# Lab 0 Prelab

Victor Perez Contreras

April 5, 2025

## Contents

## 1 Part 1 - PCB Assembly and Soldering

1. Watch Videos
2. The iron should be set to around 340 or 400$^{\circ}F$ depending on if the material is lead or lead-free respectively.
3. For Lead solder the temperature should be around 180$^{\circ}C$. For lead-free solder the temperature should be around 420$^{\circ}C$.
4. A hot weld will lead to discoloration of the material, point peaks, and overall bad spread of material at the joint. A cold weld will be weak and have very little discoloration. The aim is to be between these two exteremes to have a smooth, non-blobby soldered joint.
5. Black in the weld means the heat was excessive leading to burning in the materials. Black may also indicate oxidation.

## 2 Part 2 - ”Hello World!” on a roach

None.

## 3 Part 3 - Running the Roach Test Harness

None.

## 4 Part 4 - Roach Hardware Exploration

1. Read the roach.c file (specifically around the `c #ifdef ROACH TEST` section
2. Test Harness

- FLEFT_BUMP_MASK: Outputs and checks the current voltage level of the battery.
- FRIGHT_BUMP_MASK: Checks the roach's light sensors.
- RLEFT_BUMP_MASK: Checks the left motor
- RRIGHT_BUMP_MASK: Checks the right motor

3. Pseudocode for additional test harness:

```
void Motor_Test(void) {
    switch(keyboard_input) {
        case (w):
            // Move the roach forward
            break;
        case (a):
            // Move the roach to the left
            break;
        case (s):
            // Move the roach backwards
            break;
        case (d):
            // Move the roach to the right
            break;
        case (q):
            // Stop the test
            return;
    }
}
```

## 5 Part 5 - Event Detection

1. Pseudocode prototype of event checkers for the bump sensors and the light sensor.

```
// Note: Numbers are arbitrary and will be adjusted accordingly for the implementation
#define BUMP_SENSOR_THRESH 100
#define LIGHT_SENSOR_THRESH 200

int Check_Bump_Events(int prev_bump_state) {
    int curr_bump_state = Roach_ReadBumpers();

    // Detect an event in the bump
    if (curr_bump_state ≠ prev_bump_state) {
        // Compare both the curr and prev bump state and return a new bump
        int new_bumps = compare(curr_bump_state, prev_bump_state);

        switch (new_bumps) {
            case FLEFT_BUMP_MASK:
                printf("Front Left Bumper hit!\n");
                // Back up and turn right
                break;
            case FRIGHT_BUMP_MASK:
                printf("Front Right Bumper hit!\n");
                // Back up and turn left
                break;
            case RLEFT_BUMP_MASK:
                printf("Rear Left Bumper hit!\n");
                // Pivot right
                break;
            case RRIGHT_BUMP_MASK:
                printf("Rear Right Bumper hit!\n");
                // Pivot left
                break;
            // Multiple sensors or no sensors
            default:
                if (new_bumps > 0) {
                    printf("Multiple bumpers hit: %d\n", new_bumps);
                    // Take some action
                }
                break;
        }
    }

    return curr_bump_state;
}
```

```
int Check_Light_Events(int prev_light_state) {
    int light_reading = Roach_ReadLightSensor();
    int curr_light_state = (light_reading > LIGHT_SENSOR_THRESH) ? 1 : 0;

    // Detect an event in the light sensor
    if (curr_light_state != prev_light_state) {
        if (curr_light_state == 1) {
            printf("Light level rose above threshold: %d\n", light_reading);
            // Action for bright environment
        } else {
            printf("Light level fell below threshold: %d\n", light_reading);
            // Action for dark environment
        }
    }

    return curr_light_state;
}
```

2. Include a description of the modifications to ES_Configure.h so that the test harness will run your event checkers.

In the ES_Configure.h file there is an enum ES_EventTyp_t and an array called EventNames. To include the new test harnesses the BUMP_SENSOR_TEST and LIGHT_SENSOR_TEST must be added to the objects. They should each be included as enum in the enum and a str for the array.

Additionally, prototypes for the functions must be included in the appropriate service section (1-5).

## 6 Part 6 - Better Event Detection

1. Pseudocode prototype for ”better” event checkers for the bump sensors and the light sensor with debounce and hysteresis bounds.

```
// Bump Sensor Thresholds (Hz)
#define BUMP_SENSOR_THRESH 200
#define BUMP_SENSOR_LOW_THRESH 190
#define BUMP_SENSOR_HIGH_THRESH 210

// Time Variables
// (Note: TIMER will be initialized by the function calling it)
int TIMER;          // (in ms)
#define CLK_CYCLE 5 // 5 ms or 200 Hz cyles

int Check_Bump_Events(int prev_bump_state) {
    // Verify we have waited a significant amount of time before
    if (TIMER < CLK_CYCLE) {
        printf("Not enough time has passed since the last check.\n");
        return prev_bump_state;
    }

    int curr_bump_state = Roach_ReadBumpers();

    // Detect an event in the bump
    if (curr_bump_state ≠ prev_bump_state) {
        switch (curr_bump_state) {
            case FLEFT_BUMP_MASK:
                printf("Front Left Bumper hit!\n");
                // Back up and turn right
                break;
            case FRIGHT_BUMP_MASK:
                printf("Front Right Bumper hit!\n");
                // Back up and turn left
                break;
            case RLEFT_BUMP_MASK:
                printf("Rear Left Bumper hit!\n");
                // Pivot right
                break;
            case RRIGHT_BUMP_MASK:
                printf("Rear Right Bumper hit!\n");
                // Pivot left
                break;
            default:
                printf("Multiple bumpers hit: %d\n", new_bumps);
                // Take some action
                break;
        }
    }

    // Reset the timer
    TIMER = 0;

    return curr_bump_state;
}
```

```
// Light Sensor Thresholds
#define LIGHT_SENSOR_THRESH 200
#define LIGHT_SENSOR_LOW_THRESH 190
#define LIGHT_SENSOR_HIGH_THRESH 210
```

```c
int Check_Light_Events(int prev_light_state) {
    // Verify we have waited a significant amount of time
    if (TIMER < CLK_CYCLE) {
        printf("Not enough time has passed since the last check.\n");
        return prev_light_state;
    }

    int light_reading = Roach_ReadLightSensor();

    int curr_light_state = (light_reading > LIGHT_SENSOR_THRESH) ? 1 : 0;

    int low_to_high = (prev_light_state < curr_light_state) &&
                      (light_reading > LIGHT_SENSOR_HIGH_THRESH) ;

    int high_to_low = (prev_light_state > curr_light_state) &&
                      (light_reading < LIGHT_SENSOR_LOW_THRESH);

    // Handle light transitions
    if (low_to_high) {
        printf("Low to high (bright) environment change.");
        // Action for bright environment
    } else if (high_to_low) {
        printf("High to low (dark) environment change.");
        // Action for dark environment
    } else {
        printf("No transition");
        // No Transition
    }


    // Reset the timer
    TIMER = 0;

    return curr_light_state;
}
```

2. Description of the modifications to ES_Configure.h so that the test harness will run the new event checkers.

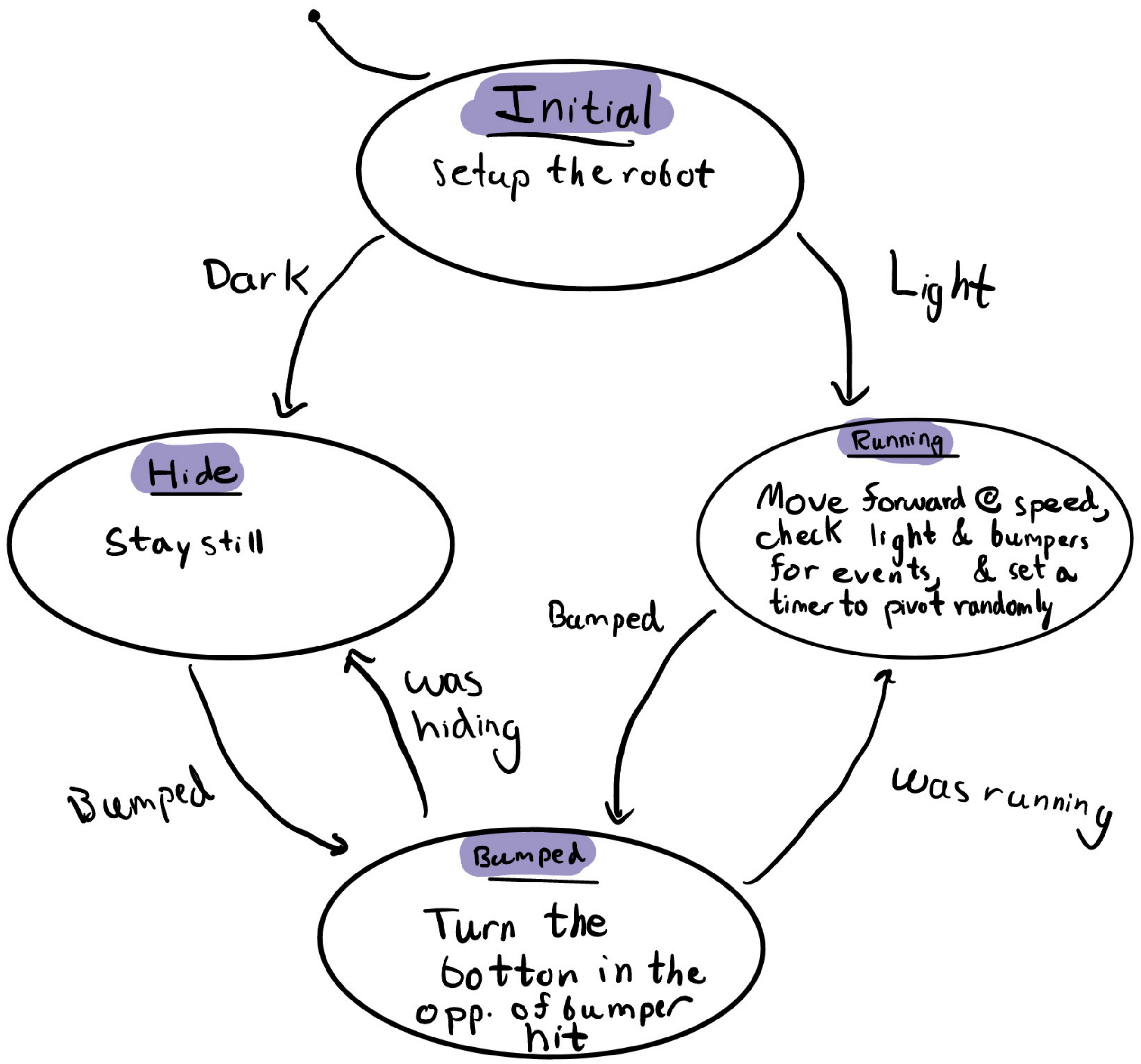
In addition to the changes from part 5, there is a section in the ES_Configure.h for timers:

```c
// ...
#define TIMER0_RESP_FUNC ((Check_Bump_Event)0)
#define TIMER1_RESP_FUNC ((Check_Light_Event)0)
// ...
```

Based on the documentation the timers should be added as such, one per function that needs a timer. Above the pseudocode defines the timer as set whenever the respective event function is called for the first time for simplicity. In actuality, the ES_Config timers should be used to properly time the various event-checkers.

## 7 Part 7 - Finite State Machine (FSM)

1. Discuss the state machine.
2. Create a good drawing of the FSM.



Initial
Setup the robot
Dark
Light
Hide
Stay still
Running
Move forward @ speed, check light & bumpers for events, & set a timer to pivot randomly
Bumped
was hiding
Bumped
was running
Bumped
Turn the botton in the opp. of bumper hit

Figure 1: Finite State Machine for the Roach

3. Create a list of the helper functions you think you will need, with a brief explanation

of what they do (refer to Roach.h for examples).

Helper Functions:

- **Hide**: Roach stays still nothing is being done. Checks the bumpers and light detection.
- **Running**: Moving forward while the light level is above threshold.
- **Bumper** Avoidance: Avoid collisions while the bumpers are activated.
- **Event** Checkers (Light and Bumper): Check for events.
- **Functions(For Movement)** : Includes all the functions for specific movements: pivots, forward, left turn, etc.

## 8 Part 8 - Hierarchical State Machine (HSM)

1. Discuss the HSM
2. Sketch the HSM

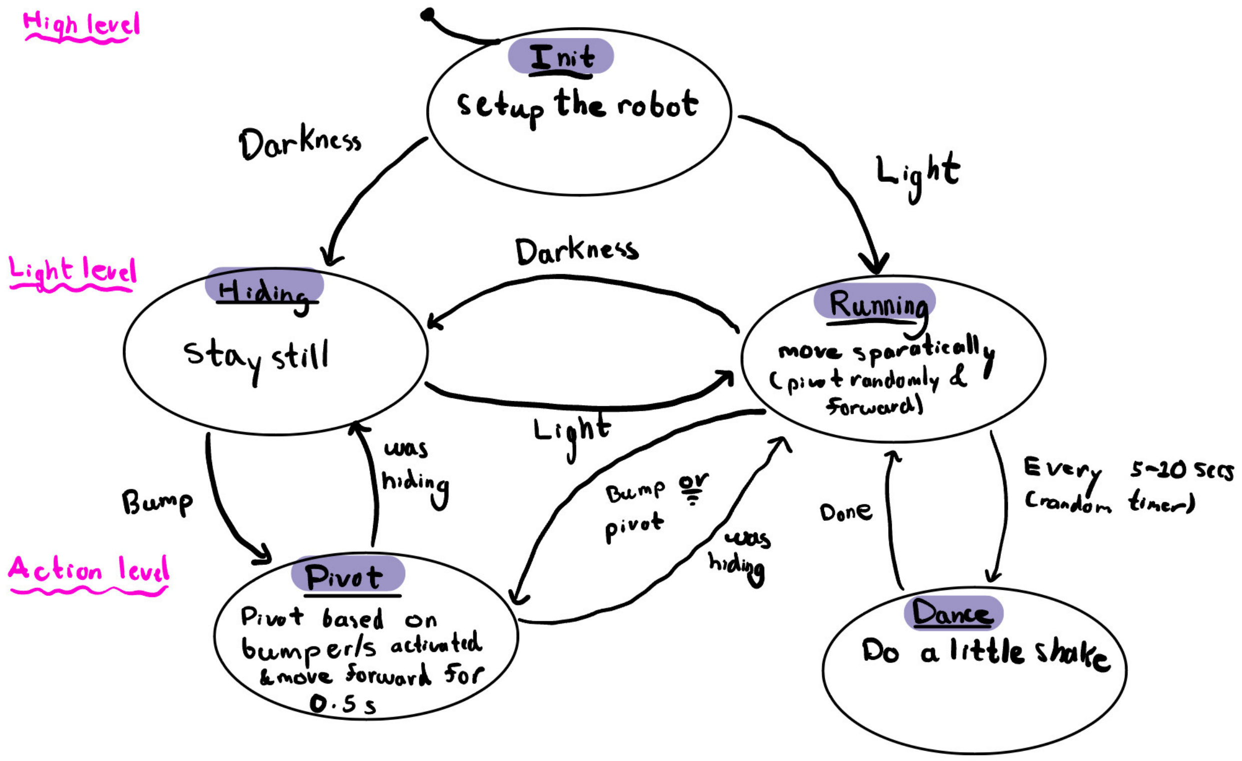


Figure 2: Hierarchical State Machine for the Roach