ME 210 Winter 2026: Laboratory Assignment 0

Pre-Lab Due at 11:59 pm on January 9, 2026
Due at 11:59 pm on January 15, 2026.

Purpose:

This lab is intended to acquaint you with programming for embedded systems, the Arduino Leonardo, and the Arduino integrated development environment (IDE). This will include:

- Learning the ropes of the IDE
- Learning how to write, debug, and run a simple embedded C program on an embedded microcontroller

Minimum Parts Required:

A computer, an Arcbotics Sparki robot, and a USB cable to program it.

References:

Textbook:

Chapter 3, *Microcontroller Math and Number Manipulation*
Chapter 5, *Program Structures for Embedded Systems*

Part 0: Pre-Lab

Background:

The pre-lab should be completed *after* you have read through the entire lab assignment, but *before* you go into the lab to begin your work there. Completing the pre-lab, which can be done without any special tools or resources, will allow you to be most efficient with your time in the lab. **When submitting your lab report, be sure to include a section for your answers to the pre-lab, even if you found no need to adjust them after the submission of the pre-lab. Assignment:**

- □ **0.1)** Brainstorm and identify the essential "Events" necessary for your desired behavior, how you will test for them in software, and how you want to respond. Your Pre-Lab write-up should explain the results of your brainstorming session in the form of a *finite state diagram*.

In the Report:

Include your answers to the exercises in the Pre-Lab.

Part 1: Setting up the Arduino Environment

Reading:

Getting Started with Arduino IDE 2: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2

Components Required:

Arduino IDE 2.2.1, downloaded from https://www.arduino.cc/en/software.

Arcbotics Sparki.

Assignment:

Complete the following exercises:

- ☐ **1.1)** Download the Arduino IDE to your computer and install it.
- ☐ **1.2)** Start the Arduino IDE while connected to the Internet, and allow it some time to download and initialize the default libraries. You may also be prompted to install system drivers that handle USB communication with the Arduino. In the"Tools/Board/ArduinoAVRBoards" menu, select "Arduino Leonardo".
- ☐ **1.3)** Download the starter project **RaptorStarter** from the `Files` section of Canvas, and save it in a directory of your choice that you can access from the Arduino IDE, which saves to your computer's `Documents` `Arduino` directory by default. **RaptorStarter** is a basic demonstration program that runs on the Arduino Leonardo, uses the Events and Services model, and controls an Arcbotics Sparki. Select "Open..." and load **RaptorStarter.ino**. If you are not prompted to select a board because your IDE workspace is already set up for Arduino Leonardo (part 1.2), select "Arduino Leonardo" now.
- ☐ **1.4)** Download the **Metro** and **Raptor** libraries from the `Files` section of Canvas. Unzip these and place them in the `lib` directory of the **RaptorStarter** project. Copy the entire unzipped directories to `lib`, not just the files.
- ☐ **1.5)** Use the USB cable to connect Sparki to your computer. **NOTE: The Arduino immediately begins executing code upon startup; be careful that Sparki does not run off any tables!** If you leave Sparki's on-off switch in the "off" position, or if you take out its 4×AA batteries, you can still access the sensors, microcontroller, and output screen by powering them from the USB cable, but the motors for the wheels and the servo will not work.
- ☐ **1.6)** Using the buttons at the bottom of the IDE, build the open project file. You should not see any errors in the output.

Part 2: Sparki Warm-Up Exercises

Components Required:

Computer with development environment installed, Arcbotics Sparki **Assignment:**

- □ **2.1)** You are now ready to download your compiled code to the Arduino on the Sparki. Click on the *Upload* button (the button with an arrow pointing to the right, located in the main toolbar). When you hover over the *Upload* button, the label "Upload" appears to the right on the menu bar. If your code compiled without errors in the previous step, and you have Sparki connected correctly, the code will be transferred from your computer to the Arduino's program memory. While this is occurring, progress and status are reported in the pane at the bottom of the Arduino IDE window.
- □ **2.2)** When the download is complete, the Arduino should automatically reset and begin executing your code. You may also reset the Arduino at any time by pressing the reset button right next to Sparki's on-off switch.
- □ **2.3)** If your program sends out serial data (e.g., using the `Serial.println+()` function), you may monitor this by opening a terminal window. Select the "Tools/SerialMonitor" menu option to open the serial monitor **and** reset the Sparki. **NOTE: The RaptorStarter code does use the Serial class and explicitly waits for you to open the Serial Monitor before it starts the main program loop!** Hit the Serial Monitor icon if nothing is happening.
- □ **2.4)** Congratulations! Hopefully, you've just compiled and downloaded your first embedded program (or, at least, your first ME 210 embedded program).
- □ **2.5)** When running our starter code, you should observe the Sparki moving forwards and backwards, with the LED blinking at 0.5Hz. However, looking at the code, the timer corresponding to the LED is set at `LED_TIMER_INTERVAL`, which is 1000ms. Why doesn't the LED blink at 0.5Hz? Write the answer to this in the lab report, and modify your code such that it blinks properly. Check-off with a TA, and keep the LED blinking at 0.5Hz for the rest of the lab! (HINT: This has to do with blocking code, which you should never use unless you have an extremely good reason to do so.)
- □ **2.6)** As part of the preprocessor macros in RaptorStarter, you'll see two constants, `LIGHT_THRESHOLD` and `LINE_THRESHOLD`, which you'll later find useful to determine whether or not the light is on, and whether or not you've hit the fence. Write a function to help you identify these thresholds, and print them out to the serial monitor. Include your thresholds, as well as the code you wrote to find them, in the report.
- □ **2.7)** In the starter code, you are given a set of (empty) functions that will help you program your velociraptor in Part 3. They are:

  ```
  TestForLightOn
  RespToLightOn
  TestForLightOff
  RespToLightOff
  TestForFence
  RespToFence
  ```

  Fill out the implementations for these functions.

- □ **2.8)** Write out the implementations and spend some time exploring the library functions provided for you. Write some simple programs to turn the motors on and off, get a character from the keyboard, print a value to the serial monitor, read the light sensor, beep the buzzer, turn on the RGB LED, and test the IR bumpers. Remember, Sparki will need 4×AA batteries and the on-off switch in the "on" position before you can move the motors. Don't walk Sparki off the table! **IMPORTANT NOTE: SAVE THESE SIMPLE TEST PROGRAMS. See the "In the Report" section below—you are asked to include these in your lab report.**

In the Report:

Include your answers to **Part 2.5** and **Part 2.6**. Also include the code for the functions you wrote in **Parts 2.7 and 2.8**.

Part 3: A mechatronic Velociraptor

Reading:

Textbook: Chapter 3, *Microcontroller Math and Number Manipulation*
Textbook: Chapter 5, *Program Structures for Embedded Systems*
Optional: *Jurassic Park*, Michael Crichton

Background:

The behavior of many simple types of creatures can be described in terms of their responses to outside stimulus. This will be your opportunity to give a mechanical creature interesting behavior.

Assignment:

Your mission, should you choose to accept it: write a program for the Arduino that implements a state machine which causes the Arcbotics Sparki to emulate the behavior of a real Velociraptor. (You must accept this mission.)

The best available reference for how Velociraptors might behave in the modern world is, of course, Stephen Spielberg's 1993 documentary, *Jurassic Park*. Robert Muldoon, the game warden for the park, says that the raptors attacked their fences whenever the feeders came. "They never attack[ed] the same place twice," he says. "They were testing the fences for weaknesses, systematically." To approximate this behavior, your raptor must:

- run around attacking fences when it detects light;
- stop and lurk, hungry and motionless, in the dark, and; - not attack the same part of the fence over and over.

The fence is represented by a black line on the ground (use Sparki's own line-following map), and weaknesses are represented by pieces of white tape laid across the line. Crossing the fence where there is no weakness will electrocute your raptor!

□ **3.1)** Design your Velociraptor program: develop a finite state diagram that describes how it should function, and plan your software using PDL (Program Design Language, a.k.a. "pseudocode").

□ **3.2)** Complete the detailed design and implementation of the program that you worked out in Part 0. *The TAs will not help debug code unless you have drawn a state diagram!*

□ **3.3)** Write and test your program to imitate a real live Velociraptor. Be sure to develop and test incrementally to avoid a huge headache!

□ **3.4) Demonstrate your working code on a Sparki to a member of the teaching staff to be checked off for this section.**

In the Report:

Include a complete description of the design and implementation of your program. This should include highlevel descriptions (such as a state diagram), PDL/pseudocode, and a listing of the final Arduino source code that resulted.

Notes on writing your code

As you will have noticed from studying the example code provided, there are a few things that you must do in order to have access to the libraries provided for programming the Sparki. In particular, notice the header files that need to be included:

```
#include <Raptor.h>
#include <Metro.h>
```

`Raptor.h` is the header file for a simple set of input and output routines provided for use on the Sparki. This library gives you access to the critical functions that control the motors, control the LED, read the state of the reflectance and light sensors, and infrared bumpers, and read the state of the light sensor. See the documentation for the Raptor library for the details about these functions.

`Metro.h` is the header file for the Metro library. This library provides access to several timers and several useful functions associated with initializing, setting, and checking them. See the documentation for the Metro library for the details about these functions. (https://github.com/thomasfredericks/Metro-Arduino-Wiring/wiki)

To read the state of the line sensors, you can either use the individual `EdgeRight()`, `LineRight()`, etc. functions, or use `ReadTriggers()` to read them all simultaneously. To use `ReadTriggers()`, treat the returned value from `ReadTriggers()` as an `unsigned char`. The individual IR bumper switches can be isolated using the following masks:

```
#define LEFT_TRIGGER 0x01
#define CENTER_TRIGGER 0x04
#define RIGHT_TRIGGER 0x10
```

Tocheckthestateofasinglelinesensor(in this example you have stored the value returned by `ReadTriggers()` in the variable `triggerState` and you are going to check to see if the left line sensor is currently below the threshold for detecting a line), use the expression:

```
(triggerState & LEFT_TRIGGER)
```

This makes use of bit masking and the C programming language's bitwise operator `&` to check the state of a single IR bumper, instead of considering all 3 bumpers at the same time.

Using Libraries with Arduino IDE

Generally, you can use libraries in the Arduino IDE by searching for them and downloading them in the *Libraries* section of the IDE. This area manages built-in libraries, and hosts a database of community-generated libraries that may prove useful to you in the future! You can also drop a software library's directory into the `lib` directory under your project. The contents of your library directory need to contain the `.h` and `.cpp` file at the very least.

FAQ, Tips and Tricks

Can my Raptor touch the fence at all?

- We understand that the line sensors are underneath your raptor's body, so as long as it doesn't completely disregard the fence and run out, it's still alive.

Does my Raptor actually need to escape?

- Generally, no—your Raptor can perform all required behavior without actually finding an exit.
- The main exception to this is if your Raptor uses a line-following strategy. Line-following strategies are allowed, but **MUST** be able to actually exit the fence.

How should I include my code in the report?

- Please use a monospaced font (Consolas, DejaVu Mono, Input Mono, Hack, Menlo, etc.)
- Alternatively, use a code syntax highlighter (e.g. http://quickhighlighter.com/) for nice fonts AND pretty colors!