Next we need to extend the robot built in Example 12.2. First we build two motor drivers and connect one to each wheel, as shown in Figure 14.9. There will be two PWM outputs: PA6 controls the right motor attached to the right wheel, and PA5 controls the left motor attached to the left wheel. The motors are classified as **actuators** because they exert force on the world. Similar to Example 12.2 we will write software to create two PWM outputs so we can independently adjust power to each motor. If the friction is constant, the resistance of the motor, *R*, will be fixed and the power is

$$Power = (8.4^2/R)*H/(H+L)$$

When creating PWM, the period (H+L) is fixed and the duty cycle is varied by changing H. So we see the robot controller changes H, it has a linear effect on delivered power to the motor.

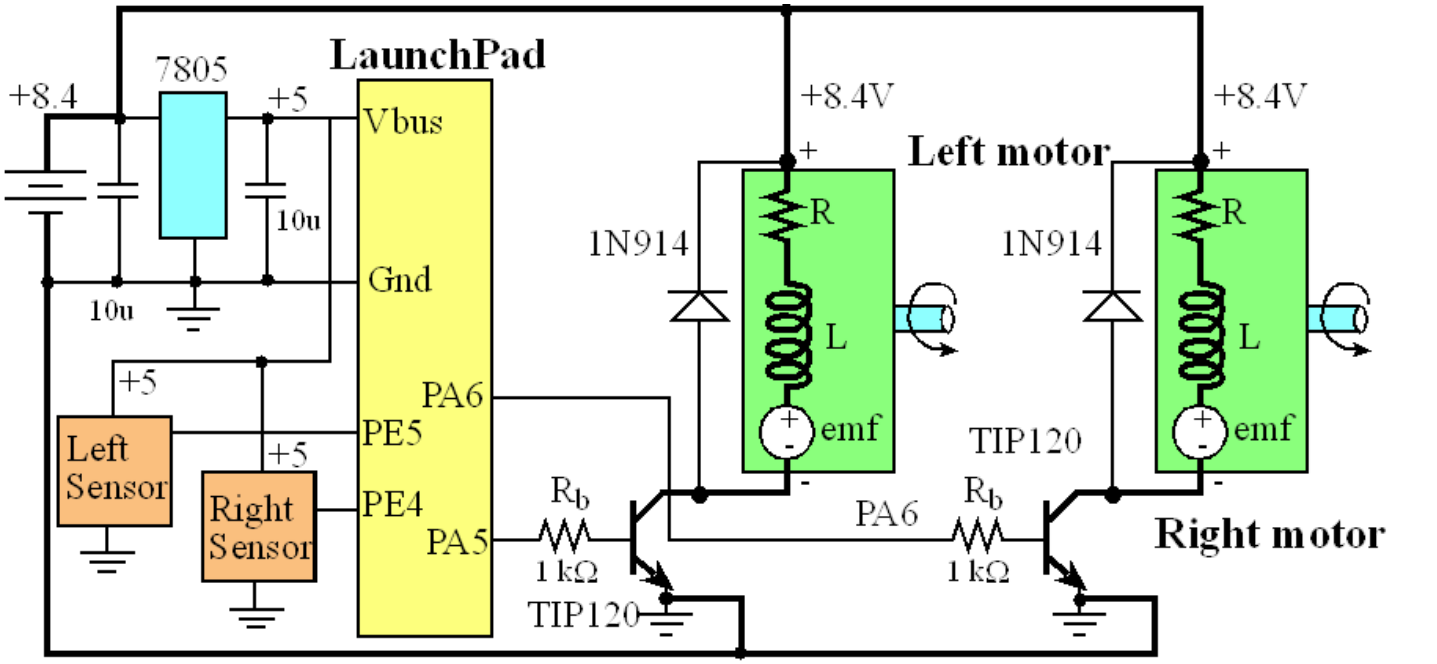

Figure 14.9. Circuit diagram of the robot car. One motor is wire reversed from the other, because to move forward one motor must spin clockwise while the other spins counterclockwise.

The currents can range from 500mA to 1 A, so TIP120 Darlington transistors are used, because they can sink up to 3 A see

data sheet (http://users.ece.utexas.edu/%7Evalvano/Datasheets/TIP120.pdf). Notice the dark black lines in Figure 14.9; these lines signify the paths of these large currents. Notice also the currents do not pass into or out of the LaunchPad. Figure 14.10 shows the robot car. The two IR sensors are positioned in the front at about 45 degrees.

The base current from the microcontroller is on the order of a few mA. The TIP120 transistor amplifies this base current by up to 1000 to be able to sink up to 3A from collector to emitter.

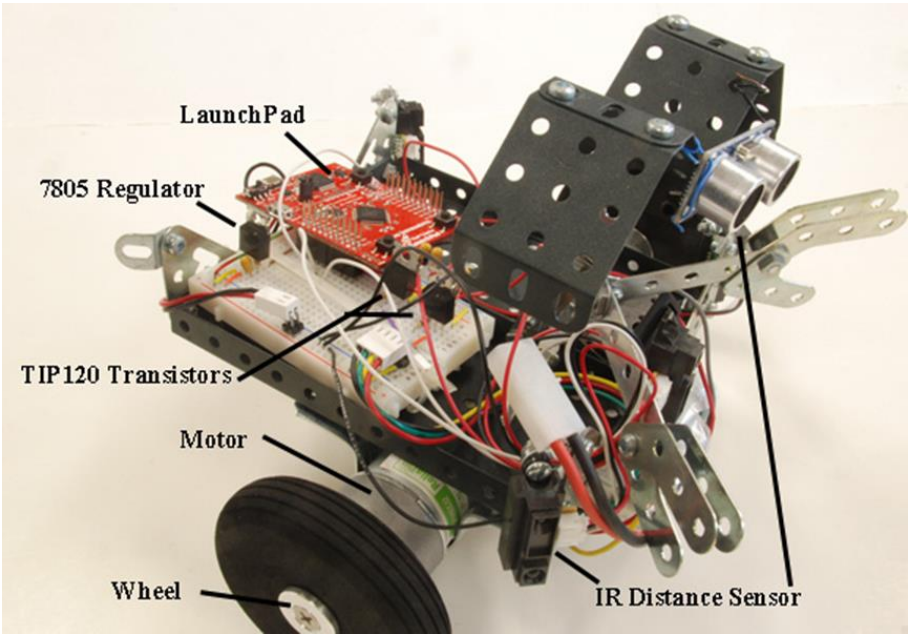

Figure 14.10. Photo of the robot car.

Figure 14.11 illustrates the feedback loop of the control system. The state variables are D_{left} and D_{right}. The two sensors create voltages that depend on these two state variables. The ADC samples these two voltages, and software calculates the estimates **`Dleft`** and **`Dright`**. **`Error`** is the difference between **`Dleft`** and **`Dright`**. The right motor is powered with a constant duty cycle of 40%, while the duty cycle of the left motor is adjusted in an attempt to drive down the middle of the road. We will constrain the duty cycle of the left motor to between 30% and 50%, so it doesn't over compensate and spin in circles. If the robot is closer to the left wall (**`Dleft`** < **`Dright`**) the error will be negative and more power will be applied to the left motor, turning it right. Conversely, if the robot is closer to the right wall (**`Dleft`** > **`Dright`**) the error will be positive and less power will be applied to the left motor, turning it left. Once the robot is in the middle of the road, error will be zero, and power will not be changed. This control algorithm can be written as a set of simple equations. The number "200" is the controller gain and is found by trial and error once the robot is placed on the road. If it is slow to react, then we increase gain. If it reacts too quickly, we decrease the gain.

```
Error = Dleft - Dright

LeftH = LeftH – 200*Error;

if(LeftH < 30*800) LeftH=30*800;  // 30% min

if(LeftH > 50*800) LeftH=50*800;  // 50% max
```

```
LeftL = 80000 - LeftH;              // constant period
```

Observation: In the field of control systems, a popular approach is called PID control, which stands for proportional integral derivative. The above simple algorithm actually implements the integral term of a PID controller. Furthermore, the two **if** statements in the control software implement a feature called **anti-reset windup**.

These controller equations are executed in the SysTick ISR so the controller runs at a periodic rate.

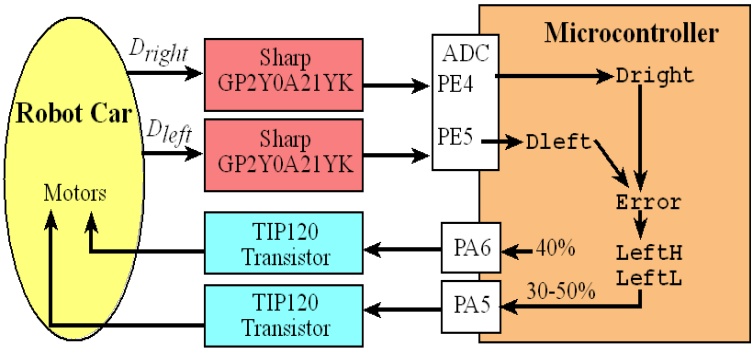

Figure 14.11. Block diagram of the closed loop used in the robot car.

Help

My real-time operating system class builds automonous robots using two microcontrollers and lots of sensors. watch as YouTube video of my class competition (https://www.youtube.com/watch?v=GKctvlvprAQ&feature=youtu.be)

Details about microcontroller-based control systems can be found in Chapter 10 of Embedded Systems: Real-Time Operating Systems for ARM® Cortex-M Microcontrollers, 2013, ISBN: 978-1466468863.