Applying Threads

We want to make a program that can control a number of appliances by starting Scenes, where a Scene is a pre-programmed sequence of operations on those appliances.

As a running example, let's consider a Garden Automation system, with a number of Sprinklers for watering a garden.
For the sake of simplicity, we assume that each Sprinkler can be controlled separately.

An example of a Scene in this system could be:

- Switch Sprinkler 1 on,
- Wait 5 minutes,
- Switch Sprinkler 1 off,
- Switch Sprinkler 2 on,
- Wait 5 minutes,
- Switch Sprinkler 2 off.

This application must run on the ESP32, using multithreading with FreeRTOS.
The minimal required functionality is:

- Store at least 3 different hardcoded Scenes on ESP32, and give each a unique number (1, 2, 3, …).
  All scenes should use different Sprinklers, so for example:
  - Scene 1 uses Sprinklers 1 and 2,
  - Scene 2 uses Sprinkler 3, and
  - Scene 3 uses Sprinklers 4 and 5.
- Enable the user to start any of these Scenes at any given moment by sending a command over the serial port from the laptop to the ESP32.
  This command can be as simple as sending just the number, so sending "3" will start execution of Scene 3.
- It must be possible to start several of these Scenes concurrently, so for example: if the user types '3', Scene 3 will start running, and if the user, two minutes later, types '1', then Scene 1 will start running, while Scene 3 is also still active.

To accomplish this, each time when the user starts a Scene, a thread must be created that performs the given operations for that Scene.

It is suggested to develop the program in this order:

1. Decide how many Sprinklers you will have (at least 3) and mimic the Sprinklers with LEDs: one LED per Sprinkler.
   Make dedicated functions to switch such a Sprinkler/LED on and off.
2. Then think of a way to store the operations of a Scene; for example: the earlier mentioned example Scene could be stored in an array of 6 strings, like:
   [“sp1 on”, “wait 5”, “sp1 off”, “sp2 on”, “wait 5”, “sp2 off”]
   Any other logical and efficient way of storing a Scene is also fine.
3. In this way, make at least 3 different Scenes, with numbers 1, 2, 3, etc., and store them hardcoded on the ESP32.
4. Make a function executeScene that can read the operations of a specific Scene from the array, and perform these operations.
   Use the function(s) you made in step 1 for switching the Sprinklers/LEDs on and off.
5. Test this function executeScene by calling it directly from main (so not using threads yet) with different Scenes; check out the LED’s to make sure that it works well.

Then we add the multithreading:

6. Make one thread, that does nothing else than repeatedly read the UART, and if it receives a number, then echo that number back to the laptop.
   This thread should never end.
7. In that thread, when a valid number of a Scene is received from the user, start another thread that performs the operations of that Scene by calling the function executeScene that you made in step 4.
   Make sure that that the thread ends itself in a controlled way when the Scene is done.
   For testing purposes you probably want to use seconds in the Wait operations, instead of minutes (so “wait 5” means wait 5 seconds instead of 5 minutes).
8. Test your program with multiple Scenes running at the same time.
   For example: first type ‘1’ and a while later type ‘2’; Scene 1 and 2 should then run together, which you can check by looking at the LED’s.

This application will be extended further in the next assignment. This one will therefore not be graded separately, but only the end result will be graded.

In the end, also a document will be required. In that document you have to describe what you did. For this week, we expect the following in the document:

- Which function(s) did you define in step 1 to switch the Sprinklers/LEDs on and off?
  Give the heading and description; full code is not needed.
- In what way are you storing the Scenes, and why did you choose to do it like this?
- Which function did you define in step 4 to perform the Scenes?
  Give the heading and description; full code is not needed.
- How did you test the program?
- Any special problems you had, and how you solved them.
- Optional: describe any special things that you added to the application, which were not mentioned in the minimal requirements above.