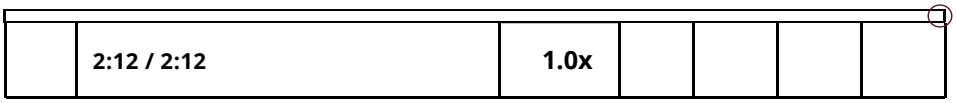
VIDEO 10.7B. ROBOT CAR - DEMO



DR. JONATHAN VALVANO: So here's our stepper motor robot.

And let's review the components.

Up here is the launch pad.

And we have two stepper motors, each one tied to a wheel.

There's one stepper motor tied to the left wheel,

and another one to the right wheel.

We have two sensors.

We have the left bumper sensor here, which is just a switch

and the right bumper switch as well.

On the protoboard here are the L293 motor drivers.

Here's the one driver, and over here is the second driver.

This pretty black chip here is the regulator,

which takes power from the battery, the 8.4 volts,

and generates 5 volts out to power the launch pad.

DR. RAMESH YERRABALLI: The two sensors are connected to port E pins 1 and 0,

and the two stepper motors are

connected to port B.

And what are those back sensors, Jon?

DR. JONATHAN VALVANO: Oh.

Well, every robot has to protect itself, and you

don't want to run into things going backwards,

so the back ones are for future development.

DR. RAMESH YERRABALLI: All right.

So let's try it out.

DR. JONATHAN VALVANO: All right, here's the on switch.

Look out.

Stand back.

It might explode.

DR. RAMESH YERRABALLI: And this an approaching from the side

Help

To make the robot move forward (states 0,1,2,3) we spin both motors. To satisfy Isaac Asimov's first law of robotics "A robot may not injure a human being or, through inaction, allow a human being to come to harm", we will add two bumper switches in the front that will turn the robot if it detects an object in its path. To make the robot move backward (states S3, S2, S1, S0), we step both motors the other direction. We turn right by stepping the right motor back and the left motor forward (states S0, S7, S8, S9). We turn left by stepping the left motor back and the right motor forward (states S0, S4, S5, S6). Figure 10.15 shows the FSM to control this simple robot. This FSM has two inputs, so each state has four next states. Notice the 1-to-1 correspondence between the state graph in Figure 10.15 and the **FSM[10]** data structure in Program 10.7.

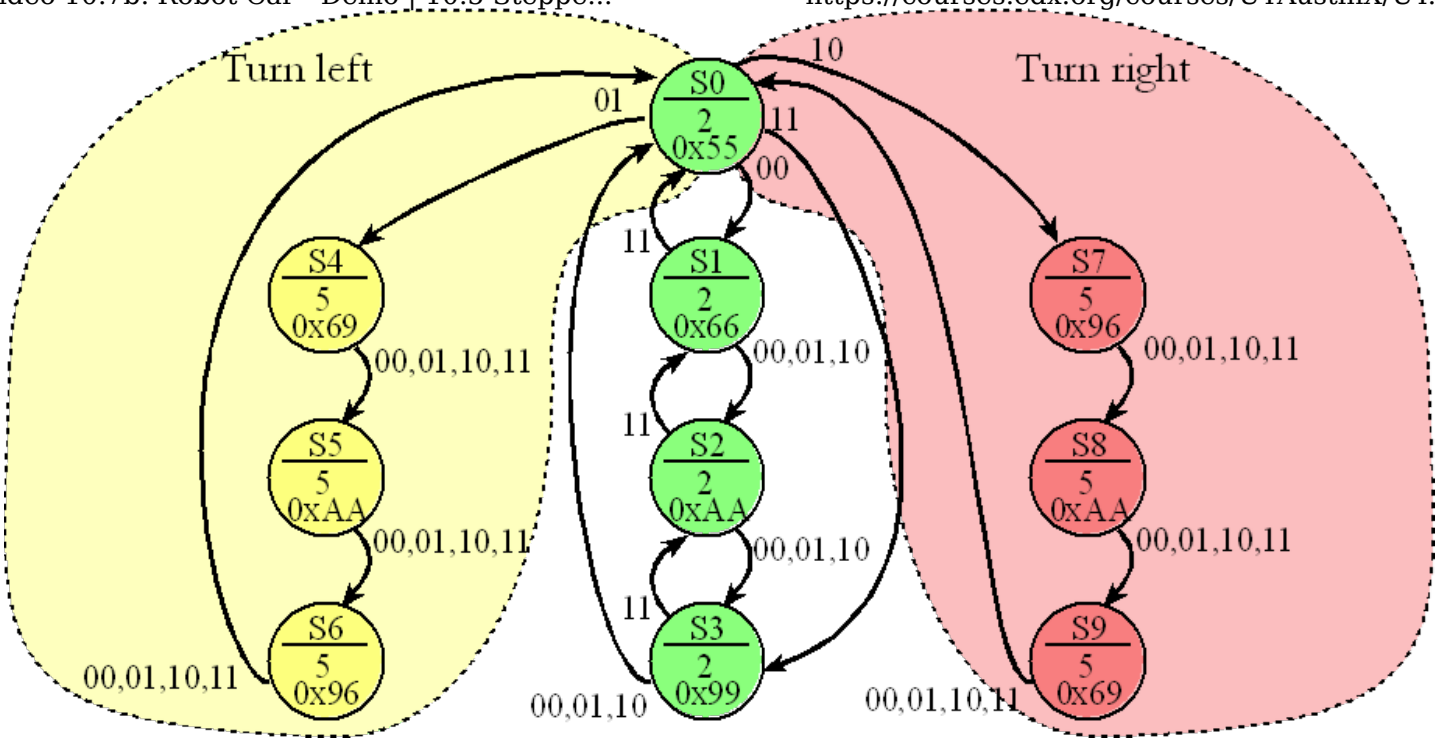


Figure 10.15. If the bumpers are not active (00) the both motor spin at 15 RPM, if both bumpers are active the robot backs up, if just the right bumper is active it turns left, and if just the left bumper is active, it turns right.

Help

The main program, Program 10.7, begins by initializing all of Ports B to be an output and PE1,PE0 to be an input. There are ten states in this robot. If the bumper switch activates, it attempts to turn away from the object. Every 20 ms the program outputs a new stepper commands to both motors. The function **`SysTick_Wait10ms()`**generates an appropriate delay between outputs to the stepper. For a 200 step/rotation stepper, if we wait 20 ms between outputs, there will be 50 outputs/sec, or 3000 outputs/min, causing the stepper motor to spin at 15 RPM. When calculating speed, it is important to keep track of the units.

Speed = (1 rotation/200steps)*(1000ms/s)*(60sec/min)*(1step/**20ms**) = 15 RPM

```
// represents a State of the FSM
struct State{
  unsigned char out;     // PB7-4 to right motor, PB3-0 to left
  unsigned short wait;   // in 10ms units
  unsigned char next[4]; // input 0x00 means ok,
                         //       0x01 means right side bumped something,
                         //       0x02 means left side bumped something,
                         //       0x03 means head-on collision (both)
};
typedef const struct State StateType;
StateType Fsm[10] = {
  {0x55, 2, {1, 4, 7, 3} }, // S0) initial state where bumpers are checked
  {0x66, 2, {2, 2, 2, 0} }, // S1) both forward [1]
```

```
    {0xAA, 2, {3, 3, 3, 1} }, // S2) both forward [2]
    {0x99, 2, {0, 0, 0, 2} }, // S3) both forward [3]
    {0x69, 5, {5, 5, 5, 5} }, // S4) left forward; right reverse [1] left
    {0xAA, 5, {6, 6, 6, 6} }, // S5) left forward; right reverse [2] left
    {0x96, 5, {0, 0, 0, 0} }, // S6) left forward; right reverse [3] left
    {0x96, 5, {8, 8, 8, 8} }, // S7) left reverse; right forward [1] right
    {0xAA, 5, {9, 9, 9, 9} }, // S8) left reverse; right forward [2] right
    {0x69, 5, {0, 0, 0, 0} }  // S9) left reverse; right forward [3] right
};

void PortB_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x02;          // 1) activate Port B
  delay = SYSCTL_RCGC2_R;          // allow time for clock to stabilize
                                   // 2) no need to unlock PB7-0
  GPIO_PORTB_AMSEL_R = 0x00;       // 3) disable analog function on PB7-0
  GPIO_PORTB_PCTL_R = 0x00000000;  // 4) configure PB7-0 as GPIO
  GPIO_PORTB_DIR_R = 0xFF;         // 5) make PB7-0 out
  GPIO_PORTB_AFSEL_R = 0x00;       // 6) disable alt funct on PB7-0
  GPIO_PORTB_DR8R_R = 0xFF;        // enable 8 mA drive on PB7-0
  GPIO_PORTB_DEN_R = 0xFF;         // 7) enable digital I/O on PB7-0
}
void PortE_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x10;          // 1) activate Port E
  delay = SYSCTL_RCGC2_R;          // allow time for clock to stabilize
                                   // 2) no need to unlock PE1-0
  GPIO_PORTE_AMSEL_R &= ~0x03;     // 3) disable analog function on PE1-0
  GPIO_PORTE_PCTL_R &= ~0x000000FF;// 4) configure PE1-0 as GPIO
  GPIO_PORTE_DIR_R &= ~0x03;       // 5) make PE1-0 in
  GPIO_PORTE_AFSEL_R &= ~0x03;     // 6) disable alt funct on PE1-0
  GPIO_PORTE_DEN_R |= 0x03;        // 7) enable digital I/O on PE1-0
}
unsigned char cState;      // current State (0 to 9)
int main(void){
  unsigned char input;
  PLL_Init();                      // Program 10.1
  SysTick_Init();                  // Program 10.2
  PortB_Init();                    // initialize motor outputs on Port B
  PortE_Init();                    // initialize sensor inputs on Port E
  cState = 0;                      // initial state = 0
  while(1){
    // output based on current state
    GPIO_PORTB_DATA_R = Fsm[cState].out; // step motor
    // wait for time according to state
    SysTick_Wait10ms(Fsm[cState].wait);
    // get input
    input = GPIO_PORTE_DATA_R&0x03; // Input 0,1,2,3
```

Help

```
      // change the state based on input and current state
      cState = Fsm[cState].next[input];
    }
}
```

Program 10.7. Stepper motor controller (C10_StepperRobot).

Students in the edX class may purchase their own Analog Discovery logic analyzer/scope at http://www.digilentinc.com (http://www.digilentinc.com/Products/Detail.cfm?Prod=ANALOG-DISCOVERY) for $99 plus shipping. This hardware debugging tool is not required for this class, but we love ours a lot. When purchasing the Analog Discovery identify your school as edX and your class as UT.6.01x. If you have any questions about the Analog Discovery logic analyzer/scope please contact Digilent at awong@digilentinc.com.

VIDEO 5 LOGIC ANALYZER

Help