**Communication Challenge 2: FreeRTOS**



A screenshot of a computer

Description automatically generated

*Circuit used: red wires are to simulate power to the component that is connected to its designated pin. The black wires are to simulate being connected to the GND pin. Each LED represents a sprinkler and they are connected to my ESP32 as show in the above circuit.*

Student: Johnson Domacasse

Date: 11 Okt 2023

Student#: 4471709

Teacher: Hans van Heumen

## Introduction:

## Procedure:

I begin by setting up what I believe to be my circuit for this entire challenge. See above circuit diagram. I have set up the pins in the

### Setup:

|  |  |
| --- | --- |
| **COMPONENT** | **PIN** |
| LED\_1\_PIN (Sprinkler 1) | D14 |
| LED\_2\_PIN (Sprinkler 2) | D26 |
| LED\_3\_PIN (Sprinkler 3) | D21 |
| LED\_4\_PIN (Sprinkler 4) | D22 |

*Table 1: Hardware Pins*

### Design choices thread:

Based off of the requirements that were given to me, I came up with the following key conclusions:

* Each sprinkler own designated LED and a function to control turning the LEDs on and off.
* Store the operations of scenes. They are hardcoded and stored in my application
* Make a function to execute a scene, by reading these operations of the hardcoded stored scenes and perform said operations.
* I need to test this function. First without threads.
* Make on thread, that does nothing but read the UART repeatedly. If it receives a number (in this case representing a scene), then end echo this number back to my laptop. This thread should never end.
* In that same 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 was defined before.
* Make sure that that the thread ends itself in a controlled way when the Scene is done.
* Test my 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.

### Thread implementation:

Now that I knew my requirements I began to work. I first defined my LED pins to simulate the sprinklers. I then made the predefined scenes. These are hardcoded arrays with messages to simulate its behaviour. I then made a simple LED controller function that takes the LED pin and the desired state. Then I implemented the executeScene function. My implementation of this is not as optimal as I would like it to be. For the time being, it’s a simple for loop, that reads the contents of the array that was given to it, and then perform the operation if it recognizes one of the operations. Finally in my loop I call that function 3 times, to test all 3 scenes. The only issue I see with this implementation is that I need to implement delays in the main function for it to have my desired behaviour.

Now I will work on the thread part of this implementation.

### Design choices mutex:

### Mutex implementation:

### Design choices queue:

### Queue implementation:

## Results:

## Issues:

## Conclusions: