RC car project documentation

Overview

The goal of this project was to build a functioning 1/24 scale remote controlled replica of my 2004 Mini Cooper. I wanted it to have more features than a cheap Wal-Mart car, including better range, a graduated throttle system that had more than two speeds, A graduated steering system that had more than three positions, working brakes, headlights, brake lights, and turn signals. I wanted the driving experience to be more like a real car, so it won’t be able to accelerate instantly to maximum speed, and if the throttle is released it will slowly come to a stop. The brakes also won’t immediately stop the car so that it feels like you have momentum. I also wanted to be able to mount an FPV camera on it, and include a head tracking system. The controller will be a sim char with a steering wheel and pedals.

Electrically, this project consists of two main parts, the transmitter and the receiver. The transmitter and the receiver each have their own Arduino board, wiring diagram, and physical hardware. The electronics need to be built and programmed simultaneously to make sure everything is working properly. The transmitter used an Arduino Uno R3, but the receiver is running on a much smaller Arduino Nano since space on the car was a major concern.

The mechanical side of the project is designed to be almost completely 3d printed. I created the car in Blender. I designed the insides, but the outside body mostly comes from a Mini Cooper model that I found for free on Thingiverse.com. I have since modified it to suit my needs, but the outward appearance remains mostly the same. The .blend file is apparently too big for GitHub, so I just uploaded all the .stl files. They’re fairly low poly and should be easy to modify if necessary.

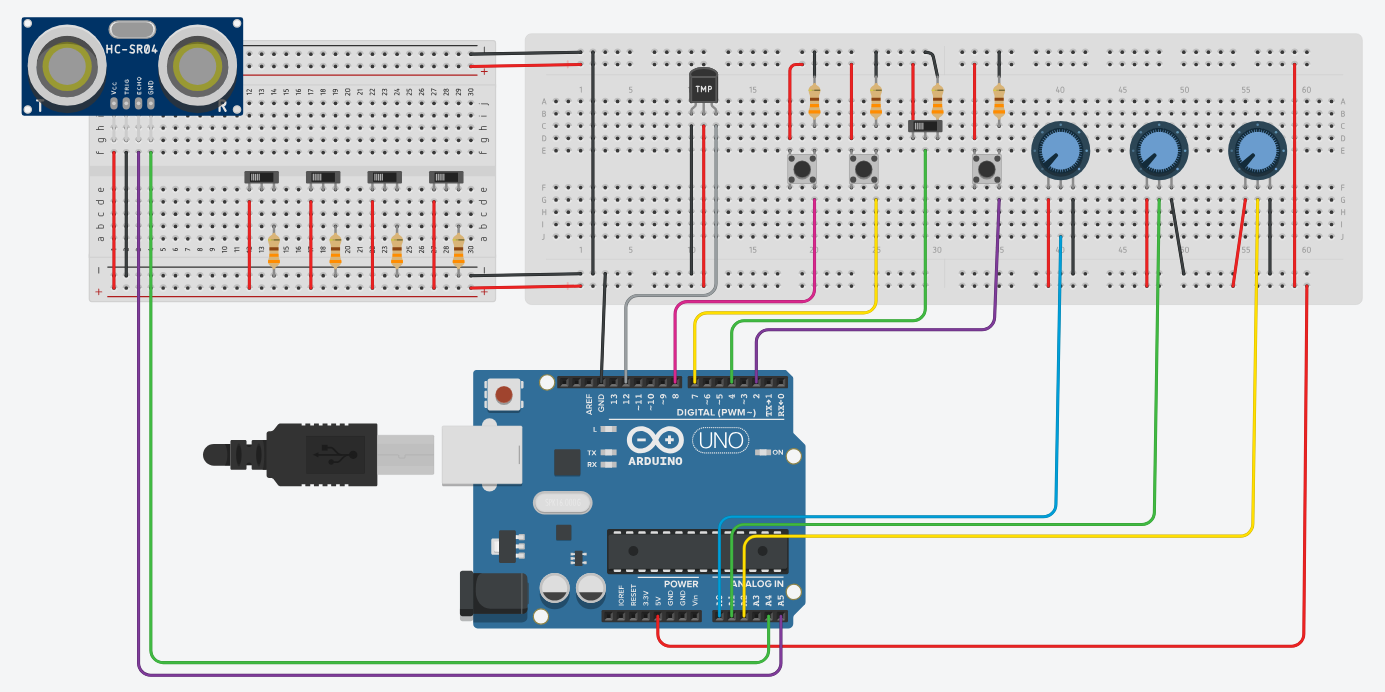
I also designed the pedals in Blender. They were made to be screwed down to a piece of plywood once assembled.

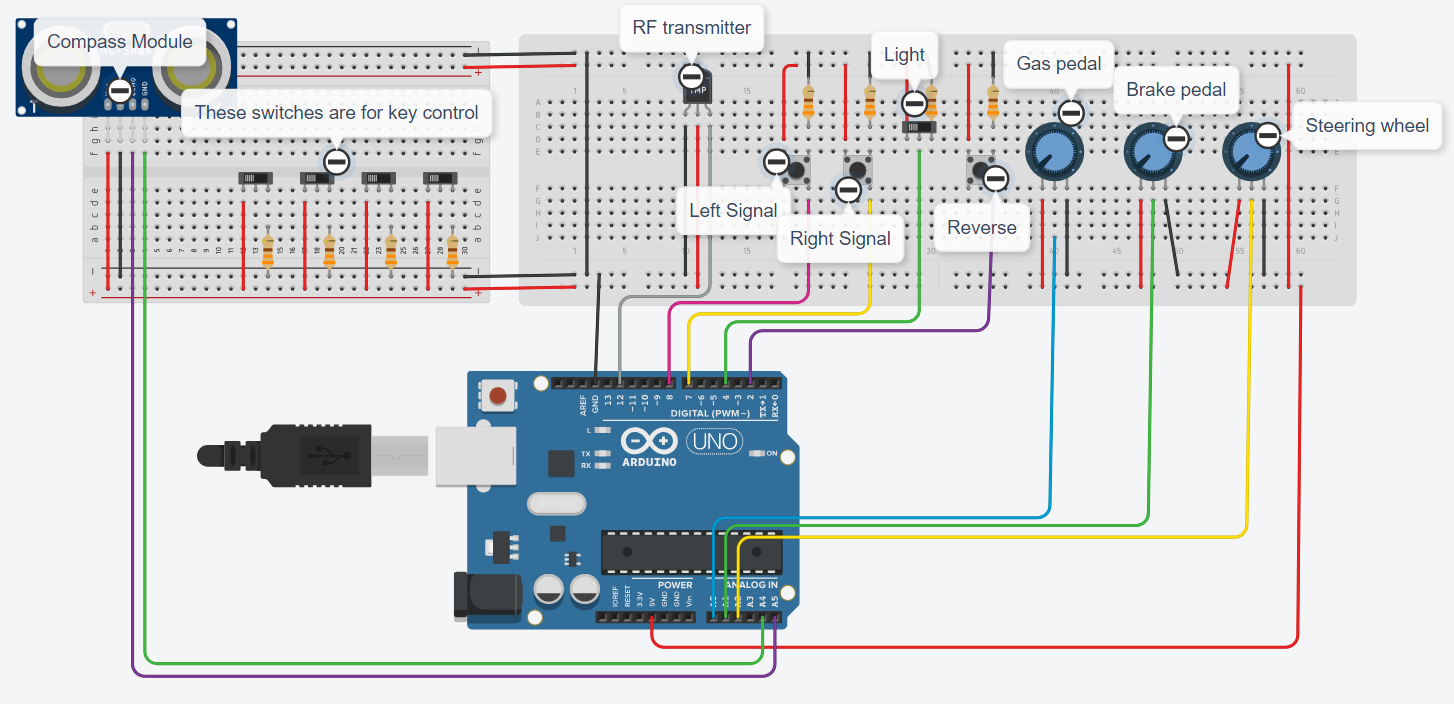
This is the link to the GitHub repo containing the files for the project

<https://github.com/KyleStalker17/RcCar_Project/tree/main>

Transmitter

Transmitter Wiring diagram:





Tinker CAD doesn’t have a symbol for the transmitter or compass modules, so I used replacement symbols to represent them. The purple wire, coming from A5, goes to the SDA pin of the compass, and the green wire, coming from A4, goes to the SCD pin. The gray wire, coming from D12, goes to the data pin of the transmitter.

The compass module is mounted on the front of the FPV goggles, and tracks your head by figuring out how many degrees off of north your head is. This means that in order for it to work properly, you have to sit facing due north.

Transmitter Parts list

1x Arduino R3

1x 10534 433 MHz RF transmitter

5x Slide switch

3x momentary button

8x 330Ω resistor

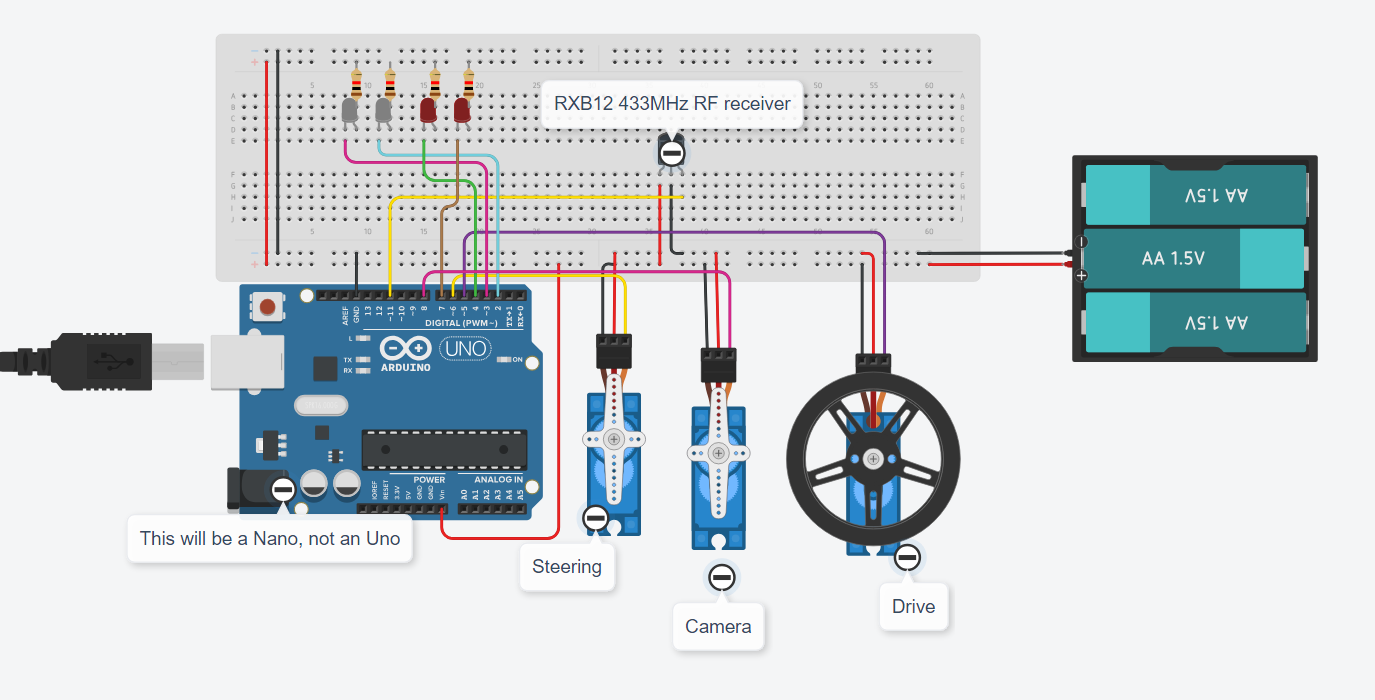
3x 10kΩ potentiometer

1x QMC5883L compass module

Prototype pictures

Receiver

Wiring Diagram



The receiver will have three servos. One to move the car forward, one to turn the wheels, and one to turn the FPV camera. Once again, I had to use replacement symbols in the wiring diagram because Tinker CAD doesn’t have symbols for Arduino Nano or RxB12 receivers.

I tried first with the cheapest receivers I could find and it was a bad idea. I did more internet research, then bought the slightly more expensive but more effective RxB12. It made a huge difference.

The servos can’t use the regular Servo library because the timer it requires is busy with the RF receiver. Instead, I had to use a library called Servo Timer 2.

The battery will be a 3.7V LiPo rather than the 3 AAs because, once again, space and weight are a major concern on the car.

Parts List

2x 180° SG90 tower servos

1x 360° SG90 tower servo

1x Arduino Nano

1x RxB12 433 MHz receiver

2x White LEDs

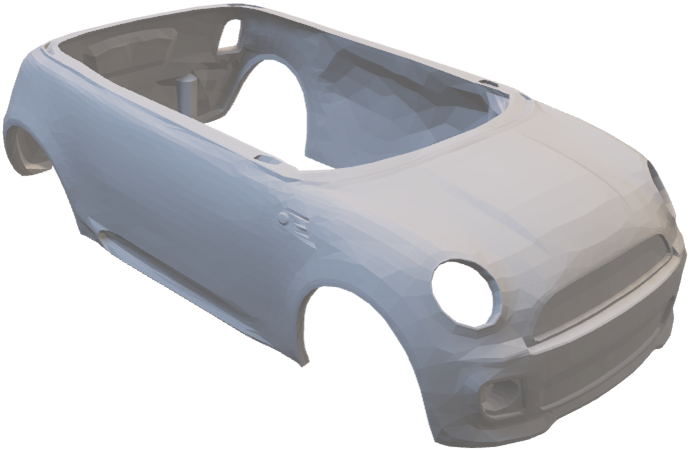
2x Red LEDs

4x 330Ω resistors

Libraries

Prototype Pictures

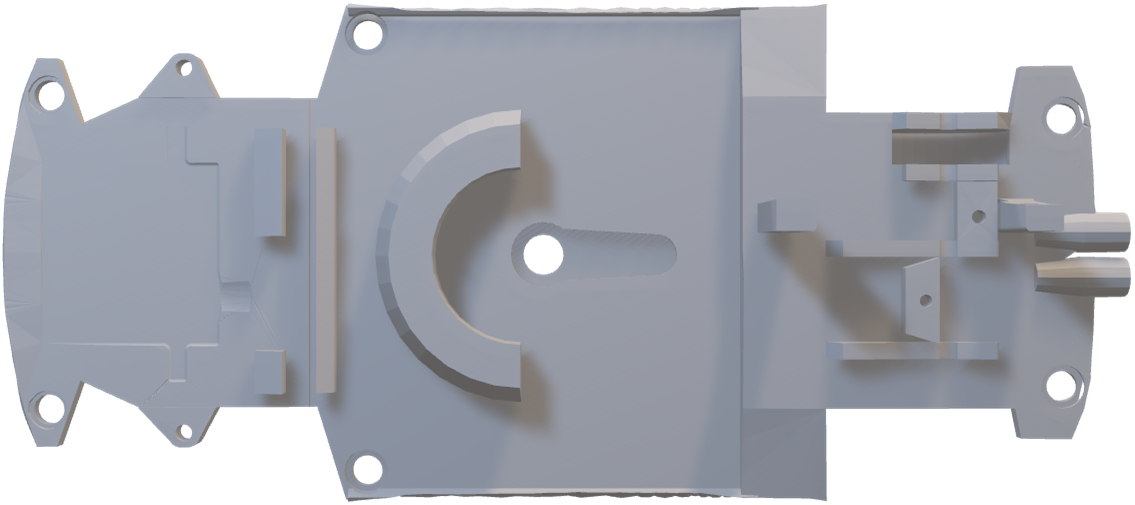
Shell





The shell will print in two parts, the main body and the roof. This will eliminate a ton of support material and greatly reduce printing time

Internals

The mechanical parts on the inside will probably also have to be printed in two parts, the main bottom and a second print with everything else

