# Lab 1 - Wed 7/18

Warm-up

* Python tutorial: <https://www.waiwaing.com/python/>
* Get acquainted with Hamster and its API. Learn about the APIs in robotAPI.txt. Methods to be used: set\_wheel(), set\_led(), set\_buzzer(), set\_musical\_note() to understand how to control Hamster using provided API.
* Understand how Hamster sensors work by using get\_proximity(), get\_floor(). Get familiar with range of sensor readings and how reading changes with distance and surface color.

Check-off 1

* Python tutorial check-off

Check-off 2

* Implement the following Hamster behaviors:
  + Moving in a square shape without using sensors
  + Using proximity sensors, implement shy, dance, and follow
  + Using floor sensors, implement line following
* Add buttons to the starter program and implement all the behaviors in one program

Optional

* Implement maze solver using side sensor attachment. Ask TAs for the attachment.

# Lab 2 - Thu 7/19

Assignment

* Implement UI for Hamster showing the sensor readings
* Implement joystick controller for Hamster

Check-off 1 - UI components

* Square representing Hamster
* Two line segments representing proximity sensor readings
* Two small rectangles on the Hamster representing floor sensor readings
  + The rectangle should be filled black if the sensor is on a black line

Check-off 2 - Joystick

* Use wasd keys as joystick to move Hamster around

Optional

* Navigate maze using UI + joystick
* Maze solver using side sensor attachment
* Grid navigation

Lab 3 - Fri 7/20  
  
Assignment - Hamster Escape

* Functionalities
  + GUI window shows a square representing Hamster. One button to start Hamster escape and one for exit program.
  + Display line segment and acknowledge (sound) when proximity sensors detect obstacle
  + Acknowledge (sound) when border is reached
  + Hamster avoids obstacle when it detects any
* Components
* Four threads (including the original thread)
  + Motion handler responds to events that are generated by Event Watcher thread. It handles Hamster motion control.
  + Event watcher is responsible for reading Hamster proximity and floor sensors and adding these events to the two event queues.
  + Alert handler responds to events that are generated by Event Watcher thread. It displays proximity sensors. This is the main (original) thread. The GUI is in this thread, and you do not need to create a new thread for this since the GUI will refresh itself.
  + Bluetooth comm thread. Spawned to connect the Hamster.
* Two event queues, one for motion and one for alert handler. Event watcher thread puts event in the queues while alert and motion handlers get events and take appropriate actions.

Check-off

* Escape while avoiding obstacles
* Display prox sensors

Optional

* Maze solver using side sensor attachment
* Grid navigation
  + Detect and acknowledge intersections by stopping and playing a tone

# Lab 3.5 - Sat 7/21

Assignment

* Finish up previous labs
* If you have finished, please work on the maze solver

# Lab 4 - Mon 7/23

Warm-up

* Study the parking ticket demo code to understand how FSMs work
* Compare the parking ticket demo code with the state diagram presented in class

Assignments

* FSM Obstacle Avoidance
  + Design (on paper!) the state diagram for obstacle avoidance, including transition of states in the machine, the event/triggers, and the actions/callbacks
  + Implement Hamster sensing and motion control callback functions for obstacle avoidance. Integrate sensing and control with FSM.
* FSM Escape
  + Extend FSM Obstacle Avoidance into FSM Escape. Same requirements as the Hamster Escape assignment.
* FSM Trash Cleaner
  + The goal is for the Hamster to push trash outside the black bordered arena
  + Design (on paper) the state diagram
  + Hamster stays inside arena at all times
  + Hamster acknowledges detection of trash with sound
  + Hamster turns around after it has finished pushing a piece of trash past outside border

Check-offs

* FSM Obstacle Avoidance
* FSM Escape
* FSM Trash Cleaner

Optional

* Extensions of FSM Trash Cleaner
  + When trash is detected, align the Hamster with the trash to minimize chance of losing it while pushing it out
  + Acknowledge when each piece of trash is pushed out
  + Acknowledge when a total of three pieces are pushed out and then stop Hamster from moving

# Lab 5 - Tue 7/24

Assignment 1 - Hamster Simulator

* Tune d\_factor in tk\_simulator\_joystick\_starter.py to accurately simulate your Hamster
* What is d\_factor? Each yellow grid in the simulation corresponds to 2 cm in the real world. This scale factor is used to map the real Hamster’s speed in the real world to the simulated Hamster’s speed in pixel space.

Assignment 2 - Collision Detection

* Design and implement collision detection for the Hamster simulator
* Design algorithm on paper, then implement inside the in\_collision function

Check-offs

* Hamster Simulator - drive the Hamster for 20 - 30 cm and show that the simulator accurately reflects the distance driven
* Collision Detection - Hamster in simulation should not be able to drive into obstacles

# Lab 6 - Wed 7/25

Assignment 1 - GridGraph

* Implement all methods in GridGraph class (starter\_grid\_graph.py)
* For a given grid, generate a simple undirected graph with no edge costs
* The grid description is dimension (# rows x # cols) and locations of nodes that are occupied by obstacles
* The node names in the graph reflect the location of the node on the grid

Assignment 2 - Display

* Implement all methods in GridGraphDisplay (starter\_grid\_graph\_display.py)
* This class has a handle to an instance of GridGraph class, and displays the grid graph, highlights the start and goal nodes, and highlights the path

Assignment 3 - BFS

* Implement BFS in starter\_bfs.py to search the graph created by GridGraph

Check-offs

* GridGraph
  + On each node, labels showing the coordinates of the node must be displayed, for example, (0-0), (0-1), (0-2), etc.
* GridGraphDisplay
  + Displays the grid graph
  + Highlights (1) start node, (2) goal node, and (3) in-between path nodes in three different colors
* BFS
  + Put the test code from test.py into starter\_bfs.py, expect 24 paths from A to G

Optional

* Implement DFS
* Grid navigation - extend GridGraph, GridGraphDisplay, and BFS to have Hamster execute path returned by BFS on real-life grid with obstacles

# Lab 7 - Thu 7/26

Assignment 1 - Grid Navigation

* Extend previously implemented GridGraph, GridGraphDisplay, and BFS
* Convert BFS path (list of ordered nodes) to a sequence of movements that are executed by the Hamster on real-life grid with obstacles

Assignment 2 - Continuous Space Motion Planning

* Note: Do NOT start this part until you have been checked-off for Grid Navigation
* Compute C-obstacles from given work space map. Hamster is simulated as a circle with a radius of 28 pixels.
* Implement cutting algorithm to compute free cells
* Compute connectivity of two free cells by locating the midpoint of overlapped free edges. A free edge is the overlapped edge shared by two free cells. These points referred to as free points and are nodes in the resulting graph.
* Compute distance connecting two neighboring free points. This distance is the edge cost in the graph.
* Generate graph using the nodes and edges from above. (code provided in graph\_with\_edge\_cost.py)
* Find a path by searching (e.g., Dijkstra) the graph for any given start and goal. (code included in the starter program)

Check-offs

* Grid Navigation
* Motion Planning