Grado en Ingeniería Telemática
Grado en Ingeniería de Sistemas Audiovisuales
Grado en Ingeniería de Sistemas de Comunicaciones

# Proyecto
# Recolector de redes inalámbricas

## 1. Recursos

- Plantillas para el proyecto
- Información sobre trabajo en equipo
- Guía de estilo
- Planilla para guiar la planificación del proyecto (dada por su profesor de laboratorio)
- 21 ficheros que contienen datos de entrada: "info_cell_NN", donde NN $\varepsilon$ {1,2,…,21}

## 2. Descripción general

Diseño de una aplicación que recopila datos sobre las redes inalámbricas que detecta un dispositivo móvil.

Se recibe en la empresa SAUCEM S.L. un encargo de un cliente que quiere una aplicación para recopilar información de las redes inalámbricas que detecta un dispositivo móvil mientras se utiliza en espacios abiertos. El cliente quiere una aplicación que el usuario pueda utilizar mientras se desplaza por un lugar y que en determinados momentos haga un barrido de las redes inalámbricas disponibles y almacene información sobre sus características. La aplicación además debe ofrecer varias operaciones sobre el conjunto de redes inalámbricas obtenidas.

Esta aplicación se necesita para poder recorrer espacios públicos sobre los que se despliegan redes inalámbricas y trazar un mapa de cobertura de estas redes, la calidad de su señal y el número de ellas que están disponibles. La información capturada con el dispositivo móvil se debe almacenar en ficheros de datos que después pueden ser procesados por otras herramientas. La información que se obtiene de cada punto de acceso a una red inalámbrica consta de los siguientes campos:

- **Identificador de Celda**: Es un número natural.
- **Dirección MAC (*Media Access Control*, o control de acceso al medio) del punto de acceso**: Es un identificador único asignado a un interfaz de comunicación, y en este caso del punto de acceso a través del que se da cobertura a la red Wifi. Estas direcciones son asignadas por el fabricante de un dispositivo de comunicaciones y están codificadas en el hardware. Este dato consta de **seis números hexadecimales de dos dígitos cada uno**. Cuando se representan textualmente se suelen separar estos grupos por ":" o "-". Por ejemplo:

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

o

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

- **ESSID (*Extended Service Set Identifier* o Identificador de Conjunto de Servicio Extendido)**: Es el nombre que identifica la red inalámbrica a la que se accede a través de ese punto. Varios puntos de acceso con diferente MAC pueden ofrecer cobertura a la misma red. El nombre estará escrito entre “’s. Por ejemplo:

```
"PTNET"
```

- **Modo**: Existen diferentes formas de funcionamiento de los puntos de acceso a redes inalámbricas. La aplicación a implementar contempla los siguientes:
  - `Auto`
  - `Ad-Hoc`
  - `Managed`
  - `Master`
  - `Repeater`
  - `Secondary`
  - `Monitor`
  - `Unknown`
- **Canal**: Un número natural que indica en qué canal (frecuencia) se emite la señal para acceder a este punto de acceso.
- **Cifrado**: Campo que codifica si la comunicación con el punto de acceso es cifrada. Valores posibles:

```
on
```

```
off
```

- **Calidad**: Dos números naturales:

```
n1/n2
```

  donde `n1` representa el nivel actual de la señal, and `n2` representa el máximo nivel esperado.

Hay varios campos adicionales que se pueden obtener de los puntos de acceso, pero la aplicación a implementar tan sólo debe considerar estos.

A continuación, se presenta a manera de ejemplo, la información acerca de un punto de acceso recogida por un dispositivo móvil:

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

## 3. Requisitos del cliente

- La aplicación se arranca desde un terminal de comandos y se manipula enteramente con el teclado.
- Tras arrancar, la aplicación muestra un menú principal de texto con las posibles operaciones, cada una de ellas identificada con un número o letra diferente. Si cuando se está esperando una orden del usuario se pulsa la combinación de teclas `CTRL-D`, la aplicación debe terminar.

```
[2019] SUCEM S.L. Recolector de redes inalámbricas

   [ 1] wificollector_quit
   [ 2] wificollector_collect
   [ 3] wificollector_show_data_one_network
   [ 4] wificollector_select_best
   [ 5] wificollector_delete_net
   [ 6] wificollector_sort
   [ 7] wificollector_export
   [ 8] wificollector_import

   Opción elegida:
```

Las operaciones que **debe incluir** el menú principal son (al menos):

1. **wificollector_quit**.- Salir de la aplicación. La aplicación debe preguntar al usuario si verdaderamente quiere terminar de trabajar con la aplicación:

   ```
   ¿Está seguro de que desea salir del programa? [s/N]:
   ```

2. **wificollector_collect**.- Recolecta información sobre Celdas. Cuando el usuario elige esta opción, debe aparecer la siguiente pregunta:

   ```
   ¿Qué celda quiere recolectar? (1 - 21):
   ```

   Las conexiones asociadas a la celda elegida se agregan a las que ya existen en la aplicación leyendo el archivo: “info_cell_NN”, donde NN es el número de la celda elegida.

   Entonces, debe aparecer la siguiente pregunta:

   ```
   Desea añadir otro punto de acceso? [s/N]:
   ```

   Si la respuesta es "s", se volverá a hacer la pregunta anterior.

3. **wificollector_show_data_one_network**.- Seleccionar y mostrar la información detallada de una red (ESSID). Es decir, la información detallada de los puntos de acceso que ofrecen su cobertura.

   ```
         Indique el ESSID (entre “”):
   ```

4. **wificollector_select_best**.- Mostrar la información detallada del punto de acceso a cualquier red que tiene mejor calidad.
5. **wificollector_delete_net**.- Seleccionar y borrar todos los puntos de acceso de una red.

```
Indique el ESSID (entre "”):
```

6. **wificollector_sort**.- Mostrar los puntos de acceso por pantalla ordenados en orden decreciente de calidad de la señal.
7. **wificollector_export**.- Guardar la información sobre los puntos de acceso en un fichero binario con nombre elegido por el usuario.

```
Indique el nombre del fichero:
```

8. **wificollector_import**.- Añadir a la información de la aplicación aquella contenida en un fichero binario de datos de un fichero dado por el usuario. Si el fichero contiene información sobre un punto de acceso sobre el que ya se tiene información en la plataforma, se ignora la información del fichero. Si no, se añade a los datos.

```
Indique el nombre del fichero:
```

Al seleccionar una operación, esta puede requerir que se introduzcan datos adicionales por el teclado. Al acabar la operación se vuelve a mostrar el menú principal a la espera un nuevo comando del usuario. Si el usuario desea volver al menú principal antes de acabar la operación, pulsará `Crtl-D`.

La aplicación debe ser **robusta**, es decir, debe poder recuperarse sin problemas si en algún punto el usuario no introduce los datos adecuados (letras cuando se espera un número, una cadena vacía, una línea vacía, un nombre de fichero incorrecto, un fichero vacío, etc.)

> **¡Aviso!**
>
> Tal y como suele suceder más a menudo de lo deseable en el contexto industrial, un cliente puede hacer ajustes a esta especificación **mientras se está desarrollando el proyecto**. Por tanto, si tal circunstancia se produce, el equipo debe tratarla con normalidad.

## 4. Requisitos de la Empresa

Además de los requisitos del cliente que tienen que ver con el producto terminado, SAUCEM S.L. tiene su propia política de desarrollo de productos orientada a mantener su imagen corporativa de empresa fiable, y a conseguir un ciclo de desarrollo eficiente. Estos requisitos son:

- **dynamic_memory**.- La solución del proyecto debe utilizar algún tipo de estructura dinámica (lista, árbol, hash, o similar).
- **company_require_divided**.- El código debe estar dividido en ficheros de tal forma que se agrupen funciones similares. Los nombres de las funciones deben ser elegidos para facilitar su entendimiento y localizar rápidamente la funcionalidad buscada.

- **company_require_templates**.- **Todos los ficheros de código** (*.c) y de definiciones (*.h) deben tener la estructura de las plantillas CFile.templ y CHeaderFile.templ respectivamente que encontraréis en el directorio Plantillas. En la cabecera se debe incluir una descripción detallada de las funciones o definiciones que contiene el fichero. Todos aquellos campos de la plantilla que no tengan contenido deberán borrarse. Por ejemplo, si tu código no contiene "macros", deberás borrar esta parte de la plantilla.

  Además, aquellas partes del código más delicadas, deben ir precedidas de un bloque de comentario que lo explique. No incluyas comentarios línea a línea, sino en bloques al comienzo de un conjunto de pasos.

- **company_require_documented**.- Todas las funciones, variables globales, `#define`, definiciones de estructuras y definiciones de enumeraciones debe estar documentadas precediendo su definición con texto de las plantillas en los ficheros con nombres Function.templ, Variable.templ, Macro.templ, Struct.templ y Enum.templ respectivamente. En el caso de las funciones se debe explicar el papel de cada parámetros así como el resultado que devuelve. Los fragmentos a incluir con sus campos por completar los puedes encontrar todos en la carpeta Plantillas de tu directorio de trabajo.
- **company_require_debug**.- Si el símbolo `DEBUG` está definido al compilar, la aplicación muestra mensajes de depuración por pantalla con las principales operaciones que realiza.
- **company_require_gccclean**.- El código debe compilar sin ningún tipo de error ni advertencia cuando se utiliza la opción `-Wall` del compilador `gcc`.
- **company_require_style**.- El código debe escribirse siguiendo escrupulosamente las pautas descritas en el documento "Lenguaje C: Guía de Estilo".
- **company_require_noleaks**.- La aplicación debe hacer una gestión correcta de la memoria dinámica, es decir, sin fugas, accesos a porciones sin inicializar, liberaciones incorrectas, etc.
- **company_require_minutes**.- En la carpeta Actas del espacio de trabajo del grupo debe haber un acta para cada una de las reuniones de grupo que se han realizado. En ella debe constar el orden del día, los asistentes, quién ha realizado de moderador, quién ha escrito el acta y las decisiones que se han tomado.
- **company_require_balanced**.- Se exige una contribución equilibrada de todos los miembros del equipo.
- **company_require_working_in_teams**.- Los equipos de trabajo deben estar formados por 3 ó 4 personas pertenecientes todas al mismo grupo reducido de la asignatura.

## 5. Trabajo en grupos

Los estudiantes deben organizarse en equipos de 4 personas, excepcionalmente pueden ser 3 o 5 pero es necesario contar con la autorización del instructor del laboratorio correspondiente. Los equipos deben tener todos sus miembros inscritos en el mismo grupo.

## 6. Hitos del proyecto

Para facilitar la ejecución, el proyecto se divide en siete "hitos.". Un hito marca el final de una fase y el comienzo de otra. A menudo los hitos de un proyecto llevan asignados también una serie de "entregables" que son fragmentos de programa, documentos, o cualquier elemento relacionado con el proyecto que debe estar listo para ser entregado. Los hitos que te

describimos a continuación son **orientativos** y pretenden ser una guía de cómo se debe desarrollar el proyecto de forma gradual. Puede que tu equipo no siga estrictamente estos hitos, pero no es importante siempre y cuando el proyecto se haga de forma gradual y la entrega parcial (hito 3) y el producto final (hito 6) estén terminados en el plazo acordado. Si un proyecto completa un hito antes de lo planificado no debe detenerse y debe continuar con el siguiente hito. Los hitos semanales propuestos para el proyecto son:

**Hito 1**. Crear un documento que incluye la descripción de los módulos en los que se dividirá la aplicación (ver la sección "Organización del código" de la guía "Guia_de_Estilo_es.pdf") así como de las estructuras de datos que se precisan. Cada módulo debe tener una descripción de la funcionalidad que se le encarga y de los datos que manipula. El equipo debe asignar a un miembro como responsable de cada módulo y mencionarlo en el documento. A lo largo del desarrollo del proyecto esta división puede ser ajustada cuanto sea necesario.

**Hito 2**. Implementación de las funcionalidades "**wificollector_quit**" y "**wificollector_collect**" de las requeridas por el cliente.

**Hito 3**. Implementación de las funcionalidades "**wificollector_show_data_one_network**" y "**wificollector_select_best**" de las requeridas por el cliente.

**Hito 4**. Implementación de la funcionalidad "**wificollector_delete_net**".

**Hito 5**.- Implementación de la funcionalidad "**wificollector_sort**".

**Hito 6**.- Implementación de las funcionalidades "**wificollector_export**" and "**wificollector_import**".

**Hito 7**. Pruebas finales. Verificación de que la aplicación compila sin advertencias, sin fugas de memoria.

## 7. Las sesiones de laboratorio están alineadas con los hitos del proyecto

Las sesiones de laboratorio están alineadas con el inicio de los hitos del proyecto y su instructor de laboratorio dará instrucciones generales de cómo afrontar el hito correspondiente, y el tiempo estimado de la realización de este.

Recuerde que su instructor de laboratorio puede guiarlo en el proyecto, pero no es su compañero de equipo.

## 8. Entrega de la versión parcial y final

La entrega de la versión parcial y final se hará a través de las tareas "Version Parcial" y "Version Final" en Aula Global. En cada caso deberás entregar el código fuente, un README en donde expliques la organización general del proyecto, cómo compilarlo y ejecutarlo.

El equipo puede obtener hasta un 10% de puntos adicionales al enviar la planificación del proyecto seguido por el equipo.

**Fechas de entrega**:

Semana 12: (22 Nov) Hitos 1 a 4 (versión parcial)

Semana 15: (13 Dic) Hitos 1 a 7 (versión final)

## 9. Aspectos de Evaluación

La siguiente tabla muestra la guía que se utiliza para evaluar las entregas del proyecto. Aquellas categorías que ocupan dos casillas, se evalúan con la de la nota más alta.

| Aspecto | Excelente (100%) | Aceptable (75%) | Discreto (50%) | Insuficiente (0%) |
|---|---|---|---|---|
| 1 Compilación del código con la opción -Wall | El código compila sin ningún tipo de advertencia o error. | | El código compila con dos avisos. | El código no compila, o compila con más de dos avisos. |
| 2 Ejecución del programa | El programa ejecuta con normalidad y acorde a la especificación. | El programa termina abruptamente en una ocasión o no cumple con la especificación en uno o dos aspectos. | El programa falla en dos pruebas o no cumple con la especificación en más de dos aspectos. | El programa falla con frecuencia o el programa no cumple con la especificación. |
| 3 Gestión de memoria | La aplicación ejecuta sin ningún tipo de anomalía cuando se analiza con Valgrind. | Se detecta o una fuga de memoria, o una liberación de memoria incorrecta. | Valgrind detecta dos anomalías al ejecutar la aplicación. | Se detectan más de dos anomalías al ejecutar conValgrind. |
| 4 Mensajes de depuración | Los principales pasos de la aplicación imprimen mensajes de depuración cuando se define el símbolo DEBUG. | | La aplicación imprime pocos mensajes de depuración. | El programa no imprime ningún mensaje de depuración |
| 5 Código en múltiples ficheros, definiciones y prototipos en .h | El código está dividido en varios ficheros y las definiciones y prototipos incluidos en uno o varios ficheros con extensión ".h". | | El código está en un único fichero, o el fichero con las definiciones y prototipos está incompleto (o incorrecto). | El código está dividido de forma arbitraria, y el fichero .h contiene información incorrecta y no se incluye de forma adecuada. |
| 6 Uso de las plantillas de ficheros y documentación en la cabecera. | Todos los ficheros siguen la plantilla, tienen todos los campos obligatorios rellenos y la documentación en la cabecera es intuitiva. | | Uno o dos ficheros no siguen la plantilla, no tienen todos los campos de documentación rellenos, o la documentación | Mas de dos ficheros no siguen la plantilla, tienen algunos campos sin rellenar, o la documentación en la cabecera no es suficiente. |

| Aspecto | Excelente (100%) | Aceptable (75%) | Discreto (50%) | Insuficiente (0%) |
|---|---|---|---|---|
| | | | en la cabecera no es suficiente. | |
| 7 Documentación de variables globales, estructuras y funciones | Todas las variables globales, estructuras y funciones están perfectamente documentadas. | En un fichero hay una variable, una función o una definición de estructura sin comentar. | Hay dos ficheros con variables, funciones o declaraciones de estructuras sin documentar. | Hay más de dos ficheros con carencias en la documentación de funciones, declaración de estructuras o variables globales. |
| 8 Estilo de codificación | Todos los ficheros cumplen con todos los requisitos de la guía de estilo. | | Algunos ficheros no cumplen con los requisitos de la guía de estilo. O algún criterio de la guía de estilo es ignorado completamente. | La mayoría de los ficheros no respetan la guía de estilo. O la mayor parte de criterios de estilo se ignoran. |
| 9 La aplicación es intuitiva | El usuario entiende sin problemas las funciones, los datos a introducir y cómo introducirlos. | | Los mensajes en general están bien, pero en alguna ocasión no se entiende bien qué hay que hacer. | El usuario se atasca frecuentemente por no saber cómo ejecutar la aplicación. |
| 10 Actas de las reuniones | El equipo se ha reunido con frecuencia (al menos una vez a la semana) y las actas detallan su contenido. | | El equipo no se ha reunido con la frecuencia deseada y/o las actas no reflejan la actividad real. | Las actas tienen muchas carencias, o hay un número muy bajo de ellas. |

Los criterios están ordenados en orden de importancia. Tener correctos los dos primeros es condición esencial para aprobar el proyecto. Si el programa no compila o falla con frecuencia, no se considera acabado y la nota es un cero.

## 10. Fuentes de "inspiración"

Cuando se emprende un proyecto de estas características es muy común el obtener información de múltiples fuentes que contribuyen a su desarrollo. Cuando el proyecto empieza y los plazos de entrega están lejos, no hay problema de este tipo. Pero cuando nos vemos con el tiempo encima, la desesperación nubla la capacidad de decisión. Lo mejor para evitar problemas es

tener muy claro en todo momento cuando esas fuentes son de uso razonable y cuando se pueden considerar una copia del trabajo. Te incluimos a continuación una serie de sugerencias.

- Comentar en el pasillo, en un aula, entre clase y clase cómo estamos resolviendo un aspecto especial de la práctica (no hay problema).
- Estoy estudiando con compañeros en la biblioteca y uno saca un papel para dibujar su problema en el proyecto y yo dibujo lo que hemos hecho en nuestro proyecto (no hay problema).
- Tras tener una de estas conversaciones, mi amigo me escribe y me dice que no le ha quedado claro. Le mando por correo una descripción textual más detallada de lo que hemos hecho (no hay problema).
- Recibo otro correo y todavía no lo ve claro. Corto y pego un trozo de nuestro código en el correo que le envío como respuesta (si, tenemos un problema). Fácil de detectar por la estructura del código. Por pequeño que sea el fragmento.
- Encuentro un trozo de código en Internet que hace lo que yo quiero, lo corto y pego, y tras unos cambios, ya funciona (no hay problema). Asegúrate de incluir la referencia en el propio código como un comentario.
- (Basado en caso real) Trabajo en una empresa y le muestro el enunciado a mis colegas. A alguno de ellos le emociona y me manda parte de la solución (si, tenemos un problema). Fácil de detectar, cada persona escribe código de forma especial, es difícil que tu colega tenga el mismo nivel de C que tú. Además, seguramente ese colega presuma de haber resuelto esto y se puede llegar a comentar a los profesores (ha pasado).
- (Basado en caso real) Mi mejor amigo ha tenido la mala suerte de que le ha tocado en otro equipo que no avanza y está desesperado. Me pide que, por favor, le mande el código, pero me promete solemnemente que solo lo va a utilizar para leerlo y que no va a copiar nada. Es mi colega, somos como hermanos, no puedo decir que no. Se lo mando (sí, tenemos un problema). Tu colega incluirá algún fragmento en la entrega o lo que es peor, entregará uno de tus ficheros “por error” (ha sucedido).
- Trabajo en el proyecto en una academia. El profesor me explica algunos aspectos de la solución, pero sin mostrar código (nosotros sin problemas, tú veras).
- En la academia se ilustra la solución del proyecto con código ejemplo (si, tenemos un problema). Recibiremos varias entregas casi idénticas, signo inequívoco de la resolución en cooperativa.

Todas estas situaciones se pueden resumir en dos consejos fundamentales. Tu código solo lo pueden **ver** los miembros de tu equipo, y tú no debes ver otro código que no sea el tuyo. Y cuando utilices fuentes de información, **cítalas** en la documentación que se incluye en la cabecera de cada fichero. En cualquier caso, ante la duda, ¡pregunta!.

En caso de que se detecte un caso de copia, como ya ha sucedido en ocasiones anteriores, se notificará a la dirección de la escuela para que abra las diligencias correspondientes y expediente a los alumnos involucrados. La nota del curso en tal caso será cero.