CECS 347 Spring 2022 Project # 1

Robot Car with Motor Control

By

Abhishek Jasti, Anand Jasti

March 2, 2022

Design a robot car with GPIO, edge-triggered interrupts, hardware PWM, PLL, power supply circuit, DC motors, and motor drivers using TM4C123G Launchpad Microcontroller.

CECS 347 Project 1 Report

Introduction

Given the launchpad and robot car we were to build the robot car to go forwards, backwards, and make turns. This project was to be done by implementing hardware PWM to control the speed. As we cycle through speeds like 30%, 60%, 80%, and 90% of the duty cycles. Using one of the switches to control the speed and another switch to determine the direction of the car. The second switch will allow us to change the direction of our robot car. With one press the car could go backwards or vice versa forwards. Also, we implemented PLL to generate a 50MHz system clock. As well as use three LEDs on the launchpad to determine the status of the robot car. Red LED indicates that the robot car is not in motion, while green LED indicates that the robot car is moving forwards, and the blue LED indicates that the robot car is moving backwards.

Operation

In the beginning our robot car should start with no motion and display the red LED on the launchpad. Next, when we press switch 1 our robot car should go through four different speeds for forward direction.

Switch 1:

- First state of speed that the robot car should start with is 30% or whatever percent the car is able to move. While moving 30% or whatever percent the car can move in the forward direction our robot car should display the green LED on the launchpad. While moving 30% or whatever percent our car can move in the backward direction it should display the blue LED on the launchpad.

- The second state of speed that the robot car should have motion with is 60%. While moving 60% in the forward direction our car should display the green LED on the launchpad indicating that it is moving forward. While moving 60% in the backward direction our robot car should display the blue LED on the launchpad indicating that the car is moving backwards.

- The third state of speed that the robot car should have motion with is 80%. While moving 80% in the forward direction our car should display the green LED on the launchpad. While moving 80% in the backward direction our car should display the blue LED on the launchpad.

- The fourth and last state of speed that the robot car should have motion with is 98%. While moving 98% in the forward direction our car should display the green LED on the launchpad. While moving 98% in the backward direction our car should display the blue LED on the launchpad.

Switch 2:

- While our car is not in motion, and this switch is pressed where it changes the direction of our car, it should display the red LED on the launchpad indicating there is no motion.

- While our car is in motion, and this switch is pressed where it changes the direction of our car, it should display the LED that is represented by the car's direction. For example, if the car is moving forwards (displaying green LED on the launchpad) and switch 2 is pressed, the car should now move backwards with the same speed as it was going forwards and display the blue LED on the launchpad indicating that the car is now going backwards. This should also work if our situation was vice versa.

Link to Demonstration video:
https://drive.google.com/file/d/16ELuZUMC9JnFVsJwvho4KFYI6OIjSCOl/view?usp=sharing

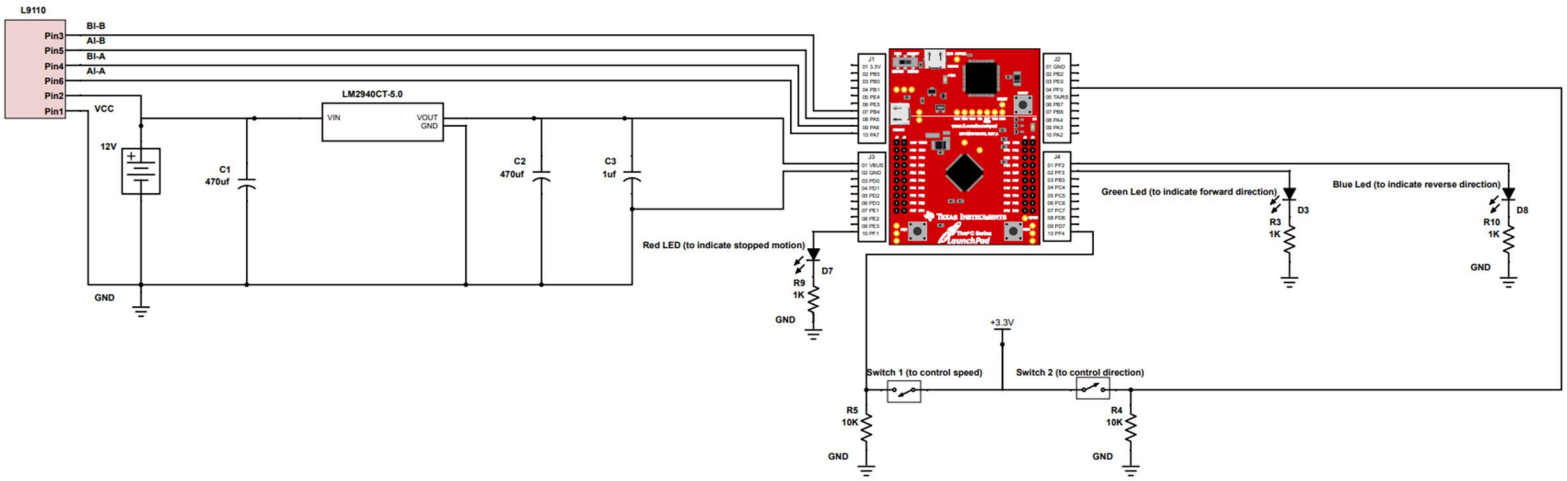
---

Theory

This project uses ARM Cortex TM4C123GH6PM Microcontroller, more specifically we used two of the six General-Purpose I/O ports (PF, PA). In port A, I used two pins for direction (PA4, PA5) and two pins for PWM signal (PA6, PA7). In port F, I used the onboard buttons pin 0 and pin 4 for change in speed and change in direction (more specifically defined in Operation and Hardware design). In this project we used GPIO pins, edge-triggered interrupts, hardware PWM, PLL, power supply circuits, DC motors, and motor drivers to implement the robot car. For the two switches we are using falling edge triggered interrupts. We also used PLL to change the system clock to 50MHz. Then we used hardware PWM on pins PA6 and PA7 to generate a 1KHz frequency to control the speed of the robot car. We also used GPIO pins PA4 and PA5 to control the direction (forward and backward) of the robot car.

Hardware design

Schematic:



Outputs:

Red LED	PF1
Blue LED	PF2
Green LED	PF3
Direction signal for Left motor (BI-B pin 3 on L9110)	PA4
Direction signal for Right motor (AI-B pin 5 on L9110)	PA5
Speed signal for Left motor (B-IA pin 4 on L9110)	PA6
Speed signal for Right motor (A-IA pin 6 on L9110)	PA7

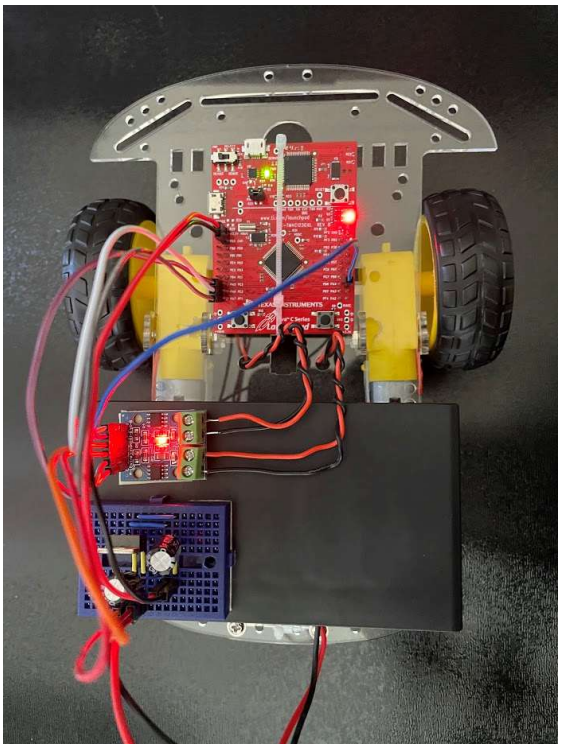
Inputs:

Switch 1	**PF4**
Switch 2	**PF0**

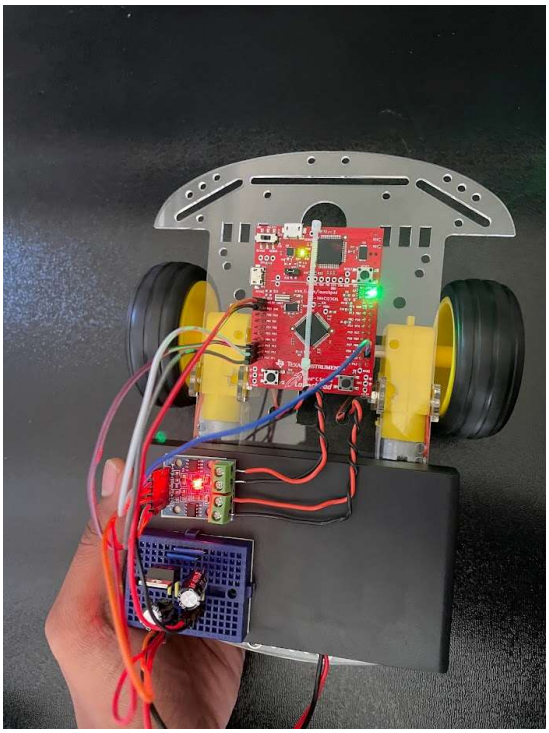
The left and right motors are being driven by L9110. The pins connected to the L9110 from our launchpad are listed above. The Red, Green, and Blue LEDs are on-board LEDs provided by the launchpad.

Pictures of Hardware System:

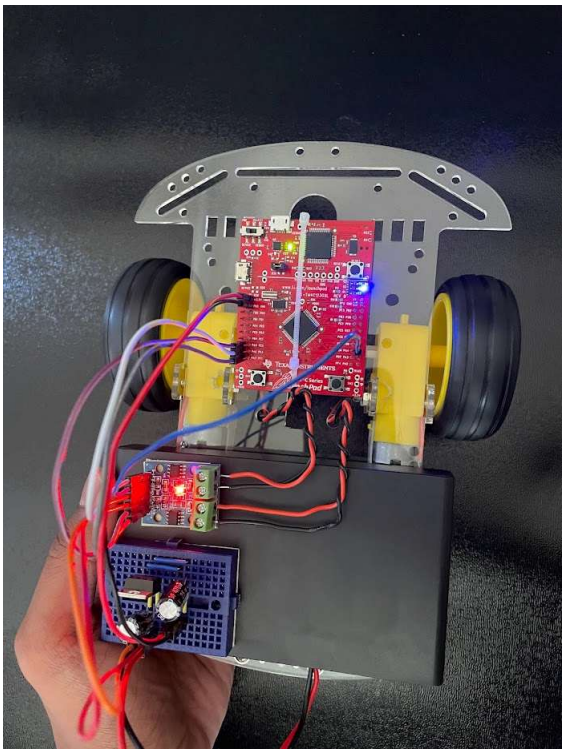
Stopped:



Forward Direction:



Reverse Direction:



Software design

For this robot car, we used PLL to change the system clock speed to 50MHz. After initializing the PLL we then initialized port F pins 1, 2 and 3 to be output using the on-board LEDs. We also initialized port F pins 0 and 4 to be inputs with falling edge triggered interrupts using the on-board push buttons. We also implemented a delay function that busy waits 20ms for debouncing. After initializing port F, we then initialized port A pins 6 and 7 to generate a PWM signal. Then we also initialized port A pins 4 and 5 to be output to control the direction of the robot car.

Conclusion

Implementing this project with the robot car was not as hard as we thought it would be. Building the robot car took the longest time in this project. Creating a power source using a small breadboard was a little tedious trying to fit everything on the breadboard but it was worth the amount of space we saved. We had the most trouble with soldering the wires to the motor, the soldering machine was not the best quality item we bought but we made it work. Fitting everything on the car was not the smartest idea, especially with a huge battery pack that takes eight AA batteries. We decided to use double sided tape to stick everything on the car instead of using zip ties or any other attaching item. Coding of this project for the robot car was simple. With the help of our previous labs using PWM and PLL helped us tremendously with coding this project for our robot car. Overall, this project didn't take as much time as I thought it would take. This project was very helpful to understand and to review the topics of GPIO, edge-triggered interrupts, hardware PWM, power supply circuits, DC Motors, and motor drivers.