**University of California, Santa Cruz**

**COSMOS Cluster 6:**

**Networking and Robotics**

**Lab 4: State Machines**

## Introduction

Now that we have a sense of how an events-and-services system works, it’s time to use that to build robot behavior.

In this lab, you will implement a very common and powerful structure for a service: The *State Machine.* Our goal is to use this structure to make our RoachBot act like a real cockroach: It will try to find a nice dark place to hide.

You will begin by implementing a relatively simple state machine that we’ve written for you. Then, you will redesign the state machine for better roach behavior.

## State Machines in code:

The phrase “State machine” is an abstract term describing the mathematical idea of persistent states connected by instantaneous transitions. There are many ways to implement this in code, however (In fact, you’ve already implemented very simple state machines when you wrote your event checkers!)

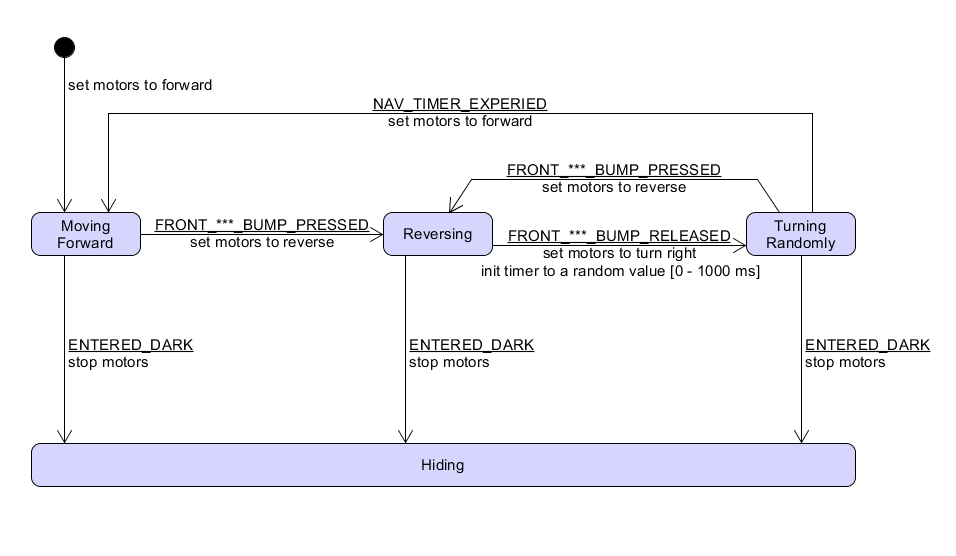
One common way in C is to use the “switch/case” structure, which exists for this purpose. It acts much like a sequence of if/else statements. Here’s how that looks:

## The Simple Roach State Machine:

One very simple algorithm for navigating an unknown territory is simply:

1. Move forward until you hit something
2. Once you hit something, take 1 step back, then turn around to a random direction
3. Repeat steps 1 and 2 until you find what you’re looking for

It’s not a great algorithm[[1]](#footnote-1), but it is easy to implement. Here’s a simple state machine to implement this algorithm:



## Instructions:

* Open the Lab 4 project and test it
  + We’ve only implemented one state transition for you (MovingForward (ENTERED\_DARK) -> Hiding). Test this on the floor!
  + Notice that there are a few new files:
    - Roach\_Events.c/.h: To keep the main file short and sweet, we’ve bundled the event checkers into a separate module.
    - Roach\_State\_Machine.c/.h: Similarly, we’ve made a separate module for your state machine code.
* Implement the rest of this state machine
  + You shouldn’t edit the State\_Machine\_main file.
    - Most of your coding will be done in Roach\_State\_Machine.c.
    - As you progress, you may find it useful to modify Roach\_Events.c.
    - If you are tempted to modify any other files, talk to the teaching staff – there’s probably a better way to accomplish this goal!
  + We recommend adding one state at a time, and bench-testing each and every transition out of that state before you proceed to the next state.
* Once you’ve implemented the state machine in code, test the state machine in the maze and on the floor. Carefully observe its behavior.

QUESTION:

Your goal: Starting from a somewhat random location and orientation on the floor, find and hide in darkness as quickly as possible. Quantify the effectiveness of the Simple Roach State Machine at achieving this goal. What are the strengths and weaknesses of this state machine?

Extra Challenges:

As usual, you can choose whichever challenges your team prefers. Whatever you do, follow these steps:

* STEP AWAY FROM THE SCREENS. The first design steps are best done on paper or on a whiteboard!
* Design a state machine. Write it down with pencil and paper.
* Show your state machine to a member of the teaching staff for feedback.
  + Make modifications and then take a photo
* Ok, NOW you can start coding!
  + Write one state at a time, and test as you go.
  + As you go, you will inevitably make changes to your state machine. Make sure that your paper (or UMLet file) stays in sync with your code!

## Better Roach State Machine:

Modify (or completely re-write) the state machine from the first part of the lab to make the roach more effective at finding darkness on the. Once you’re happy with it, design and perform an experiment that lets you quantify how much better your version of the state machine is.

## Chemotaxis

The roach is a pretty good approximation of a single-celled organism searching for a place with a high concentration of food. Design and implement a state machine that allows the roach to find and remain near a bright spot on the floor. Quantify its effectiveness.

## Maze Crawler

We’ll put a maze from last year’s COSMOS project in the lab. The goal is to start at a somewhat random point at the maze and reach the dark corner as soon as possible. Design a state machine to accomplish this goal. Quantify its effectiveness.

1. It’s not useless: many single-celled organisms use this algorithm to perform *chemotaxis* (ie, finding food). [↑](#footnote-ref-1)