# RC Racing Simulator

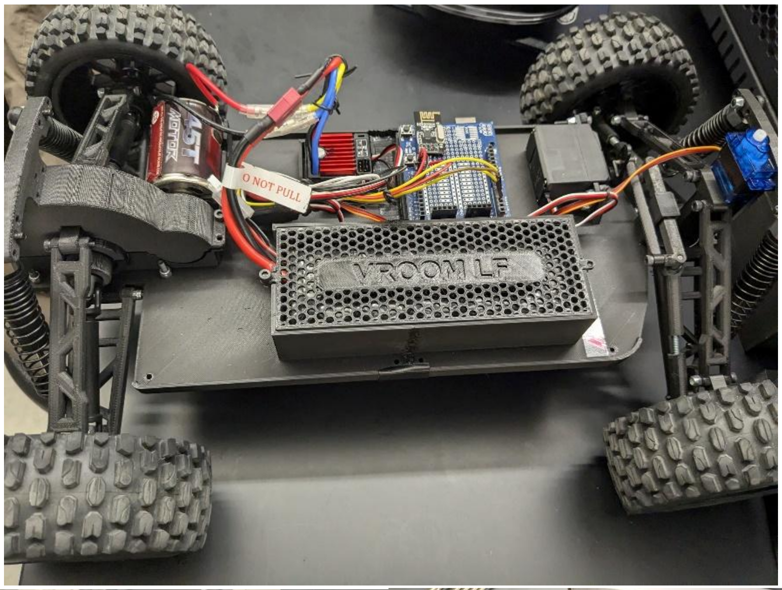
Vic Cuatico
Parsa Hashemi
W. Booth School of Engineering Practice and Technology
August 16, 2024

## Contents

## Introduction

This report will walk through the RC Racing Simulator's project details. Each component and their wiring will be shown clearly, and additionally the code for each microcontroller will be included.

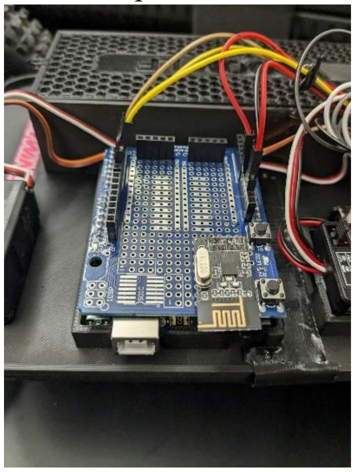
The main objective of this project is to use Arduino and nRF24-L01 chips to allow a commercial RC simulator to seamlessly communicate with a Learning Factory RC car.



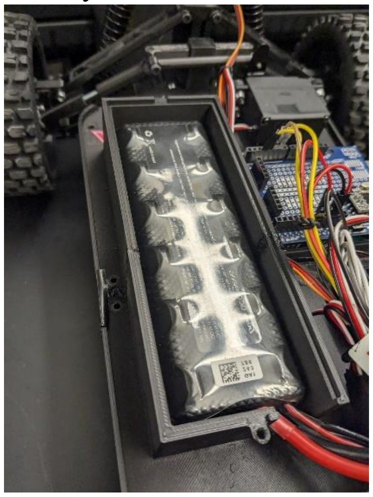
## Components

### RC Car

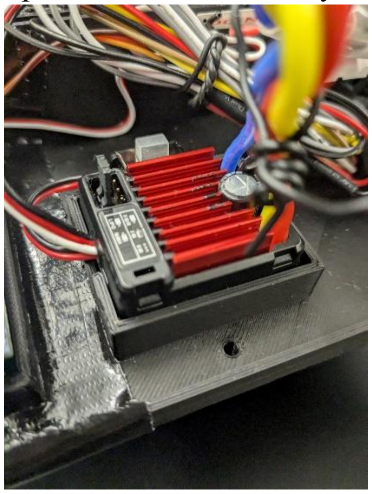
1. Microcontroller – Arduino Uno
2. Uno Soldering Shield
3. Radio Chip – nRF24-L01



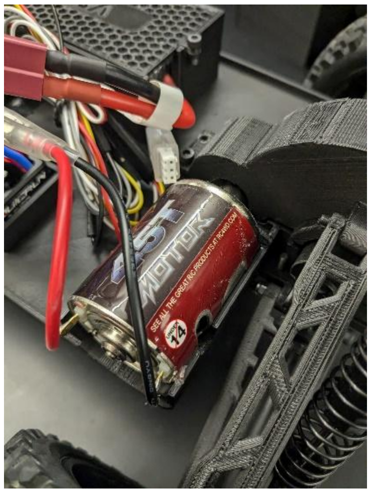
4. Battery – Gens Ace 4000mAh 2S1P 7.4V 45C LiPo



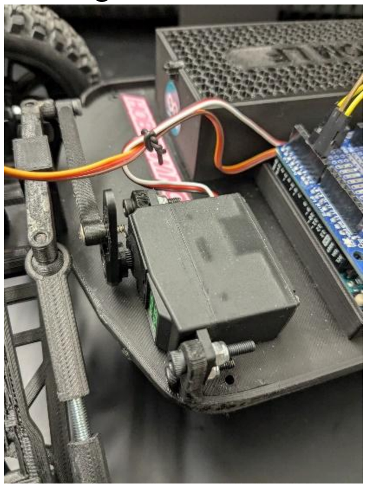
5. Speed Controller - Hobbywing Quicrun 1060 Brushed Electronic Speed Controller



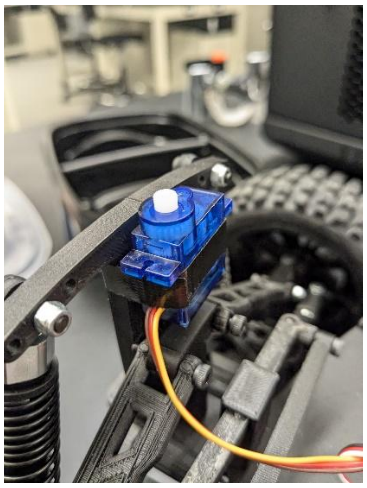
6. Motor – 540 Crawler Brushed Motor 45T



7. Steering Servo – ECO Power Standard Ball Bearing Servo



8. Camera Servo – Servo Motor SG90

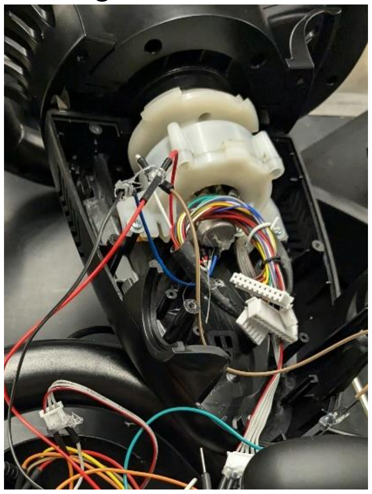


9. Camera – Skydroid Camera

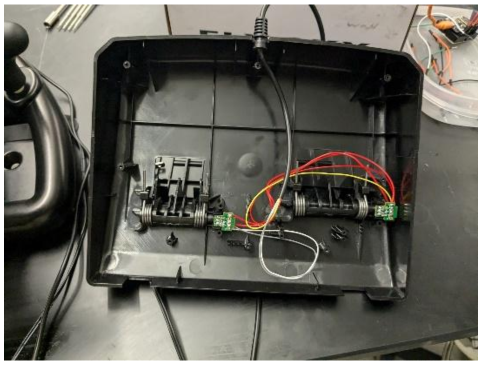
## Racing Sim

PXN racing wheel → https://www.amazon.ca/Racing-PXN-Universal-Steering-Nintendo/dp/B07XK6F14F

1. Steering Wheel – 5K Potentiometer

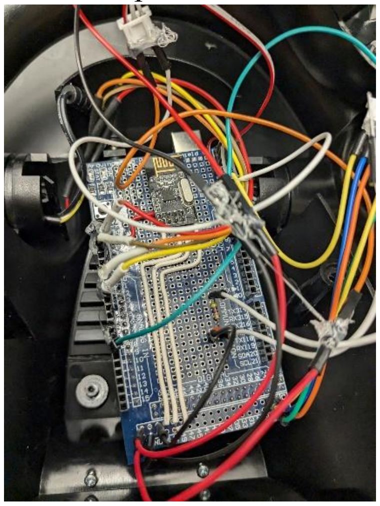


2. Shift Knob – 2 Buttons (Shift down left disconnected)
3. Forward Pedal – Potentiometer
4. Reverse Pedal – Potentiometer



5. Microcontroller – Elegoo Arduino Mega
6. Mega Soldering Shield

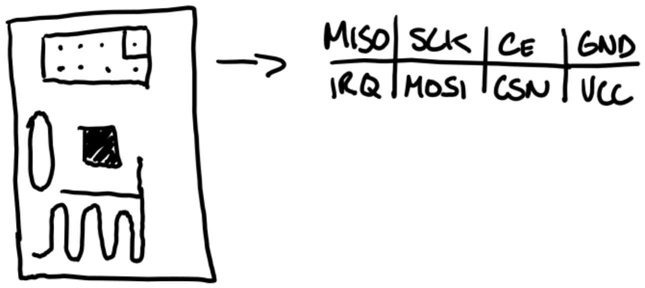
7. Radio Chip – nRF24-L01



## Wiring

### Radio Chip

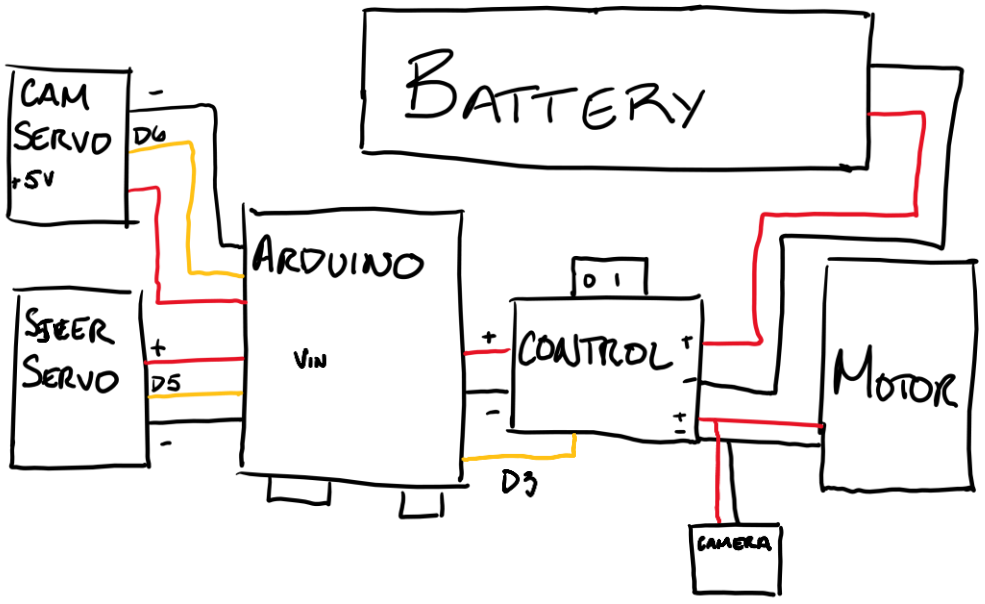
The Chip being used is the nRF24-L01 chip. Capable of receiving and transmitting data. The pins on the chip are as shown in the following diagram:



Please refer to the RC Car and Racing Sim wiring tables to find which pins the radio chip is connected to.

### RC Car

Drawn Schematic of the RC Circuit:



Pins to the Arduino Uno:

| | |
|---|---|
| Speed Controller | +5V out -> VIN<br>Ground -> Ground<br>Serial -> D3 |
| Steer Servo | VCC -> 5V<br>Ground -> Ground<br>Serial -> D5 |
| Camera Servo | VCC -> 5V<br>Ground -> Ground<br>Serial-> D6 |
| Radio Chip | VCC -> 3.3V<br>Ground -> Ground<br>CE -> D9<br>CSN -> D10<br>SCK -> D13<br>MOSI -> D11<br>MISO -> D12<br>IRQ -> Leave disconnected |

## Racing Sim

Pins to the Elegoo Mega:

| | |
|---|---|
| Wheel | VCC - > 5V<br>Serial -> A8<br>Ground -> Ground |
| Forward Pedal | VCC -> Reverse 5V Out<br>Serial -> A1<br>Ground -> Reverse Ground |
| Reverse Pedal | VCC -> 5V<br>Serial -> A2<br>Ground -> Ground |
| Shift Knob | VCC -> 3.3V<br>Serial 1 -> Leave Disconnected<br>Serial 2 -> D13 |
| Radio Chip | VCC -> 3.3V<br>Ground -> Ground<br>CE -> D10<br>CSN -> D11<br>SCK -> D52<br>MOSI -> D51<br>MISO -> D50 |

## Code

### RC Car Receiver Code

```
/*
  Module // Arduino UNO
    GND    ->   GND`
    Vcc    ->   3.3V
    CE     ->   D9
    CSN    ->   D10
    CLK    ->   D13
    MOSI   ->   D11
    MISO   ->   D12
  RC Car // Arduino UNO
    Steer Servo -> D5
    Motor       -> D3

  Whole Sim Code RC CAR
 */

#include <SPI.h>                   //Radio
#include <nRF24L01.h>              //Radio
#include <RF24.h>                  //Radio
#include <Servo.h>                 //Servo

                 //Send Data
struct Data
 {
  int Speed = 0;                 //Value transmitted
  int Direction = 90;
 };

                 //Global Variables
//Customizeable variables
const int NumOfTries = 7000;

//Global Var
int Tries = 0;

//Pins
const int Motor = 3;

//Radio
const uint64_t pipeIn = 0xE8E8F0F0E1LL;
RF24 radio(9, 10);                 // CSN and CE  pins
```

```
Data Sent;

//Motor
Servo MotorSpin;
Servo STEER;
Servo Camera;

                  //Prototyoe
void RecordData();
void MotorWrite();

void setup()
{
  Serial.begin(9600);

  radio.begin();
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.openReadingPipe(1,pipeIn);
  radio.startListening();

//Attach servos
  MotorSpin.attach(3);
  STEER.attach(5);
  Camera.attach(6);


  MotorSpin.writeMicroseconds(1497);
  delay(1000);

  while(!radio.available())
  {
    STEER.write(0);
    delay(1000);
    STEER.write(180);
    delay(1000);
  }
  Sent.Direction = 90;
}

void loop()
{
  RecordData();

  if (!radio.available())
  {
    Tries++;
  }
  else
  {
    Tries = 0;
  }

  if (Tries >= NumOfTries)
  {
    Sent.Speed = 0;
    MotorWrite();
    delay(1000);
    Sent.Speed = -100;
    MotorWrite();
    delay(2000);
    Sent.Speed = 0;
    MotorWrite();
    while(!radio.available())
    {

    }
  }
}

void RecordData()
{
  while (radio.available() )
  {
    radio.read(&Sent, sizeof(Data));
    Serial.println(Sent.Speed);
    MotorWrite();
    Tries = 0;
  }
}

void MotorWrite()
{
   Sent.Speed = map(Sent.Speed, -255, 255, 1000, 2000);    //Map the speed
  MotorSpin.writeMicroseconds(Sent.Speed);
  STEER.write(Sent.Direction);                            //Write the direction
to the Steer servo
  Camera.write(map(Sent.Direction, 45, 135, 60, 120));    //Write the direction
with the smaller range to the camera servo
}
```

## RC Car Transmitter Code

```
/*
  Radio Module // Arduino MEGA
    GND     ->    GND
    Vcc     ->    3.3V
    CE      ->    D10
    CSN     ->    D11
    SCK     ->    D52
    MOSI    ->    D51
    MISO    ->    D50

  June 2024 Working RC Simulator
*/

//Include Libraries
#include <SPI.h>                  //Radio
#include <nRF24L01.h>             //Radio
#include <RF24.h>                 //Radio

//Data Stricture
struct Data
{
  int Speed = 0;                  //Speed Value
  int Direction = 90;             //Direction
};

//Pins
const int RVRS = A2;                      //Backwards Pin
const int FWRD = A1;                      //Forward Pin
const int WHEEL = A8;                     //Steering Wheel Pin
const int SHFT = 13;                      //Shift Up pin

//Radio
const uint64_t pipeOut = 0xE8E8F0F0E1LL;  //Radio Address
RF24 radio(10, 11);                       //CSN and CE
Data Send;                                //Innitialize

//Variables
int MapFWRD = 0;                          //Forward Mapped
int MapRVRS = 0;                          //Reverse Mapped
int range[4] = {120, 150, 190, 255};      //Gear speed ranges
int gear = 0;                             //Current gear
bool shifted = false;                     //Boolean to avoid held shifting
```

```
//Prototypes
void SendData(int PVal);                    //Radio Send
void PedalRead();                           //Read Pedals
void WheelRead();                           //Read Steering Wheel
void ShiftRead();                           //Read Shift Knob

void setup()
{
  Serial.begin(9600);

  //Pin Initialize
  pinMode (FWRD, INPUT);
  pinMode (RVRS, INPUT);
  pinMode (WHEEL, INPUT);
  pinMode (SHFT, INPUT);

  //Radio Intialize
  radio.begin();
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.openWritingPipe(pipeOut);
}

void loop()
{
  //Value read
  ShiftRead();
  PedalRead();
  WheelRead();

  while (MapFWRD > 0)
  {
    PedalRead();
    SendData(MapFWRD);
  }

  MapFWRD = 0;

  while(MapRVRS < 0)
  {
    PedalRead();
    SendData(MapRVRS);
  }

  SendData(0);
```

```
}

void SendData(int PVal)
{
  ShiftRead();
  WheelRead();
  Send.Speed = PVal;
  radio.write(&Send, sizeof(Data));
  Serial.println(String(Send.Speed) + "\t\t" + String(Send.Direction));
}

void PedalRead()
{
 MapFWRD = analogRead(FWRD);
 MapRVRS = -analogRead(RVRS);

 if (MapFWRD > 800)
  {
    MapFWRD = 800;
  }

  MapFWRD = map(MapFWRD, 0, 800, 0, range[gear]);
  MapRVRS = map(MapRVRS, 0, -1023, 0, -range[gear]);
}

void WheelRead()
{
  Send.Direction = map(analogRead(WHEEL), 0, 1023, 180, 0);
}

void ShiftRead()
{
  if ((digitalRead(SHFT) == HIGH) && shifted == false) {
    if (gear == 3) {
       gear = 0;
    }

    else {
      gear += 1;
    }

    shifted = true;
  }

  else if (digitalRead(SHFT) == LOW) {
    shifted = false;
  }
}
```

## Brief Code Explanation

The code uploaded onto the RC car is left short and simple to use less battery usage due to having less processing to perform. It receives the data from the RecordData(); function then remaps it for the various motors through the MotorWrite(); function.

The code uploaded to the Racing Sim has a lot of processing due to the differences in readings from the Racing Sim. The PedalRead();, ShiftRead();, and WheelRead(); functions read each sensor currently being used. The information is then sent to the RC car through the SendData(); function.

The radio chips use a structure to send multiple numbers at the same time. They are currently set to communicate at 250kBPS through the address 0xE8E8F0F0E1LL. It is important to ensure that the two radio chips use the same address and frequency so they can properly connect.

Additionally, when the receiver does not find a connection for 7000 tries, it will slowly reverse on the same direction as the last input until it is back in range.

## Problems

### <u>Shifter</u>

In its most basic form, the shift knob is just two push buttons. The physical knob itself uses springs and levers to press one of the buttons whether its pushed upwards or downwards. When giving power to one wire and reading the others on a separate Arduino it worked fine reading 0s and 1s depending on which button is pressed. When adding it to the Arduino controlling the whole wheel the down shift caused inaccuracies in the pedal and steering readings. Eventually it was found that downshifting caused some sort of short circuit or large draw in current grounding everything which then caused the Arduino to fully reset itself sometimes.

### <u>Lights</u>

Initially the lights worked perfectly fine. However, one day they just stopped working. Even connecting to them to a separate Arduino board just turning on the lights was not possible. The LEDs could’ve burnt out or when the shifter was resetting the board something could’ve happened causing the LEDs to stop working.

### <u>Steering Potentiometer</u>

Over time the original potentiometer in the steering wheel started to become more and more inaccurate. The wiring connections were starting to become weak from the number of times the wheel was open and closed to fix other components within the wheel. After numerous attempts of soldering stronger connections to the potentiometer is eventually just stopped working in certain positions. The potentiometer used was a generic potentiometer so replacing it was not an issue. This just means that the electronics in the wheel is very delicate and requires us to be more careful with what we are doing.

## Future Plans

### Remote Driving

Now having access to more features of the Arduino IoT, it is possible to create an online platform for people to drive the car remotely. This of course includes web development and more research done on the Arduino IoT API. Additionally, research must be put into finding the best way to host this server with the least amount of delay.

### Data Analysis

The next step with making this car more efficient can be adding sensors that measure speed and power consumption. This can allow us to work around problems with the components overheating and battery life concerns especially when adding more smart components to the car. This can provide thoughtful insights to take into account in the future and can even benefit the design of the RC car as well.

### Autonomous Driving/ Driving Assist

Adding additional sensors around the car to add driving assist features. This can be anything like detecting obstacles, blind spot indicators, parking assist, and positioning cameras around the car to generate a birds-eye view image of the car. After this the project can shift towards autonomous driving using artificial intelligence and video recognition. The car can detect lanes on a pathway, stop/yield signs, and intersection lights.

## Conclusion

In summary, this project not only achieved its primary goal of integrating an RC car with a commercial racing simulator. It also provided valuable insights into the challenges and solutions associated with wireless communication and real-time control in a hobbyist setting. This project opens up possibilities for further enhancements, such as adding more sensors for autonomous driving, data analytics, improving response times, and increasing the communication range, ultimately contributing to a more immersive RC racing simulation.