# Instructions - Laboratory assignment 6 WS22

*Johan Holmgren, 2018-10-06*
*Johan Holmgren, 2020-10-13 - Modification to work with VSCode, PlatformIO, and ESP32* board.

## Overview and purpose of task

In this assignment you will practice designing and implementing a finite state machine. In particular, you will implement a finite state machine for an elevator controller using the function pointer approach.

You will conduct this assignment either indivually or in pairs of two students; however please note that the examination is individual.

## Required hardware components

- ESP32 board
- Breadboard
- Three LEDs
- One push button
- Some cables

## Preparation

Before you start with the assignment, you are recommended to complete the following steps:

- Study the finite state machine lecture material, including the finite state machine implementation that was presented during the lecture.
- Make sure you have all the required hardware components (see above).
- Make sure that your code for assignments 2 and 4 works properly as you might need to use this code in the current laboratory assignment. However, you will most likely need to make some minor modifications of the previous code.
- Read this assignment specification carefully before the scheduled laboratory session. It is possible, and recommended, to start work on this assignment before the scheduled

laboratory session.

# Description of task

As mentioned above, the purpose of this laboratory assignment is that you should implement a finite state machine representing an elevator, which should run on your ESP32 board. The push button and the three LED's will be used as the interface towards the "simulated" elevator, as will be described in more detail below. You should implement your elevator using the C programming language and the function pointer approach.

In addition, you should apply the following assumptions about the elevator.

- The elevator should operate in a house consisting of three levels: UPPER. MIDDLE, and LOWER.
- Only one person can travel in the elevator at the same time.
- Each traveler has an origin level and a destination level. For simplicity, you do not have to simulate that the travelers push buttons inside the elevator.
- The elevator should serve the travelers in the order they pushed the outside button.
- If the elevator is full and there is people waiting to enter, it is assumed that these persons will push the button again and wait for the elevator.
- It always takes 5 seconds for the elevator to travel between two adjacent level.
- It takes 5 seconds from when the elevator stops until it is ready to go again. During these 5 seconds, the passengers enter and leave the elevator.

Please note that in case there are missing specifications regarding the operation of the elevator, you are expected to make your own assumptions, which you should clearly document in the code.

You will use your three LEDs to show the current location of the simulated elevator. Each level is represented by one LED and the LED representing one of the levels should be switched on only when the elevator is located at the level it represents. All three LEDs should be switched off when the elevator is traveling between levels.

To make it easy to follow the movement of the elevator, you need to position your LEDs in a straight line (upper, middle, and lower) on your breadboard in a way that the middle LED represents the middle level and the two other LEDs represent the upper and lower level, respectively.

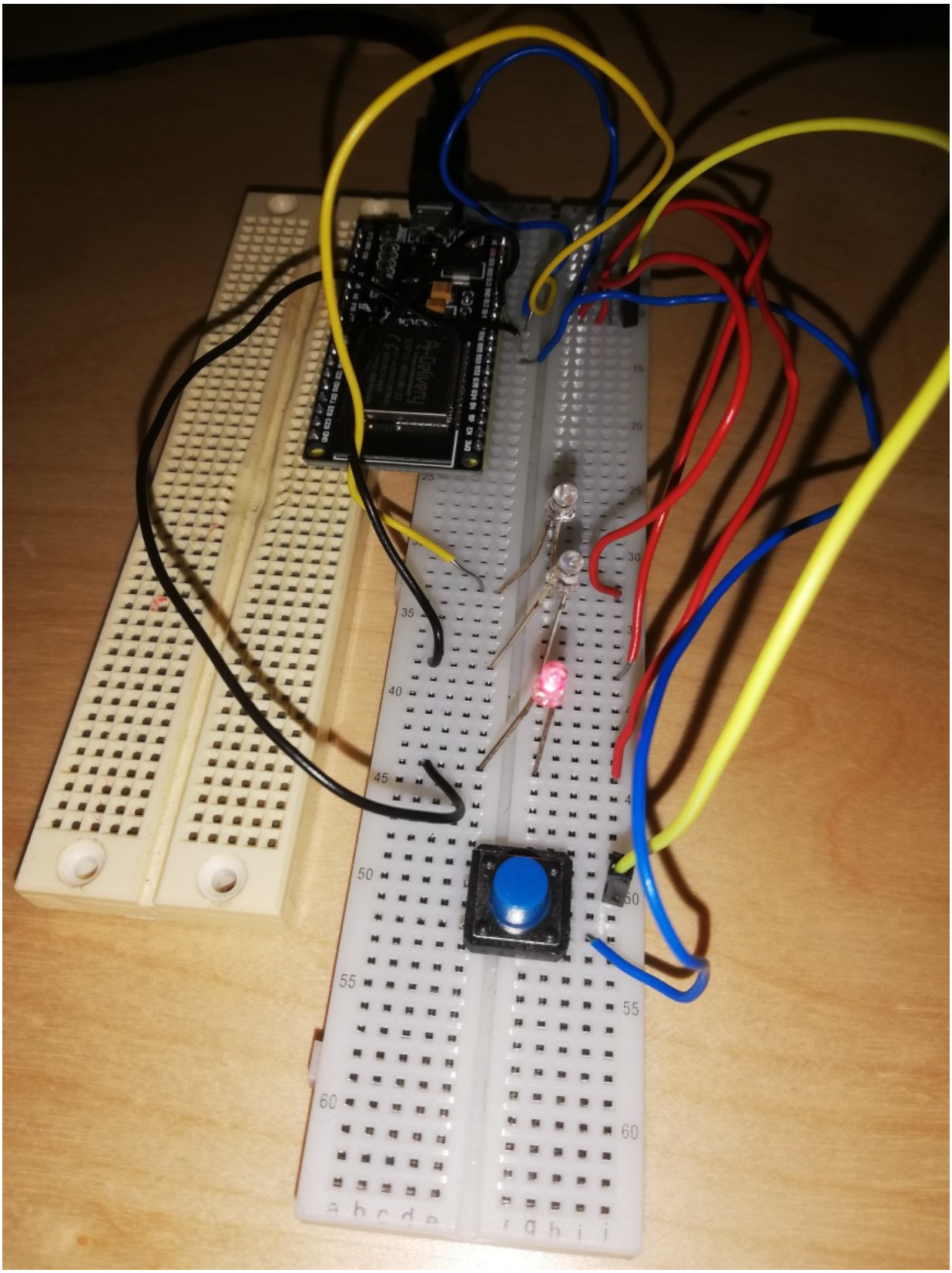
The push button is used to simulate that a traveler pushes the button outside the elevator. For simplicity you are provided a list of 50 randomly generated travel needs (see startup code), each consisting of:

1. The level the traveler wants to travel from (i.e., origin)

2. The level the traveler wants to travel to (i.e. destination)

Each time the button is pushed, you should proceed to the next travel need in the provided list of travel needs. This means that you need to keep track of the travel needs that have been generated (using the push button) but not yet served by the elevator.

See the image below for an illustration of how the components might be positioned on the breadboard. In this image, the LED connected closest to the EPS32 board (connected to PIN 12) represents the upper level, the LED in the middle (connected to PIN 14) represents the middle level, and the LED connected closest to the push button (connected to PIN 27) represents the lower level. The push button is connected to PIN 26.



To help you get started, you are provided a visual studio code project using PlatformIO with the following settings:

- platform:  espressif32
- board:  nodemcu-32s
- framework: espidf

- monitor_speed: 115200

In addition, you will get some source code to help you get started. You find the "startup code" in a zip file on Canvas.

### Testing your solution

In addition to designing and implementing an elevator controller, you also need to test your solution. You can test your solution in many different ways, and it is up to you to decide how you should test your solution.

As mentioned above, the elevator is driven by button pushes, or rather passenger travel wishes, which are included in the provided source code. As part of your testing, you might want to create your own test cases by modifying the provided list of travel needs.

## Submission and examination

When you are done with the assignment, you should upload the `src` folder of your visual studio code project (in a zip file) using the submission page for assignment 6 on Canvas. You should also include in your zip-file a short report where you graphically illustrate and briefly describe your finite state machine.

You should also demonstrate and discuss your solution with the examiner of assignment 6, which you do during one of the scheduled presentation times for laboratory assignment 6 (examination times are published on Canvas). During the discussion, you should be able to show that your code works correctly, explain the code, and answer some basic questions about your solution. Please make sure that the original list of travel needs is available in your code during examination.