Physics 124 Project Report
Bryan Liang and Biyang Yu

Self Correcting Vehicle

Motivation and Concept

Self-driving and correcting vehicles have always been a significant aspect in automotive industry. An important task for the vehicle to accomplish is to sense its surroundings and react correspondingly. Fundamentally, the vehicle should be able to detect “obstacles” and try its best to avoid it or come to a stop without interference of human operators. If a collision cannot be avoided, it should reduce the velocity as much as possible. In this sense, we will need to attach peripheral sensors to the vehicle. Proximity sensors on the car will analyze the vehicle’s surroundings. If one side of the car is too close to an object, the car will turn away at increasing angles based on the distance from the object.

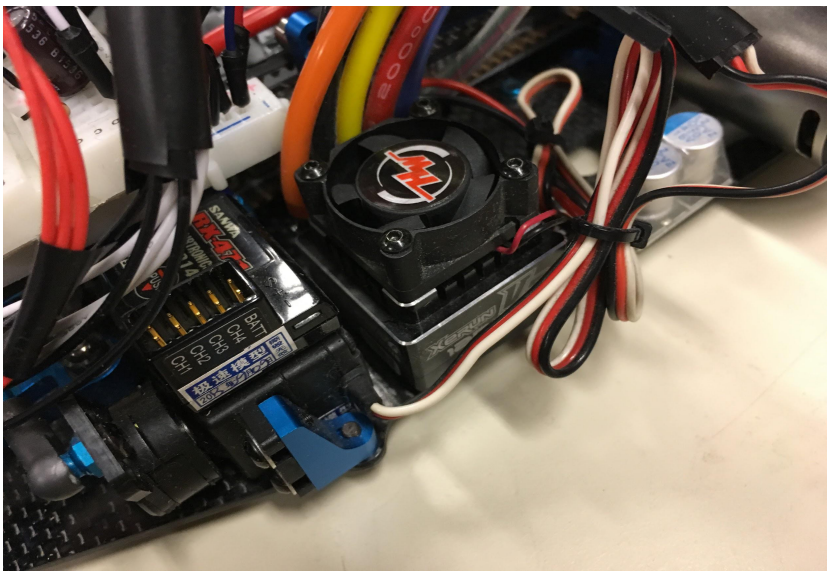
Functional Definition

Our proximity sensors will determine distances toward various or a large obstacle in 3 directions(front right, front, front left). The car will be moving at a constant speed (adjusted for the space in the lab) and will slow down when it needs to avoid an obstacle. Arduino will accelerate the car after successfully avoid the obstacle.

Sensors

And IR distance sensor includes cable (10cm-80cm) - Sharp GP2Y0A21YK, this device is to be supplied with 0 and +5 V, and output 0.4V to 3.3V depend on the distance its sensing. A 120A ESC was used as well.

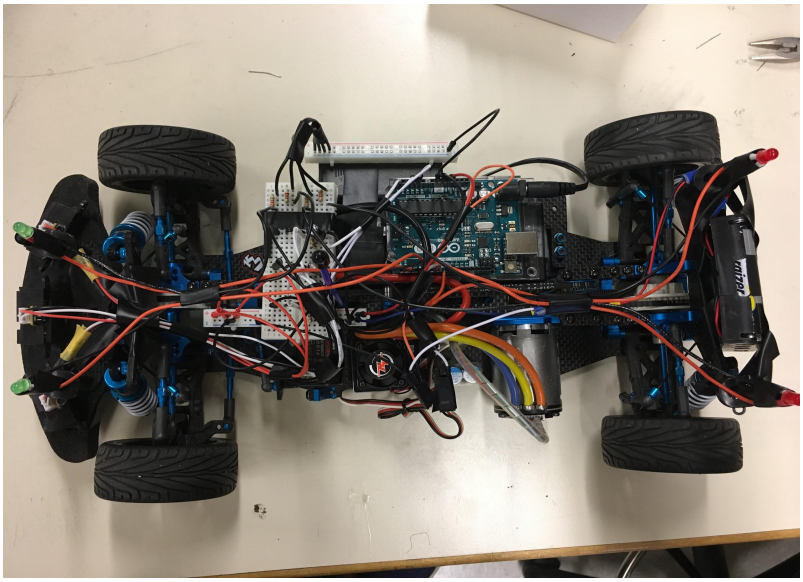
120A ESC:



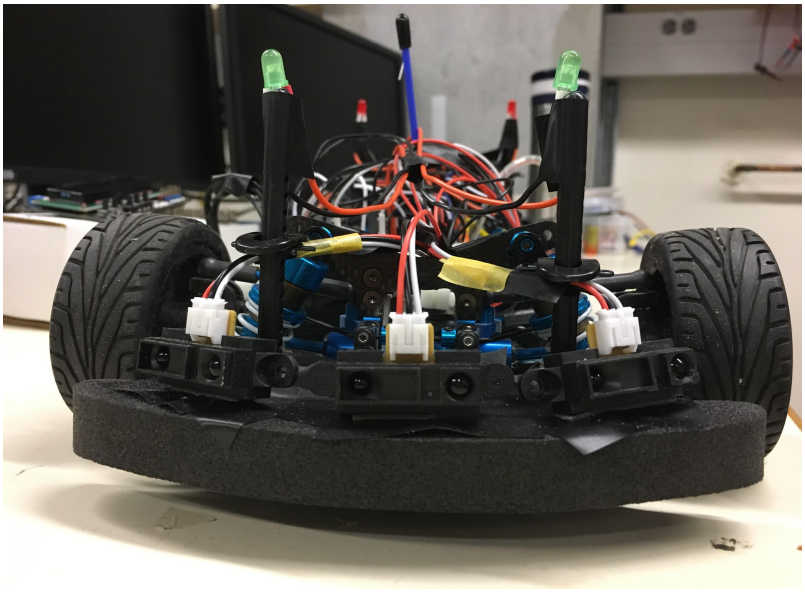
Mechanical Considerations

ESC needs to be carefully calibrated according to arduino duty cycle first. This is a tricky step because ESC will take in full throttle/reverse/neutral according to pre set time. All wires need to be secured(duct tape, zip ties) for maximum reliability. Arduino and sensors also need to be secured via double sided tape and duct tapes so that they will not move around during motions. Three IR sensors will be positioned forward. Intervals between three front IR sensors need to be very close for maximum accuracy. Battery packs for arduino and servo also need to be secured onto the chassis.

Whole Car Setup:



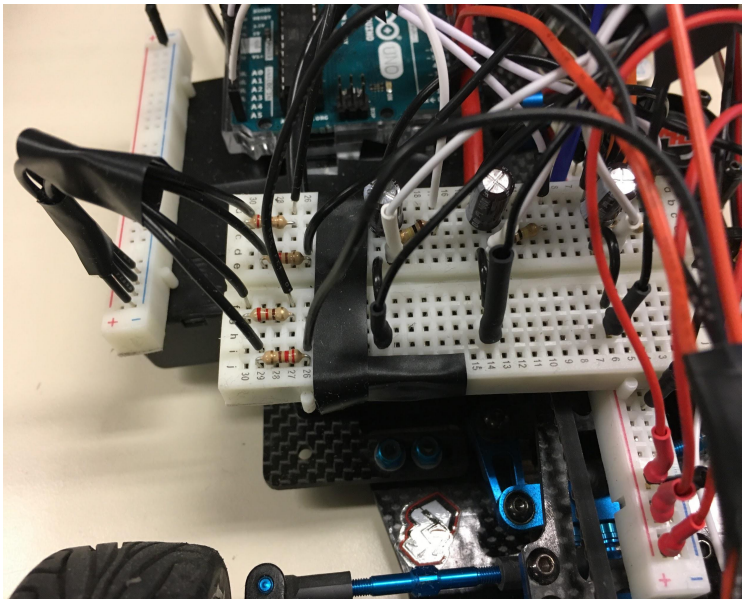
IR Sensor:

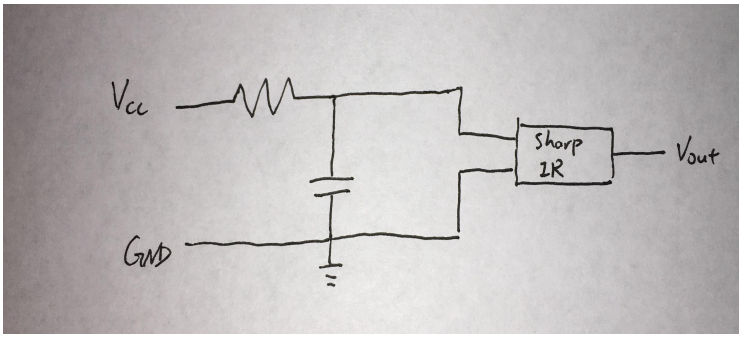


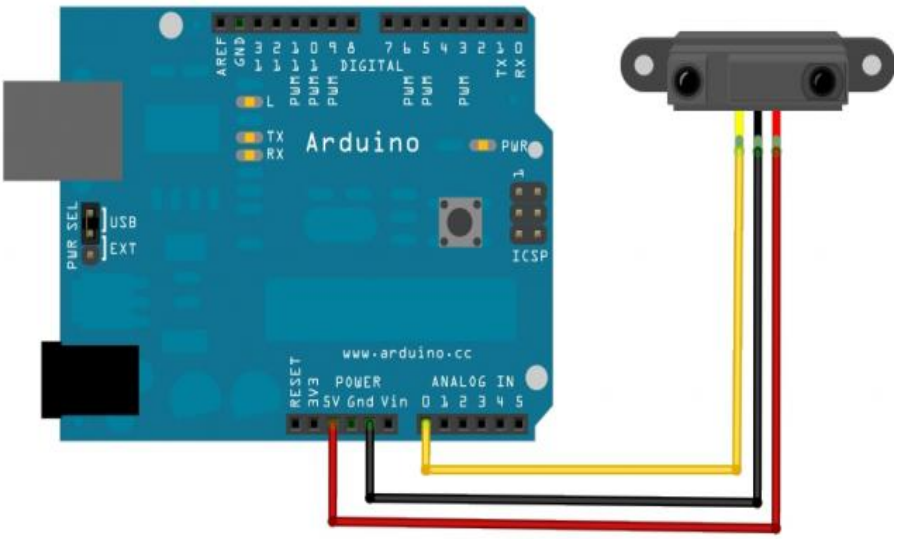
Electrical Considerations

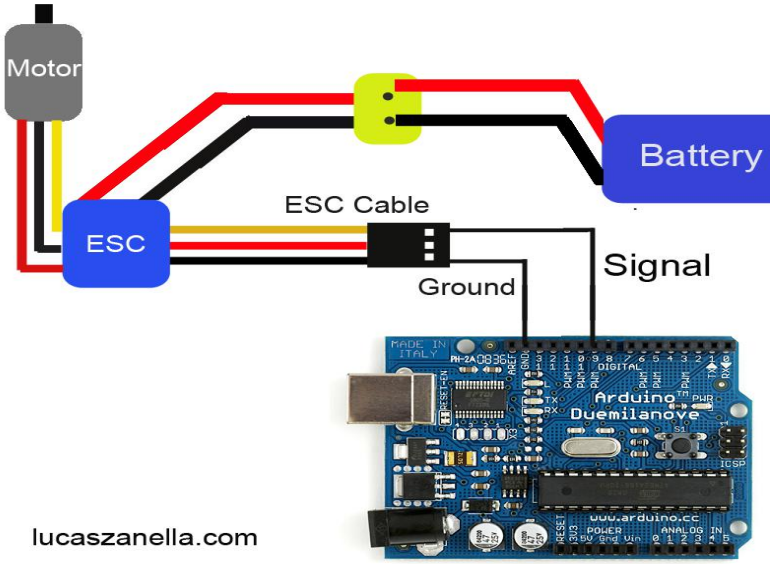
The distance sensor needs +5 V and ground, plus one signal connection. The servo needs a +5V, ground and a signal out. Since the parts are in motion, we will minimize the number of cables going to the platform. Everything needs to be powered by battery as this is a moving car. We have a battery box for the Arduino which will also power the servo. We have a separate battery for the DC motor as that takes 12V instead of 5V. Additionally, since Sharp IR sensors are very sensitive to change in supplied voltages, we used a separate 6V battery pack to power the servo, as it draws a lot current from arduino and causes instability. Moreover, IR sensors are hooked with corresponding low-pass filter (16Hz) to minimize any noise issue(particularly voltage spike).

Low-Pass Filter









Interface

We used the Arduino Uno, although other boards such as the Mega would also be suitable. During the debugging phase, we will use the Serial Monitor to track behavior and sensor feedbacks. Once operating satisfactorily, we will speed up the operation and comment out the Serial transmissions.

2 pins (pin5 and pin9) will be used to control the servo and ESC in the mechanical set.

We used pin A1,A2,A5 for IR sensors. We thought different analog pins would help with the noise, but that was not the problem.

We also used 4 digital pins to control 4 LED to act as right turn, left turn, and reverse.

Software

We will program in C, using the Arduino IDE. This application requires no manipulation of the ATMega chip beyond what the standard Arduino library provides. We will use the Servo Library to control the motor, but the other interfaces are as simple as analogRead(), analogWrite(). We will use the Servo library to control the motors on the RC car.
We will take the inputs and send an output signal according to what will be the best time to adjust the car. These timings will be acquired through trial and error during our tests.
We are using multiple sensors, so we will have to play around with the code quite a lot to get the car to do what we want it to do. It will contain mostly if() functions and a lot of else if() functions.

We created separate functions outside the loop like

```
void forward(){
        function
}
```

So we could call this function instead of writing it out every time we needed the car to move forward.

Testing

We tested the car a lot in the past few weeks. We had to find the right speed and distance in order for the car to turn without hitting the obstacle. We adjusted the positions of the sensors slightly upward so the ground would not be considered an obstacle. We can move the obstacles into random positions and adjust our program until the car avoids any potential collisions. We found that non-flat surface obstacles proved hard for the sensors to pick up a consistent value. Most of this project was adjusting the code so that the car navigated smoothly through the obstacles.

Safety

This system operates at low voltage, moderate speed, without any flame, sharp objects, poisonous gasses or chemicals, or other hazards that would require special consideration. The worst thing that is likely to happen is that the wires will get tangled or repeatedly be pulled out of the breadboard, but this does not constitute a safety concern. A RC car can't run anyone over.

*In ESC calibration sequence, if values were inputted, the car will automatically run at maximum speed, which is very dangerous and caused some trouble for us. Luckily, we were able to control the situation and assembled the car back into its original shape and no one was hurt.

Parts Required, and Reusability

We are using Tamiya TRF417 1/10 RC chassis as our platform, along with hobbywing ESC and a brushless motor. We are using a futaba digital servo for steering. We calibrated the servo to middle steering position and added/subtracted same value to reach left/right turning positions. We will control the D.C. Motor with an Electronic Speed Control (ESC). We need a few proximity detectors to detect the distance between some obstacle and the car. We are thinking of 3 in the front so the RC car will know when to turn based on the ratio of each sensor. If we need more sensors, then we will have to move onto the Arduino Mega since the Uno doesn't have enough analog pins.

Problems Occurred:

Very first problem we encountered is to figure out the center position of our servo. Since our hobby servo, unlike other servos, do not turn more than 120 degrees. We looked up its duty cycle spec and adjusted the mechanical parts on the chassis to center the car.

Another problem we met is with ESC. Since different ESCs take in setting signals based on different order, we had to look up the spec sheet and repeat the calibration sequence over and over again with different time delays until we successfully set Brake/Neutral/WOT(wide open throttle) as 0/90/180.

When we moved on to the IR sensors, we realized that its readings are not very stable. It produces a reasonable error. Therefore, we had to set a sample size(30 readings) and used its average. It smoothed out the readings by a lot, however, sudden change still occur occasionally. We then employed a set of low-pass filters for further optimization.

After we got the car running, turning and IR sensors reading, we found out in the test that the servo, combined with 3 IR sensors, draw too much power from the arduino board. The 9V battery pack powering the board runs out very fast. The power shortage also lead to incorrect readings from IR sensors(Each time servo turns, a spike in IR reading will occur, which leads to servo turning again). We then utilized a separate 6V battery pack mounted on the back of the car to power the servo.

Lastly, the car does not brake in time. We have utilized different methods(maximum braking for short period of time, lower braking threshold reading and reversing when car is still moving forward, however, due to ESC’s rather weak braking force and lengthy braking sequence, we were unable to achieve an optimal braking function.

Comments on Performance:

After we sorted out the problems listed above and secured all mechanical parts(boards, wires, sensors etc.), the car runs smoothly and will turn in time to avoid distinguishable objects. However, since the car sits particularly low, roughness on the road surface sometimes causes misreading in IR sensors. Moreover, since Arduino requires 30 sample readings every time it makes a decision, we cannot run the car too fast otherwise it will crash into obstacles. We have tested the car in various situations and it handled them well

Additional Pictures:

12V battery pack for ESC and motor + 9V battery for Arduino + 6V for the Servo: