

Project

Real-Time systems [ELEC-H-410]

Pandemic, the RTOS game

2019–2020

In order for you to work remotely, we changed this year project into a game. The game simulates a pandemic situation, very much like the one we are seeing right now. As you have seen, real-time response is a critical component of handling a pandemic efficiently.

It is by no means our intention to take the current epidemic lightly. However, we thought that this project assignment would allow to lighten the mood a little, as everyone's morale will also be taking a serious hit during this period of confinement. Keep your spirits high and make sure you give your courses your best (all of them, not just this one). We will need talented and bright engineers in the aftermath of this crisis.

To play this game, you don't need any equipment other than the PSOC itself. The game is entirely coded inside one FreeRTOS task (*gameTask*). It sends "events" and we ask you to make a code that will automatically respond to those "events".

Depending on your response time you will increase or not some variables.

Game description

In this game you will try to beat the current pandemic propagation by researching for a vaccine, producing medicine and containing sudden contamination burst. The only objective is to find a cure before the population get totally infected.

- The percentage of healthy population, `populationCntr`, starts at 100%. **You lose** when it reaches 0%.
- The vaccine counter, `vaccineCntr`, starts at 0%. **You win** if it reaches 100% before the whole population is infected.
- The amount of medicine pills available (in million), `medicineCntr`, will help you to slow down the disease propagation.

You have at your disposal one research lab. It can either work on a cure or produce medicine. It cannot do both at the same time and when it starts a job it should work until completion¹.

There are three types of event

Vaccine Research Every 3s, the *gameTask* will drop a *clue*. If you manage to make the lab work on the *clue* and return its result within 3s then the `vaccineCntr` is increased by 3.

Spread of the disease Every 5s, the percentage of healthy population is decreased by 5–`medicineCntr`. The pills can only be used once. This is why you should keep producing them. When it is not working on a *clue* the research lab should produce as many pills as possible.

Contamination The *gameTask* will randomly contaminate an individual. If you do not *quarantine* the population within 10ms, 20% get infected instantly.

¹The lab is a shared resource.

Specifications

Open the project `pandemic`.

The whole game is coded in `pandemic.c`. Read its documentation in the header file (`pandemic.h`) to better understand how to use it.

You should only write your code in `main.c`. You may use as many task as you want as long as their priority is lower than the `gameTask`.

Write a task that print the `populationCntr`, `vaccineCntr` and `medicineCntr` on the left side of the display. The last 8 characters of the LCD are reserved to the `gameTask`.

When the `gameTask` sends a contamination it calls the `releaseContamination()` function. You should override this function in `main.c` in order **to communicate with your own tasks**. The `quarantine()` function should be launched from your own task. **The following is strictly forbidden** (otherwise it would make things too easy):

```
|| void releaseContamination(){
||     quarantine();
|| }
```

The `gameTask` releases clues for the vaccine in a similar manner, (see `releaseClue(clue)`). You should again override this function in the `main.c` in order **to communicate with your own tasks**. When this happens your code should call the functions `assignMissionToLab()` and `shipVaccine()` from within your own tasks.

```
|| // [...]
|| vaccine = assignMissionToLab(clue);
|| shipVaccine(vaccine);
|| // [...]
```

To produce medicine pill use:

```
|| // [...]
|| medecine = assignMissionToLab(0);
|| shipMedicine(medecine);
|| // [...]
```

Guidelines

During the labs, you have seen how to handle inter-task communication, resource sharing, synchronisation, ... Use this knowledge in a clever way.

We ask you to work remotely, you don't need physical access to the device to contribute. We advise you to use Microsoft Teams as you all already have an account and you can use the Remote Desktop Control feature to compile and run code from a distance. If for some reason that doesn't suit you, keep in mind that you only need to share the `main.c` file to code. A simple Pastebin² might do the trick.

Start by making a list of all the task that you need. Then use timing diagrams, block diagrams, state machines or other operating/functional schemes to share your ideas with your team.

Good luck!