Bluetooth Remote Controlled Car
CS 193: Design Project (4 Units)
Summer Study Abroad 2014
Final Report

Hector Dominguez
September 29th 2014

For my project for CS193, I wanted to extend my knowledge of embedded systems that I acquired from the introductory course CS120B. I also wanted to explore a new area of computer science and apply it into this project. Therefore, I wanted to do a Bluetooth Remote controlled Car which is controlled via an Android application that I created. I expanded my skills by using a Bluetooth module to send and receive signals to get more in depth experience using UART and sending PWN signals to DC Motors for increasing/ decreasing speeds or move the wheels left/right or forwards/reverse. However, I did not have any prior experience in Android development or using the Java and XML languages. These were the new technologies I experimented and applied into my CS193 project.

Technologies and components used in Project

- AVR Studio 6
- ATmega1284
- BreadBoard Wb -106
- 330Ω resistors
- L293D Motor controller
- 5V regulator
- HC-05 Bluetooth module
- 9V DC motors
- 9V Batteries
- LED lights
- Android SDK
- Eclipse
- Car Wheels and Frame

Features

- ✓ The car has the ability to move forward and reverse.
- ✓ Can go left or right while going forward/reverse.
- ✓ Can increase/decrease speeds while in use.
- ✓ 5-Second Turbo button
- ✓ Wheel alignment dedicated buttons
- ✓ Breaking lights
- ✓ Front light

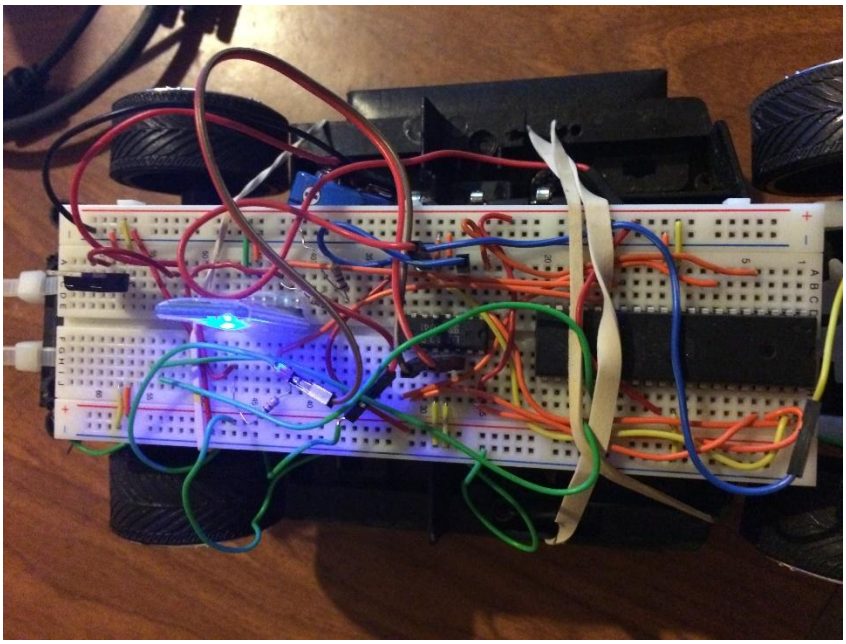
Project Overview

The Hardware

The RC Car was implemented using one ATmega 1280 microcontroller, one L293D Motor controller, one 5V regulator, one HC-05 Bluetooth module, two 9V DC motors, two 9V Batteries, some LED lights and resisters. The Bluetooth is connected to the microcontroller via USART and constantly waiting for a connection to be made. Once the Bluetooth receives a connection, it begins to wait until it receives some data. The microcontroller does not know when the Bluetooth module receives a connection or when it begins receiving data. The micro controller only knows when the Bluetooth module sends data to the microcontroller. Once the data is sent to the microcontroller, the data is then decoded and process.

For example if the data is meant for the DC motors, to move the car forward, for example, then the process consists of attaining the correct PWM signal, sending the signal to the DC Motor Controller to convert the 5V PWM signal into a 9V PWM signal and then send it to the appropriate DC Motor. The car features stopping lights when going in reverse and also front lights to imitate a real car. Therefore, if the data is meant for the car's light then the microcontroller decodes the data and sends it to the appropriate pins that manage the lights.

The microcontroller uses two pins (PB3 and PB6) to send two independent PWM signals to the two DC motors. However, since the microcontroller is run by 5 volts then this means it creates a PWM signal using 5V logic. The DC motors runs at 9 volts, therefore I needed to convert the 5V logic into 9V logic. To do this, I used a L293D motor IC. The outputs are then connected to the DC motors.



The Software:

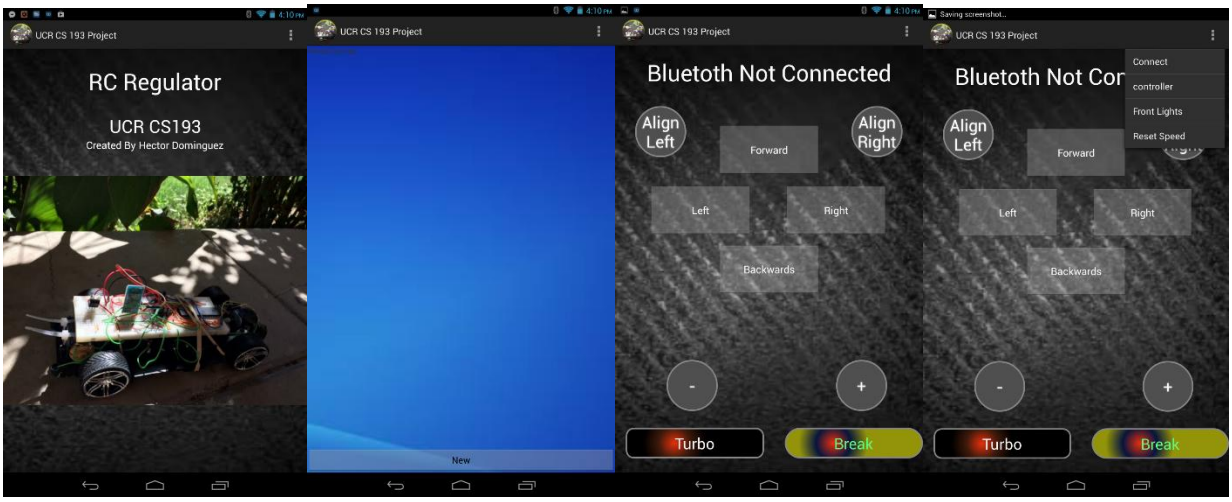
The Car logic was implemented using ATMEL Studio. During the process of implementing the logic, I did not know how to work with PWM signals. This was something I did not fully understand from my introductory CS120B course. Thus, I was not able to re-use the PWM code from one of the labs of CS120B. I started researching on the web regarding PWM and all of its sub topics and managed to acquire a better understanding of how it works. I created custom functions that turn on the PWM settings on pin PB3 and PB6 and also to calculate the duty cycle for both PWM pins to be sent to the DC motors.

Controlling the DC motors was another aspect of the project I had to learn. DC motors only have two inputs, like a speaker. Therefore, I had no idea of how it works and did not wanted to burn them out in the process. I then learned that it uses a Binary table to turn the DC motor and the speed varies according to the duty cycle provided.

Remote Controller:

The RC Car is controlled via an android application that I created that sends Bluetooth signals to the RC car. The application features turning on Bluetooth within the app if the Bluetooth is off on the device. Also, once a connection is established, the application displays a custom remote controller that enables the user to move the car forward/reverse, left/right, increase/decrease the speed, Break dedicated button, turn on front lights, reset speed to default speed, align front wheels, and a 5-second turbo button.

This was my first Android application and my first time using the Java and XML languages. Therefore, it took me a few days to learn the two languages syntax and the android API's.



Struggles:

During the process of this project, I encountered many unforeseen difficulties that mainly relied on material that I had not leaned or material I did not fully understood in previous courses. I did not fully understood PWM in CS120B since the code was given to us during lab. I only knew about the duty cycle is what controls the speed of the motors. Using the code given to us for the speakers in CS120B lab was not working for the motors because it was created specifically for the speakers. I did some research and re-learn more in depth this topic and became familiar of PWM in general and also about the port variables in the ATmega1284 microcontroller.

When dealing with the DC motors, i managed to run them using regular outputs from the microcontroller. However, this was limiting me of running them at that speed throughout the whole project. Since the microcontroller run at 5 volts that meant the motors weren't running at full speed neither. Once I figured this out, I started looking online on how to supply more external power to the DC motors and at the same time converting the 5V logic into 9V logic. Later, I found a DC motor IC that enables me to do this.

Writing the Android application was easier than I thought. However, one of the main struggles was getting the Bluetooth API to work. When trying to connect to other devices, the application kept crashing. I could not understand why it was crashing. And then I noticed that I was not assigning the Bluetooth device struct to the device I connected with. I followed many tutorials to understand the API. After many modifications to my code I managed to make the application sync with a Bluetooth device and send data.

Extra:

I did not add the LED matrix because I created the Android application. I felt like the LED matrix would have been useless and had no use for.

Conclusion:

Overall, the project was a success. I was able to extend my knowledge of embedded systems by learning how to use Bluetooth modules and more in depth knowledge using USART and PWM. I also obtained experience that will be helpful in the future such as Android development and small experience using the Java and XML environments.