Project 9: Finite State Machines I
Project 9: “Your Mission”

Produce the following behavior:

• Wait for the switch to be pressed.
• Record the current orientation as your goal.
• After a 5-second delay, ramp up the middle fan to a point where the craft begins to turn (as measured by the gyro).
• (optional) Slightly drop the middle fan thrust.
• Move forward until a wall is detected to the front.
• Stop
• Make a 90 degree turn to the left
• Move forward until another wall is detected to the front.
• Stop
Implementation

We are using a Finite State Machine to implement this entire sequence

- Use a FSM diagram to plan your machine

Code:

- New task: fsm_task (with fsm_step())
- Use an enumerated data type State to capture the different possible states
- Define behavior for each state:
  - What are the events, actions and transitions?
- Implement and test incrementally
Finite State Machine Implementation

```c
fsm_step() {
    static State state = STATE_START;  // Initial state

    switch(state) {
        case STATE_0:
            <handle state 0>
            break;
        case STATE_1:
            <handle state 1>
            break;
        case STATE_2: ...
    }
```

Andrew H. Fagg: Embedded Real-Time Systems: Project 9
Finite State Machine Implementation

• All sensing and low-level control will be addressed by other tasks
• Communication between tasks through global variables:
  • Sensors include: IMU, distance, velocity_smoothed, theta_error
  • “Actuators” include: theta_goal, velocity_goal
Notes

• Implement and test the FSM incrementally
• You can test your code while holding onto the craft
  • Person holding simulates the sequence of movements
• We have a partial field set up now; a full set of walls will be installed soon
• Surface: we are trying something new today and tomorrow