Lab 24 Line Tracking Robot

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, published by Brooks-Cole, copyright © 2000.

Goals
- Interface a DC drive motor using an on/off MOSFET driver circuit,
- Interface a DC steering solenoid using a plus/off/minus H-bridge circuit,
- Implement variable duty cycle squarewave with output compare interrupts,
- Measure period with input capture interrupts,
- Develop a line-tracking algorithm using a finite state machine.

Review
- Sharp GP2L01 data sheets,
- Valvano Section 2.4.3 on Moore finite state machines implemented with linked data structures,
- Valvano Section 2.7 on device drivers,
- Valvano Chapter 6 about input capture, period measurement, and pulse-width modulation,
- Valvano Sections 8.2.1, 8.4, and 8.5 on LEDs, transistors, and MOSFETs,
- Valvano Chapter 13 about incremental, PID, and fuzzy logic control systems.

Starter files
- MOORE12.*, OC3TEST.C, OC3.C, OC3.H

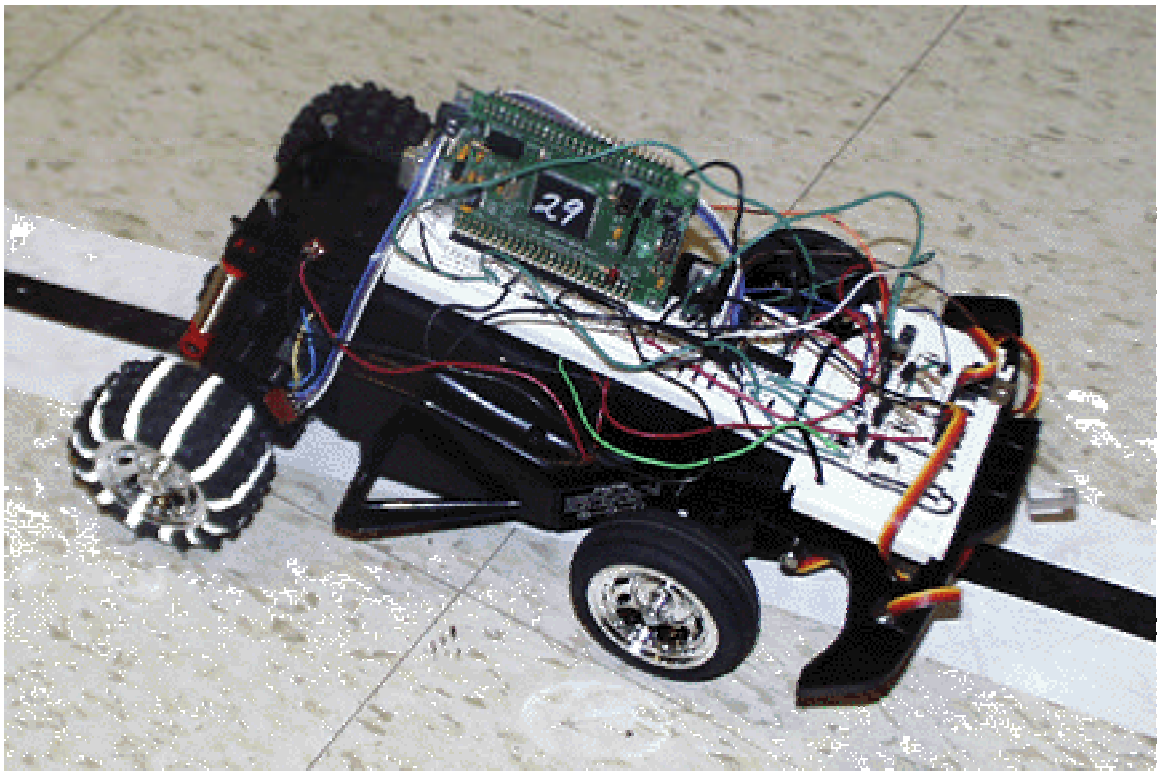

Figure 24.1. Completed Robot System.

Building the Track

The overall goal is to design a robot that can follow a line. The track is created on a white (reflecting) surface. White shiny tape can be used. A black line is created with electrical tape (optically absorbing.) You should make the turning ratio smooth enough, so your robot won't have to back up while following the line. The contrast between the white background and the black line should be large enough so that the line sensor can easily determine if the robot is over white or black. You could configure one track for a timed race, or you could make two closed-loop tracks for head-to-head racing. Your robot will have two optical sensors placed close together on the bottom. Each sensor will return a boolean signal depending on whether it senses black or white. If both sensors return black, then the robot is properly positioned over the track. When the left sensor observes a white signal (and the right sensor still sees black) then the software knows the robot is veering off to the left. Similarly, when the right sensor observes a white signal then the robot is veering off to the right. It will be important to place the two sensors close

enough so that both will be black when the robot is on the track, but far enough apart to prevent optical crosstalk. Crosstalk occurs when the infrared transmission of one sensor is sensed by the receiver of the other sensor.

Building the Robot

The robot are can be easily modified from a standard radio frequency controller toy car. Even though the robot can move backwards, this feature will not be used. The steps in building the robot include:

1) remove the RF receiver circuit
2) attach wires to the battery power pack, the drive motor(s), and the steering motor(s)
3) add a front bumper switch
4) create a wheel tachometer using a Sharp GP2L01 sensor, and marking a tire with white stripes
5) provide a line tracker sensor using two Sharp GP2L01 sensors on the bottom of the robot
6) drill holes in the plastic body so that screws can be used to hold the prototyping board(s)

Connector scheme we used

green/blue stranded wires are the rear motor, 40 Ω, goes forward and reverse
green/blue solid wires are front steering, 40 Ω, turns left and right
red is +10 or +12 volts depending of if you use 8 NiCad or 8 alkaline batteries
orange is +5 or +6 volts depending of if you use 4 NiCad or 4 alkaline batteries
black is ground (switched through the on/off switch on the robot)
tachometer is a 4 wire ribbon cable to the Sharp GP2L01 positioned near the back wheel

pin 1 white	infrared LED cathode
pin 2 gray	infrared LED anode
pin 3 purple	infrared-sensitive transistor emitter
pin 4 blue	infrared-sensitive transistor collector

the line tracking sensor also uses two 4 wire ribbon cables to the two Sharp GP2L01s on the bottom

pin 1 yellow	infrared LED cathode
pin 2 orange	infrared LED anode
pin 3 red	infrared-sensitive transistor emitter
pin 4 brown	infrared-sensitive transistor collector

The 6812 microcomputer requires +5±5% volts to operate. During debugging, you should use the regular method to power the Adapt812. Use a voltmeter to verify the Adapt812 can supply the 125 mA (5V/40Ω) current required to spin the motor. If the Adapt812 regulator gets hot or the voltage drops below 4.8 V, then use an external power supply to supply the current to the motor. Unloaded, the motor spins at about 1 rotation in 70 ms (850 rpm). The wheel circumference is 22 cm, resulting in a speed of about 300 cm/sec. There are five white marks on the wheel. Divide the wheel circumference by the number of stripes to determine the distance traveled per tachometer pulse. You could use input capture to interface the tachometer.

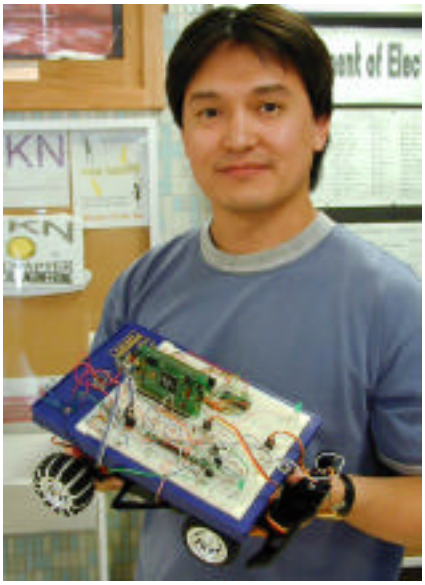

Figure 24.2. Award-winning Line-Tracking Robot.

Although stranded wires provide better stability for cables, solid wires could be used because they are easy to insert into a prototyping board. One approach to using stranded wire with prototyping boards is to

1) strip the insulation off the end of the stranded wire which will be inserted into the prototyping board,
2) twist tightly the stranded wire into a straight cylinder shape,
3) melt a little solder into the stranded wire end to give it mechanical strength.

Controlling Motor Speed

The interface developed in this section will allow the software to control the motor speed using pulse-width modulation. One possible interface is illustrated in the following figure. Notice the Adapt812 and motor circuits have the same power and ground connections, so please don't forget the diodes. Although the IRF540 MOSFET gate does not require a lot of current to maintain its present state, large currents are required to switch it from off to on and from on to off. To switch the MOSFET on, a high voltage (V_{OH} of the 74HC14) and a positive current (I_{OH} of the 74HC14) is applied to pin 1. To switch the MOSFET off, a low voltage (V_{OL} of the 74HC14) and a negative current (I_{OL} of the 74HC14) is applied to pin 1. Do not use an open collector gate like the 7405 in place of the 74HC14 because it had no I_{OH}. You could design a different interface using a darlington transistor like the TIP120.

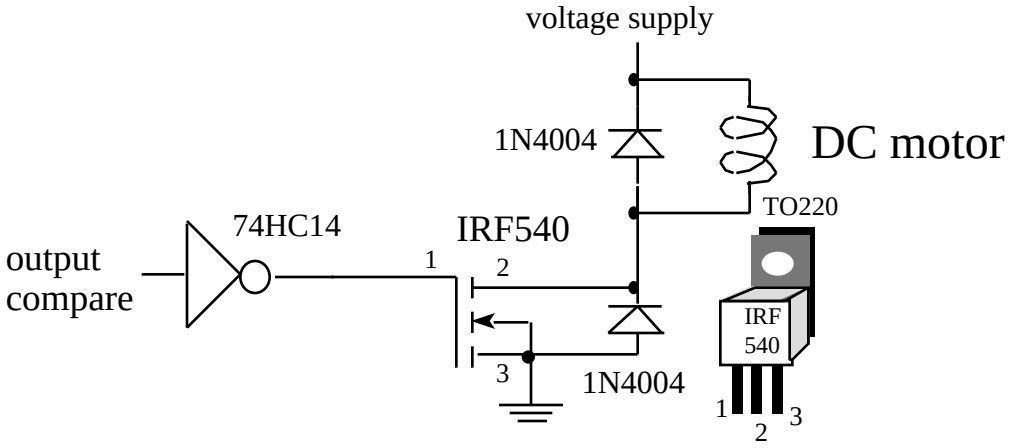

Figure 24.3. Typical motor control circuit.

The output compare interrupts will continuously maintain a squarewave with variable period and variable duty cycle. You should observe each signal on a scope. You will control the motor speed by varying the duty cycle. The period of the squarewave should be fixed. Selecting the squarewave period that is a tradeoff between motor smoothness and duty cycle precision. In general, the period must be much shorter than the motor time-constant, so for a fixed duty cycle the motor spins at one constant speed. If the period is too large, the motor will speed up and slow down during each phase of the squarewave. On the other hand, longer periods provide higher precision (more possible duty cycles) and lower software overhead (fewer interrupts.)

Measuring the motor speed

When 5 V is applied to the motor, it spins at about one rotation in 70 ms (850 rpm) when the rear wheels are off the ground. The wheel circumference is about 22 cm, resulting in a speed of about 300 cm/sec. When the robot is on the ground, it moves much slower. Unfortunately, there is no simple relationship between the squarewave duty cycle and resulting robot speed, because battery voltage and friction are unknown. If we wish to move the robot at the desired speed, a closed-loop control system with tachometer is required. If there are five white marks on the wheel, then the robot speed in cm/sec is 22•f/5, where f is the frequency of the tachometer in cycles/sec. Each Sharp GP2L01 has a transmission diode that illuminates the track with infrared light and an infrared sensitive transistor that detects reflected light. Refer to the Sharp GP2L01 data sheet for circuit suggestions and important parameters like suggested input current (I_F), typical forward voltage (V_F), optimal distance from sensor to object, and frequency response.

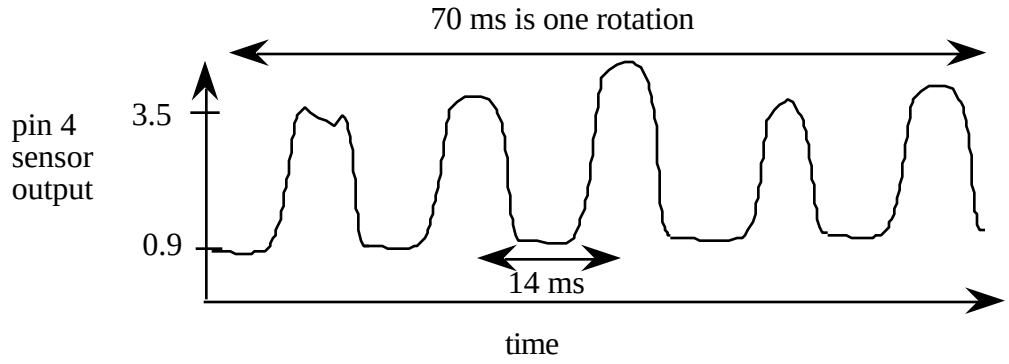

Figure 24.4. Typical tachometer output from a Sharp GP2L01.

Using a current-limiting resistor, you apply continuous power to the illumination LED. This resistance is chosen in the same manner as other LED interface, i.e., using I_F V_F diode parameters. Be careful not to pass too much current into the illumination diode. A pull-up resistor on the Sharp GP2L01 receiver collector (emitter is grounded) results in a voltage that is a function of the reflectivity (white versus black) of the surface. The above figure shows a typical sensor collector output. The output voltage levels depend on the current applied to the infrared transmitter, the reflectivity of the surface, the distance to the surface, and the pull-up resistor connected to the output.

There are two approaches to interfacing the Sharp GP2L01 sensor. You could design an analog to digital interface that converts the tachometer signal into a clean digital squarewave. Hysteresis must be added to prevent false triggers. The digital signal could then be connected to an input capture port, the software could measure period, and then it could calculate speed from the period measurements. If you were to measure frequency directly, you will find either it takes too long to get a good measurement, or the speed resolution will be unacceptable.

The second approach would be to connect the raw sensor signal to an ADC port. A high speed DAS could sample the sensor signal and detect presence of pulses. Hysteresis could be implemented in software using two thresholds. For example the signal must go above 2.5 V then go below 1.5 V to be counted as a new pulse. The time (TCNT value) in between pulses would be the period of the wave. The software would then calculate speed from the period measurements.

Interfacing the line tracking sensors

The two optical sensors located on the bottom of the robot provide position information. The two hardware interfaces are similar to the tachometer interface described above. Just like the tachometer, there are two approaches to interfacing the line tracking sensors. The first approach is to design two analog to digital interfaces. Depending on the high-level line-tracking algorithm, hysteresis may or may not be needed. The two digital signals could then be connected to any input port, and the software reads the input port to determine the robot's position relative to the line. Again, depending on your high-level line-tracking algorithm, you may wish to generate interrupts on the rise and fall of these two signals.

The second approach uses 2 channels of the ADC port. The software could sample the two sensor signals to determine the robot's position relative to the line. If needed, hysteresis could be implemented in software using two thresholds.

Interfacing the steering solenoids

There are three states for the steering mechanism. When no current is applied to the steering solenoid, the robot wheels will point straight. When current is applied in one direction, then the wheels will point left. When current is applied in the other direction, then the wheels will point right. The speed of the robot and the duration of the applied current will determine the magnitude of the change in robot direction. An H-bridge is an effective approach to interfacing the steering solenoid. The H-bridge employs four transistors. The two transistors T1 T2 source current from the power supply, and the two transistors T3 T4 sink current to ground. To steer left, you activate T1 and T4. To steer right, you activate T2 and T3. To go straight, all four transistors are deactivated. Be very careful, to prevent a direct path from power to ground. In other words, regardless of the software commands (i.e., during a reset, while downloading new software, after a software crash), T1 and T3 should never be simultaneously active. Similarly, T2 and T4 should never be simultaneously active. The diodes are extremely important because they block the back EMF caused by the dI/dt across the inductance of the solenoid coil.

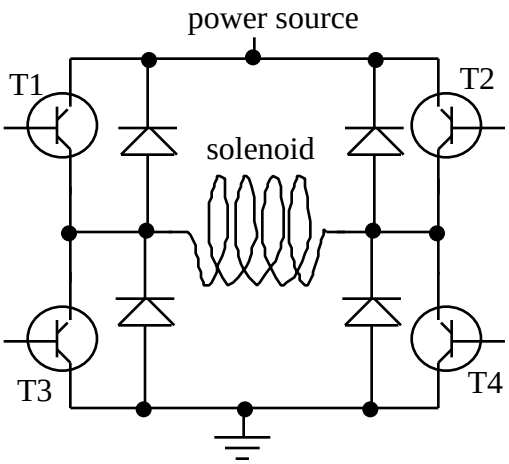

Figure 24.5. Four transistors are used to build an H-bridge.

Interfacing the front kill-switch

The system uses the front bumper switch to detect the situation where the robot is so far off the track that it strikes a wall. You could interface this switch so that it creates an interrupt when activated. The ISR should shut off the motors.

Line-tracking algorithm

The highest-level software is the line-tracking algorithm. You are free to design and implement this software task in any manner you wish. You should consider using a Moore finite state machine implemented with a linked data structure. The inputs to the FSM would be the two line-tracking sensors and the front kill switch. The outputs of the FSM would be the desired motor speed and the steering control. Whichever algorithm you choose document the important issues of "separation of mechanism from policy" and "software abstraction" as discussed in Chapter 2 of Embedded Microcomputer Systems.

Battery power allows for stand-alone operation

After all other aspects of this system are designed, built, and tested, you should convert the system to battery operation. The 6812 microcomputer needs +5±5% volts to operate. The red/black power wires originate from the 8 batteries in the battery compartment. The orange/black power wires originate from 4 of the batteries in the battery compartment.

Option 1. If you use eight NiCad batteries (+1.25 V each), then you should connect the red/black 2-wire (+10 V) cable to the power source for the rear wheel motor and connect the orange/black cable (+5 V) to the Adapt812 power on H1. With NiCad batteries there is no connection to the regulator chip on the side of the Adapt812 board.

Option 2. If you use four NiCad batteries (+1.25 V each), then you should connect the orange/black (+5 V) both to the Adapt812 power on H1 and to the power source for the rear wheel motor. With four batteries you will not use the red/black power 2-wire cable.

Option 3. If you have eight alkaline batteries (1.5V each), then you should connect the red/black 2-wire (+12 V) cable to the power source for the rear wheel motor and connect the orange/black 2-wire cable (+6 V) into the Adapt 812 power regulator. The 2 wire (red/black) connector plugs into the Adapt812 with the black wire on top:

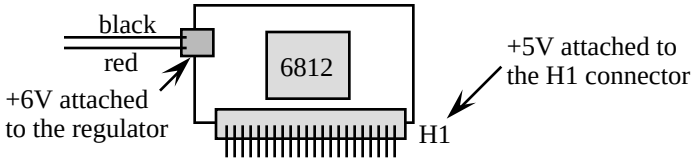

Figure 24.6. The Adapt812 board can be powered from the regulator (+6V) or from the H1 connector (+5V).

Option 4. If you have four alkaline batteries (1.5V each), then you should connect the orange/black 2-wire cable (+6 V) to both the Adapt 812 power regulator and the power source for the rear wheel motor. With four batteries you will not use the red/black power 2-wire cable.

Checkout (show this to the TA)

Minimally intrusive software debugging instruments are required for the first part of the demonstration. The instruments will allow you to show the interaction between the hardware and software. In the first part of the checkout, you should be able to demonstrate correct operation of the individual components of the system:

- observe/explain on the scope the hardware signals at strategic places in the circuit,
- demonstrate the variable-duty-cycle constant-speed motor controller,
- holding the robot in the air over a track, show the operation of the line-tracking algorithm,
- demonstrate the operation of the front kill-switch.

In the second part of the checkout, you need to demonstrate the robot running on the track.

Hints

1) Make sure your TA checks your hardware diagram before connecting it to the 6812. Use a scope to look at the voltages along the circuit. Please test the hardware before connecting it to the 6812. Then test the system with a fixed resistor in place of the motor. After all hardware and software parts are thoroughly debugged, then connect the motor with the 6812.

1) Debug this lab in very small steps.

2) Check for back EMF on the motor coil before connecting to the 6812. It can be huge!

3) See the incremental, PID, fuzzy control examples in Chapter 13 of Embedded Microcomputer Systems.

4) Don't use an open collector gate (like the 7405) in place of the 74HC14 for the MOSFET interface. You could use other regular output HC gates (e.g., 74HC00, 74HC02, 74HC04, 74HC08 or 74HC10.) We will use HC gates, because the eventual goal is to run under battery power. The MOSFET needs positive current (when the 74HC14 output is high) to turn on, and negative current (when the 74HC14 output is low) to turn off.

If you are performing this lab in two parts, then divide the tasks into two parts. For example,

Part 1 tasks

Hardware

1) interface the motor (on/off). use two diodes to eliminate positive and negative back EMF voltages.
2) interface the steering allowing three states (move to the left, move to the right, same)
3) interface the three Sharp GP2L01 sensors
4) interface the front “kill” switch
5) any other hardware used for debugging (e.g., LED, LCD)

Software

1) variable duty cycle to PWM output, rear wheel drive motor
2) low level software for line sensors, software output is two true/false
3) tachometer sensor interface, calculate current speed (rpm, rps, or µs/rotation)
4) any other software used for debugging

Part 2 tasks

Hardware

1) any additional hardware used for debugging (e.g., LED, LCD)
2) conversion to battery power

Software

1) incremental, PI, or Fuzzy motor speed controller for the rear wheel speed
2) line tracking algorithm
3) handling a front end collision
4) any other software used for debugging