Technologies for Connectivity: Guardian system design

Lisa Booms
Eindhoven University of Technology, the Netherlands
l.m.booms@student.tue.nl
s166434

Reka Magyari
Eindhoven University of Technology, the Netherlands
r.magyari@student.tue.nl
s167006

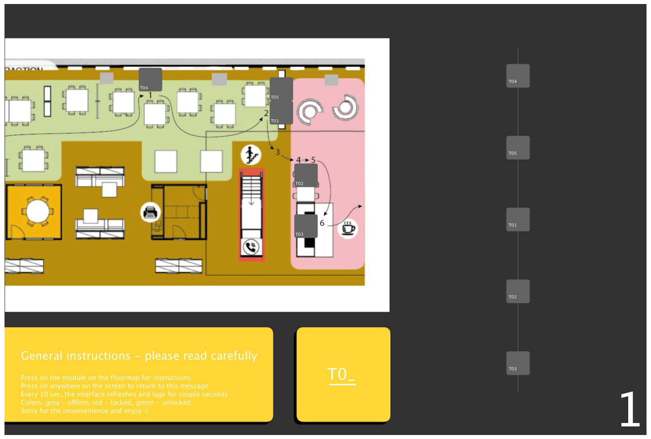
Bas Lazlo van der Meer
Eindhoven University of Technology, the Netherlands
b.l.v.d.meer@student.tue.nl
s165438

Glenn Mossel
Eindhoven University of Technology, the Netherlands
g.i.mossel@student.tue.nl
s165438

Zihao Yang
Eindhoven University of Technology, the Netherlands
z.yang@student.tue.nl
s144209

Challenge description

The phantom thief is walking towards the elevators of Atlas. As he heard all six infinity stones are now to be found on the second floor of this building. Stepping into the elevator he starts his tablet which will help him get through all the obstacles there will be before he can get to the stones. The second floor, also called the Tyria world, contains of five modules serving as a protection system for the infinity stones. Each module uses different technologies and require various trigger actions to unlock the system. The tablet with the interface shown in Figure 1.1 is guiding the phantom thieves through each of these technologies and let the thief trigger the system which unlocks the module. The main part of the interface is the map where all the modules are indicated with a red square.


General instructions – please read carefully

Press on the module on the floormap for instructions.
Press on anywhere on the screen to return to this message
Every 10 sec, the interface refreshes and lags for couple seconds
Colors: grey – offline, red – locked, green – unlocked
Sorry for the inconvenience and enjoy :)

Figure 1.1 - Interface of the tablet

When starting the system this interface will provide you with a route of the three routes made possible. The route will be displayed with arrows from module to module. These arrows indicate the route through all the five modules to fulfil. Below the map there is a small description of the actions that need to be done. When arriving at the first module the name of the module (e.g., T-01) is shown and a small explanation is displayed besides this to guide the phantom thief. The last component of the interface is on the right of the screen and shows the phantom thief what the current state of the level is. At the beginning all the squares are red, just as the squares in the map. When a module is offline the square turns to grey. When a module is unlocked correctly, the square turns into green and gives the thief the notion to go to the next module. When finishing all the modules they will turn green one by one and when they are all unlocked the thief will be able to grab the infinity stones from under the kitchen sink.

The use of other modules in Tyria world

Before going more into depth oF our module, first the other four modules will be explained. Most of the other groups worked with movement sensors while we opted for a sound sensor and another group picked an infrared sensor. The modules will be explained in order of the Group names, but in the routes the modules are in a different sequence. Group T01 designed a module containing a mug with a movement sensor inside. The action to trigger the module was to shake the mug three times, but not too quickly. To do this unsuspiciously the thief could act like taking a sip out of the mug or to act like to check if there is anything inside the mug. Both movements are not too suspicious and the sensor was easy to implement. The mug was placed inside the box of our module to create a connection between the different modules.

Group T02 was the second group to use a movement sensor which they implemented beneath the high table on the side of the kitchenette. They placed two sensors within a distance of 1 meter. To unlock this module the phantom thief should stand before the first sensor for four seconds, walk to the second sensor and stand before this one four seconds as well. This movement sensor was quite sensitive to the surface that was standing before it, so it was recommended to use a flat surface like a book or lid to hold before the sensors. Not making this movement obvious could be done by checking your phone or tablet.

Group T03 used an infrared sensor in the kitchenette itself. This sensor reacted to objects in front of it. It would be unlocked by a certain programmed time sequence of standing in front or away of the sensor. This time sequence to trigger the module was to stand before the sensor for about ten seconds, then move away for four seconds and return to the sensor again for six more seconds.

Group T04 decided to use a simple screen which they used in a small box on the wall. There was a question displayed on the screen which asked the question "Books or movies?". There were two buttons in the box as well which gave the option to push a button for either one of these two answers. Which answer the phantom thief pushed the button for does not matter at all, it was just the push on the button alone that was needed to proceed. After following the instructions on the LCD screen there were one, two, or three dots presented. How many dots were shown determined what action could trigger the module to unlock. When seeing one dot the thief should press the left button three times. When seeing two dots on the screen, the right button should be pussed on three times. And when there are three dots shown, the thief has to push the left button one, followed by the right button once and ending with pussing the left button once more. When unlocking all these modules, including our module, the infinity stones could be picked up inside the cabinet of the kitchen.

For completing the whole world level a lot of collaboration was needed between the different groups. The first thing to make sure was that every group had a functioning code for the individual modules with which they could unlock their module and most important, return a true boolean value. All the communication between the modules was going to be identical, since there would be randomized routes, so the world decided that a few people from different groups should focus on making a networking code in arduino. This code was made in a way so that every single module-code could be implemented into the rest of the code. All the information was sent to an OOCSI channel. To visualize the data received, a processing code was made as well which shows whether a module is unlocked, locked, offline or online. This could be seen for every module separately, so the thief could notice which module worked and which not. In order to maintain functionality, the networking code was made such that it could work completely independent without any external help and would be ready for any errors in the network. To make sure the thief is not getting stuck if a module does not work, there is always checked which modules are off- or online in the route. The last thing decided for this networking world code, was that the security should be maintained. To do this, the system will not be able to be unlocked if two or more modules are offline in the route.

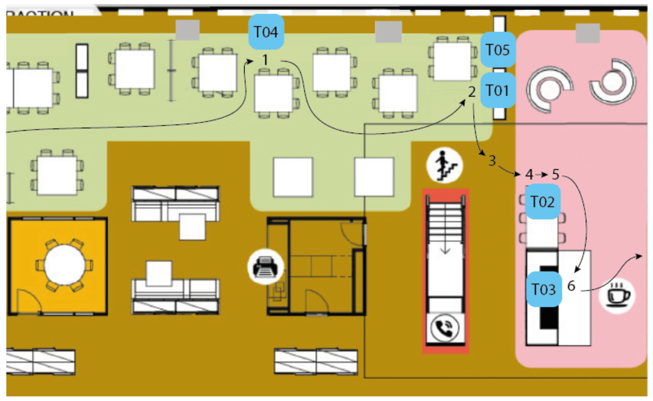


Figure 2.1. floorplan made by other team

Description of our sound module

After naming the other four modules now the module of T05, using a sound sensor, will be explained. To contribute to the world challenge of Tyria, our module from Challenge 1 will be used and redefined in order to fit the purpose. The module includes a microphone, which can detect sound and distinguish among them based on set thresholds. Therefore, the challenge required us to come up with a natural sound action that could trigger the microphone sensor. Hence the brainstorming resulted in the decision of the required action. First this decision was to use a coughing sound, but this turned out to be too hard to trigger. Thus we decided to use the lid to create pressure inside the box when closing the lid. Causing the membrane of the microphone to move more than what other "natural" noise could cause. These boxes are assigned to students of Industrial Design and standing on shelves on floor 2. The sound module measures the sound of the environment on the shelf. Compared to the previous challenge, the module does not store or transform any data anymore, but only compare the measured value to a value that represent the sound of closing the lid of the box. The module will only start "listening" whenever it should be interacted to be unlocked. This means whenever the previous module is unlocked, it will push a notification to our module to start "listening".

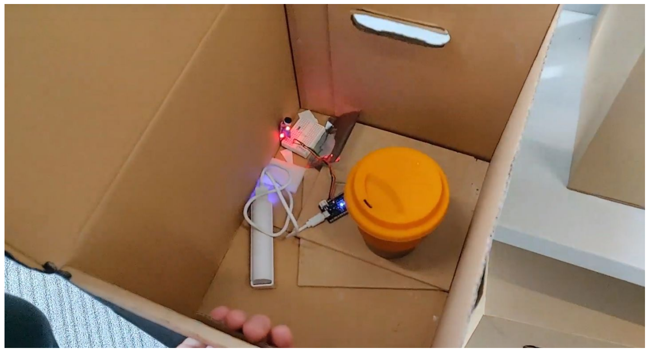


Figure 3.1. pictures of physical setup and monitor result difference of coughing and slamming

As for contribution to set up the system, a collaboration with Kirsten from T02 and Sylvan from T01 has been set up to be responsible for the communication protocol.

The communication protocol of the Tyria world is created as a decentralized system. This means that each module is equally responsible for contributing to a global chain of behavioral actions; in order to unlock the system. The decentralization of the system ensures that the system continues to work when one module is not working and therefore not able to contribute to the system.
The system does this by having each module check the OOCSI server to see if the other modules are connected to it. The return value is stored locally on the module and is then acted upon.

Because of this, the dependency of each module is minimal. Not to forget that each module is designed to be unlocked individually. The idea behind this system is that if one of the five modules is offline, the next module in the route can be unlocked. However, to improve the security of this system, a set of rules has been made to prevent unlocking the system when two or more modules turned are offline. In that case, the system will make sure that the other modules will fall back to a “soft shutdown”, meaning they will not be able to be unlocked.

API description of our module

The plan to define the threshold of closing the lid of the box is to manually create a set of measurements by using a custom code. This would also be done with additional “loud” noises such as the door slamming or people shouting to prevent false positives. For detecting the lid-closing sound, the module will simply match up the value with the predefined threshold value. All of this is put in a boolean function called module. Unlocked and returns a boolean of whether it is unlocked or not. The module is unlocked, it sends the unlock state to the oocsi channel as true . The module will only execute this function when

The whole world decided to use OOCSI for communication. To keep the communication with the other modules consistent, all teams agreed upon the following naming for consistency:

```
OOCSI channel:           “Tyria_World”
OOCSI name:              “T05”
Module unlock state:     (“T05_Unlocked”,
true)
Current route:           (“route”, 0)
World unlock state:
(“World_unlocked”, false)
```

Within the communication API, the modules check which other modules are online. They do this by sending a contains Client message to the server. If two or more of the other modules are offline, the module will stay idle and wait for the other modules to come online. If four or more modules are online, the module will check its own position in the route. When it is the module’s turn, the module will go into an active state and will be able to be unlocked. After finishing the module’s unlock pattern, a message is sent to the OOCSI channel, indicating a successful unlock.

In the rare case that the module a module offline, the module right after that module will notice that (from checking the online status) and go into the active mode itself. The route will then be continued skipping the module before.

The representation of the user interface as shown in Figure 1.1, will be done through processing with its functions to display images and a video. The interface is connected with the OOCSI for receiving the events of the module status (locked/unlocked/offline/World_unlocked/route/Setup_finished) through:

```
if ( !T01_unlocked )
      T01_unlocked =
event.getBoolean("T01_unlocked", false);

currentRouteNumber = event.getInt("route",
0);
```

```
World_unlocked =
event.getBoolean("World_unlocked", false);

//checks whether the module has gone through
the setup phase
if ( event.getSender().equals("T01") )
Setup_finished[0] =
event.getBoolean("Setup_finished", false);
```

Also the interface checks every 10 second whether every module is online or not. This way of information collection is depend on the module communication method.

```
T01_online =
oocsi.getClients().contains("T01");
```

Code with its documentation of the communication protocol can be found in the github link down below:
https://github.com/Glivmo/OOCSI-Networking

Code with its documentation of the interface can be found in the github link down below:
https://github.com/danyng/OOCSI-System-UI

Also the code of the simple UI that was used by others during the testing and demo:
https://github.com/danyng/OOCSI-Simple-UI

Demo-day

Networking

On the demo day, a few problems were encountered. Firstly, the module of group T03 would not work correctly with the communication protocol. This was most likely due to overlapping variable names that were not communicated correctly. This module was therefore not used in the route and had to be manually activated.

The second problem was encountered in the second route. The first route was executed correctly. After completing the first route, all modules should have restarted and started the second route. However, for the first module in the second route the battery was depleted right after restarting. This caused the second module in the route thinking that the first module was online, which resulted in that the second module would stay in the inactive state. This caused that the entire system could not be unlocked. To encounter this problem a manual failsafe should have been created.

Interface

What went wrong was that the interface oocsi key name for the route number was not consistent with the modules.

```
currentRouteNumber =
event.getInt("currentRouteNumber", 0);
```

instead of:

```
currentRouteNumber = event.getInt("route",
0);
```

- Not enough visuals of the states of each module, thus not clear of what was happening and which module was currently “active”.

- Not enough general instructions on how the system works. Assumed that the thief already knew how to unlock it.

What was going right:

- The colors, order of the sequence and instructions were clear
- The interface was responsive when it didn't had to refresh

Video

Link to video: https://youtu.be/GFJeIFOpsok