(https://www.edx.org)

**UTAustinX:** UT.6.01x Embedded Systems - Shape the World

**KarenWest (/dashboard)** ▼

**Courseware (/courses/UTAustinX/UT.6.01x/1T2014/courseware)** **Course Info (/courses/UTAustinX/UT.6.01x/1T2014/info)**

**Discussion (/courses/UTAustinX/UT.6.01x/1T2014/discussion/forum)** **Progress (/courses/UTAustinX/UT.6.01x/1T2014/progress)**

**Questions (/courses/UTAustinX/UT.6.01x/1T2014/a3da417940af4ec49a9c02b3eae3460b/)**

**Syllabus (/courses/UTAustinX/UT.6.01x/1T2014/a827a8b3cc204927b6efaa49580170d1/)**

## VIDEO 10.7A. ROBOT CAR - SOFTWARE

**Help**

**4:01 / 4:01** **1.0x**

DR. JONATHAN VALVANO: So let's look at our software for our robot.

The hardware is such that PB7-PB4 is to the right stepper.

PB3 to PB0 is to the left.

PE1 has the left bumper.

PE0 has the right bumper, both positive logic.

There's initialization, which is similar to all the other initializations we've

been doing so far, where port B is an output, port E is an input.

We will use a struct to define the components of a state.

Each state will have an output, will have a time

to wait in 10 millisecond units, and four next states.

Again, the four possible next states are zero zero means, we're OK.

Full speed ahead.

Zero one means the right bumper just hit something,

zero two means the left bumper hit something,

and zero three means a head-on collision,

both bumpers at the same time.

DR. RAMESH YERRABALLI: So the indexes in this area respond to the inputs

that you have to respond to.

DR. JONATHAN VALVANO: Absolutely.

An important thing about finite state machines, the reason it is so powerful

is because there is a one-to-one mapping between the structure that we drew,

the state transition graph, and this data

structure that we will store in the computer.

Every state had exactly one output, every state had a time to wait,

and every state had four next states, shown here as indexes,

depending upon whether the input is zero, one, two, or three.

So there is a one-to-one mapping.

That means no more, no less information in this than the state transition

graph.

This data here completely specifies what the machine will do.

We will store the current state in this variable called

cState, so will have a number between zero and nine

specifying the state number that we are currently in.

The initialization is to turn on the clock and the SysTick and the ports

and initialize the state to be the first state, or state zero.

And then we'll repeat the engine over and over again.

The first step is exactly like we saw with Dr. Yerraballi's, and that is,

we're going to output depending on the current state.

So cState is our current state, and so we're

going to fetch the output field from that state and output it to port B.

That will be one of the numbers we saw in the finite state machine.

The next, to make it run at the speed we want,

is we were going to delay for a certain amount of time.

So a delay of two will mean it will wait 20 milliseconds.

DR. RAMESH YERRABALLI: So this will control

the speed at which the motors will rotate?

DR. JONATHAN VALVANO: Yes.

So if we're waiting 20 milliseconds, that

will be 1.8 degrees for 20 milliseconds.

Next, we'll fetch the input.

And so we're going to fetch the input from PORT E bit 0 and 1.

And so this variable here is going to have the values zero, one, two,

or three, depending upon whether we hit something or not.

And lastly, we're going to change state, depending

on both the input and the current state.

The current state will specify which one

Help

**Solution:** We choose a stepper motor according to the speed and torque requirements of the system. A stepper with 200 steps/rotation will provide a very smooth rotation while it spins. Just like the DC motor, we need an interface that can handle the currents required by the coils. We can use a L293 to interface either unipolar or bipolar steppers that require less than 1 A per coil. In general, the output current of a driver must be large enough to energize the stepper coils. We control the interface using an output port of the microcontroller, as shown in Figure 10.13. Motors require interface circuits, like the L293, because of the large currents required for the motor. Furthermore, most motors will require more current than the USB or LaunchPad can supply. In this system the motors are powered directly from an 8.4V battery. Motor current flows from the battery to the L293, out 1Y, across the coil of the stepper motor, into the 2Y-pin of the L293, and finally back to the battery. Notice the motor currents do not flow across the LaunchPad or in/out the USB. The circuit shows the interface of two bipolar steppers, but the unipolar stepper interface is similar except there would be a direct connection of +8.4V to the motor (see Figure 10.14). The front of the robot has two bumper switches. The 7805 regulator allows the LaunchPad to be powered from the battery.

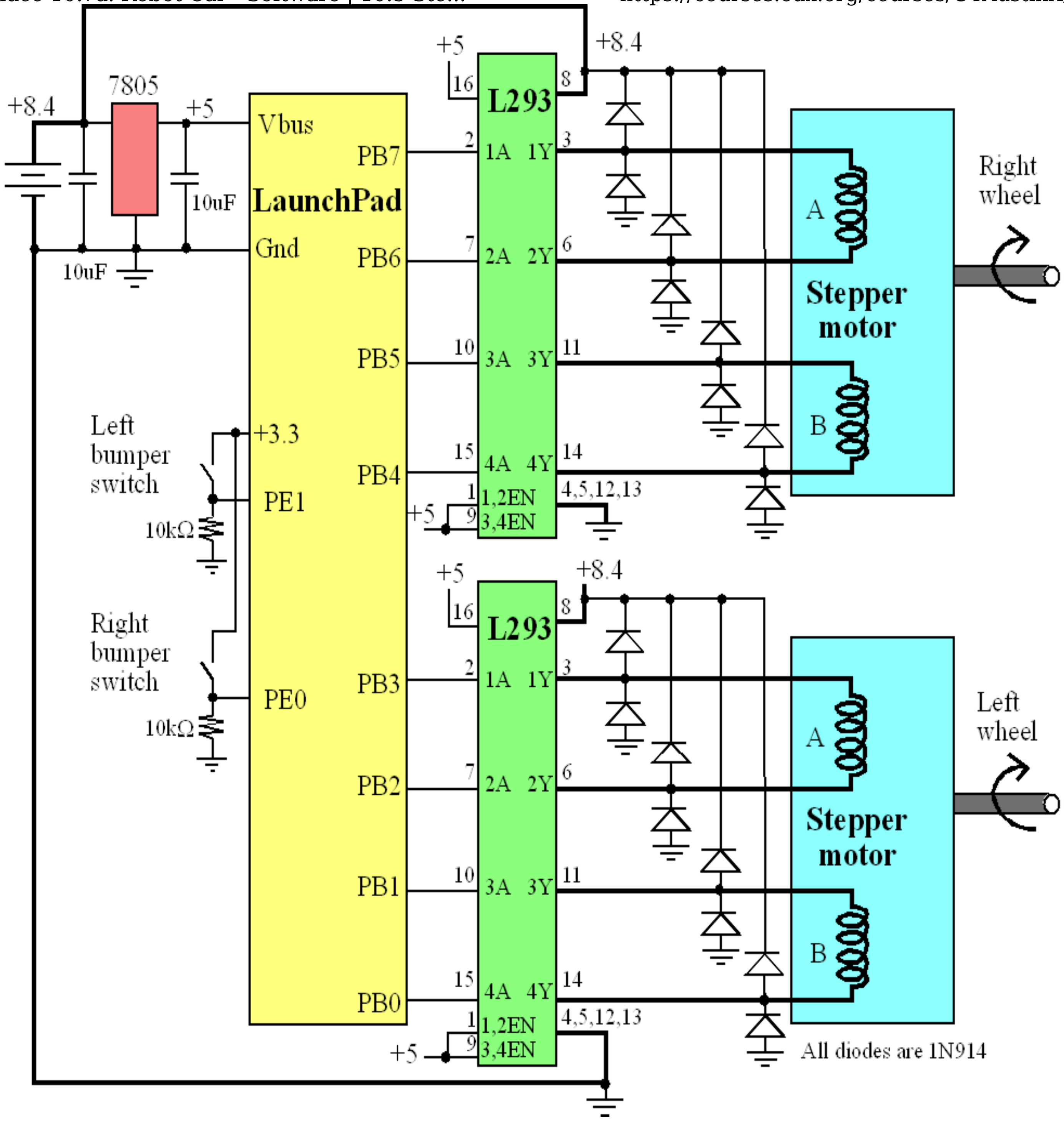

*Figure 10.13. Two bipolar stepper motors interfaced to a microcontroller allow the robot to move. The dark lines carry large currents. Bipolar stepper motors have 4 wires.*

**Warning** R9, R10, R25, and R29 are zero-ohm resistors on the LaunchPad.  R9 shorts PD0 to PB6, R10 shorts PD1 to PB7, R25 connects PB0 to the USB device header, and R29 connects PB1 to +5V. So make sure R26 and R29 are removed. If you use both PD0 and PB6, remove R9. If you use both PD1 and PB7, remove R10.

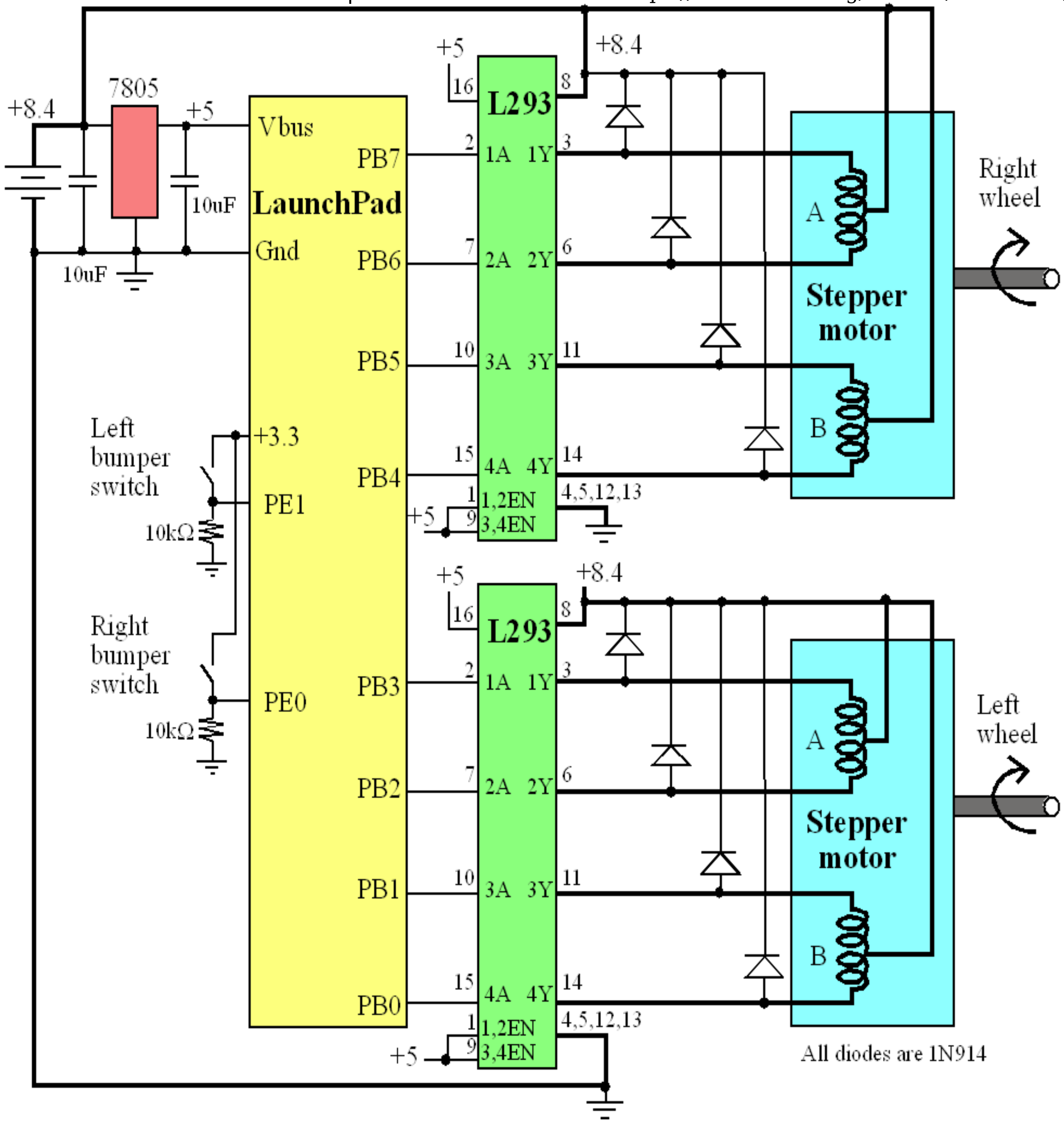

*Figure 10.14. Two unipolar stepper motors interfaced to a microcontroller allow the robot to move. The dark lines carry large currents. Unipolar stepper motors have 5 or 6 wires.*

---

Bill of materials for stepper motor robot

1) Two stepper motors,

- size less than 50 mm on a side, with mounting holes

- 4 or 5 mm shaft (get hubs to match)

- voltage depends on the battery (you loose about 2V in the L293 driver)

------ an 8.4V battery can drive a 5 to to 6V motor

------ an 11.1V battery can drive an 8 to 10V motor
------ a 14.4V battery can drive a 10 to 12V motor
Examples
Jameco Part no. 237519, 42BYG017-R, $21.95 each
Jameco Part no. 238538 42BYGH404-R, $21.95 each
www.AllElectronics.com, SMT-373U, 103-591-0155, $9.50 used

2) Three pieces of 1 by 2 in wood for base, built like a capital I,
- backbone is about 9 inches long
- front and back are about 5 inches long

3) Hardware for mounting
- 4 wood screws to attach front and back to backbone
- 2 screws to attach the two sensors
- 2 screws to attach protoboard (I drilled a hole in the ends of my protoboard)
- 2 bolts and nuts to attach two motors. Must fit into mounting holes of the motor, and belong enough to go through backbone and two motors.
- some way to attach the LaunchPad (I used an Erector set, but you could use rubber bands)

4) Wheels hubs and a caster
- Tamiya 70144 Ball Caster Kit
- Two wheels and two hubs to match the diameter of the motor shaft
www.robotmarketplace.com
http://www.robotshop.com

5) Two snap action roller switches
www.bgmicro.com SWT1117
www.AllElectronics.com, SMS-231, SMS-197

6) Battery
- 8.4V NiMH or 11.1V LiIon. I bought the 8.4V NiMH batteries you see in the video as surplus a long time ago. I teach a real-time OS class where students write an OS then deploy it on a robot. I have a big pile of these 8.4V batteries, so I used a couple for the two robots in this class. NiMH are easier to charge, but I suggest Li-Ion because they store more energy/weight. For my medical instruments, I use a lot of Tenergy 31003 (7.4V) and Tenergy 31012 (11.1V) (internet search for the best price). You will need a Li-Ion charger. I have used both of these Tenergy TLP-4000 and Tenergy TB6B

7) Electronic components
- two L293 motor drivers
-16 1N914 diodes
-2 10uF tantalum caps
- 7805 regular
-2 10k resistors

Help