TTK4145, exercise 5: **Module design**

Hanne-Grete Alvheim, Maren Keini Haugen,
Iselin J. Nordstrøm-Hauge

February/March 2021

Contents

1 The implementation

1.1 Modules in Go, general modules and ”our modules”

In Go, modules are a collection of related packages that are put together as a single unit. These are versioned together semantically. For our project, we believe implementing this will be a bit excessive, since there will not be ”a lot” of packages with source files, and there will not be ”a lot” of different versions.

Generally, modules are containers of ”something that does something”. A module should have everything necessary to achieve a certain functionality. With this in mind we can view functions, goroutines, source files and packages as modules. For us to describe our system by referring to all of these as modules will give little information about what we are actually planning to do. We will therefore be using go packages as our modules.

1.2 Interaction between modules

The modules are a part of the same software system. Packages in go are like directories, see Figure 1, and we plan to make one package for each extensive functionality in our system. These packages will contain related source files, meaning that their contribution to our total system is recognised as similar. The packages will interact with each other through imports. Furthermore, the content in the package source files will communicate with other packages' source file content mostly by channels. This since most of the content will be goroutines, and gouroutines share memory by communicating through channels.

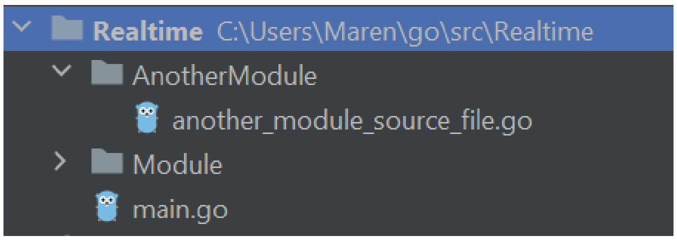

Realtime C:\Users\Maren\go\src\Realtime
AnotherModule
another_module_source_file.go
Module
main.go

Figure 1: What we will use as our modules.

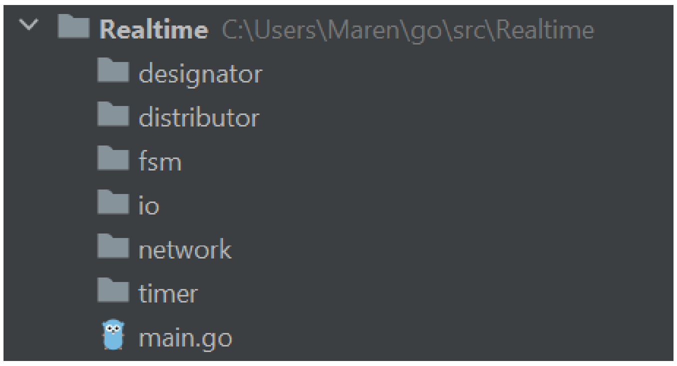
2 The contents

2.1 The modules

The modules we need are:

- Order designator module
- Order distributor module
- FSM module
- I/O module
- Network module
- Timer module

How the modules will be presented in our project can be seen in Figure 2.



```
Realtime C:\Users\Maren\go\src\Realtime
    designator
    distributor
    fsm
    io
    network
    timer
    main.go
```

Figure 2: Modules in our project.

When planning our modules we have thought about functionality, and named them accordingly. The designator module will take care of designating orders (cost function etc.). The distributor module will take care of distributing orders. The FSM module will take care of the states of the node. The I/O module will take care of hardware functionality, and thereby inputs and outputs to the node. The network module will take care of networking, sending and receiving messages etc. The timer module will take care of the timers for different processes.

Additionally, because we have chosen a P2P network topology, where the node who gets the order on its hall panel becomes the ”order assigner”, and all the nodes needs the same modules. Also, we want to keep as much code out of the main script as possible. This is something we keep in mind while designing the modules.

2.2 Inputs, outputs and states of the modules

An overview of inputs and outputs for our modules can be seen in Figure 3. Inputs, outputs and states for all modules are also stated with bullet points below.

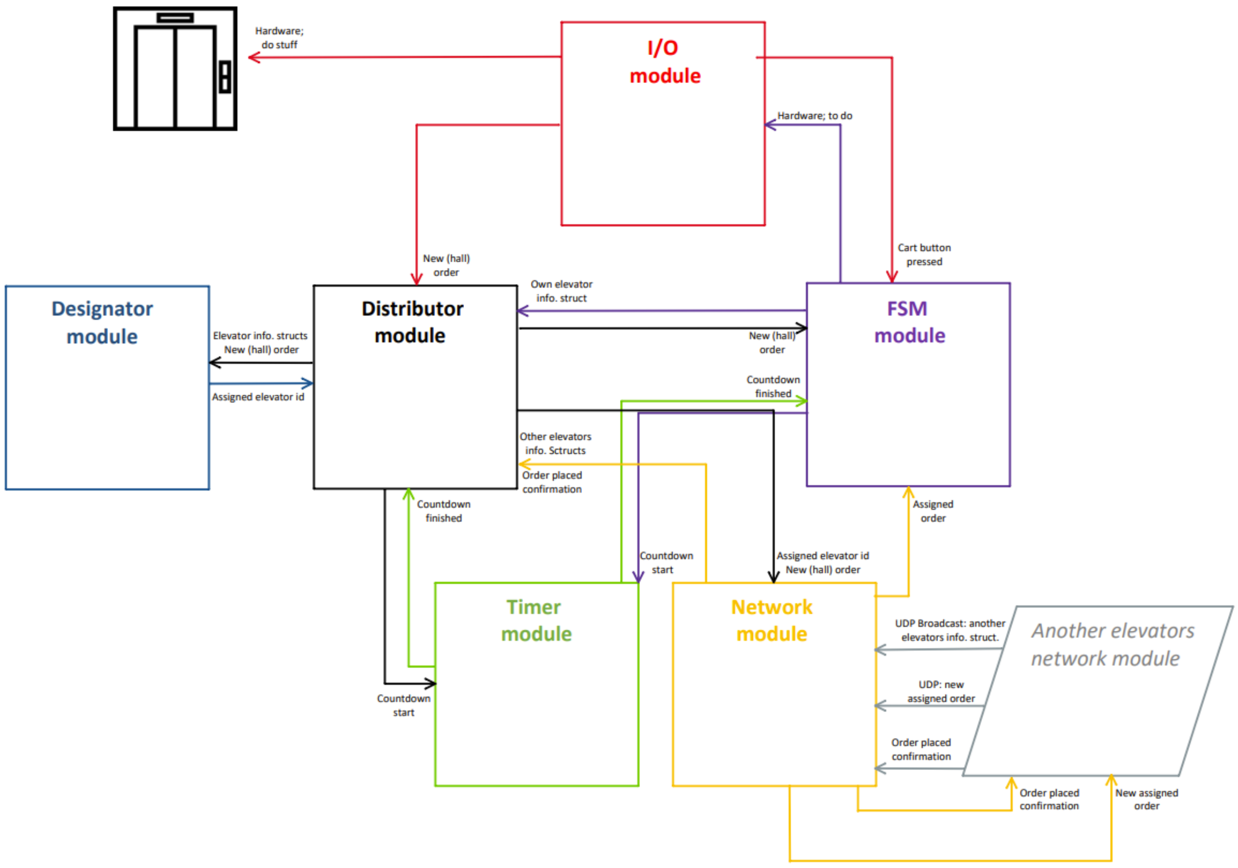


Figure 3: Inputs and outputs for our modules.

We are playing to the strengths of Go by using packages and channels to communicate data between (and within) our modules.

Order designator module

Inputs:

- Elevator info. structs [from distributor module].
- The new order [from distributor module].

Outputs:

- Elevator id [to distributor module].

States:

- No state.

Order distributor module

Inputs:

- New hall order [from I/O module].
- If changed: Its own elevator info. struct [from FSM module].
- Constant input: The other elevators info. structs [using the network module].
- Assigned elevator id [from designator module].
- order placed confirmation message [through network].
- countdown finished [from timer module].

Outputs:

- New hall order + elevator info. structs [to designator module].
- If assigned elevator != itself: Assigned elevator id and the new hall order [with network module].
- If assigned elevator == itself: new hall order [to FSM module].
- start countdown [to timer module].

States:

- Update on its ”own” elevator info. struct from the FSM module → Update stored elevator information.
- Update on another elevators elevator info. struct using the network module → Update stored elevator information.
- New hall order through I/O module → use designator module.
- Designator module assigned order to its ”own” elevator → send order to its own FSM module.
- Designator module assigned order to another elevator → send order using network module.
- Order sent using network module → Checking for order placed confirmation message from network.

FSM module

Inputs:

- Assigned order [from its ”own” distributor module].
- Assigned order [through network module].
- Countdown finished [from timer module].

Outputs:

- If changed: Its own elevator info. struct [to order distributor module and through network module].
- What to do with hardware [using I/O module] (motor dir., lights off/on etc.).
- Countdown start [to timer module].

States:

- States for elevator behaviour.
- Combine input output here, if something is stored.

I/O module

Inputs:

- What to do with hardware [set by FSM module].

Outputs:

- Do stuff with hardware.
- New hall order [to order distributor module].
- Cart button pressed [To FSM module].

States:

- Elevator fuctionality (up/down, doors open/closed etc.).

Network module

Inputs:

- New assigned order [from another elevators network module].
- New order to send + id for receiving elevator [from distributor module].

- Other elevators elevator info. structs [from other elevators network modules].
- Order placed confirmation message [from other elevators network modules].

Outputs:

- New assigned order form another node [to FSM module].
- Other elevators info. structs [to order distributor module].
- Order placed confirmation message [to other elevators network module].
- Order placed confirmation message [to Order distributor].
- New assigned order [to another elevators network module].

States:

- No state.

Timer module

Inputs:

- Start countdown [from order distributor].
- Start countdown [from FSM].

Outputs:

- Countdown finished [to order distributor].
- Countdown finished [to order distributor].

States:

- Countdown.