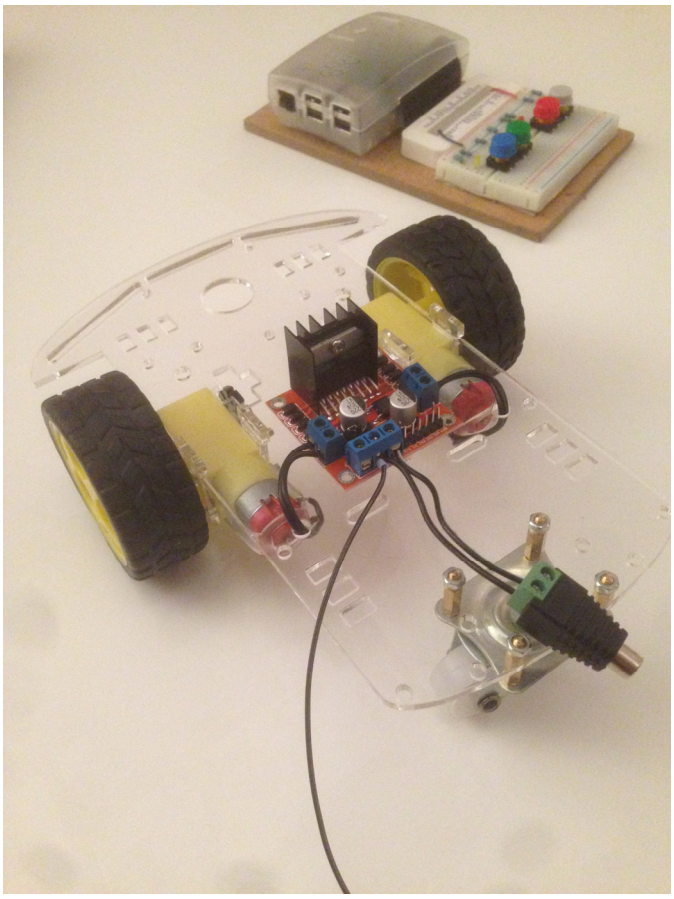
Lab 4: Develop a Simple Car

Farhang Nemati, April 2016



In this lab you will develop a simple car. You will practice towards development of a complex real-time embedded application. You can reuse the concepts, code and practices that you have achieved in previous labs whenever it is possible.

Controlling Motors

A Motor Driver Board (MDB) is installed on the car platform that can control two motors independently (Figure 1). The MDB has an input for power supply (5V or 12V, and GND – in Figure 1, the blue part in the front), 2 parts where each of them controls one motor (2 blue parts on the sides).

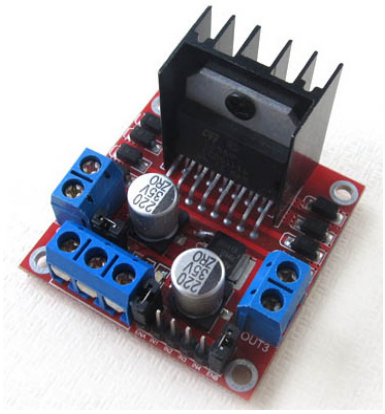


Figure 1

The motor driver has an input port which consists of 6 digital pins; 3 pins for each motor (Figure 2). The 2 pins at both ends (the left most and right most pins) are for controlling the speed of their corresponding motors. These pins have to be connected to an output GPIO pin controlled by PWM. Next to each PWM pin there are 2 pins that are used for indicating the direction of their corresponding motor. To run a motor in a direction one of its 2 direction pins has to be high and the other one has to be low. The motor runs in different directions depending on which of its direction pins has high value.

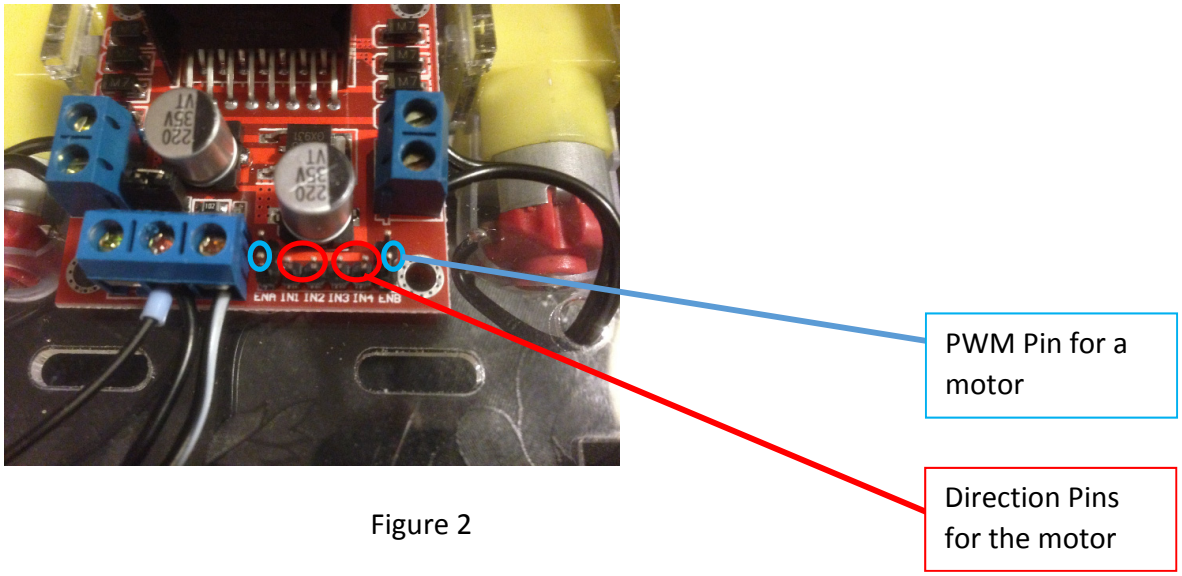


Figure 2

A. Controlling Motors

In this part you will write a program that controls a motor with different speeds and directions using your Raspberry Pi and the PWM function that you implemented in Lab 2. Connect the PWM pin of each motor to a GPIO pin that is controlled by pwm. Write a function that takes a motor (motor1 or motor2), a speed level (0 to 100), and the direction (forward or backward) as parameters. When the function is called it will run the given motor with the given speed level and in the given direction. As a hint, define a motor object (e.g., as a *struct*) which holds its pwm pin and direction pins in its corresponding data members.

B. Implement the Car

In this section you will write a program which will control the car using the inputs given by the buttons on the breadboard. Ask the lab assistant for mounting the required buttons and LEDs as well as for connecting the car platform to the Raspberry Pi. Observe the way the assistant does the setup so that you can do it yourself later if needed.

There are 4 buttons (blue, green, red, and white) and 4 LEDs (1 red, 1 green, and 2 yellow) on the breadboard (Figure 3). The buttons are used to input different commands and the LEDs will show the current status of the car.

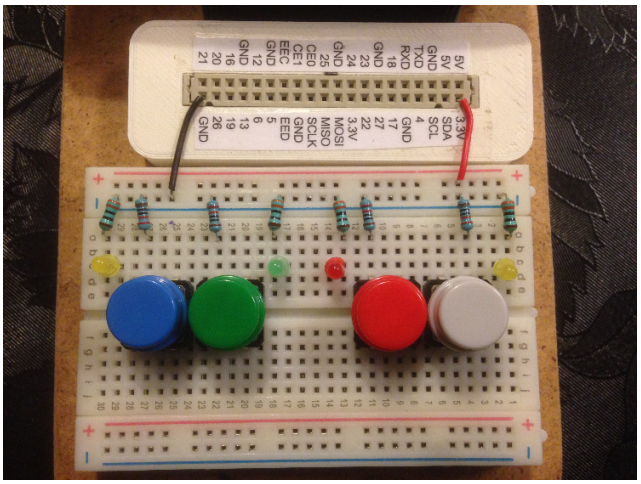


Figure 3

Your implementation has to satisfy the following requirements:

1. The commands have to be buffered. This means if a sequence of commands are entered while the car is executing another command, the new commands have to be added to a queue. The car should then pick the commands from the queue and execute them.
2. When the program starts, the green LED has to shine in breathing mode (reuse the breathing LED logic from Lab 2). The other three LEDs have to be off. This state is neutral gear.
3. The car has 4 speed levels (4 gears) and backward gear. The gear shift will be done by the green button. When the green button is pressed once the car (both motors) has to go with speed level 25 (25% of full speed – grear1), when the green button is pressed again the motors have to increase their speed to 50% (gear 2), if it is pressed for 3rd time the speed will be increased to 75% (gear3), when the button is

pressed for 4th time the car should run with full speed (top gear – gear4), and when the button is pressed for 5th time the car goes to the neutral gear and stops.

At any stage if the green button is pressed and it is hold down for at least 3 seconds, the car has go backward. Next time the button is pressed the car should go to neutral gear.

When the car is in gear1 (25% speed) the green LED should blink 2 times per second. In gear2 the green LED should blink 4 times per second, in grear3 10 times per second and in gear4 20 times per second. When the car goes backward the LED should be off and when the car is in the neutral gear again the green LED will go back to breathing mode.

4. When one of the buttons on the sides (blue or white) is pressed it means the car has to turn left or right. When the car gets the command of turning right or left it will stop the left or right motor (to turn right or left respectively). The corresponding yellow LED has to blink 2 times per second. At this state if a gear shift command is received (the green button is pressed) the car starts from gear1.
5. The red button sends an emergency stop command. This command should have the highest priority in the command queue and the car should stop immediately (no later than 100ms). In this state the red LED has to start blinking. In this case all existing commands should be deleted from the command queue. The car stays in this state until a new command is entered. In this state if red button is pressed again the car should turn off meaning that the program will do clean ups (destroy the GPIO pins etc.) and exit.

Examination

To pass the lab hand in a short report where you briefly explain your design in a high abstraction level (with text and/or diagrams) and your code (only in Blackboard!). You also need to demonstrate the program for the lab assistant. The lab assistant may ask you questions related to lab task.