[Tran, Huynh]

Lab 4: Follow Me Remote Control Car

Richard Tran: 1924764
Huy Huynh: 1925597
10-June-2022
Assignment: ECE474 Lab 4

Project Idea

We will be designing an RC car with the purpose of item transportation. It will follow the user forward with the use of an ultrasonic sensor and IR remote to move left and right (skid steering) depending on the user input.

In order to meet the criteria, we will be making a list of materials that we will be using

- MEGA 2560 Controller Board
- L298N Motor Driver (New)
- 4 Motors (New)
- Ultrasonic Sensor HC-SR04
- IR Receiver Module
- Remote Control
- LEDs
- Wheels, plastic, etc (For the mechanical side of creating the car)

We will be performing high-speed operations by collecting data from the ultrasonic sensors that will constantly and rapidly collect and return data in order for us to determine where the human (the follow target) is. We will be making use of multiple new materials from the Elegoo starter kit and a motor kit.

This is an idea that can be translated into a bigger scale where we can create robots (or RC cars in this case) that can transport loads for us.

A user interface would be through the usage of the remote control where the user will be able to decide if they want the car to move right or left. This will also light up the corresponding LED to their respective sides.

Introduction

The lab will be two parts. The objective of the first part of the lab will be to use the FreeRTOS OS package for Arduino in order to schedule our tasks. We will also be importing the FFT library, which we will use in one of our tasks. The FreeRTOS OS package offers similar basic functions to our schedulers created in lab 3, like delay and halt to manage our tasks. In addition, FreeRTOS introduces a queue that can be used to share data between tasks with a first in first out structure.

[Tran, Huynh]

Next, we will be planning and making a creative project that includes any new material that we did not use in this lab. We will be using FreeRTOS to schedule our tasks as well as importing additional libraries that we need. We will have to satisfy the project criteria as mentioned in the Lab 4 document.

Methods and Techniques

Similar to previous labs, we organized our code by using separate files and using header filers to share necessary code between each file alongside using Doxygen to document our code as well as producing visual websites to read. We also used (pre-defined) #defines for all registers and bits we needed for the timers and compares values.

For the chassis of the RC car, we will be making use of SolidWorks to create a 3D model and 3D print it. This will be the foundation of the car that holds all of the materials used in this project.

We made sure to get all the different materials working before having specific tasks controlling what it will be doing. For the ultrasonic sensor, we made use of the built-in timer to generate a fast PWM signal that activates the ultrasonic sensor to send out sound waves every 20ms. Then we use a pin change interrupt to measure how long the output pulse of the ultrasonic sensor indicates how much time it takes the sound wave to travel. Afterward, we use an equation to convert the time to distance based on the speed of sound. For the remote control and the IR receiver module, we imported an Arduino library that handles detecting and converting IR signals from the remote control. The signal is decoded into a HEX representation of the buttons we are pressing on the remote. This will be used in order to determine if the car will skid to the right or left. For our motors, we made use of the L298N driver that allows us to use GPIO pins to control the motors. We looked up the L298N datasheet in order to understand which wire needs to turn on in order to generate the correct wheel spin direction. In a similar fashion, we used PinMode() and digitalWrite() in order to light up the LEDs.

In order to check what information that was being read, we made use of Serial print functions. For example, we used it on the ultrasonic sensor in order to see what information is being passed through. Also when checking on the wiring of the motors, we used our multimeters to check if the voltage and direction of the current are correct. When we upload our code and watch the car run, we are able to understand the mechanical functions of the car, such as turning and going straight. We are also able to test if our code is working as a whole.

[Tran, Huynh]

Experimental Results

Part 1.3:

To test the LEDs and speaker, we time the LED blinking and listen to the speaker. We observed that the speaker is playing correctly and stops after 3 times and the LED blinks at the correct timing. In order to test the computation time of the FFT, we print out time on the serial monitor. For a sample size of 64, 128, 256 we got a time of 24, 54, 118 ms respectively. We verify the times with online sources and it confirms our data.

Project:

We created 3 different tasks that consist of getting the distance from the closest object in front of the RC car, getting and deciphering the IR remote inputs, and determining the movement that the RC car would perform. All the tasks were created with the same priority because we want the task to go in order when we call xTaskCreate(). This allows us to get the information from the ultrasonic sensor and the remote inputs first in order to decide the movement of the car.

[Task 1: Distance]

We initialize the ultrasonic sensor with the usage of pinMode() to our defined pin 10. We created a queue with xQueueCreate() of size 10. We set the size to 10 in order to help against glitches that will be explained in Task 3: Movement. Since the ultrasonic sensor returns a time value, we used our equation to calculate the distance equivalent and sent it to the back of the created queue with 0 ticks because we want the information immediately.

[Taks 2: Remote Input]

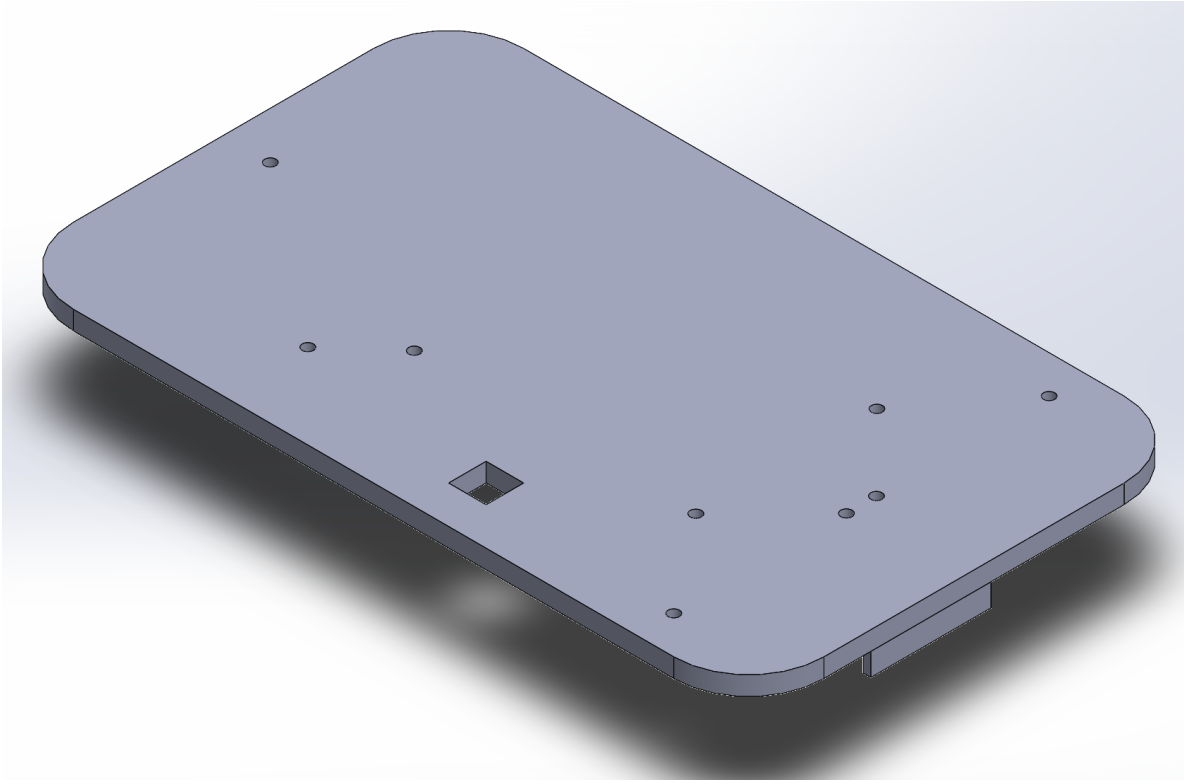
We decode the signals from a button press into HEX representations in order to store it into a variable that we can use as a condition for setting specific integers, “turnRight” and “turnLeft”, used to determine the movement. We will also be using digitalWrite() alongside vTaskDelay() in order to turn on a corresponding change of direction. When one of the buttons is pressed, their respective LED will stay on as long as the vTaskDelay(). This time frame will be the same as how long it takes the car to change direction. As well as change an integer representing turning left or right to true. If no button is pressed, it will turn off the LED and set the integers to false.

[Task 3: Movement]

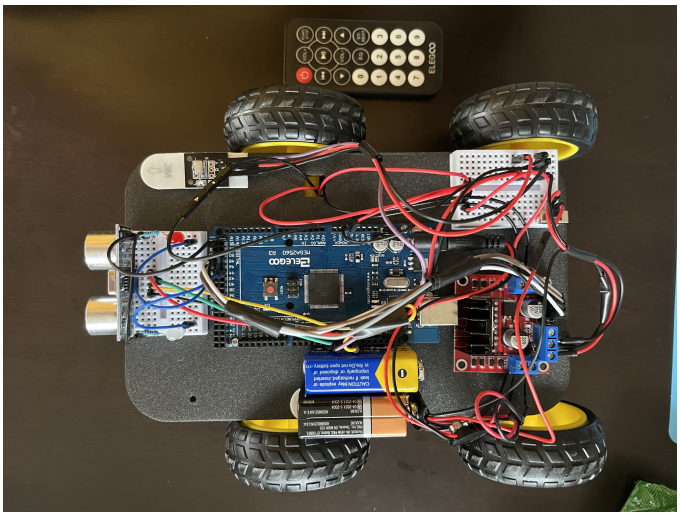
We will not allow interrupts for doing the main objective of this task with the usage of noInterrupts() and Interrupts(). We will be using xQueueReceive() from the queue we created in Task 1: Distance, in order to get the distance which we will be taking the average from the next 10 values, passed to help against small glitches (such as getting distance from the wall between our feet). If there is a possible movement, it will prompt the motors to spin in the correct direction with digitalWrite() on pins 46-49. If “turnRight” or “turnLeft” is true, it will turn its respective direction for 0.5 seconds and then set itself back to false. This time window of when the motor is turning will give us almost a 90 degree turn. Whenever this value is within 20 to 120

centimeters, this will prompt the car to forward, else the car will idle (motors not moving). Understanding the correct GPIO pins to control comes from the L298N datasheet.

.Chassis Assembly:



Hardware: Separate 9V batteries were used in order to power the MEGA 2560 board and the motor driver. The motor driver has a switch in order to turn on and off determining whether the RC car will drive or not. To not burn the LEDs, we used two resistors when wiring them. The ultrasonic sensor and the IR receiver are both wired up to the MEGA 2560 board and the 4 motors are wired up to the L298N motor driver.



Code Documentation (Doxygen)

For the first part of the lab we re-implement blinking LEDs and the speaker playing the theme like the previous lab, but now using the FreeRTOS API instead. We used the imported FFT library to compute the FFT on a random generated signal. The random signal is generated by getting two random integer numbers using the Arduino random function, casting them to double and dividing them to get a random double. In order to time the FFT computation, we used millis function to get the time by subtracting the time before and after FFT is called. TaskRT3p0 generates the random signal, initializes the queues and starts tasks TaskRT3p1 and TaskRT4. TaskRT3p1 and TaskRT4 suspend after running 5 times, print the time FFT takes each time.

The flow of our project is to first convert the data from the ultrasonic sensor if it is ready and add it to the queue. The queue has a max size of 10 data points. Next, detect if there is an IR signal and decode which button is pressed. If the correct button is pressed, turn on the appropriate turn left /right flags. Only accept a new button press every one second. The last step is to determine the next movement of the RC car based on the flags and ultrasonic sensor data. Turn right or left if the flag is turned on and if there is an object in a 20cm - 120cm range then go forward, otherwise stay still. The distance is computed by taking the average of the 10 data points in the queue added in the get distance function. See the zip file html folder for all the code documents of the project.

Overall Performance Summary

The performance of the ultrasonic sensor is accurate to plus or minus 5 cm when there is a flat object with a large surface area. However, legs are harder for the ultrasonic sensor to detect consistently during the demo. The RC car correctly moved toward an object if it was in range of the sensor (20cm - 120cm) and halted if out of range. Also, we were having issues with getting the RC car to go straight. We are unsure if it is due to the uneven power distribution to the motors or misalignment of the wheels. The RC car correctly responds to input from the IR remote and the LEDs accurately display when turning left or right. The RC car is able to turn left or right by about 90 degrees. Overall, we successfully accomplished our learning objectives of using FreeRTOS to schedule tasks that measure sensor data and based on the collected data control actuators.

Teamwork Breakdown

1. Hardware: Huy-50% Richard-50%
2. Software: Code in pair Huy-50% Richard-50%
3. Debugging/Testing: Huy-50% Richard-50%

[Tran, Huynh]

Discussion and Conclusion

Overall, implementing the new materials into our creative project was a fun and learning experience where we got more practice with the engineering process. As well as seeing our creation come to life was an exciting feeling. We were pretty proud about hearing that this was a very interesting project and something that is exceeding expectations from our TA when we did our demo.

The most challenging part of this project was finding a way to detect if the object is moving left or right. Initially, we planned on using two PIR sensors on the left and right sides of the car. But the PIR sensors were not accurate enough and it was detecting things moving when we did not want them to cause the car to turn right or left at the wrong times. We then tried using the sound sensor module where we made use of FFT to determine the amplitude of the frequency of the sound. We were planning on using specific tones of our voices to determine the change in direction but that proved to be more difficult when trying to get consistent tones produced from our voices. So we settled with the IR receiver and the remote in order to facilitate the right and left movement.

Something that we learned in addition to the learning objectives is further understanding the applications of signal processing, most specifically the FFT in this case. This got us thinking more about what else we can use FFT such as decoding speech in order to execute specific tasks we want it to. We also learned how remotes work in the real world through our implementation of our own with the IR receiver module. Another thing we learned about skid steering from our TA is that it would be better to make use of treads as our movement assistant instead of wheels as it would be able to better facilitate the skid motion.