Autonomous RC Car

**David Jackson - Summer 2016**

Objective

Develop an autonomous remote-controlled (RC) car using an Arduino and PID controllers.

Introduction

The idea of developing an autonomous RC car was conceived by Dr. Peters and initially developed by Dwarkesh Iyengar in October of 2015 [(link)](http://dianelpeters.com/DSCC_2015_Development_of_a_Miniaturized_Autonomous_Vehicle.pdf). Iyengar used a microcontroller unit to control the car. The microcontroller would receive commands and transmit data to a myRIO controller unit using Xbee radio transmitters. Building on Iyengar’s work, the goal of this project was to place the controller in the RC car, removing the dependency on the myRIO controller and making the car fully autonomous.

Iterations

This project went through three design iterations during the author’s term.

Iteration 1: Servo Mounted LIDAR

The first design iteration centered around using one LIDAR sensor mounted to a servo motor to measure the distance to the vehicle’s surroundings. The servo motor would rotate 90 degrees, obtaining a distance reading at every degree interval. Wiring for the Lidar/Servo assembly and the Arduino can be seen in Figure 1. This concept was first tested by placing the LIDAR/Servo assembly in 12 inches away from a wall. The servo motor looped through 45 and 135 degrees and back with different delay times between degrees ranging from 5 and 50 milliseconds. This was repeated 10 times for data validation.

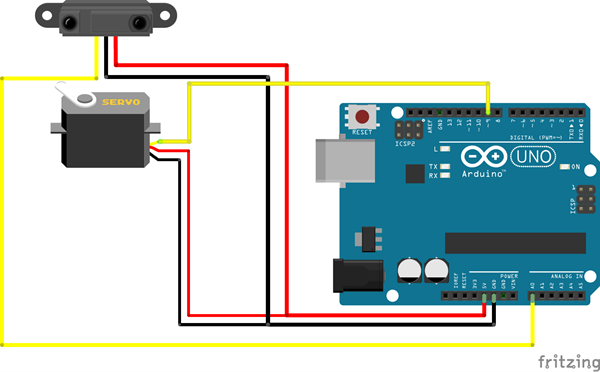


Figure 1: LIDAR/Servo Wiring Schematic

The results of the LIDAR/Servo test showed a deviation of around 10 cm between measurements at all speeds tested. This result was not within the accuracy needed for this project. A closer look into the specification sheet for the sensor showed a range between 0 and 40 meters and an accuracy of 0.025 meters. The LIDAR sensor was meant for larger-scale use where the difference between centimeters was not as important. IR distance sensors were implemented as an alternative to LIDAR, which is described in Iteration 3.

Iteration 2: Radio Transmission Data Collection

In an attempt to collect real-time analytics from the RC car without the need of a wire, two radio transmitters where considered. Wiring schematic for this iteration can be seen in [Figure 2](https://jbt.github.io/markdown-editor/#fig-2). An Arduino Nano was used as the controller. The controller was connected to an accelerometer/gyroscope/ magnetometer board for acceleration readings and the steering potentiometer for steering direction readings. Additionally, the controller was connected to a bi-directional motor controller IC, also called a H-bridge to control the speed and direction of the steering and drive motors (in Iteration 3 this was replaced with the Adafruit Motor Shield, which is based on a similar chip). Finally, he controller was then connected to the radio transmitter, which would send acceleration, steering direction, and motor speed data to the radio receiver, seen in Figure 3, which would then be able to be seen on a computer.

During testing, the Arduino controller took approximately 1.5 seconds to collect and send the data to the radio receiver. This delay was most likely due to the lack of processing power on the Arduino and could be avoided with a more powerful controller. This large of a delay in sensor data acquisition would be detrimental to the performance of the controller, so the concept of relaying data using radio was tabled and another iteration was developed.

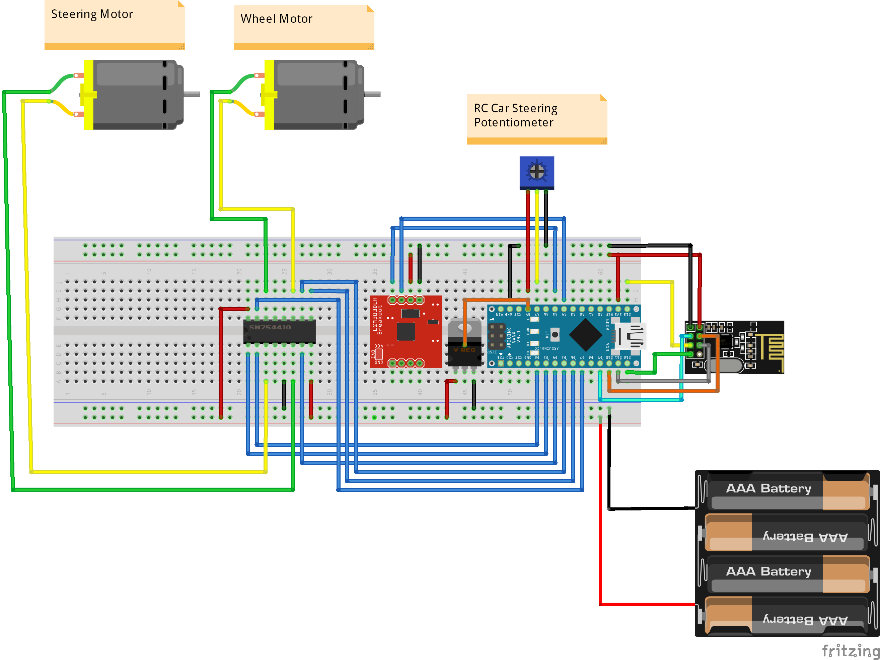


Figure 2: Radio Transmitter Controller Wiring Schematic

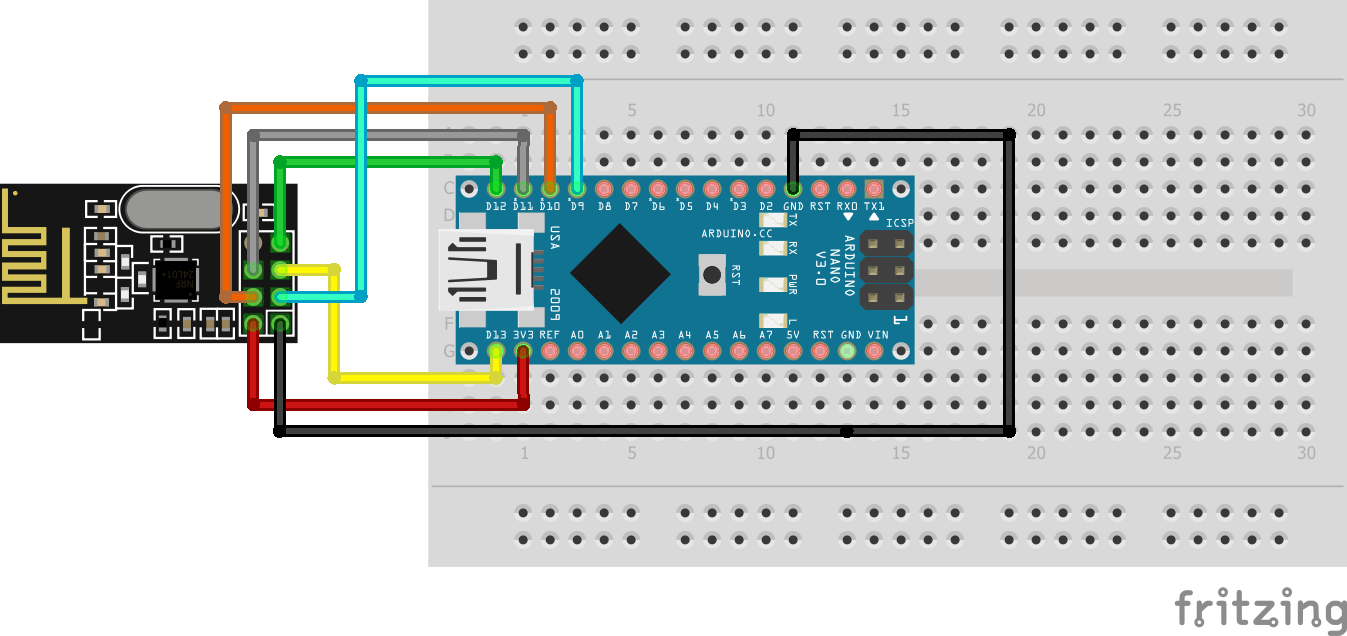


Figure 3: Radio Receiver Module

Iteration 3: RC Car with IR Distance Sensors

The final design of the autonomous RC car consisted of three [IR distance sensors](https://www.amazon.com/GP2Y0A21YK0F-Sharp-Distance-10-80cm-Compatible/dp/B00IMOSEJA), an Arduino, and two PID controllers. The wiring schematic for this design can be found in Figure 4. This iteration was completely self-sufficient and did not require any radio transmission or outside control. Adapted from Microsoft’s Internet of Things [Github repository](https://github.com/ms-iot/wod-autorc) on an autonomous RC car, the vehicle would determine its surroundings using the IR sensors and decide whether to turn, stop, or continue straight.

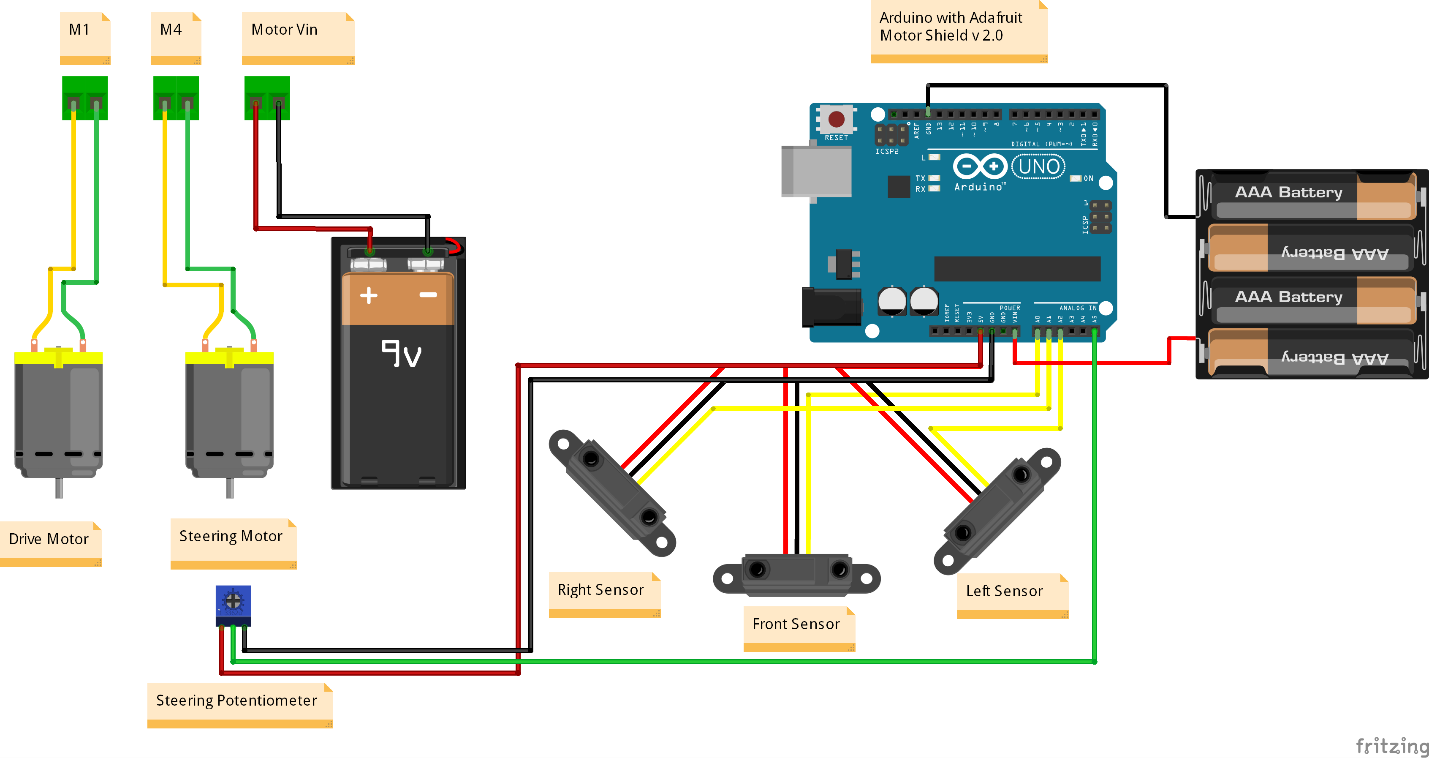


Figure 4: RC Car with IR Distance Sensors Wiring Diagram

This design was similar to the radio transmitter iteration, except with an Arduino Uno instead of an Arduino Nano, an Adafruit Motor Shield v2.0 instead of the H-bridge Motor Driver IC, and with the addition of three IR distance sensors mounted to the front of the vehicle to measure direction straight ahead, to the right of the vehicle, and to the left of the vehicle. M1, M4, and Motor Vin seen in Figure 4 represent the terminal blocks on the motor shield. It is also important to notice that the Arduino and motors are being powered by two different power sources to eliminate sensor noise produced by the motors’ energy consumption.

**Arduino PID Code**

Below is all the code relating to the Arduino PID controller library. For a complete code file, see the code in the project folder.

Include the Arduino PID Library.

#**include**<PID\_v1.h>

Next, define the variables used for the Drive PID Controller.

**float** Kp\_drive = 0.8; // Proportional Gain

**float** Ki\_drive = 0.07; // Integral Gain

**float** Kd\_drive = 0.12; // Derivative Gain

**double** Setpoint\_drive, Input\_drive, Output\_drive;

Initilize the Drive PID Controller.

PID **drivePID**(&Input\_drive, &Output\_drive, &Setpoint\_drive, Kp\_drive, Ki\_drive, Kd\_drive, DIRECT);

Define Variables for the Steering PID Controller.

**float** Kp\_steer = 0.8;

**float** Ki\_steer = 0.07;

**float** Kd\_steer = 0.12;

**double** Setpoint\_steer, Input\_steer, Output\_steer;

Initilize the Steering PID Controller.

PID **steerPID**(&Input\_steer, &Output\_steer, &Setpoint\_steer, Kp\_steer, Ki\_steer, Kd\_steer, DIRECT);

In the setup() function, set the Drive PID setpoint, the PID mode, and the output limits. In this case the output is going to be the speed of the motor, which we want to be between -50 (reverse) and 50 (forward). NOTE: the maximum motor speed is -255 and 255.

Input\_drive = readPosition();

Setpoint\_drive = 300;

drivePID.SetMode(AUTOMATIC);

drivePID.SetOutputLimits(-50, 50);

Now do the same for the Steering PID controller, with the motor speed being limited to between -40 and 40.

Input\_steer = readPosition();

Setpoint\_steer = 650;

steerPID.SetMode(AUTOMATIC);

steerPID.SetOutputLimits(-40, 40);

In the loop() function, read angle of the steering wheels from the potentiometer, compute the PID response, and set the motor speed accordingly.

Input\_steer = readSteering();

steerPID.Compute();

setSteerSpeed(Output\_steer);

Read angle of the steering wheels from the middle IR distance sensor, compute the PID response, and set the motor speed accordingly.

Setpoint\_drive = 300;

Input\_drive = readPosition();

drivePID.Compute();

**if** (stopped) {

setDriveSpeed(0);

} **else** {

setDriveSpeed(Output\_drive);

}

Conclusions

This independent study was able to design and create an autonomous RC car using an Arduino. The Arduino uses IR distance sensors to determine its surroundings and a potentiometer to determine the angle of the front wheels. The Arduino PID library was successfully implemented on the Arduino using two controllers: one for the control of the angle of the front wheels, and one for the speed of the driven rear wheels. In the continuation of this project, the author recommends fine tuning of the PID controllers and to add a data collection unit to the Arduino. The drive PID controller should be fine-tuned to an over damped controller to eliminate any overshoot when approaching an object. Additionally, fine-tuning the steering controller will also eliminate any steering overshoot and will guarantee the steering will reach steady-state. The addition of a data collection unit would be very beneficial to this project to analyze vehicle acceleration and sensor effectiveness. Data collection can be added to the Arduino using a [Data Logger shield](https://www.amazon.com/SD-Shield-for-Arduino-v4-0/dp/B00BPGPNH8) or even an [Ethernet Shield](https://www.amazon.com/Ethernet-Shield-Network-Expansion-Arduino/dp/B00AXVX5D0/), which both include the ability to read and write files to an SD card.