Arquitectura de Sistemas
Grado en Ingeniería Telemática
Grado en Ingeniería de Sonido e Imagen
Grado en Ingeniería de Comunicaciones
Móviles y Espaciales

# Project: Wifi Collector
# 2022-2023

## 1. Resources

- Notes on how to work as a team
- Style guide
- 21 files with input data: "`info_cell_NN`" where NN ε {1,2,…,21}

## 2. General description

Design and implementation of an application that collects data about the wifi networks detected by a mobile device.

SAUCEM Inc. receives a customer request to design an application to collect information about the Wifi networks detected with a mobile device while used in open spaces. The customer wants an application that the user is executing while moving through a place and in certain times, it scans the available Wifi networks and stores information about their features. Additionally, the application allows the execution of several operations over the set of obtained nets.

This application is needed to be able to traverse public spaces on which wifi networks are deployed and create a coverage map for these networks, the signal quality and the number of them that are available. *The information captured with the mobile device will eventually be stored in data files to be suitable for processing by other tools*. The information obtained for each access point to a Wifi network is as follows:

- **Cell identifier**: It is a natural number.
- **MAC (Media Access Control) address of the access point**: It is a unique identifier assigned to every communication interface, which in this case is the access point offering the Wifi coverage. These addresses are usually assigned by the manufacturer and encoded in the hardware. This data is represented by **six hexadecimal numbers of two digits each**. When represented as text, each of the six numbers are separated by ":" or "-". For example:

  `01-23-45-67-89-AB`

  or

  `01:23:45:67:89:AB`

- **ESSID (Extended Service Set Identifier)**: Name that identifies the wireless network that the access point is offering coverage. Several access points with different MAC may offer coverage to the same network. The name is enclosed between "s. For example:

  `"PTNET"`

- **Mode**: There are various configurations for access points to wireless networks. The application to create considers the following:
  - `Auto`
  - `Ad-Hoc`
  - `Managed`
  - `Master`
  - `Repeater`
  - `Secondary`
  - `Monitor`
  - `Unknown`
- **Channel**: A natural number indicating in which channel (frequency) is the signal to access the access point.
- **Encryption key**: Field that encodes if the communication with the access point is encrypted. Possible values:

  `on`

  `off`

- **Quality**: Two natural numbers:

  `n1/n2`

  where `n1` represents the actual signal level, and `n2` represents the maximum level expected.

There are additional fields that can be obtained from an access point, but the application to implement has only to consider these ones.

As an example of the information recolected by the mobile device about one access point we have:

```
Cell: 01
Address: 00:01:38:1F:CB:3E
ESSID: "PTNET"
Mode: Master
Channel: 2
Encryption key: on
Quality: 70/70
```

## 3. Customer requirements

- The application starts from a command terminal and is manipulated entirely with the keyboard.
- After starting, the application shows a main text menu with all possible operations, each of them identified with a different number or letter.

```
[2022] SUCEM S.L. Wifi Collector

[  1] wificollector_quit
[  2] wificollector_collect
[  3] wificollector_show_data_one_network
[  4] wificollector_select_best
[  5] wificollector_select_worst
```

```
 [ 6] wificollector_delete_net
 [ 7] wificollector_sort
 [ 8] wificollector_export
 [ 9] wificollector_import
 [10] wificollector_display
[11] wificollector_display_all

 Option:
```

The operations that the main menu **must have** are (at least):

1. **wificollector_quit**.- Exit the program. The application should offer the posibility of asking the user if s/he wants to exiting.

   ```
   Are you sure you want to exit? [y/N]:
   ```

2. **wificollector_collect**.- Collect network information of a Cell. When the user chooses this option, the following question must appear:

   ```
   What cell do you want to collect? (1 - 21):
   ```

   The connexions associated to the cell chosen are added to the ones already existing in the application by reading the file: "`info_cell_NN`", where NN is the number of the cell chosen.

   Then, the following question must appear:

   ```
   Do you want to add another access point? [y / N]:
   ```

   If the answer is "`y`", the previous question will be asked again.

3. **wificollector_show_data_one_network**.- Select and show the detailed information about a network (ESSID). That is, the detailed information about the access points offering its coverage.

   ```
   Indicate the ESSID (use ""):
   ```

4. **wificollector_select_best**.- Show the detailed information about the access point to any network that offers the highest quality.
5. **wificollector_select_worst**.- Show the detailed information about the access point to any network that offers the lowest quality.
6. **wificollector_delete_net**.- Select and delete all the access points of a net.

   ```
   Indicate the ESSID (use ""):
   ```

7. **wificollector_sort**.- Show the access points on the screen sorted by decreasing value of the signal quality.

8. **wificollector_export**.- Store the information about the access points in a binary file with name chosen by the user.

```
Indicate the name of the file:
```

9. **wificollector_import**.- Add to the information contained in the application the one contained in a binary data file with name given by the user. If the file contains information about a data point which is already present in the application, the file information is discarded. Otherwise, the information is added to the application.

```
Indicate the name of the file:
```

10. **wificollector_display**.- Print the information of the cells stored in the application.

```
Indicate the number of the cell for which you want to
know its information (1 - 21):
```

If information is requested from a cell that has not been added using the **wificollector_collect** command, an error message will be given.

Then the following question should appear:

```
Do you want to print the information of another cell? [y
| N]:
```

If the answer is "`y`", the previous question will be asked again.

11. **wificollector_display_all**.- Print the information of all cells stored in the application.

After selection an operation, additional data may be introduced through the keyboard. When the operation finishes, the main textual menu is shown again and the application waits for a new command from the user.

The application must be **robust**, that is, it should recover with no problems if at some point the user introduces incorrect data (letters when numbers are expected, an empty string, an empty line, an incorrect file name, an empty file, etc.)

> **Warning**
>
> Although more frequent than desired, in an industrial context sometimes the customer makes adjustments to the specification **while the project is being developed**. Thus, if such situation arises, the team should treat it normally.

## 4. Company Requirements

Aside from the customer requirement that are related to the finished project, SAUCEM Inc. has its own product development policy oriented toward maintaining an image of a reliable company, and to assure an efficient development cycle. These requirements are:

- **company_require_divided**.- The code must be divided into files such that similar functions are grouped. Function names have to be chosen to increase code readability and to quickly locate certain functionality.
- **company_require_documented**.- All functions, global variables, `#define`, structure declaration and enumeration definitions must be documented.
- **company_require_debug**.- If the symbol `DEBUG` is defined when compiling, the application shows on the screen debugging messages with the main operations executed.
- **company_require_gccclean**.- The code must compile with no error or warning when compiled with the option `-Wall` in `gcc`.
- **company_require_style**.- The code must be written strictly following the rules described in the document "C Language: A Style".
- **company_require_noleaks**.- The application must perform a correct dynamic memory management, that is, no leaks, access to uninitialized portions, incorrect frees, etc.
- **company_require_balanced**.- We required to have a contribution balanced among the team members.
- **company_require_working_in_teams**.- The work teams must be formed by 2 people belonging to the same reduced group of the course.

## 5. Working in Teams

Students must organize themselves in teams of 2 people.

## 6. Submissions

The submissions of the three versions for review will be done through through the tasks "Submission 1", "Submission 2", and "Submission 3" in Aula Global. In each case you must deliver the source code, a README where you explain the general organization of the project, how to compile and execute it.

**Submission 1** (10%)

The first submission must implement the following operations:

- wificollector_quit
- wificollector_collect
- wificollector_display
- wificollector_display_all

wificollector_collect must be implemented using arrays (maximum size: 200). For any string you need to define use as maximum size: 80.

**Submission 2** (15%)

The second submission must implement the following operations:

- wificollector_quit
- wificollector_collect
- wificollector_display
- wificollector_display_all
- wificollector_select_best.-
- wificollector_select_worst

For this submission you must use dynamic arrays, initially with a size of 5 and use the function realloc() to increase in 5 new elements each time that you do not have enough memory.

**Submission 3** (15%)

The third submission must implement the following operations:

- wificollector_quit
- wificollector_collect
- wificollector_display
- wificollector_display_all
- wificollector_select_best.-
- wificollector_select_worst

For this submission you must:

- use dynamic lists
- deallocate all dynamic memory before exit

Deadlines:

| | |
|---|---|
| Friday of Week 5 (the 7th of October, 2022) at 23:55: | Submission 1 |
| Friday of Week 9 (the 4th of November, 2022) at 23:55: | Submission 2 |
| Friday of Week 14 (the 9th of December, 2022) at 23:55: | Submission 3 |

# Wifi Collector

| | INPUT | OUTPUT | |
|---|---|---|---|
| 1. Quit | scanf (1,2) | printf (stdout) (1,2) | |
| 2. Collect | getchar (1) | putchar | |
| | ~~gets~~ not recommended | puts | puts("Hello") |
| 10. Display | fgets | | printf("Hello\n") |
| | getline | sprintf (2) | |
| 11. Display_All | | | |