the RC car. The controller reads analog signals from its joystick peripherals via ADC pins and transmits this data over Bluetooth Low Energy (BLE) to the microcontroller (MCU) on the RC car. The RC car's MCU processes this data and utilizes PWM and GPIO pins to control motor driver modules, which, in turn, regulate the vehicle's movement. Additionally, the PWM duty cycle was employed to adjust the speed of the motors, enabling precise control of the car's motion.**

Index Terms – RC car, Bluetooth Low Energy, nRF5340 DK, ADC, PWM

I. Introduction

Controlling model cars has long been a great intermediate embedded systems project idea. This is because the project idea requires the designer to be proficient with embedded design fundamentals, forcing the designer to interface with many internal peripherals, sensors, and actuators to implement an exciting design. There were two main design motivations for this project. The first was the previous class of EGEC 558B 2022, and seeing how they could use these fundamentals and concepts to control a model car for their projects. This inspired me that I could do the same as well. The second motivation was seeing how much helpful information was online for completing this task. Both reasons were my motivation for pursuing this project idea.

This paper shows the implementation of a remote-controlled car using Bluetooth Low Energy as its wireless communication protocol. There are many online hobbyist websites and videos showing the implementation of similar RC model vehicles. However, in them, it's common to find the use of Arduino hardware and the use of the Arduino IDE. There are many advantages that come with using Arduino, like its ease of use and vast community support, but many of its disadvantages come from its simplicity [1].

For example, the Arduino IDE is very limited; it doesn't support modular or object-oriented programming, which removes lots of programming optimization. The Arduino hardware is also limited in both its scalability and real-time performance. Many of Arduino's inner works are either completely abstracted away, so you can't adjust them, or they are very limited in how much can be changed [1].

In this implementation, the project design was split into two primary parts: the remote controller design and the RC car design. The controller acted as the transmitter device, sending analog readings from two joystick peripherals, one of which was for the throttle control, and the second was for the steering control. The transmitter used an nRF5340 DK from Nordic Semiconductor as its MCU. The nRF5340 DK has an onboard 2.4 GHz antenna and supports Bluetooth Low Energy. For this reason, no external hardware was needed for wireless transmission.

The RC car will act as the receiver. So, after the Bluetooth Low Energy packet has been set over the connection, the receiver will pick it up on the other end. The receiver also uses the nRF5340 DK as its processing unit and as its Bluetooth communication device. On the RC car, the nRF5340 DK will use the data in the packet to determine how quickly the motors should spin and in what direction. This is done through the use of PWN and GPIO pins. The hardware and software setup and use will be explained in more detail in sections II and III.

II. Circuit Description

This section will discuss the hardware used, the internal peripherals used, and how the hardware was set up for both the controller and the car for this implementation.

To begin, Figure 1 below illustrates the system block diagram for the Bluetooth RC model car's remote-control system. At the core of this system is the nRF5340 DK, the primary microcontroller responsible for processing inputs and controlling the vehicle. To provide additional context, Figure 2 presents a detailed block diagram of the nRF5340 DK, showcasing its functional architecture for reference [2]. In the remote-control design, two joysticks serve as secondary input devices, each generating analog signals that are transmitted to the nRF5340 DK. The microcontroller utilizes its internal ADC peripheral to process these signals—specifically the VRx and VRy outputs—allowing it to determine the joystick's position along the x-axis and y-axis. Each joystick achieves this functionality because each can be seen as just two potentiometers whose values can be read from these output pins.

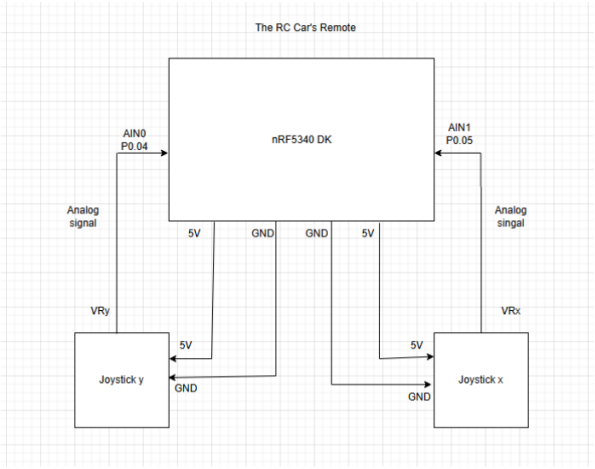

Figure 1. System Block Diagram of the RC Car's Remote

The VRy signal, representing the vertical position of the joystick, controls the car's speed in forward and backward directions. In contrast, the VRx signal, indicating the horizontal position, controls the steering for the left and right movements. The VRy signal is connected to the P0.04 pin, and the VRx signal is connected to the P0.05 pin, which corresponds to the DK's first and second ADC channels. Additionally, the DK provides the required 5 volts and ground connections for the joysticks, which serve as the reference points that the ADC will be reading.

Bluetooth Low Energy (BLE) will be used to transmit data between the devices. In this setup, the remote will act as the Bluetooth central device, scanning for advertising packets, while the car will act as the Bluetooth peripheral device, sending out these packets. Although BLE allows for two-way communication, with both devices capable of sending messages, this system is designed so that only the central transmits messages while the peripheral receives them.

Another essential feature of the remote control design is its power source. This will be discussed more in detail later in section III, but because of the way the ADC is configured, the nRF5340 DK needs to be powered by a 5-volt supply in order for the joystick output to make sense when the firmware is parsing its position.

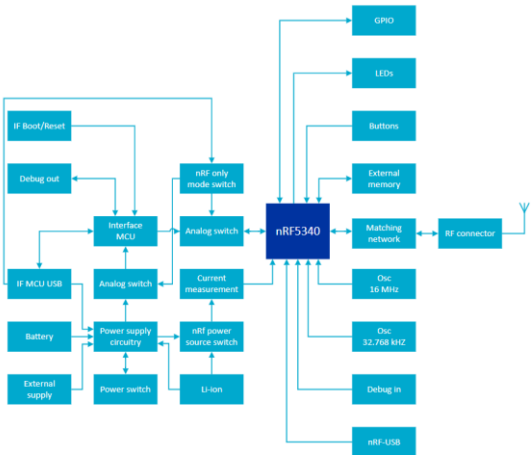

Figure 2. nRF5340 DK system block diagram [2]

Next, Figure 3 shows the system diagram for the RC car itself. The nRF5340 DK is again used as the central processing component in this system. In this system, the DK is used to control the car's motors through the use of the L298n motor driver. When the DK receives the joystick position over a Bluetooth low-energy message, it will process the data and send the correct signals to the motor drivers. In the diagram, we can see GPIO pins being used to send signals to the driver's E1, M1, E2, and M2 pins. The E1 and M1 pins are motor control pins for the left motor, and the E2 and M2 pins are for the right motor. In this implementation, the E pins are sent a PWM signal for control and output voltage to the motors, and the M pins are for controlling if the motor spins clockwise or counterclockwise. The motor drives are supplied with a 6v battery that is fed into its VS input screw terminal. This is the voltage that is provided to the motors but is fed through according to the pulse width modulation. Lastly, all the drivers and the batteries are connected to a common ground that is supplied by the DK.

The RC car design uses two separate 2S 35C Lipo batteries as power sources. Due to this, the RC car can also be said to have a split architecture. The top two motors and motor driver are connected to one of these Lipo batteries, and the other two motors and motor driver are connected to the other LiPo battery. The reason for the use of LiPo batteries is to be a precaution. In this design, four DC Gearbox Motors were used. In the specification for these motors, they can be seen to draw up to 1.5 amps when stalled at 6 volts [3]. For this reason, it was the safe option to purchase a battery for the system that can handle a relatively high burst of current if necessary. And because LiPo 2S batteries can supply a voltage of 7.4 [4], this gives the system a more extensive range of values to feed the motors.

Similarly to the nRF5340 DK used by the remote, the DK on the RC car also needs to be supplied with a constant 5 volts to output the correct PWM voltage properly. Fortunately, each of the L298Ns comes with four on-board 5-volt pins [5]. The nRF5340 DK has many different ways of powering it. One of which is through the external source pins. So the external pins on the DK are connected to one of the driver's 5v pins and the GND screw terminal of the driver.

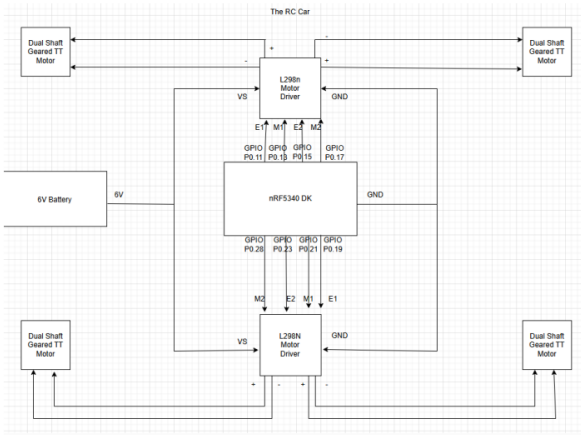

Figure 3. System Block Diagram of the RC Car

III. Software Description

This section discusses the software setup and implementation for this project. For this project, the controller and RC car each required separate firmware. This is, of course, because they perform different functions in the system, but it's also because when it comes to Bluetooth low energy, the wireless connection protocol used for this system they have a different role in the connection.

In BLE, there are two roles in performing a one-to-one connection: the central and the peripheral. In this system, the Central role is given to the remote, and the peripheral role is given to the RC car. The job of the peripheral is to perform advertising, which means that it sends out packets of data in the hopes that they will be received by a listening central device. The job of the central is to initiate the connection by first scanning for advertising packets. After the central processes the packet, it can begin the connection by sending a connection request to the peripheral. This will then establish a bi-directional connection channel. Figure 4 shows this process.

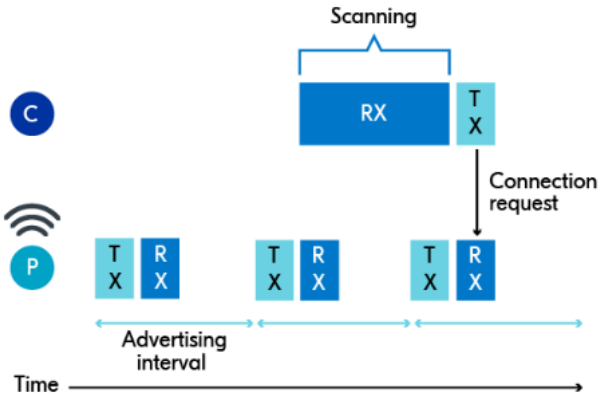

Figure 4. The processes of establishing a one-to-one Bluetooth Low Energy connection.

When the bi-directional connection has been created, both devices will 'sleep,' meaning they will turn their radios off and not communicate until the connection interval. The connection interval is the time they agree on sleeping until they wake up. When they wake up, a connection event occurs where the central sends a packet to the peripheral. During this time, the two devices can send many packets, but when they stop, they will have to wait for the next connection event to send more. Even if there is no useful data, the peers must send empty packets to sync their clocks. If they want to send more data than there is time for in one connection interval, it will be split over several connection intervals.

In order to implement BLE in this project, the Nordic UART Service (NUS) was used. This Bluetooth low-energy GATT service is a custom service that receives and writes data and serves as a bridge to the UART interface. For this project, UART interfacing was not used; instead, it was used because of its emulation of a serial port over BLE.

The NUS service is split into two parts: the NUS client and the NUS. The regular NUS runs the GATT server with the NUS services. A GATT server in BLE is a device that stores data locally and provides access to that data to a remote GATT client, so it essentially acts as the data source that a client can read from or write to over a BLE connection. The NUS client interacts with the NUS by using the GATT discovery manager library. This is used so the client can properly use the services of the NUS GATT server and communicate with it.

For this project, the remote control used the NUS client, and the RC car used the NUS GATT server. As explained before, the purpose of using NUS services was to be able to mimic a serial port over BLE. The controller needs a way to transfer the ADC readings to the RC car to give it instructions on how to move. And that was done through the use of NUS and its APIs.

The program for the remote control begins by setting up and enabling Bluetooth and initializes the device for scanning and as a central. Next, the NUS client is initialized, and during this process, data sending and receiving callback functions are set. The device then begins scanning for advertising packets and finishes its initialization by setting and configuring the ADC and the nRF5340's internal hardware timer.

The Program then waits in an infinite loop until data is ready to be sent. Data is ready to be sent when an item is placed into the program's sending fifo. In this program, an item is periodically set into the sending FIFO every 400 milliseconds. This is because the internal time gets an ADC reading every 50 microseconds and stores that value in a sample buffer. This sample buffer has a size of 8000, so because it fills it every 50 microseconds, the buffer will be filled in 400 milliseconds.

When the buffer is filled, a function is called. In this function, the buffer's values are averaged, and this value is sent over the BLE connection using the bt_nus_client_send() API.

For the RC car, its program begins similarly by enabling Bluetooth and initializing nus, but for the server side, After this, the device starts advertising its presence and services. After this, it sets up its PWM and GPIO pins to control the motors. The program will then stay in an infinite loop, waiting for data to be sent over the connection. When it does, it will trigger a callback function. In this function, the ADC data from the remote control is used in an if-else chain. This chain determines the PWM and GPIO values, which in turn determines the movement of the model vehicle.

IV. Measurements & Results

Due to the non-intuitive nature of simulating or measuring the performance of an RC car, this discussion will focus on the model vehicle's weight, carrying capacity, and power source current draw.

To begin, the final weight of the model vehicle is approximately 800 grams. The individual components contributing to this weight are listed in Table 1. However., the primary contributors are the car's chassis, motors, and two

LiPo batteries, which power the motor drivers and the motors. Specifically, the chassis, four DC motors, and two LiPo batteries weigh 190 grams, 122.4 grams, and 198 grams, respectively [3][4].

The total weight of an RC car can vary significantly depending on the components used in its construction. In this design, weight is a critical consideration for two main reasons. First, additional components generally increase costs. Second, increased weight necessitates more robust motors and higher-voltage power supplies, further driving up costs. Since this implementation is intended to be a low-cost RC car, minimizing weight is essential for maintaining affordability in the final design.

TABLE 1
List of RC car components and their weights [3][4][5]

Components	Weight in grams
Chassis	190
DC gearbox motor (4)	122.4
OVONIC 2S LiPo Battery (2)	198
nRF5340 DK	62.5
Motor driver (2)	58
Wheel (4)	120
Half breadboard	50

The carrying weight capacity of the model vehicle is another critical aspect of the design to measure, as it directly relates to both the weight of the vehicle (discussed earlier) and the speed at which it can move. The carrying capacity is primarily determined by the motors used in the design. In this case, four 3-to-6-volt DC gearbox motors were selected. The technical specifications for these motors are as follows [4]:

Rated Voltage: 3~6V
Continuous No-Load Current: 150mA +/- 10%
Min. Operating Speed (3V): 90 +/- 10% RPM
Min. Operating Speed (6V): 200+/- 10% RPM
Stall Torque (3V): 0.4kg.cm
Stall Torque (6V): 0.8kg.cm
Gear Ratio: 1:48
Body Dimensions: 70 x 22 x 18mm
Wires Length: 200mm & 28 AWG
Weight: 30.6g

The most relevant specifications for evaluating carrying weight are the minimum operating speeds and stall torques at 3 volts and 6 volts. While these values are provided by the manufacturer, they do not account for factors such as load on the motors or variations caused by different wheel sizes. Therefore, while these specifications serve as useful references, they do not provide a complete picture of the vehicle's carrying capacity or maximum speed. There are many complicated equations that can give a good approximation of max weight and speed given these metrics, but in this design, experimentation was used rather than equations.

During testing, it was observed that the motors stalled when powered with 3V from the batteries. This issue was likely due not to insufficient voltage alone but to voltage drops occurring at the motor drivers. By increasing the input to 4V using pulse-width modulation (PWM), the car could move, albeit slowly. When powered with 6V, the vehicle moved significantly faster and more efficiently.

Additional testing was conducted by adding artificial weight to the vehicle. At 4V, the motors stalled when a further 300 grams was added. At 6V, the car could handle more weight but stalled completely with an extra 700 grams. These tests highlight the limitations of the motors under load and emphasize the importance of considering such constraints during the design phase.

The final metric examined was the current draw of the RC car. The nRF5340 DK is connected to one of the LiPo batteries in the system; as stated before, it is connected to the 5V pin on the motor driver. This means the system has three main current-drawing components: the motors, motor driver, and nRF5340 DK.

The system's current draw can be divided into two parts because the design uses two separate LiPo batteries. The bottom two motors and their motor driver draw current from one LiPo battery, while the top two motors and their motor driver and the nRF5340 draw current from the other LiPo. According to the motor driver specifications, the logical part of its operation requires a current of 36 mA [5]. For the DC motors, the specifications indicate a no-load current of 150 mA at 3 volts, increasing to 1.1 amps when fully stalled. At 6 volts, the no-load current is 160 mA, and the stalled current is 1.5 amps [4].

The current draw of the bottom system is summarized in Table 2. The data shows that the measured current draw aligns closely with the expected values from the datasheets. Specifically, when 4 volts is applied, the bottom system draws 0.852 amps, and when 6 volts is applied, it draws 1.137 amps. These measurements highlight the significant influence of voltage on the system's current demands.

TABLE 2
List of current draws in the system from different voltages (at 800 grams load)

Voltages	Current draw
4	0.852 A
5	0.972 A
6	1.137 A

V. Conclusion.

In conclusion, this paper demonstrated the hardware, software, and integration of various components to successfully create a functional RC car using Bluetooth Low Energy (BLE). Developing this system required the application of key embedded systems concepts. For instance, configuring and utilizing the ADC and its channels were essential for reading the joystick's analog output values. Similarly, PWM and GPIO pins were employed to control external hardware, such as the motor driver, while proper power supply selection considered current draw requirements

to ensure reliable operation. Additionally, implementing a wireless communication protocol like BLE, specifically a BLE GATT service such as NUS for serial port emulation, was crucial to achieving seamless communication. These fundamental concepts were integral to the successful implementation of the design.

REFERENCES

[1] H. K. Kondaveeti, N. K. Kumaravelu, S. D. Vanambathina, S. E. Mathe, and S. Vappangi, "A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations," *Computer Science Review*, vol. 40, no. 1574-0137, p. 100364, May 2021, doi: https://doi.org/10.1016/j.cosrev.2021.100364.

[2] "TechnicalDocumentation," *Nordicsemi.com*, 2024.https://docs.nordicsemi.com/bundle/ug_nrf5340_dk/page/UG/dk/intro.html

[3] A. Industries, "DC Gearbox Motor - 'TT Motor' - 200RPM - 3 to 6VDC," www.adafruit.com. https://www.adafruit.com/product/3777

[4] "2×OVONIC 2S Lipo Battery 2200mAh 2S1P 35C 7.4V RC LiPo Battery Short S," Ampow, 2024. https://www.ampow.com/products/ovonic-7-4v 2200mah-2s-35c-short-lipo-battery-with-deans-2pcs

[5] "MD1.3_2A_Dual_Motor_Controller_SKU_DRI0002DFRobot," wiki.dfrobot.com.https://wiki.dfrobot.com/MD1.3_2A_Dual_Motor_Controller_SKU_DRI 0002

[6] "Connection process - Nordic Developer Academy," *Nordic Developer Academy*, May 02, 2024. https://academy.nordicsemi.com/courses/bluetooth-low-energy-fundamentals/lessons/lesson-3-bluetooth-le-connections/topic/connection-process/

[7] "Technical Documentation," *Nordicsemi.com*, 2024. https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/libraries/bluetooth/services/nus.html