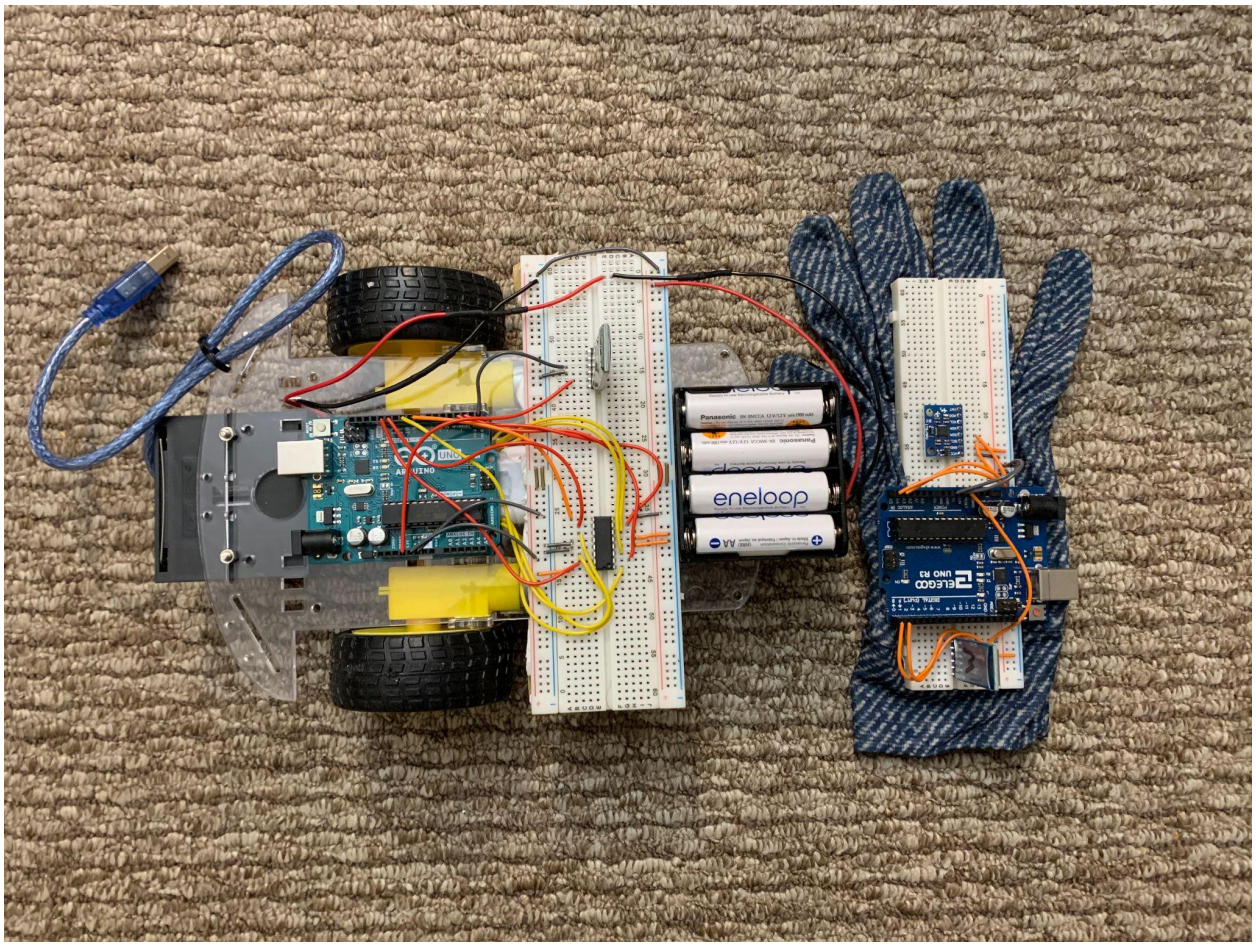
Hand Controlled RC Car

Anuj Patel

https://devpost.com/software/hand-controlled-rc-car



Final Project, Spring 2021

ESE350: Embedded Systems & Microcontroller Laboratory

University of Pennsylvania

Table of Contents

Abstract

For this project, I created an RC car that is controlled by the movements of your hand. By tilting your hand forwards, backwards, or side to side, you are able to wirelessly control the movement of a small RC car. This way of controlling an RC car makes the experience much more immersive and enjoyable. To accomplish this, I used an accelerometer which would track the position of your hand and send signals over bluetooth to tell the car which direction to move. The car works just as intended, but there are a few issues regarding controllability. First, the motors are low power 5V motors. These motors failed to provide enough torque as well as speed to the wheels. Another issue with these motors was that the speed was not easily tunable using PWM. Also, the project was not fully seen through as the controls were not mounted to a glove.

Motivation

Traditional RC car controls are boring and force you to bring a controller everywhere you go. This project aims to change that by having your hand be the controller. Although you will still need to have the glove controller with you, the controls will be much more intuitive and interactive.

Goals

A. Milestone

For the milestone, I expect to have the code working for the accelerometer that will be located in the glove. I will use the serial monitor to ensure that the code is working as well as to start mapping the ADC values to movement for the car. I also want to have the bluetooth communication working and again use the serial monitor to ensure that I am receiving the correct input depending on the position of the accelerometer.

B. Final Demo

For the final demonstration, I will have a fully working car that can be wirelessly controlled. My basic goals are to have the car move at a static speed forwards, backwards, left, and right. If I have the time, I will add variable speed.

Methodology

I first approached the problem by getting my controller to work properly. This was the main component of the car so I wanted to get it right. I worked with the accelerometer to make sure I had it finely tuned so that it would be easy to control the car. The next step was to get the bluetooth to work properly. We haven't worked with bluetooth in any of the labs so this was completely new to me. I had used bluetooth modules before using Arduino code, so I had a basic understanding of how they work. Once I had those two components working together, I finally connected both components of the project and got the motors to move depending on the position of the accelerometer.

Results

Overall, the project was a success. The car was able to be controlled using the movement of my hand. I initially set out with the goal of being able to complete an obstacle course with the car. The course I built included a small ramp to drive over as well as a figure-8 path to follow. The car was able to do both with a reasonable amount of precision. I was also able to use three class topics: timers, serial communication, and wireless communication.

The main issue I faced with the final product was controllability. This came down to 2 design choices. First was the choice of motors. The motors I ran on the car came with the chassis I received and were inexpensive 5V motors connected to small gearboxes to increase torque and reduce RPMs. The problem with these motors was that it was very difficult to control the speed of them precisely using PWM from the Arduino. The motors would fail to turn on below a 50% duty cycle with no load on the wheels. This made it extremely difficult to have precise movements with the car which led to a lot of oversteering when I was trying to turn the car. The second design choice that led to poor control was the chassis that I bought. This chassis was not designed very well for my application. I chose the two wheel design with the caster wheel in the back so that I could make 360 degree turns with no turn radius, however, due to the low quality bearing in the caster wheel, the wheel would not turn as freely as I needed it too and would often drag along the surface I was driving on which would throw the whole chassis off course. To try and remedy this, I taped the wheel to fix it in one position and just have it drag. This helped slightly, but it also limited the car to only drive on very smooth surfaces.

Conclusion

In conclusion, I was happy with the outcome of the project regardless of its shortcomings. I learned a lot about how the bluetooth module works and communicates. I also learned a lot about serial communication with the accelerometer. The controller design went very well and I was able to get accurate positions from the accelerometer. I gained a lot of knowledge on how to use external libraries with embedded programming and even how to modify those libraries to fit my needs. I had to change my approach a little bit when I began testing the car and realized that the motors were not strong enough for the car. I couldn't operate the car at the exact speeds I wanted and I also had to supply them with more power. A few things I could've done differently would be to buy better motors as well as create a continuous mapping for accelerometer position to motor speed instead of a discrete one. The obstacles I encountered that I did not expect were the poor performance of the motors and the chassis of the car. The next step would be to build a new chassis with better motors. With more time, I could laser cut my own chassis as well as research better quality components. I could also work on the code more to make the power delivery more seamless with a continuous mapping for the accelerometer.

References

I2C Basics: https://embedds.com/programming-avr-i2c-interface/

Appendix A

1. UART Library
    a. I used UART to communicate with the bluetooth modules. I used the exact library that we previously used for the labs, but I needed to add a receive function so that I could send as well as receive with bluetooth. To do this, I added a simple function to the library which checks the UCSR0A register to see if any data was received. Once data is received, it returns the value in the UDR0 register. This function returns one byte at a time and is casted to an unsigned char. Because of this, I needed a loop in my code that continued to call the receive function I added until a null character was found so that an entire string could be received.
2. I2C Library
    a. I used the I2C library to communicate with the accelerometer. To use this library, I first had to write to registers on the accelerometer to set up the

values that it would need to output. I did this by creating an initialization function and using the I2C_start and I2C_write functions contained in the library. I then used the I2C_Read_ACK function to get the raw value output from the accelerometer and did some simple calculations to these values so they stayed within a range that was usable for my application.