



**UNIVERSIDAD DE BUENOS AIRES
Facultad de Ingeniería
Seminario de Sistemas Embebidos**

Memoria del Trabajo Final:

Módulo bluetooth para equipo de control de acceso

Autor:

Sr. Ivan Mariano Di Vito

Legajo: 95.722

*Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires,
entre marzo y julio de 2017.*

RESUMEN

Tener información sobre el ingreso y egreso a ciertas áreas de personal y equipamiento es algo muy importante para ciertas organizaciones. Sin embargo el costo de un sistema que permita esto es muy elevado para alguna de ellas. La tecnología bluetooth de bajo consumo energético, así como el uso del formato de comunicación beacon, permite disminuir los costos de un sistema de estas características.

En esta memoria se detalla el desarrollo de un sistema embebido capaz de sensar la presencia de beacons bluetooth, almacenar dicha información, alertar sobre la presencia o ausencia de los mismos y estimar su cercanía. Además el mismo cuenta con una interfaz USB por la que es capaz de transmitir información así como de recibir comandos de configuración. También se desarrolló un programa de PC para facilitar la comunicación con el sistema embebido.

Este trabajo representa el primer paso hacia un sistema completo capaz de suplir las necesidades de monitoreo y administración de espacios de manera eficiente y económica.

Índice General

| | |
|--|-----------|
| Índice de figuras | 5 |
| Índice de tablas | 6 |
| Registro de versiones | 7 |
| Introducción General | 8 |
| 1.1 Problema a resolver | 8 |
| 1.2 Solución propuesta | 8 |
| 1.3 Objetivo del presente trabajo | 9 |
| Introducción Específica | 10 |
| 2.1 Tecnologías y herramientas utilizadas | 10 |
| 2.2 BLE | 11 |
| 2.2.1 BLE beacon | 11 |
| 2.3 Módulo BLE | 12 |
| 2.3 EDU-CIAA-NXP | 13 |
| 2.4 Programa de control | 14 |
| Diseño e Implementación | 15 |
| 3.1 Diseño de Hardware | 15 |
| 3.2 Diseño de Software | 16 |
| 3.2.1 Diseño de interfaces | 16 |
| 3.2.2 Diseño del firmware del módulo BLE | 17 |
| 3.2.3 Diseño del firmware de la EDU-CIAA-NXP | 17 |
| 3.2.3 Diseño del programa de control | 18 |
| Ensayos y Resultados | 20 |
| 4.1 Pruebas de hardware | 20 |
| 4.2 Pruebas del software | 20 |

| | |
|---|-----------|
| Conclusiones | 22 |
| 5.1 Conclusiones generales | 22 |
| 5.2 Próximos pasos | 22 |
| Bibliografía | 23 |
| Anexos | 24 |
| A. Diagrama esquemático del poncho diseñado | 24 |
| A. Diagrama pcb del poncho diseñado | 25 |

Registro de versiones

| Revisión | Cambios realizados | Fecha |
|-----------------|---------------------------|--------------|
| 1.0 | Creación del documento | 25/06/2017 |
| 1.1 | | |
| 1.2 | | |

CAPÍTULO 1

Introducción General

1.1 Problema a resolver

Existen muchas organizaciones para las cuales la seguridad en sus edificios es muy importante. Para resolver este problema existe un mercado en el que empresas de seguridad ofrecen soluciones que involucran personal, instalacion de camaras y centros de monitoreo. La necesidad que esta industria pretende suplir es la de regular el acceso a ciertas áreas así como la conocer en tiempo real la presencia de personal y equipo en las diferentes zonas a monitorear.

Las solución propuesta por la industria de la seguridad tienen varios problemas. Las mismas tienen un costo fijo elevado dada la necesidad de una gran cantidad de personal profesional. Además, al estar levemente automatizada, son muy susceptibles al error humano.

1.2 Solución propuesta

Se propone como solución a este problema aumentar el nivel de automatización de la seguridad generando un sistema capaz de controlar el ingreso, presencia y egreso del personal, visitas y equipamiento de una institución. Esto requiere de un sistema completo que cuenta con varios subsistemas:

- Dispositivos portátiles que de manera inalámbrica comuniquen su posición. Los mismos serian entregados al personal o fijados a los equipos a controlar.
- Instalación de infraestructura que permita ser activada por los dispositivos anteriores. Por ejemplo puertas que permitan el acceso a zonas restringidas y por lo tanto solo se abran ante la presencia de los dispositivos portátiles autorizados.
- Dispositivos fijos en las áreas de control que reciban la información de los portátiles y actúen en consecuencia o informen a una capa superior del sistema.
- Un sistema centralizado que coordine la información de los dispositivos fijos y la almacene. Además su función sería la de informar al personal de seguridad ante cualquier movimiento no autorizado en las zonas a controlar

1.3 Objetivo del presente trabajo

Este trabajo pretende iniciar el desarrollo del sistema completo descrito en la [sección 1.2](#). Se decidió comenzar por el dispositivo fijo (a partir de ahora dispositivo central) capaz de recibir información sobre la posición de los dispositivos portátiles (a partir de ahora dispositivos periféricos). La razón de esto es que el mismo puede ser generado y probado sin la necesidad de los demás subsistemas anteriormente nombrados.

También se desarrollaron las herramientas secundarias necesarias para el correcto testeo y uso del sistema. Un ejemplo de esto es una interfaz de control, funcionando sobre una pc, capaz de revisar el estado del sistema así como configurar el mismo.

Por último también se requirió el estudio de las diferentes tecnologías y herramientas utilizadas a lo largo del proyecto. Un resumen de las mismas puede encontrarse en la [sección 2.1](#).

CAPÍTULO 2

Introducción Específica

2.1 Tecnologías y herramientas utilizadas

En esta sección se realizará un resumen de las tecnologías y herramientas utilizadas. Para una explicación más detallada de su funcionamiento así como la justificación de su uso es necesario leer el resto del [capítulo 2](#).

Las tecnologías utilizadas fueron:

- BLE (*Bluetooth Low Energy*, Bluetooth de baja energía) con formato iBeacon para la comunicación inalámbrica entre los dispositivos periféricos y el dispositivo central.
- Módulo basado en el SOC (*System-on-a-chip*, sistema en chip) Nordic nRF51822 para la recepción de las señales BLE.
- EDU-CIAA-NXP para el procesamiento de la información proveniente del módulo BLE. También se realizó una implementación que permita comprobar el correcto funcionamiento del sistema.
- Interfaz de comunicación UART para el intercambio de información entre el SOC BLE y la EDU-CIAA-NXP.
- Interfaz USB para la comunicación entre la EDU-CIAA-NXP y una pc de control.
- JVM (*Java Virtual Machine*, máquina virtual Java) para la ejecución del programa de control desarrollado.

Las herramientas utilizadas para el desarrollo fueron:

- Kicad para el desarrollo del circuito esquemático y pcb necesario.
- CLion IDE (*Integrated Development Environment*, entorno de desarrollo integrado) para el desarrollo del firmware en lenguaje C presente en la EDU-CIAA-NXP y en C++ presente en el SOC Nordic nRF51822.
- Programador ST-LINK V2 y OpenOCD para la programación de SOC Nordic nRF51822.
- LPCXpresso IDE para la programación y debugeo de la EDU-CIAA-NXP.

- IntelliJ IDE para el desarrollo del programa de control de pc en lenguaje Kotlin junto con el framework TornadoFX para el desarrollo de la interfaz gráfica.
- GIT como software de control de versiones para todos los archivos del proyecto
- Gitlab y GitHub como servidor remoto de GIT.
https://github.com/ivandivito/control_acceso

2.2 BLE

BLE, también conocido como Bluetooth Smart, es una tecnología de radio digital capaz de transmitir y recibir información de hasta 100 metros de distancia teóricos con un throughput de 0.27 Mbit/s. Aunque estas especificaciones son similares a la de la tecnología Bluetooth clásica la principal ventaja de BLE es su bajo consumo. Esto permite que un sistema embebido que cuente con esta tecnología es capaz de funcionar durante un tiempo mucho mayor que si el mismo está alimentado con una batería. Por ejemplo usando un dispositivo en modo beacon el mismo puede durar hasta 2 años con una bateria tipo boton. Esta característica es la principal razón para usar esta tecnología. Esto permite que los dispositivos periféricos móviles puedan funcionar una gran cantidad de tiempo, sin necesidad de mantenimiento, volviendo económicamente viable al sistema propuesto.

Otra ventaja es que la misma es muy común en dispositivos como teléfonos celulares o computadoras portátiles por lo que sería posible utilizar los mismos como parte del sistema. Un posible ejemplo es el desarrollo de una aplicación de celular para que el mismo funcione como el dispositivo periférico móvil descrito en el sistema.

2.2.1 BLE beacon

Se conoce como beacon a los dispositivos inalámbricos portátiles que utilizar la tecnología BLE sólo para comunicar su presencia a otros dispositivos con la misma tecnología. Los mismos no aceptan conexiones para transferencia de datos como normalmente lo hacen otros dispositivos BLE. Su única función es enviar de manera periódica un paquete de datos mínimo que contiene, además de todo los datos que el formato BLE requiere, un identificador del dispositivo y unos pocos bytes mas de informacion.

Existen distintos formatos de beacons que definen cómo utilizar los pocos bytes que se tiene de informacion extra. Alguno de ellos son AltBeacon, URIBeacon, Eddystone e iBeacon. Se decidió usar este último ya que el mismo es el más sencillo y al mismo tiempo

el más utilizado. El formato de los paquetes puede verse en la figura 2.1. En los mismos la información extra utilizada son:

- UUID: dirección de 16 bytes que identifica únicamente al hardware que emitió el paquete.
- Major Number: identificador de 2 bytes del grupo de dispositivos
- Minor Number: identificador de 2 bytes de este dispositivo particular dentro del grupo definido por Major Number
- Tx Power. 1 byte que define en complemento a 2 la potencia de la señal que emite este dispositivo, medida a 1 metro de distancia. Este valor permite al dispositivo que recibe el paquete estimar de manera muy grosera la distancia del emisor al cruzar este dato con la potencia real medida de la señal.

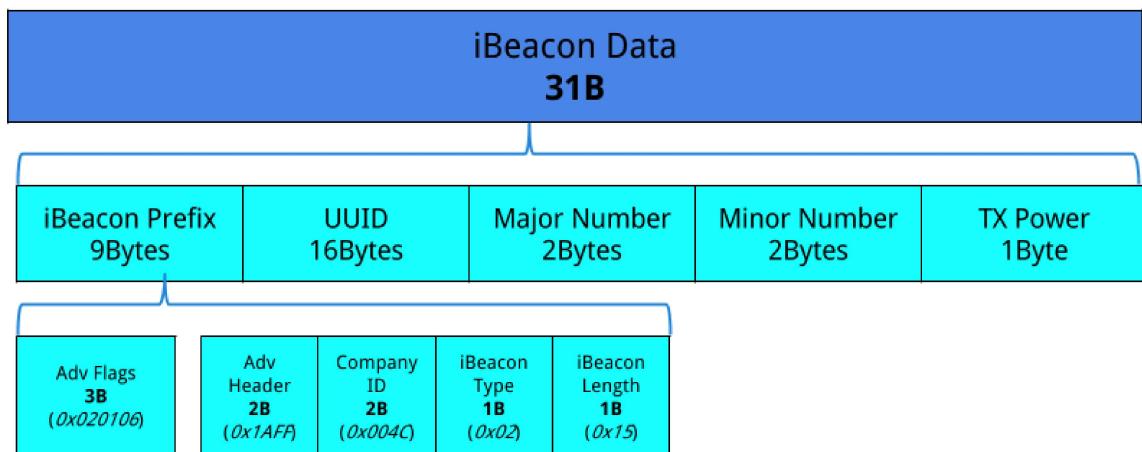


Figura 2.1: Imagen que representa el formato de un paquete de datos BLE beacon, tomada de la página oficial de MBED¹.

2.3 Módulo BLE

El módulos BLE elegido para implementar la recepción de paquetes con formato Ibeacon es el Nordic nRF51822. Puede verse una imagen del mismo en la figura 2.2. Este cuenta con un SOC en el cual se encuentra presente un procesador cortex M0 programable. Al mismo tiempo el módulo contiene el circuito de RF (radiofrecuencia) necesario para realizar transmisiones de paquetes BLE.

¹blog de desarrolladores mbed disponible:
<https://developer.mbed.org/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/>

Una gran ventaja de utilizar este SOC es que el mismo cuenta con una gran cantidad de bibliotecas desarrolladas que facilitan el desarrollo del firmware. Las mismas están pensadas para ser usadas dentro del framework MBED en lenguaje de programación C++.



Figura 2.2: Foto del módulo bluetooth de baja energía elegido para la recepción de datos.

2.3 EDU-CIAA-NXP

La mayor parte de este trabajo consistió en desarrollar un firmware sobre la EDU-CIAA-NXP que permitiera administrar la información provista por el módulo BLE y actuar a partir de ella. La misma es la versión educativa de una plataforma de desarrollo para aplicaciones industriales que tiene la ventaja de ser totalmente abierta. Cuenta con un microcontrolador dual-core ARM Cortex M4/M0, 136 KB de memoria RAM y 1 MB de memoria Flash. Puede verse una foto de la misma en la figura 2.3.

Para programarla se utilizaron varios frameworks:

- sAPI: Esta biblioteca actúa como HAL (*Hardware Abstraction Layer*, capa de abstracción de hardware), lo cual permite simplificar el acceso a los recursos de hardware de la plataforma. Esta biblioteca interactúa directamente con la provista por el fabricante del microcontrolador LPCOpen.
- FreeRTOS: Es un RTOS (*Real-time operating system*, Sistema operativo de tiempo real) pequeño y sencillo que otorga al firmware la capacidad de administrar la distribución del tiempo de cómputo a las distintas tareas así como la administración de memoria dinámica.



Figura 2.3: Foto de la plataforma EDU-CIAA-NXP.

2.4 Programa de control

Se desarrolló un programa de control que permite la comunicación entre la EDU-CIAA-NXP y una pc. La comunicación entre ambos se realiza emulando un puerto serie a través de la conexión USB de la plataforma de desarrollo con la pc. El programa fue desarrollado en lenguaje Kotlin. El mismo permite compilarse a un archivo .jar que puede ser ejecutado por cualquier JVM. En el desarrollo se utilizaron varias bibliotecas:

- RxJava y RxKotlin para utilizar el paradigma de programación reactive.
- TornadoFX para la generación de la interfaz gráfica de usuario
- jSerialComm para la comunicación a través del puerto serie virtual con la EDU-CIAA-NXP.

CAPÍTULO 3

Diseño e Implementación

3.1 Diseño de Hardware

Fue necesario el diseño y fabricación de un poncho (más conocido como *shield* para otras plataformas de desarrollo) capaz de conectar al módulo BLE con la placa de desarrollo. Dicho poncho debe entregarle alimentación al módulo BLE al mismo tiempo que permitir programar y conectar uno de sus puertos series con los pines correspondientes de la EDU-CIAA-NXP. También se agregó al mismo un led y un pulsador conectados al módulo Ble con el propósito de utilizarlos durante el desarrollo del firmware.

El diseño esquemático y de Pcb se realizó en KiCad. Luego el mismo se fabricó utilizando una fresadora CNC casera. Puede verse una imagen del circuito terminado en la figura 3.1.

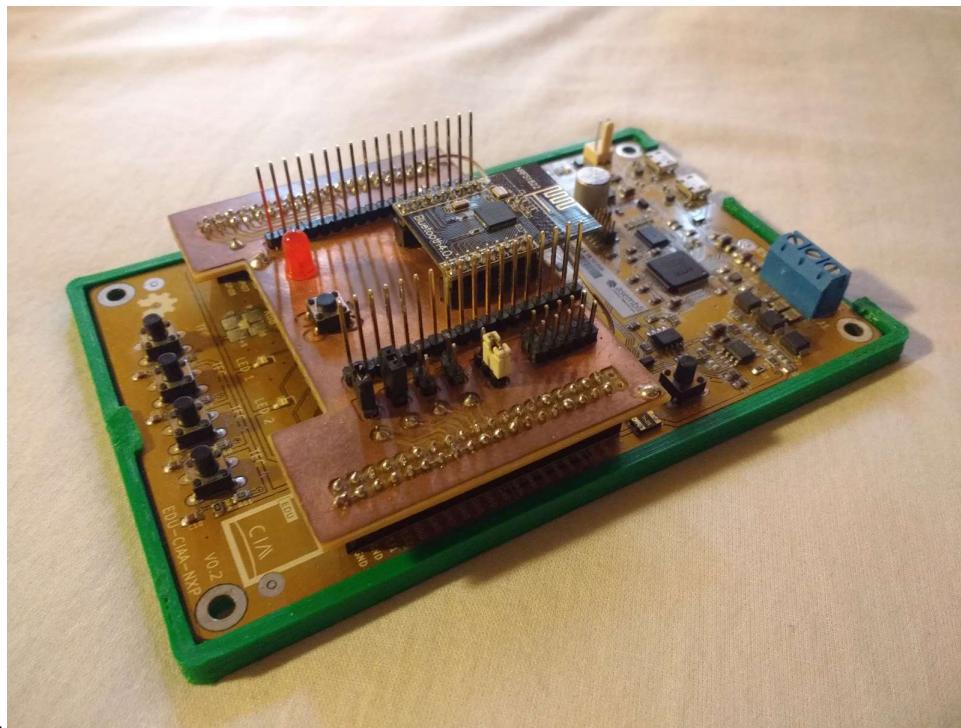


Figura 3.1: Foto de la EDU-CIAA-NXP con el poncho diseñado y módulo BLE

3.2 Diseño de Software

El diseño de Software puede particionarse en las 3 plataformas que requieren programación: el módulo BLE, la EDU-CIAA-NXP y el programa de control. Otro aspecto importante del diseño cuando se tienen múltiples programas que tienen que funcionar de manera coordinada es determinar el formato de comunicación entre ellas. Todos estos puntos serán desarrollados en las próximas subsecciones.

3.2.1 Diseño de interfaces

Existen 2 interfaces entre programas. Una de ellas se encuentra entre el módulo BLE y la EDU-CIAA-NXP. El primero debe informar por la misma cada vez que se recibe un paquete de datos IBeacon, mientras que el segundo debe poder enviar el comando de inicio y fin del escaneo. Para el envío de comandos se eligió el siguiente formato: “\$x\n” en donde x es un carácter único para cada una de las órdenes posibles y \n es el carter de fin de línea que permite determinar que se completo el envío del comando. En esta interfaz existen 2 posibles comandos: ‘S’ para inicializar el escaneo y ‘E’ para finalizarlo.

En el otro sentido de comunicación de esta interfaz se requiere representar cada paquete de información de la forma más compacta posible. Esto es porque ante la presencia de múltiples beacons la cantidad de información a transmitir puede ser muy grande. Por este motivo se decidió enviar un paquete de actualización de 7 bytes. El primero y el último de los mismo son caracteres que delimitan el paquete y permiten comprobar si el mismo es válido, mientras los 5 intermedios contienen el identificador único de ese beacon, así como la distancia, a la que se estima, que se encuentra en metros.

La segunda interfaz es la que se encuentra entre la EDU-CIAA-NXP y el programa de control. La misma tiene requerimientos distintos a la primera. Se tomó como criterio de diseño que no necesariamente se va a encontrar del otro lado de la placa de desarrollo el programa desarrollado. Es posible que otro sistema o un usuario de manera directa, a través de una terminal serie, quiera interactuar con el dispositivo. Por esto se decidió que el formato en el que la placa de desarrollo responde al exterior sea apto para ser leído por humanos. Los comandos que pueden enviarse son los siguientes:

- “\$S\n” y “\$E\n” para iniciar y detener el escaneo de beacons respectivamente.
- “\$D\n” para borrar la información almacenada en la EDU-CIAA-NXP
- “\$R {major} {minor}\n” para pedir el estado actual del dispositivos con dirección mayor {major} y dirección menor {minor}.
- “\${led} {mode} {major} {minor}\n” para configurar una respuesta en el caso de que se reciba o deje de recibir información del beacon con dirección {major} {minor}.

El parámetro {led} es un número entre 0 y 3 que representa a un led de la EDU-CIAA-NXP y {mode} es un número que vale 0 en el caso de que se quiera que el led indique la presencia del beacon o 1 para que muestre la distancia al mismo.

Ante estos comandos existen las siguientes posibles respuestas del sistema:

- Ante el comando de inicio de escaneo se recibe "start ble scan\r\n" y ante el de fin "end ble scan\r\n"
- Ante el comando de pedido de estado la respuesta del sistema es de la forma "report: addr: {major} {minor}, presence: {presence}, last distance: {distance}\r\n" en donde {major} y {minor} corresponde a la dirección pedida, {presence} puede tener el valor de "present" si se están recibiendo paquetes, "lost" si se dejaron de recibir o "missing" si nunca se recibieron.
- Ante un error en el formato en el comando se recibe "format error\r\n"

3.2.2 Diseño del firmware del módulo BLE

Para la programación del módulo ble se utilizó el framework MBED de ARM. El mismo se encuentra desarrollado en C++ por lo que se utilizó este lenguaje de programación para el desarrollo. El mismo provee las herramientas para de manera simples se puedan utilizar los recursos de hardware presentes en el módulo, como el puerto serie o la antena BLE. Además el framework permite utilizar un sistema de programación basada en eventos. El mismo consiste en definir funciones para que sean llamadas ante distintos eventos definidos en el sistema. En particular el programa desarrollado indica al framework que se ejecuta una función en particular al recibir un paquete BLE y otra ante la llegada de información por el puerto serie. La primera determina si el paquete corresponde a un IBeacon y si es el caso estima su distancia e informe por la UART con el formato explicado en la [subsección 3.2.1](#). La segunda revisa si se recibió por la interfaz un comando válido y en caso afirmativo realiza las acciones pertinentes. Por último el programa presente en este módulo hace parpadear el Led presente en el poncho diseñado en caso de encontrarse activo el escaneo de datos.

3.2.3 Diseño del firmware de la EDU-CIAA-NXP

El diseño del software presente en la EDU-CIAA-NXP puede dividirse en 2 partes: una biblioteca capaz de generar y operar con una base de datos de los dispositivos Beacons cercanos y otra donde se implementan las cuestiones específicas de la plataforma utilizada.

La primera fue implementada de manera tal que pueda utilizarse en cualquier otro dispositivo programable. A parte de su función básica descrita anteriormente, la misma se

encarga de actualizarse con el paso del tiempo para marcar a los dispositivos de los que no se reciben paquetes como perdidos. En el caso de que se modifique una entrada una señal es emitida para que el sistema que la contiene reaccione ante ese cambio.

En cuanto al diseño de la segunda parte se utilizaron 5 tareas dentro del RTOS utilizado. Las mismas fueron

- “uartUsbOutputTask” encargada de enviar informacion a travez del puerto serie virtual USB. La prioridad de la misma es 1 (menor prioridad).
- “uart232OutputTask” encargada de enviar informacion a travez del puerto que funciona como interfaz con el módulo BLE. La prioridad de la misma es 1 (menor prioridad).
- “cmdManagerTask” encargada de procesar los comandos provenientes del programa de control. La prioridad de la misma es 2 (prioridad intermedia).
- “beaconManagerTask” encargada de procesar los comandos provenientes del módulo BLE, procesarlos y enviarlos a la biblioteca que maneja la base de datos. La prioridad de la misma es 3 (mayor prioridad).
- “beaconManagerPeriodicTask” encargada de informar a la biblioteca que maneja la base de datos sobre el paso del tiempo para que la misma se actualiza. Esta tarea también se encarga de persistir cada 10 segundos el estado del sistema de forma tal que se pierda una cantidad de información mínima ante una falla temporal del suministro eléctrico. La prioridad de la misma es 3 (mayor prioridad).

3.2.3 Diseño del programa de control

Para facilitar el uso del sistema se desarrolló un programa de pc capaz de enviar instrucción y recibir las respuestas previamente detalladas. El mismo cuenta con una interfaz gráfica en la cual se puede elegir con qué dispositivo conectarse, se puede ver los comandos enviados y recibidos, se pueden enviar comandos personalizado o enviar los predefinidos por el programa. Dicha interfaz puede verse en la figura 3.2.

El programa fue realizado utilizando la técnica reactive. En la misma se modela al programa como una serie de streams de eventos asincrónicos y cómo responde el programa a los mismos. Esto es ideal para un programa con interfaz gráfica que además se comunica con un dispositivo externo, una situación con muchos asincronismos.

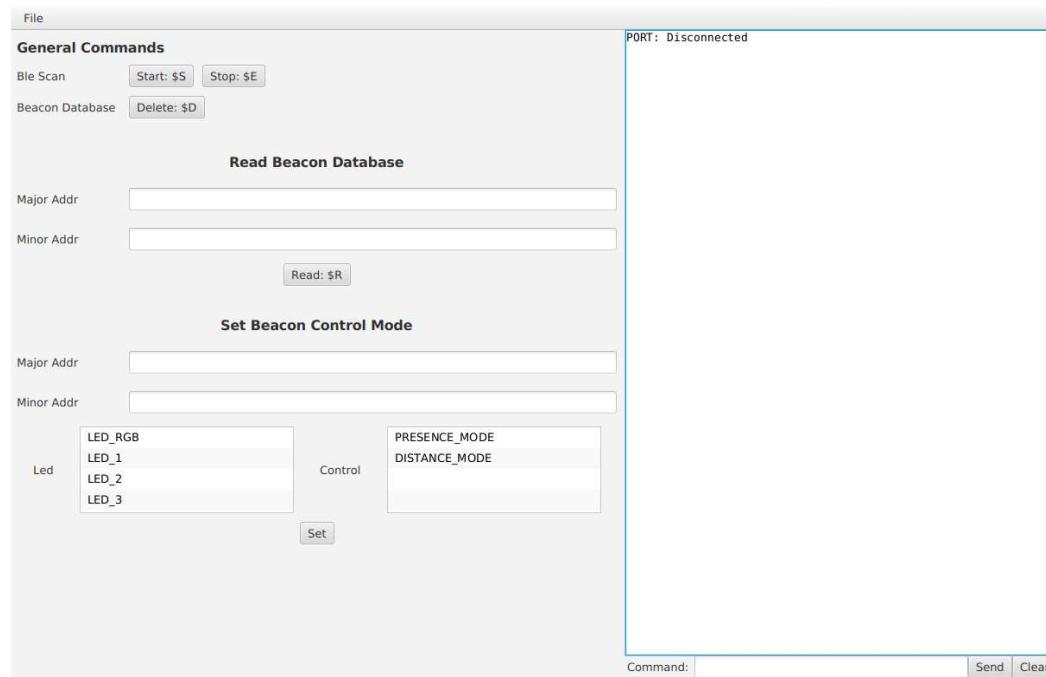


Figura 3.2: Interfaz gráfica del programa de control desarrollado

CAPÍTULO 4

Ensayos y Resultados

4.1 Pruebas de hardware

Para el testeo del poncho diseñado y construido se realizaron las siguientes pruebas:

- Se revisó que existiera continuidad entre los terminales de las pistas y que las mismas no tuvieran cortos entre sí.
- Se instaló el poncho en la EDU-CIAA-NXP sin el módulo BLE y se revisó que las tensiones en los pines fueran las correctas.
- Se conectó el módulo BLE y el programador y se cargo un programa de prueba vacío para revisar que el circuito de programación fuera correcto.
- Se cargo un programa que utilizara el LED y el botón presentes en el poncho para testear su funcionamiento.

Todas las pruebas anteriormente explicadas tuvieron resultados satisfactorios.

4.2 Pruebas del software

Este conjunto de pruebas se realizó en 4 etapas. La primera fue el testeo del módulo BLE de manera individual. Para esto se utilizó un conversor USB serie de forma tal de poder ver la salida directa del módulo sin pasar por la EDU-CIAA-NXP. Este conjunto de pruebas fueron:

- Se envió el comando de inicio y fin de escaneo y se revisó que el LED indicador respondiera correctamente.
- Se utilizó la aplicación para teléfonos celulares android “Beacon Simulator” para simular la presencia de un IBeacon. Esta aplicación permite setear los parámetros enviados en el paquete de datos. Con esta información se revisó la información recibida fuera correcta

El resultado de este conjunto de pruebas fue mayormente satisfactorio aunque la precisión de la distancia estimada fue menor a la esperada. Sin embargo se descubrió que la causa de esto no era un problema de implementación sino la propia falta de precisión del método de estimación.

La segunda etapa de pruebas consistió en conectar el módulo BLE a la EDU-CIAA-NXP. En este caso utilizando las funciones de debug presentes en el microcontrolador se revisó cómo se modificaban las variables internas del programa a medida que pasaba el tiempo, se recibe información desde el módulo BLE o desde comandos enviados por puerto serie.

La tercera etapa consistió en comunicarse con la EDU-CIAA-NXP utilizando el programa de control. Nuevamente se probaron todos los comandos presentes en la interfaz gráfica y se comprobó que los mismos llegaran a la plataforma utilizando las funciones de debug.

Para finalizar se realizó una serie de pruebas integrales en las que se comprobó la experiencia del usuario del sistema. En las mismas sin utilizar ninguna herramienta de debugo se configuró al sistema de diferentes formas y se comprobó que los LEDs se modifiquen correctamente ante modificaciones en los beacons cercanos así como ante una conexión y desconexión repentina del sistema.

Todas estas pruebas tuvieron resultados satisfactorios por lo que se puede afirmar que, segun estas pruebas, el sistema completo se comporta como fue especificado.

CAPÍTULO 5

Conclusiones

5.1 Conclusiones generales

En este trabajo se lograron los objetivos propuestos durante su planificación. El sistema diseñado e implementado cumple con todas las especificaciones esperadas. Además de esto durante su desarrollo se adquirieron conocimientos y experiencia con las distintas tecnologías y herramientas involucradas.

También es para destacar que este proyecto contaba con 3 módulos, cada uno en una plataforma de hardware independiente, y la coordinación de los mismos funcionó de la manera esperada.

5.2 Próximos pasos

Los próximos trabajos asociados a estos tendrían que estar dedicados al desarrollo de los demás módulos planteados en la [sección 1.2](#). Esto permitirá que el sistema completo existiera y se pudiera ofrecer el servicio dentro de la industria de la seguridad que disparó esta idea. Sería muy recomendable que el siguiente paso contemplara la realización del sistema que centraliza la información de varias estaciones como las implementadas en este trabajo.

Si lo que se pretende es mejorar este módulo en particular serían deseables las siguientes modificaciones:

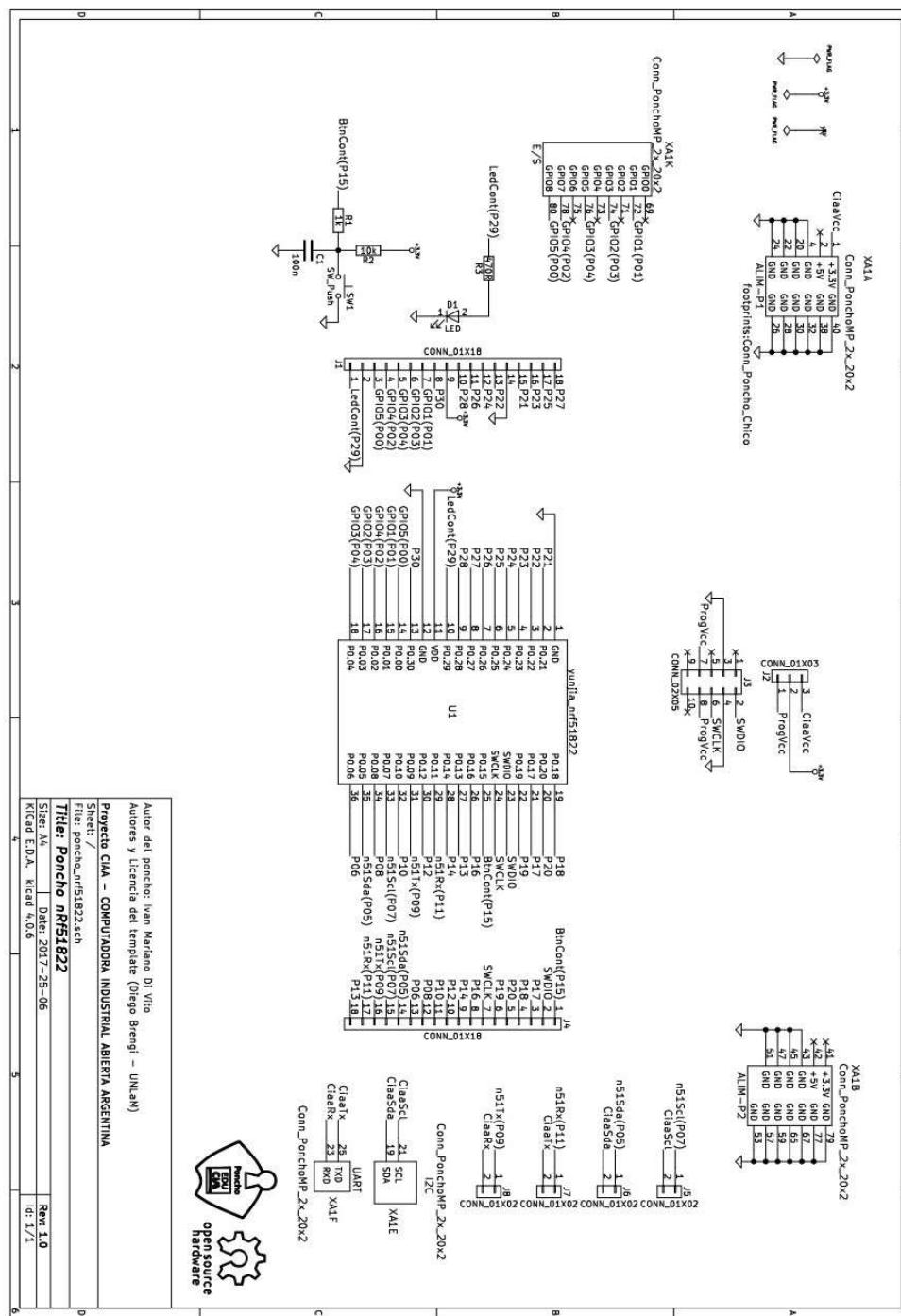
- Implementar los demás formatos de Beacon especificados en la [sección 2.2.1](#) para aumentar la flexibilidad del sistema.
- Se podría intentar fusionar el software de la EDU-CIAA-NXP con el del módulo BLE. Esto bajaría en gran medida los costos del sistema. Sin embargo esto requiere un previo análisis ya que las especificaciones del microprocesador del módulo BLE son más modestas que las de la placa de desarrollo.

Bibliografía

- [1] Proyecto CIAA (2016, Jun 26). EDU-CIAA-NXP [Online]. Available: <http://www.proyecto-ciaa.com.ar/devwiki/doku.php>
- [2] MBED documentation [Online]. Available: <https://docs.mbed.com/>
- [3] CIAA Firmware_v2 github page[Online]. Available: https://github.com/ciaa/firmware_v2
- [4] FreeRTOS documentation [Online]. Available: <http://www.freertos.org/RTOS.html>
- [5] Kotlin documentation [Online]. Available: <https://kotlinlang.org/docs/reference/>
- [6] TornadoFX Guide [Online]. Available: <https://edvin.gitbooks.io/tornadofx-guide/content/>
- [7] Repositorio del proyecto[Online]. Available: https://github.com/ivandivito/control_acceso

Anexos

A. Diagrama esquemático del poncho diseñado



A. Diagrama pcb del poncho diseñado

