

Localización basada en beacons

Informe de avance



7 de diciembre de 2017

Iglesias Matías, 828/5 – Stranieri Jorge 917/5

Taller de Proyecto II

# Propuesta original del proyecto.

El objetivo del proyecto era desarrollar la detección de las señales (emitidas por Beacons que utilicen el protocolo Eddystone) a través de cada microcontrolador; luego, con esta información en una notebook (que actuará como servidor) se realizaría una aplicación web que, al menos, permita ingresar un ID y muestre cual es el microcontrolador que se encuentra más cerca del ID (Beacon) ingresado.

Dentro del alcance de este proyecto no se pretendía desarrollar los Beacons (emisores de señal) porque sería innecesario dado que todos los Beacons que sigan el protocolo Eddystone emiten un ID que es suficiente para identificar el objeto/persona (esto no quiere decir que sea inútil desarrollarlos, pero hacerlo aumentaría el costo de tiempo y dinero). Además, se propuso no comprar ningún Beacon, sino utilizar alguna de las aplicaciones móviles disponibles para que el celular se comporte como un Beacon.

Para el testeo del correcto funcionamiento se requerían dos microcontroladores con sus respectivos módulos bluetooth; aun así, se pretendía que el sistema funcione con la cantidad que se desee de acuerdo al caso de uso específico.

# Correcciones y cambios de la propuesta.

## Indicadas por la cátedra.

En lo que respecta a la presentación de propuesta entregada anteriormente, se nos marcó que:

* Sólo utilizáramos un Arduino y un módulo Bluetooth.
* El sistema informe cuando se ingresa desde el servidor web un ID de una Beacon que no se encuentra reconocida por el módulo.

## Definidas por el avance y disponibilidad.

### Correcciones en base al módulo Bluetooth:

El módulo comprado (CC-41-A) es una imitación del HM-10 que contiene un firmware distinto y entonces trae muchos menos comandos (entre ellos, no se encuentra el comando para "escanear") y además posee muy escasa documentación. Las funciones que disponemos por parte de éste módulo nos hace imposible desarrollar el proyecto que teníamos previsto. Por otro lado, tuvimos la opción de flashear el módulo para que el firmware del mismo contemple las funcionalidades del módulo BT HM-10 con el cual partimos con la idea de este proyecto. Gracias a que el flasheo del módulo tuvo éxito podemos seguir con esta propuesta, aunque aun así hay pequeñas cosas que cambiaron.

El cambio más importante es que en un principio teníamos pensado utilizar el protocolo de beacons llamado Eddystone, pero los comandos que permiten interactuar con el módulo HM-10 al escanear solamente detectan los beacons que utilicen el protocolo iBeacon. Este cambio no es un limitador de la funcionalidad del proyecto, dado que (como se dijo antes) en el alcance del proyecto solo se requiere conocer el identificador del beacon, cosa que está disponible para ambos protocolos; pero habíamos elegido Eddystone dado que, en caso de querer expandir el proyecto, es más flexible y permite enviar otros datos (una URL, por ejemplo) dentro del paquete de información.

### Mencionamos los protocolos para mostrar las diferencias:

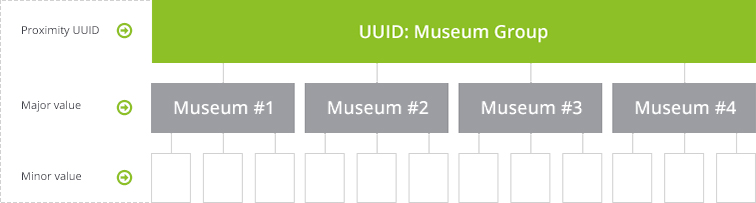
Protocolo Eddystone. Es un proyecto de código abierto desarrollado por Google. A diferencia de iBeacon, tiene soporte oficial para iOS y Android. Un beacon configurado con este protocolo puede emitir uno de los siguientes tipos de paquetes:

* Eddystone-UID: contiene un identificador de un beacon.
* Eddystone-URL: contiene una URL.
* Eddystone-TLM: es emitido con los paquetes anteriores y contiene el estado de salud de un beacon, como el nivel de batería, por ejemplo.
* Eddystone-EID: contiene un identificador encriptado que cambia periódicamente.

Protocolo iBeacon: creado por Apple, fue el protocolo que introdujo la tecnología BLE mundialmente y define 3 parámetros:

* UUID: identifica un grupo.
* Major: identifica un subgrupo de beacons dentro de un grupo más grande.
* Manor: identifica un beacon específico.

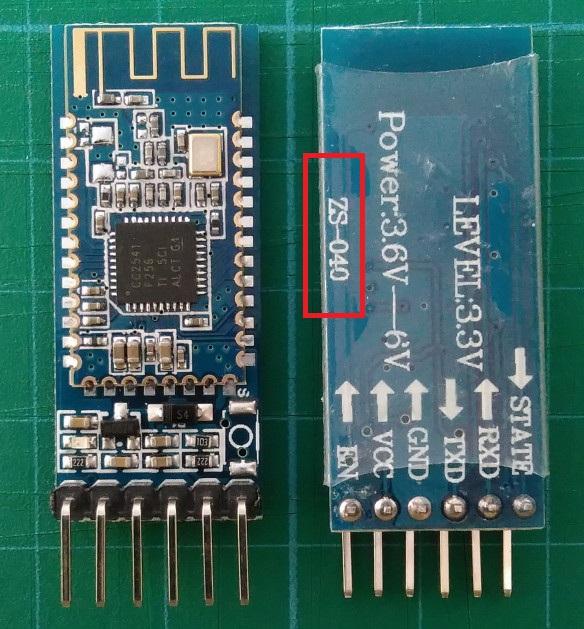
En la siguiente imagen se puede ver un ejemplo para el uso de los parámetros de este protocolo.



Si bien lo mencionado anteriormente son las convenciones adoptadas por Apple para el desarrollo de sus iBeacons, en nuestro proyecto no tenemos la necesidad seguir esa categorización a través de los parámetros, por lo que sólo los consideraremos como parámetros identificación extras. Por el momento solo tenemos la intención de utilizar el parámetro UUID.

### Diferencias entre el módulo CC-41-A y el HM-10:

Dejamos alguna de las diferencias del módulo que compramos con el HM-10 que creíamos haber comprado. Hay diferencias a simple vista de los módulos que pueden hacer que se evite este inconveniente que tuvimos. El módulo comprado se llama CC-41-A y es fabricado por la compañía [Bolutek](http://www.bolutek.com), está identificado en la placa como “zs-040” cómo se puede ver en la siguiente foto:



En el momento de la compra de algún módulo Bluetooth es importante fijarse en este detalle, ya que el módulo HM-10 va a tener su nombre escrito en la placa. La recepción de los comandos AT enviados por Arduino requieren que se agreguen los caracteres de final de línea (\r\n), en algunas terminales se puede configurar esto para que se envíen por defecto, por el contrario con el HM-10 no es necesario. La versión del Firmware del CC-41-A que viene por defecto es la V3.0.6.

# Descripción de hardware y conexiones.

## Conexión general.



Corregir: No se ven bien las conexiones, que es una de las razones básicas para pedirles un esquemático. Si no me acuerdo mal, están usando Arduino Mega así que este gráfico es incorrecto por más que “conceptualmente” esté bien. Un esquemático se pone en parte para entender y en parte para replicar y con este esquemático no se podrí replicar luego lo que uds. hicieron de la forma en que lo hicieron.

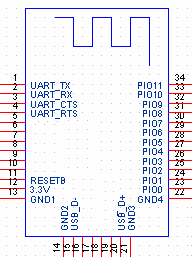
diagrama esquemático general + foto de conexiones general. Aclarar q Arduino-pc va por usb y ya ta

## Conexión normal para comunicación con el módulo.

## Conexión para proceso de flasheo.

Esta conexión es utilizada solamente en el proceso que se encuentra explicado en el apartado “Proceso de flasheo”, de la sección 6.1. Se requieren 5 cables tipo jumper, soldador, algo de estaño y el cable USB para el Arduino.

Se debe conectar el Arduino directamente al circuito integrado CC2541 que contiene el módulo CC-41-A. Para esto se debe realizar el soldado de un cable tipo jumper a cada uno de los tres pines que podremos ver a continuación en el esquemático del módulo. Esta tarea fue realizada por uno de los profesores de la cátedra debido a nuestra nula experiencia en soldado. Cada uno de estos cables se conectaron a tres pines del Arduino, como se detalla a continuación:

Mejorar: Agregar una foto con el módulo con los pines soldados y su conexión a la placa

Además de estos tres pines soldados, lógicamente se utilizaron también las conexiones Vcc y GND del módulo, siendo conectadas al Arduino en el pin de 5V y a uno de los pines GND, respectivamente. No se utilizaron en este proceso los pines de Rx y Tx. Finalmente, se conecta el Arduino a la computadora con el cable USB.

# Descripción funcional.

El esquema general de nuestro proyecto consiste en poder recibir señales BLE transmitidas por las Beacons, procesar la información transmitida de la señal y ser mostrada si así es requerida mediante el servidor web.

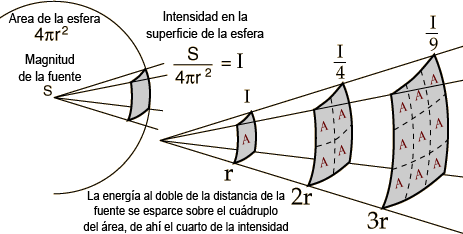
La información será captada por el módulo Bluetooth que va a estar escaneando el entorno debido a las Beacons que estarán transmitiendo datos cada cierto intervalo de tiempo. Una vez captada la información por el módulo Bluetooth éste transmitirá los datos mediante el puerto serie UART de la placa Arduino, en esta instancia no definimos todavía si el proceso de los datos se hará desde la placa o directamente en la computadora antes de ser guardados o enviados al servidor web. Esto se debe a que tenemos que tener en cuenta la velocidad en que se puedan procesar los datos y las ventajas o desventajas de hacerlo en un dispositivo como en otro. Esperamos que la velocidad de muestreo sea aproximadamente 1 segundo. Igualmente se procesen los datos o no, una vez que se envíen los datos serán recibidos en la computadora a través de un programa escrito en lenguaje Python, el cual los guardará en un archivo de texto para luego ser consumidos por el servidor web una vez estén disponibles. En cualquier momento un usuario puede ingresar al servidor para requerir información sobre la Beacon que desee, ingresando el ID de la misma. El ID que se ingresará en la página web será un ID de 8 caracteres hexadecimales (es decir, permitiendo valores de 0 a 9 y A a F). El ID que se ingresa corresponde al UUID de 32 caracteres de la Beacon. Decidimos acortar la cantidad de caracteres para trabajar con mayor simpleza, ya que para el alcance del proyecto no son necesarios los 32 caracteres. Igualmente, el ID de 8 caracteres se formará a partir del UUID de la Beacon.

## Cálculo de la distancia.

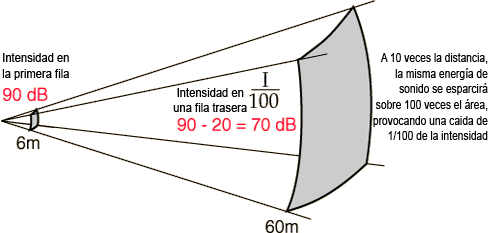
Para determinar la distancia en metros a la cual se encuentra la Beacon, que será el transmisor de la señal medida en potencia en escala de dBw(Decibel-miliwatt), lo haremos teniendo como base la Ley del Inverso del Cuadrado para la relación de éstas dos magnitudes.

#### Ley del inverso del cuadrado.

Cualquier fuente puntual que extiende su influencia en todas las direcciones por igual, sin límite de alcance, se regirá por la ley del inverso del cuadrado. Esto proviene de consideraciones estrictamente geométricas. La intensidad de la influencia en cualquier radio dado “r”, es la intensidad de la fuente en el origen, dividida por el área de la esfera. La ley del inverso del cuadrado se aplica a diversos fenómenos como fuerzas de gravitación, campos eléctricos, luz, sonido o radiación.

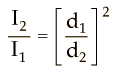


En la siguiente imagen mostramos un ejemplo de esta ley aplicado a la intensidad del sonido y su atenuación.



Este es un ejemplo hecho en base al desplazamiento del sonido y la atenuación de su potencia, y como se detalló antes, esta ley se puede aplicar a varios fenómenos físicos como las ondas electromagnéticas que son de nuestro interés en el proyecto.

Expresada como ecuación la ley del Inverso al cuadrado toma la siguiente forma.



Siendo “I1” la intensidad de la señal medida en dB a una distancia “d1” medida en metros (medida de referencia). Y lógicamente “I2” será la intensidad de la señal obtenida a una distancia “d2”, la cual es el valor a calcular. Para esto, despejamos “d2” de la ecuación anterior, y obtenemos la siguiente fórmula.

Aun teniendo en cuenta esta fórmula, es posible que se deban hacer ciertos ajustes en el algoritmo debido al ruido u otras interferencias posibles.

## Dispositivos disponibles.

Los dispositivos que utilizaremos para nuestro proyecto son:

#### *Notebook*

Específicamente estaremos usando una Notebook Acer Aspire E 14, donde se la utilizara como un servidor web que se montará en localhost, aunque de ser posible el servidor que sea accedido de cualquier dispositivo. Ya tenemos la prueba hecha de montar un servidor web como se hizo en la práctica 1 de esta materia.

#### *Arduino MEGA 2560*

El modelo del microcontrolador de la Arduino Mega es el Atmega 2560. Esta placa es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Individualmente la placa funciona correctamente, además de que la integramos con el módulo Bluetooth comprado para testear que funcionen correctamente ambos juntos. Además, la placa se conecta mediante USB a la computadora mediante la cual se puede interactuar directamente con el software de Arduino.

#### *Módulo bluetooth CC-41-A (HM-10)*

El módulo Bluetooth comprado es el CC-41-A, que es una imitación del módulo Bluetooth HM-10. Igualmente, gracias a que se pudo flashear el firmware del HM-10, el módulo comprado está actualizado (luego de realizar todo el procedimiento que se explica más adelante en este informe) con los comandos AT del mismo, que son los requeridos para llevar a cabo el proyecto. Lo probado hasta ahora son las funcionalidades del mismo módulo chequeando la respuesta en el monitor serie del IDE de Arduino. Características técnicas del módulo:

* + Protocolo Bluetooth: Bluetooth V4.0 BLE.
  + Límite: pueden alcanzar hasta 70 metros en línea de vista.
  + Basado en el integrado CC2541 de Texas Instruments.
  + Bluetooth clase 2.
  + Módulo de nivel industrial SPP Bluetooth.
  + Antena embebida.
  + Tamaño: 26.7 x 13 x 2 mm.
  + Rango de temperatura: -40 ℃ a 85 ℃.
  + Frecuencia de Funcionamiento: Banda ISM de 2,4 GHz.
  + Método de modulación: GFSK (Gaussian Frequency Shift Keying).
  + Ganancia: - 84 dBm o Pérdidas BER 0,1%.
  + Velocidad de transferencia: asíncronos: 6 KBPS; síncronos: KBPS.
  + Servicios de apoyo: central y periférico UUID FFE0, FFE1.

#### *Celulares*

Los smartphones que vamos a usar en el proyecto serán principalmente para emular una Beacon. Esto quiere decir que nuestro celular simulará ser algún objeto que se puede localizar mediante otro dispositivo Bluetooth BLE que se encuentre buscando señales. Esto se puede hacer gracias a diversas aplicaciones disponibles, como “Beacon Simulator”, utilizando el protocolo iBeacon de Apple. Y desde el celular se podría ingresar al sitio web, así de esa forma podríamos ver la información de nuestro “objeto”. Entonces al celular lo tomamos directamente como una Beacon, que es un dispositivo hardware que solamente transmite un paquete de información en forma periódica. Este paquete puede ser recibido por todos los dispositivos compatibles que estén al alcance de la señal, para luego procesarla como deseen.

Todos los beacons poseen las siguientes características configurables independientemente del fabricante o el protocolo utilizado:

* + Tx power: es la potencia con la que los beacons emiten la señal que contiene la información. Esta señal viaja por el aire y por lo tanto disminuye su intensidad debido al ruido de otras señales, objetos que obstruyan su paso o al aumentar la distancia. Cuanto mayor sea esta potencia, más batería consumirá el beacon, pero tendrá un mayor alcance y resistencia al ruido.
  + Intervalo de emisión: este valor define el período con la que emite la información un beacon. Cuanto más bajo sea este intervalo de emisión, más consumirá el beacon.

## Identificación de las partes del proyecto.

### Alimentación del dispositivo/placa de desarrollo:

Por parte de la alimentación de los dispositivos, tenemos dos para remarcar. La placa Arduino Mega y el módulo Bluetooth CC-41-A conectado en la misma.

La placa Arduino Mega permite un voltaje operativo de 5V, por esto decidimos proveer la alimentación a la placa desde la PC mediante la conexión de un cable USB (que entrega justamente 5V).

El módulo Bluetooth CC-41-A puede funcionar a 5V de alimentación, que la placa Arduino le entregará conectando el pin VCC del módulo con el pin 5V de la placa, el pin de GND del módulo al pin de GND de la placa Arduino.

### E/S de la placa con el exterior.

La base de nuestro proyecto en lo que respecta a entrada y salida de la placa con el exterior, se basa en la comunicación mediante Bluetooth 4.0 BLE. Esto es posible a la conexión del módulo BT a la placa Arduino, así mismo la entrada/salida de la placa con el módulo es a través de la conexión UART. Para esto el pin de Transmisor(Tx) de la placa se conecta al pin Receptor del módulo (Rx) y el pin Rx de la placa al pin Tx del módulo. A continuación, en el informe se detalla mejor ésta conexión. SI PONEMOS TODO ESTO EN EL 4, EL DETALLE ESTARIA ANTES, EN EL 3 (NO A CONTINUACION)

### Comunicación de la placa Arduino con la PC.

La comunicación de la placa Arduino con la PC va a ser mediante un cable USB, que funciona tanto de alimentación para la placa, como el medio de transferencia de los datos, tanto desde la PC a la placa como de la placa a la PC.

### Sistema/interfaz web.

El sistema web pensado para este proyecto, es un sistema el cual el usuario puede usar para verificar de manera amigable características del objeto conectado, como la intensidad de la señal entre los dispositivos. A partir de la intensidad de la señal entre los dispositivos vamos a calcular y mostrar la distancia entre ellos; en caso de que no se pueda precisar la distancia, se podrá informar si se encuentra cerca, relativamente cerca o lejos. Contará con un formulario para que el usuario ingrese el ID del objeto a localizar o si es posible que pueda ingresar el nombre asignado a la Beacon.

El sistema web lo programaremos en Python utilizando Flask para facilitar el manejo del montado y las respuestas a los diferentes métodos HTTP requeridos por parte del Usuario. Los métodos de petición los cuales utilizará nuestro servidor son: “GET” y “POST”. El índice (index) del sitio web estará conformada por componentes describiendo el uso de la misma y un formulario para el ingreso del ID de una Beacon para mostrar su localización. Si el ID es correcto el servidor dispondrá de los datos de dicha Beacon, sino lo es se informará al usuario de esto.

Creemos que no es necesario agregar ninguna vista además de la propia asociada al index ya que se tratará de una interfaz web responsive, pero en caso de esto no ser suficiente, se podría agregar alguna vista con información más detallada sobre la Beacon referenciada.

Además de disponer la información de manera adecuada al usuario, el sistema web se encargará de la captación de los datos a partir de un archivo de texto, y el procesamiento de éstos. Las funciones del sistema y lo que se mostrará son: la distancia, ésta se dispondrá en metros para el usuario, pero cabe destacar que el servidor cumplirá con la función de calcular y determinar la distancia a partir de la intensidad de la señal expresada en Decibeles (dB), y se detallará además una descripción de proximidad de tres niveles (Cerca - medianamente cerca - Lejos).

Mejorar: aquí sería más que bueno y en parte necesario aclarar si hay uno o varios scripts/procesos Python. Quizás no lo pensaron aún, pero en el informe final seguramente habrán implementado alguna alternativa y sería necesario que lo aclaren. Normalmente con un gráfico esquemático de procesos en ejecución alcanza. Si no se entiende esto, me dicen y se los explico personalmente.

## Protocolos utilizados.

### Bluetooth 4.0 LE.

Bluetooth 4.0 Low Energy (LE) se fundamenta en la reducción del consumo y, por tanto, en minimizar la potencia de transmisión de la señal radio utilizada y el radio de cobertura. Con estas premisas, una velocidad de conexión 1 Mbps y códigos de redundancia para minimizar las transmisiones erróneas. Se pueden transmitir paquetes de datos pequeños y no es posible que soporte transmisión de audio. Los smartphones que soportan esta conexión depende de la versión de sistema operativo que tengan, además de las características de hardware propias del dispositivo. BLE posee un módulo de indexación más alto y necesita 2Mhz de ancho de banda, en lugar 1Mhz del clásico Bluetooth. La conexión entre dispositivos con BLE es mucho más rápida, con BLE se toman 6 ms y el bluetooth clásico hasta 6 segundos. A partir de estas versiones los smartphones soportan Bluetooth LE:

* Android 4.3 (aun así, debido a la aplicación que escogimos para utilizar en este proyecto, se requerirá al menos la versión 5.0).
* Apple iOS soporta BLE desde el iPhone 4S.

### Conexión serial UART.

La conexión serial estándar UART (Universal Asynchronous Receiver-Transmitter) es la que nos permite comunicar todos los datos que capta el módulo a la placa Arduino mediante los pines de Transmisión(Tx) y Recepción (Rx). Las funciones principales del chip UART son: manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa. El chip UART actúa de intermediario, ya que los dispositivos de interfaz separados lo usan para convertir las señales de nivel lógico del UART hacia y desde los niveles de señalización externos. En nuestro caso el chip UART es el intermediario entre los bits de información recibidos del módulo y los bytes que se envían a la computadora.

### Conexión USB.

En un Arduino usamos el puerto USB para tres funciones: cargar nuestro programa, conectarnos al puerto serie (UART) predefinido para comunicarnos durante la ejecución del programa y adicionalmente alimentar la placa.

### Protocolo iBeacon.

iBeacon es un protocolo de comunicación. Como se comentó, las beacons envían con determinada frecuencia señales bluetooth, y el protocolo de comunicación describe el formato que genera la señal.

### Protocolo HTTP.

El Protocolo de transferencia de hipertexto (Hypertext Transfer Protocol) es el protocolo de comunicación que permite las transferencias de información en internet. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es importante remarcar los métodos de petición que HTTP define para su utilización. Estos métodos permiten definir acciones específicas dependiendo de los eventos y los datos que el usuario demande. Cada método indica la acción que desea que se efectúe sobre el recurso identificado. Lo que este recurso representa depende de la aplicación del servidor. Estos son algunos de los métodos de petición que se usan generalmente:

* GET: Pide una representación del recurso especificado.
* POST: Envía los datos para que sean procesados por el recurso identificado. Los datos se incluirán en el cuerpo de la petición.
* HEAD: Pide una respuesta idéntica a la que correspondería a una petición GET, pero en la respuesta no se devuelve el cuerpo. Esto es útil para poder recuperar los metadatos de los encabezados de respuesta, sin tener que transportar todo el contenido.

## Uso del beacon.

El único software desarrollado por terceros que utilizaremos en el proyecto, es una aplicación Android que nos ayuda a convertir el celular en una Beacon. La aplicación se llama “Beacon Simulator” y como su nombre lo indica su función es la de transformar a cualquier smartphone que la utilice en una Beacon virtual que transmite información. Esta aplicación nos permite crear una colección de configuraciones de las Beacons para usarlas en cualquier momento para emular una Beacon física.

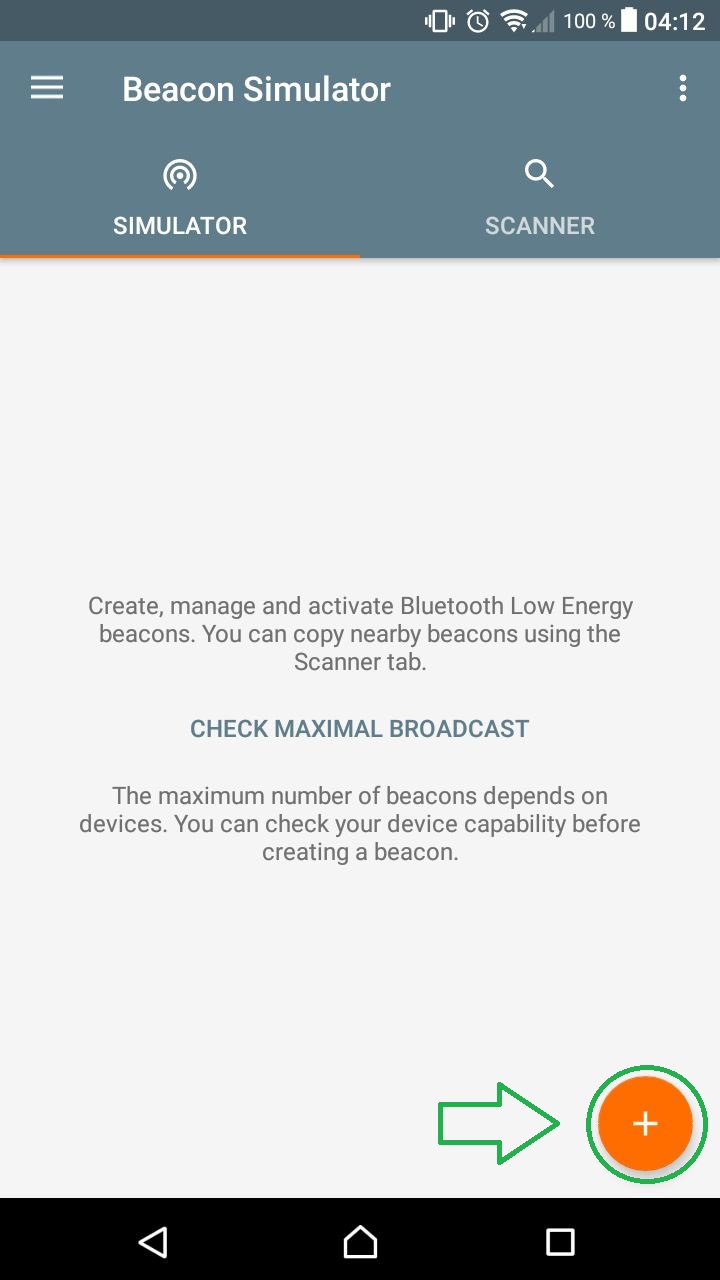
Beacon Simulator es una aplicación que se puede descargar gratis desde el Playstore de Android. Los únicos requerimientos de la aplicación son:

* Versión de sistema Android 5.0 y superiores.
* El smartphone debe ser compatible con el protocolo Bluetooth LE (generalmente esto se cumplirá siempre).

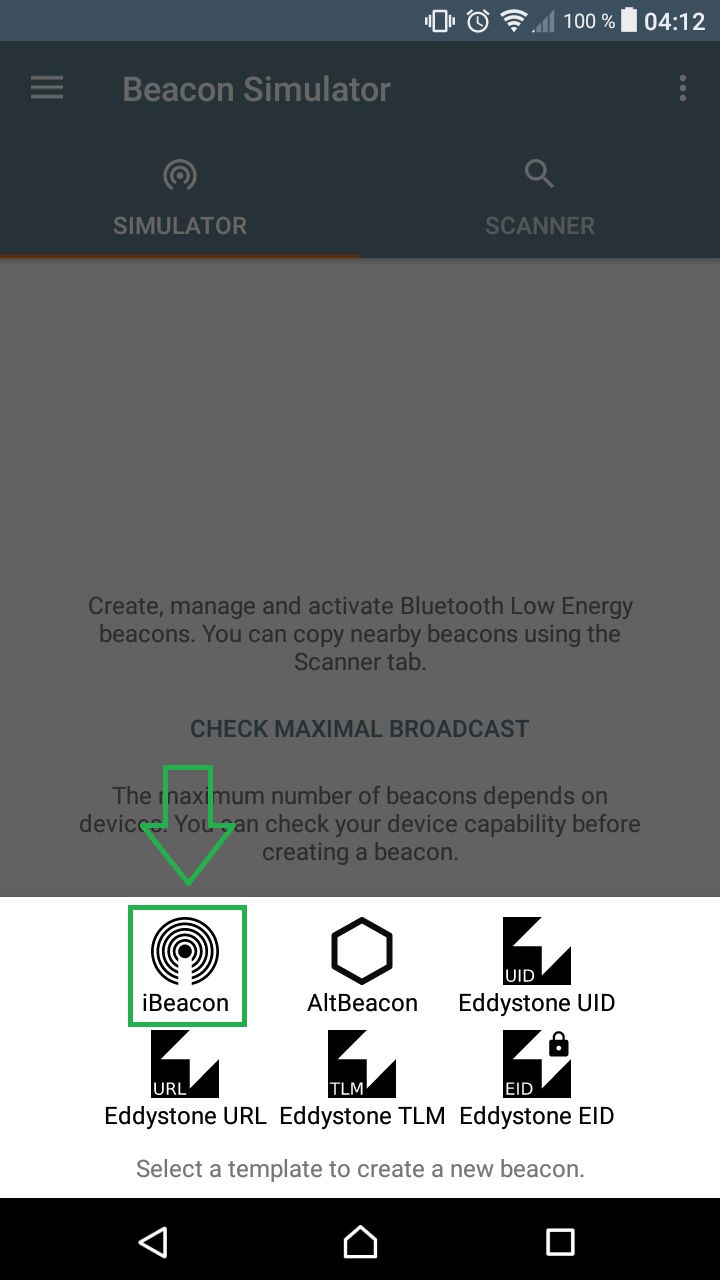
Luego de descargar la aplicación, detallamos los pasos a seguir para configurar una iBeacon virtual que será la encargada de estar transmitiendo información cada cierto intervalo.

### Configurar iBeacon

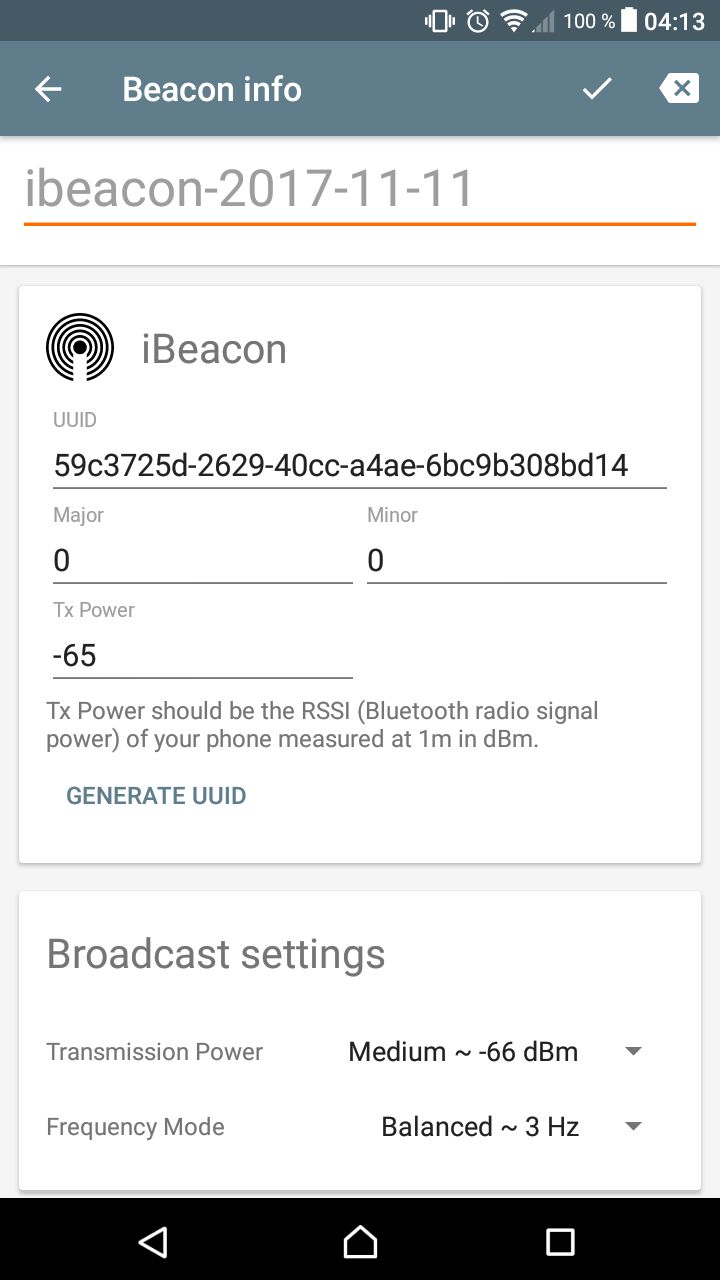
Lo primero que se debe hacer, previo a abrir la aplicación para configurar una iBeacon, es activar el Bluetooth del dispositivo. Abrimos la aplicación y nos encontraremos con dos funcionalidades de la misma: “Simulator” y “Scanner”. En nuestro proyecto sólo utilizaremos el “Simulator” para configurar las Beacons virtuales. Para esto debemos apretar en el símbolo “+” como se indica en la siguiente captura de pantalla:



Ahora tenemos que elegir el tipo de protocolo de Beacon a ser simulado, como se explicó en este informe, utilizaremos el protocolo iBeacon, así que seleccionamos esta opción del menú emergente.



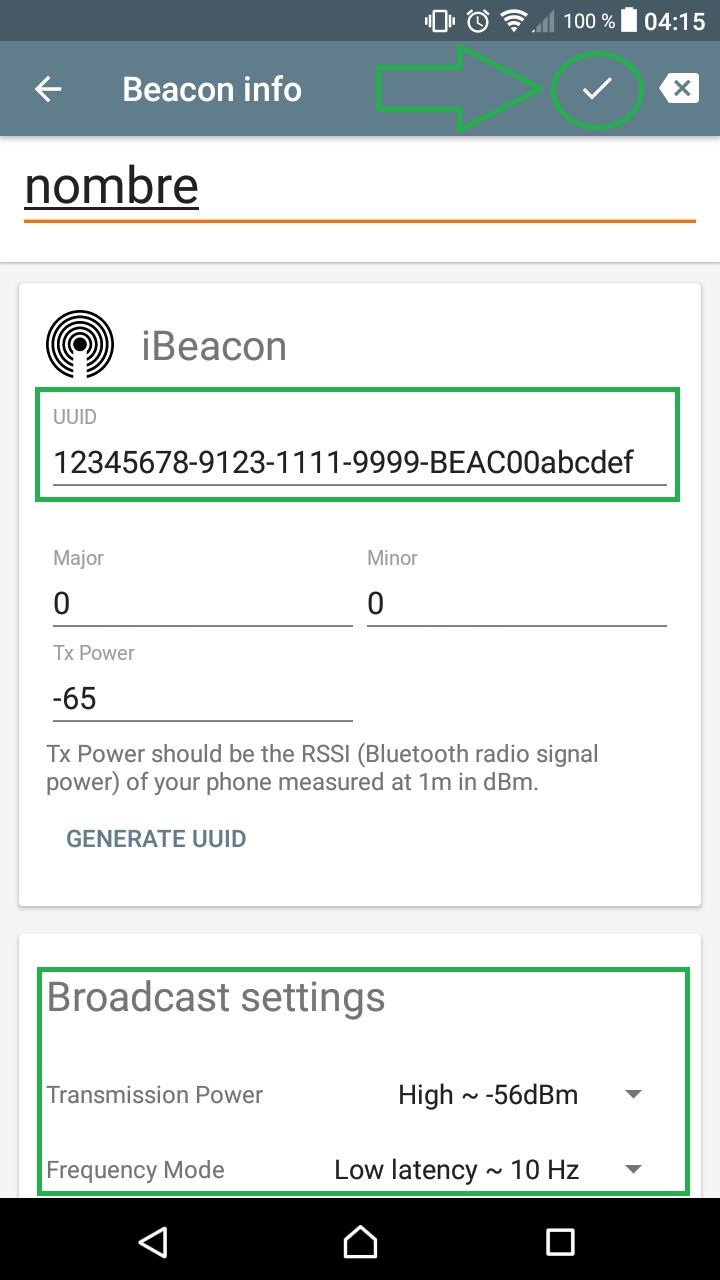
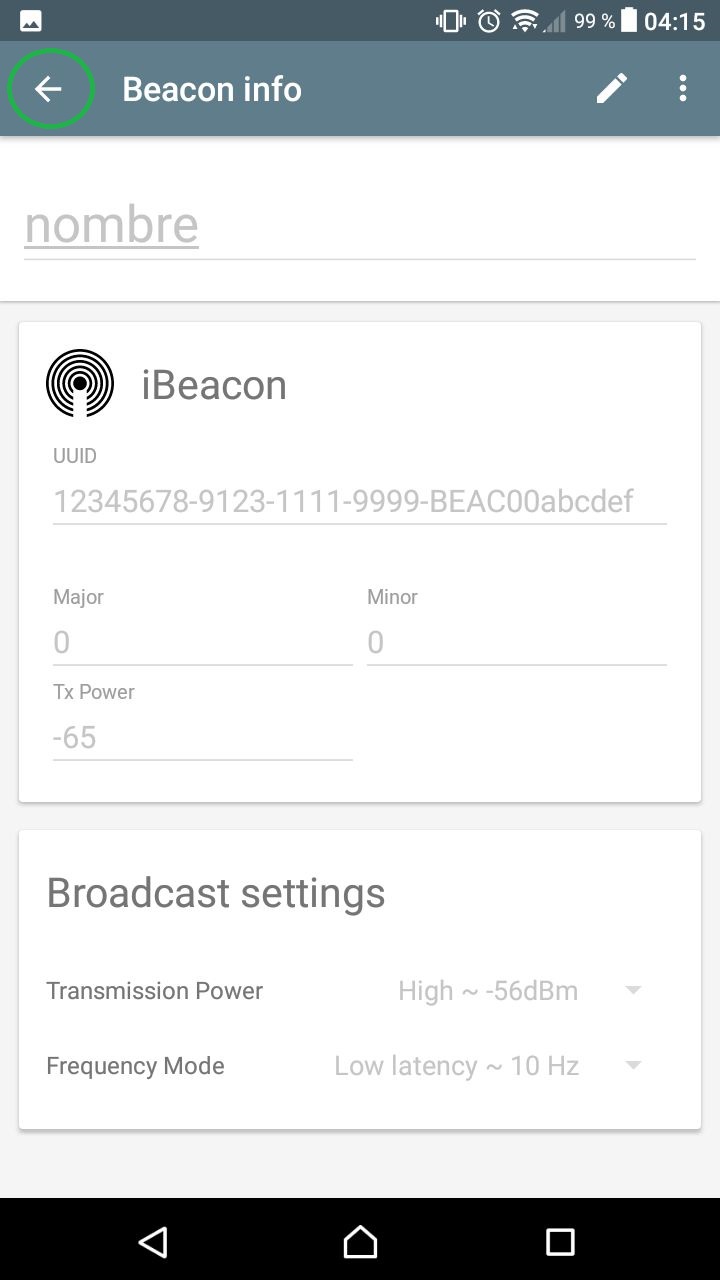
En esta nueva pestaña que se abre, se puede configurar los parámetros de la iBeacon. Como se muestra en la imagen siguiente, la aplicación contiene una configuración estándar a ser usada, de este modo podemos usar directamente los parámetros que nos brinda la aplicación.



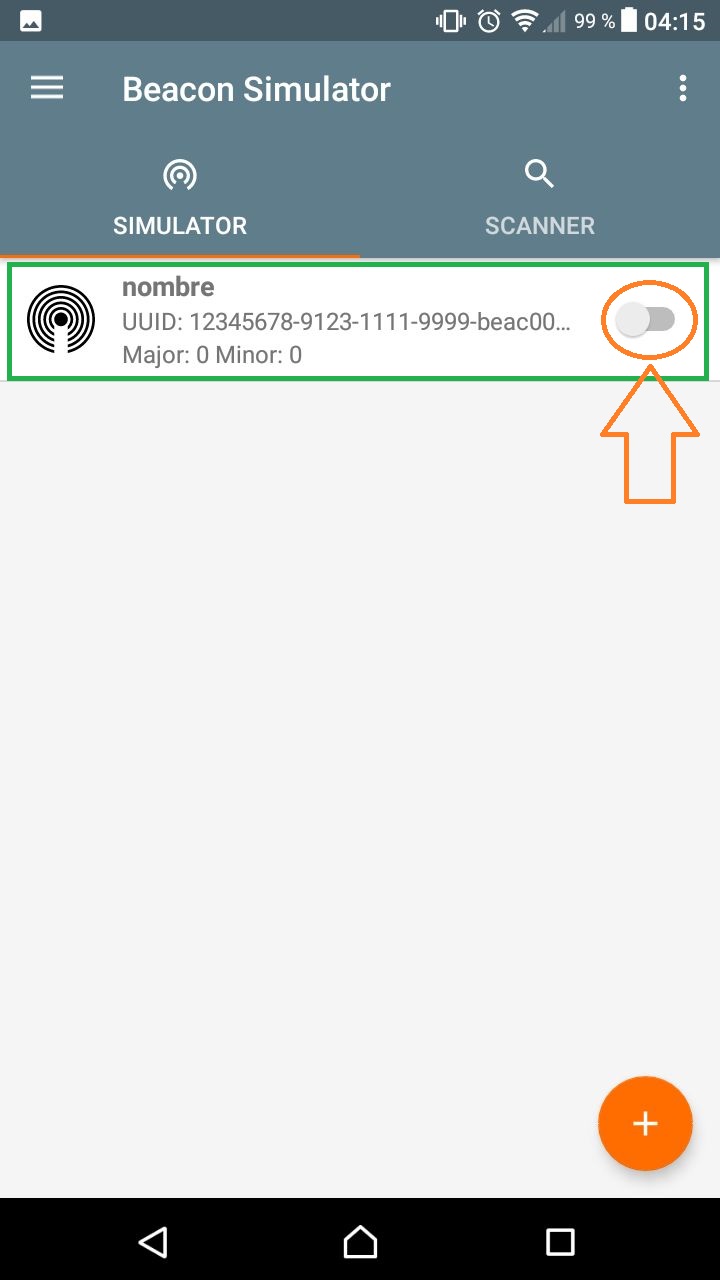
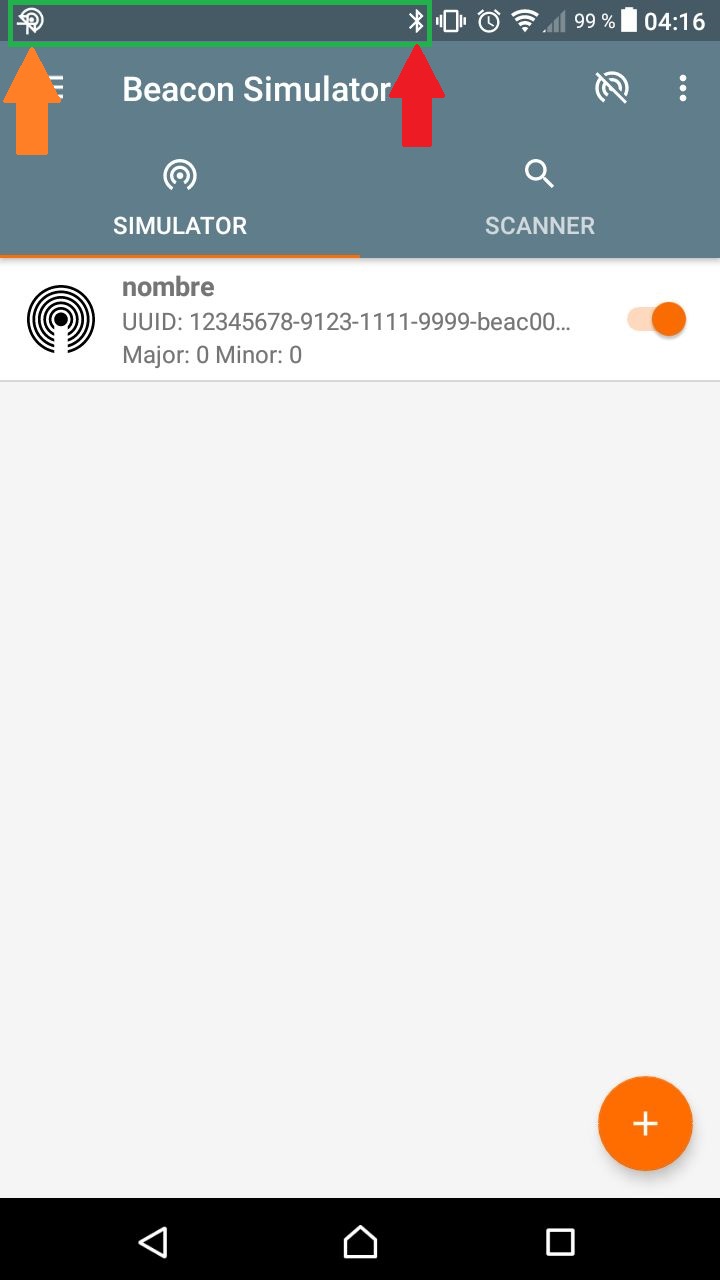
Los parámetros que se pueden configurar son: Nombre de la Beacon virtual, UUID, Major, Minor, Tx power; y luego las opciones de transmisión que la aplicación nos permite elegir entre algunos valores predeterminados. Los parámetros que nos interesa cambiar para el manejo de la Beacon en nuestro proyecto son:

* UUID: Está compuesto por 32 caracteres hexadecimales, separados por guiones que no hay que ingresarlos en la aplicación. Podemos ingresar cualquier cadena de caracteres que queramos, normalmente la única cadena que no se ingresa como UUID es lo que se llama una cadena nula que está compuesta por todos, pero sí se quiere usar, podemos hacerlo.
* Transmission Power: Define en dBm(Decibel-Miliwatt), una unidad de potencia, la intensidad de la transmisión de la Beacon. Para nuestro proyecto utilizaremos la intensidad más alta, “High ~ -66 dBm”.
* Frequency Mode: Define la frecuencia con la que la Beacon transmitirá la información. En nuestro proyecto utilizaremos un modo de frecuencia de “Latencia Baja ~ 10Hz”.

La configuración para nuestra Beacon Virtual quedaría de la siguiente manera:

Luego apretamos la tilde para que esta configuración de la Beacon Virtual a utilizar se guarde en la aplicación. Como se ve en la imagen de la derecha la configuración queda guardada y puede ser editada próximamente, luego queda apretar la flecha hacia atrás como se indica en la imagen para salir de la pestaña. Esto vuelve a la pestaña de inicio de la aplicación donde se puede ver el perfil de la configuración que guardamos:

Como se muestra en la imagen de la izquierda, para iniciar la simulación de la Beacon Virtual queda con apretar el botón indicado. En la imagen de la derecha podemos ver el estado de la simulación con el icono de la aplicación en la barra de estado del smartphone, y como dijimos antes, el Bluetooth tiene que estar activado para la transmisión.

# Descripción del software.

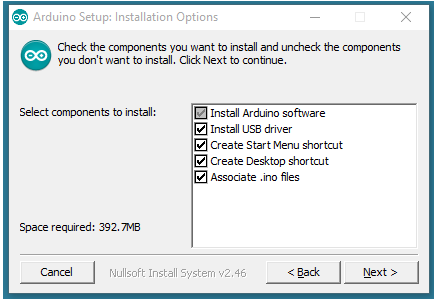
# Guía de instalación del ambiente de desarrollo y documentación del código.

## Ambiente de desarrollo.

### IDE de Arduino.

Para trabajar con Arduino se debe de instalar el IDE del mismo, que nos facilitará la comunicación de la computadora con la placa, la carga del código, y los drivers necesarios a ser instalados. Primero que nada, desde el sitio <https://www.arduino.cc/en/Main/Software> se puede descargar el IDE de Arduino gratuitamente. Se puede instalar en cualquier sistema operativo, nosotros instalamos el IDE en Windows de la siguiente manera:

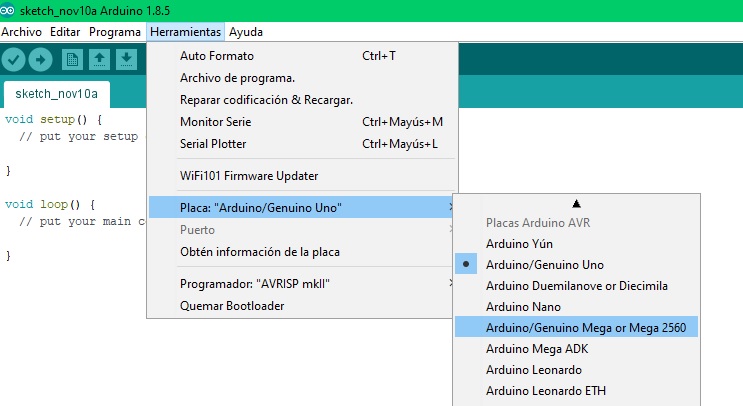
Una vez descargado el instalador, iniciamos la instalación y si el sistema operativo alerta de que se requiere una instalación de drivers, proceder con dicha instalación, igualmente una vez instalado el IDE, se pueden instalar los drivers necesarios la utilización de la placa.



En el paso siguiente paso nos permite elegir la carpeta de destino de instalación, que nosotros dejamos por defecto y de esta forma está terminada instalación.

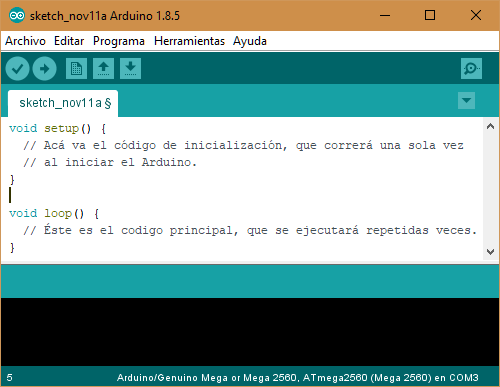
#### Configuración y uso del IDE.

Primero abrimos el IDE para configurarlo adecuadamente con la placa Arduino en específico.Para esto vamos a “*Herramientas > Placa:*” seleccionamos “Arduino/Genuino Mega o Mega 2560” como se ve en la siguiente imagen:

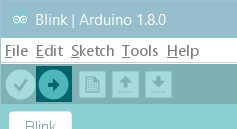


De la conexión con la placa resta indicar el dispositivo serie, para lo que necesitamos tener conectada la placa Arduino a la computadora. En “*Herramientas > Puerto”* se nos muestra una lista de los dispositivos seriales que podemos elegir. Normalmente suelen ser COM3 o superiores debido a que COM1 y COM2 están usualmente reservados. Se puede desconectar la placa y volver a abrir el menú, y el puerto que desaparece de la lista debería de ser el puerto de la placa Arduino.

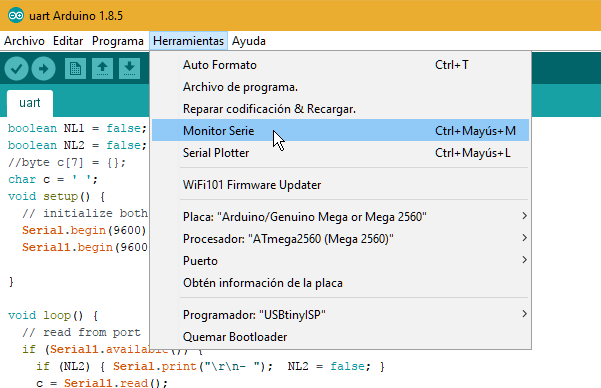
Vemos entonces algo similar a la siguiente imagen, siendo este el entorno de desarrollo donde se escribirá el código a ejecutar por el Arduino. A este código se lo llama “sketch”.



Una vez escrito el sketch deseado, se carga el programa al Arduino a través del botón “Subir”. En la parte inferior se verá una barra de progreso y un mensaje indicando el estado del proceso de carga.



Además, en la comunicación por comandos con el módulo se utilizará el monitor serie, que se puede abrir de la siguiente forma.

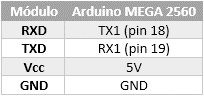


### Módulo CC-41-A.

#### Comprobación del módulo obtenido.

Como hemos comentado anteriormente, nos encontramos con el problema de que el módulo comprado CC-41-A no era el esperado HM-10, por lo cual necesitamos flashear el firmware del HM-10 en el módulo comprado para poder tener todos los comandos que se necesitan en nuestro proyecto. Si bien en el inciso 2.2 explicamos haber identificado el módulo por sus características físicas, reafirmamos esto probando el módulo.

Del módulo solamente utilizamos los pines RXD para la señal de entrada del módulo (comandos a ejecutar) y TXD para la señal de salida del módulo (respuesta a los comandos); lógicamente los pines GND y Vcc también se utilizan para la conexión a tierra y fuente de energía de 5V, respectivamente. A estos cuatro pines se les conectaron cables tipo jumper, que del otro extremo estarán conectado al Arduino de la siguiente forma:



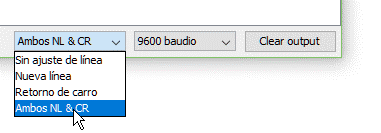
Una vez realizada esta conexión, conectamos el Arduino a la computadora a través del cable USB, abrimos el IDE de Arduino (hay que configurarlo como ya se explicó, en caso de ser necesario) para cargar el sketch de comunicación serie al Arduino y abrimos el monitor serie.

Corregir: aclarar la sección y página en la que está el sketch explicado en este informe. Además, adjuntar el sketch al informe.

Este sketch de comunicación serie se encuentra documentado más adelante en este informe, pero lo que hace es simplemente enviar al módulo el comando que ingresamos por el monitor serie, y luego imprimir en el monitor la respuesta que el módulo devuelve.

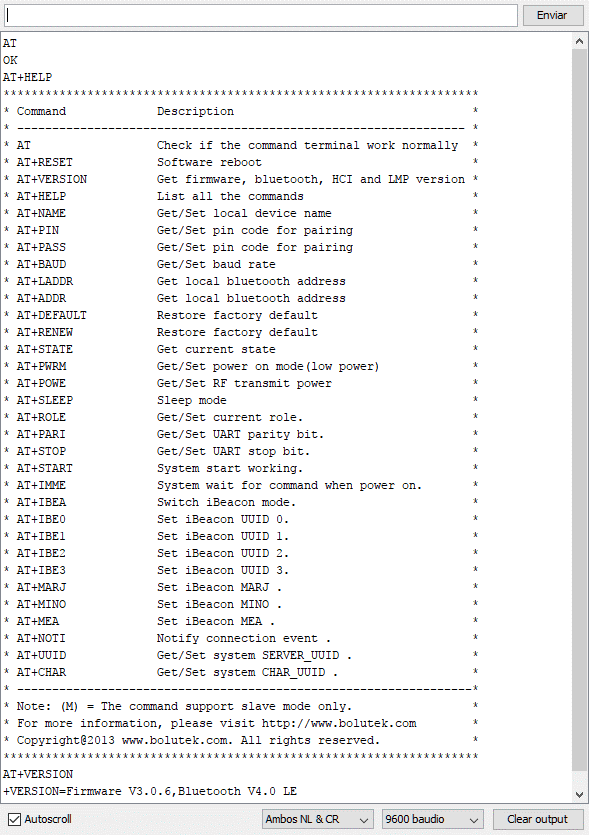
Según la documentación (y experiencia de otros usuarios) hallada sobre ambos módulos sabíamos que el CC-41-A, a diferencia del HM-10, requería que al final de cada comando se le agregue un retorno de carro (“\r”) y una nueva línea (“\n”). Por lo cual, sin agregar dichos caracteres, probamos enviar el comando “AT” que sirve para verificar que la conexión es correcta y devuelve “OK” en caso de serlo; como era de esperar, no recibimos respuesta alguna.

Continuamos probando agregando estos caracteres al final de cada comando. Recomendamos para simplificar esta tarea, seleccionar la siguiente opción que se encuentra en la parte inferior derecha dentro de la terminal serie (solo para usar en este firmware, dado que al flashear ya no se requerirán agregar).



Esta vez sí, al enviar el comando “AT” se recibió correctamente la respuesta “OK”. Probamos con otros de los comandos que se encuentran en la hoja de datos del CC-41-A (si bien la mayoría son similares o iguales a los del HM-10, no todos son idénticos). Al ingresar el comando “AT+HELP”, recibimos como respuesta una lista con todos los comandos que supuestamente (realmente ni siquiera son todos) admite el firmware actual, donde vemos que según las descripciones, no se encuentra ninguno que permita realizar el escaneo.

Además, al final se puede ver el sitio web del fabricante (Bolutek) del dispositivo CC-41-A.



Una vez despejadas todas las dudas acerca de la utilidad de este módulo y su firmware, optamos por realizar el intento de flasheo del firmware del módulo HM-10.

Mejorar: dado que lo que sigue es un proceso de varios pasos, conviene poner antes un resumen o listado de pasos que luego va explicado detalladamente. El detalle de cada paso está muy bien, pero no sé si lo entiendo del todo:

1. Flash del firmware
2. ¿Actualización del firmware?

Sería bueno documentar con una tabla que identifique qué es necesario en qué momento para hacer qué cosa. Quizás el problema es que no lo entiendo, pero justamente eso es un problema. La documentación debería ser clara para todos, en particular para el siguiente grupo/proyecto que replique esto para hacer el proyecto siguiente usando esto como base.

#### Proceso de flasheo:

Cabe aclarar que, como todo flasheo, es un proceso que peligra la integridad del módulo.

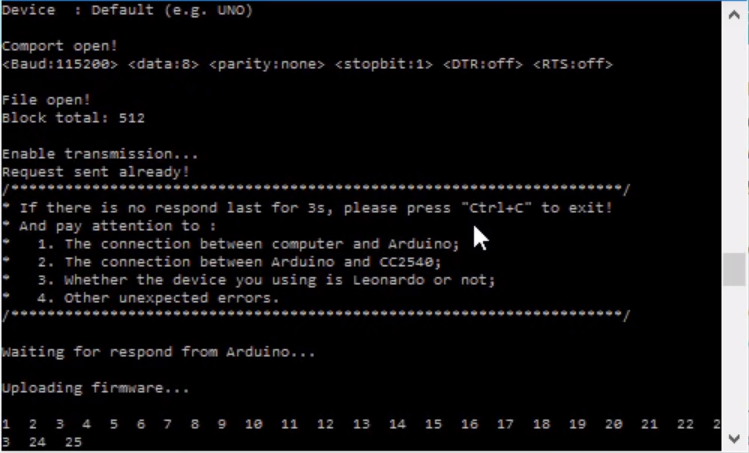
Para realizar este proceso se utilizó el siguiente software:

* IDE de Arduino.
* Sketch Arduino “CCLoader.ino”.
* Software “CCLoader.exe” para Windows.
* Firmware versión 540 del módulo HM-10.

El módulo se conectó al Arduino de la forma descripta en la sección 3.3. Luego se le cargó a través del IDE de Arduino (de la misma forma que se cargaría cualquier sketch) el sketch “CCLoader” mencionado en los requerimientos. A partir de aquí el módulo se encuentra listo para ser flasheado a través del software “CCLoader”.

Abrimos la consola de Windows (inicio -> ejecutar -> cmd) y dentro de ésta debemos navegar hasta la carpeta que contiene al archivo “CCLoader.exe” (el archivo del firmware “CC2541hm10v540.bin” debe ser ubicado en la misma carpeta que el “CCLoader.exe”). Para esto utilizamos el comando “cd /d C:\carpeta\de\ejemplo”.

Finalmente, ejecutamos el siguiente comando, que es el encargado de realizar el flasheo: “CCLoader.exe <NumeroDelPuertoCOMUtilizado> CC2541hm10v540.bin 0”. Veremos algo como la siguiente imagen:



Como se puede ver en la imagen, en caso de que no haya una respuesta del Arduino durante tres segundos se debe cancelar el flasheo pulsando “CTRL+C”. Particularmente a nosotros nos sucedió que no haya respuesta, pero luego de desconectar el Arduino de la computadora, volver a conectarlo y volver a intentar, el proceso se completó correctamente. Los números que se van imprimiendo debajo de “Uploading firmware…” irán subiendo hasta llegar a 512 que es cuando finaliza el proceso; como referencia podemos decir que esta subida demoró no más de tres minutos.

Una vez finalizado, se podrá desconectar el dispositivo de los pines 4, 5 y 6 del Arduino, y volver a conectarlo para su uso normal a través de los pines Rx y Tx.

#### Actualización del firmware (a versión 603):

En el proceso de flasheo se utilizó el firmware en su versión 540, pero en esta versión el comando de escaneo está hecho (sin posibilidad de modificarlo) para escanear durante 3 segundos, por lo que la frecuencia de muestreo del RSSI de cada beacon detectada sería muy baja; recién a partir de la versión 542 se agregó un comando para poder modificar la duración de el escaneo. Si bien esta limitación la habíamos averiguado incluso antes de realizar el flasheo, resulta que el archivo de los firmwares que publicados en la página oficial del HM-10 (<http://www.jnhuamao.cn>) no contienen dentro de ellos el bootloader requerido para correr el resto del firmware; el único firmware disponible en internet que también contiene el bootloader es el la versión 540. Es debido a todo esto que se utilizó dicha versión en el proceso de flasheo y no la 542.

Vale remarcar, que al igual que los firmwares sin bootloader no pueden ser flasheados, este método solamente sirve para una actualización de un módulo con firmware HM-10, es decir, en nuestro caso era obligatorio flashear primero para luego poder actualizar.

Dicho esto procedemos a explicar el proceso de actualización, el cual es proveído oficialmente por los fabricantes del módulo HM-10. Solamente se requieren los archivos llamados “HMSoft.bin” y “HMSoft.exe” contenidos en el archivo Zip del siguiente enlace:

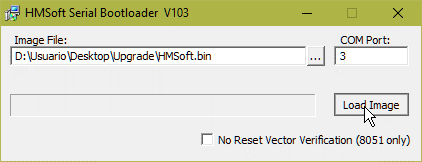
<http://www.jnhuamao.cn/rom/HMSoft-10-2541-V603.zip> (aunque se podría utilizar cualquier versión de la sección de descargas del fabricante, al momento de redactar este informe ésta versión es la más nueva).

Se conectó el módulo al Arduino de forma normal con los pines Tx y Rx, ya sin utilizar los pines soldados. El Arduino debe fue conectado al computador y se cargó el sketch para comunicación serie mencionado anteriormente. Luego, de forma similar a como se hizo cuando el módulo contenía el firmware del CC-41-A, se abrió la terminal configurada en el puerto COM (3) pero ahora no se debe incluir el retorno de carro (\r) y nueva línea (\n) como se hacía anteriormente.

Vale aclarar, que según la información provista por el fabricante del módulo HM-10, a partir de ingresar el siguiente comando el módulo entrará en un “modo de actualización”, del cual solamente se podrá salir realizando una actualización en forma exitosa (no saldrá de este estado a través de comandos, y ni siquiera cuando se lo desconecte de la fuente y vuelva a conectar). En nuestra experiencia quedamos trabados varias veces (unas 5) en este “modo de actualización”, pero afortunadamente logamos salir realizando el proceso de flasheo y volver al estado por defecto del firmware. Ingresamos entonces el comando “AT+SBLUP” en la terminal, y recibimos como respuesta “OK+SBLUP” indicando que se entró correctamente al “modo de actualización” (se debe recordar de cerrar la terminal para que no quede una comunicación abierta).

El problema por el cual nos quedamos trabados varias veces en este estado radicaba en la velocidad de la conexión (bps); si bien el fabricante indica que la conexión debe ser realizada a 9600bps, al hacerlo así el software que utilizaremos a continuación nos devolvía un error por exceso de tiempo (timeout). Por lo cual, se modificó el valor (9600) de iniciación de comunicación en el sketch a 115200 (valor mayor a 9600 escogido en forma aleatoria entre las opciones de configuración provistas por el IDE Arduino) y se lo cargó al Arduino.

Luego, solamente se tuvo que abrir el archivo “HMSoft.exe”, elegir el archivo “HMSoft.bin”, ingresar el puerto COM por el cual se encuentra conectado el Arduino y hacer click sobre “Load Image”, como se muestra en la siguiente imagen:

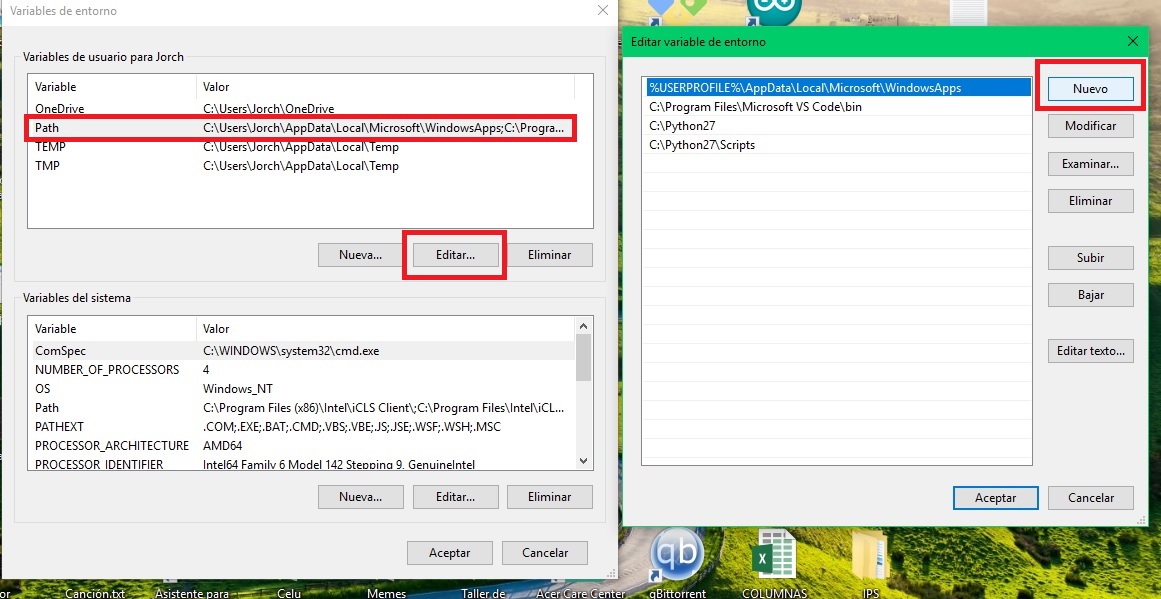


Una barra muestra el progreso, y no demora más de dos minutos. Al finalizar, aparecerá un cartel indicando una “Descarga completada exitosamente”. Ya se encuentra entonces el dispositivo con el último firmware actualizado y todos sus comandos disponibles. Se debe recordar que la comunicación que viene configurada por defecto en el módulo sigue siendo a 9600bps, sólo se la cambió a 115200bps para la actualización, por lo que para ingresar comandos se deberá volver a cambiar el sketch y cargarlo al Arduino.

### Sistema web.

#### Python:

El desarrollo del servidor web principalmente y otras funcionalidades se harán en este lenguaje de programación. Para la instalación de Python en Windows entramos en este sitio web <https://www.python.org/getit/> . En nuestro caso utilizamos la versión 2.7.6 para Windows. Una vez descargado el archivo lo instalamos dejando por defecto la ruta de instalación. Luego de esto, en Windows hay que agregar la carpeta en donde se instaló Python como variable de entorno, para mayor comodidad a la hora de usar los comandos desde la terminal. En Windows 10, abrimos el Explorador de archivos, apretamos click derecho sobre el icono “Este Equipo”, y luego “Propiedades”. Se nos abrirá una ventana de las “Propiedades del Sistema”, clickeamos en “Variables de Entorno”. Esto nos abrirá una nueva ventana en donde se nos muestra las variables de Usuario, entre ellas tiene que estar la variable “Path”, la cual clickeamos en editar. En la nueva ventana que se nos abre, clickeamos en “Nuevo” y nos permitirá ingresar, en este caso, la ruta en donde se tiene instalado Python, que sí es por defecto seria: “C:\Python27”.



Con éste mismo método agregamos a la variable de usuario “Path” la ruta “C:\Python27\Scripts” que nos facilita la instalación de paquetes Python.

#### Servidor web:

El servidor web lo implementaremos con Python, usando un micro-framework llamado Flask que nos facilita la programación a la hora de montar un servidor web.

Flask es un framework ligero escrito en Python y concebido para facilitar el desarrollo de aplicaciones web bajo el patrón MVC (Model View Controller), similar a Sinatra de Ruby o Express de Node.js. El término “ligero” quiere decir que su núcleo dispone de un número reducido de características, las mínimas para implementar las abstracciones del protocolo HTTP y su manipulación. Esto significa que no tiene ORM (Object-Relational Mapping), gestión de formularios, autenticación, etc.

#### Instalando Flask:

Para incluir Flask a nuestro proyecto, se puede instalar de diferentes formas, además depende del sistema operativo que estemos usando. A continuación, explicamos las dependencias necesarias para instalar Flask en Windows. Las dependencias necesarias para poder levantar un servidor con Flask son dos:

* Instalar Python 2.7 o superior
* Una vez que tenemos Python instalado hay que instalar “pip**”**, esta es una herramienta que nos permite hacer la instalación de forma sencilla de los paquetes para Python y desde la consola. Con pip luego instalamos Flask.

Para instalar “pip”**,** primero, descargamos get.py desde el sitio web <https://bootstrap.pypa.io/get-pip.py> . Si el navegador lo abre en lugar de descargar el archivo, podemos presionar CTRL+S (CTRL+G) en la pestaña abierta para guardarlo. Segundo una vez situado en la terminal en donde esté guardado el archivo “get-pip.py”, ejecutamos el siguiente comando:

Python get-pip.py

Una vez terminada la instalación, para comprobar que se instaló correctamente, en la terminal ejecutar el comando “pip”.

Luego ejecutamos en la terminal:

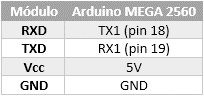
pip install Flask

De esta forma ya podemos implementar programas en Python con la facilidad de Flask para capturar los requerimientos que lleguen al sitio web y correr la aplicación en un puerto específico.

## Documentación de código.

### Sketch para comunicación serie:

Para poder realizar la comunicación entre la computadora y el módulo CC-41-A utilizamos la placa Arduino con las conexiones usando cables tipo jumper ya mencionadas anteriormente:



Para la elección de la conexión se aprovechó que el Arduino MEGA 2560 cuenta con 4 puertos serie (pines Tx y Rx de 0 a 3). Gracias a esto se pudo conectar a la computadora usando el cable USB, para lo que Arduino utiliza (“reserva” podríamos decir) el puerto serie 0, dejando inutilizables los pines 0 y 1 de la placa (y llamados RX0 y TX0, respectivamente); y para la conexión del módulo al Arduino se utilizó el puerto 1.

Entonces el sketch para la comunicación resulta bastante simple, ya que para enviar comandos solo se debe enviar los bytes que se reciben en el puerto 0 (por USB) y enviarlos al puerto 1 (módulo); para recibir la respuesta por consola es lo análogo, recibir bytes por el puerto 1 y enviarlos al puerto 0 (por USB, y se mostrará en el monitor serie).

Desarrollamos entonces el siguiente sketch:

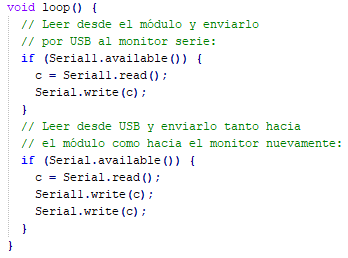
Para el manejo de la comunicación se utiliza la librería “Serial” que viene incluida en el IDE de Arduino. Ésta se encarga de enviar a los puertos serie los parámetros que le enviemos a las funciones, y de encolar los bytes que entran a un puerto para luego ser leídos a través de las funciones.

El método “setup()” es el que se ejecutará una sola vez al encender el Arduino. En nuestro caso, lo utilizaremos para (a través de la librería Serial) inicializar la comunicación de los puertos 0 y 1. Según esta descripto dentro de la hoja de datos del módulo, éste se comunica (por defecto) a 9600bps, por lo cual debemos indicarle eso a los puertos.



Como se puede ver, se le indica al puerto 0 (Serial) que inicie la comunicación a 9600bps; y luego lo mismo con el puerto 1 (Serial1).

Luego de haberse ejecutado el método “setup()”, automáticamente se ejecutará el método “loop()”, que se encargará de ejecutar su contenido repetidas veces e indefinidamente. Los comentarios puestos en el código (en verde) describen claramente lo que realiza cada porción del código.



Ya sea ejecutado a través de Serial o de Serial1, el método “available()” devuelve como resultado un valor verdadero en caso de encontrarse algún byte encolado en determinado puerto. De igual forma, el método “read()” devuelve el primer byte que se encuentre en la cola del puerto.

En el caso del método “write(c)”, se encarga de enviar el parámetro (“c” de tipo char, definido al inicio del sketch) a través del puerto indicado. Como se puede ver, dentro de la segunda sentencia “if” se encuentra invocada la función “write(c)” tanto para Serial1, como para Serial; esto logra una interfaz más amigable en el monitor serie, está hecho para que al ingresar un comando en el monitor serie, podamos ver cuál es el comando que se ha ingresado.

### Configuración del módulo:

Recordar que en forma similar a cuando aún no se había flasheado el firmware, los comandos se ingresan de la forma “AT+RESTODELCOMANDOparametro” (sin comillas, sin retorno de carro “\r” y sin nueva línea “\n”); sea el comando que sea, el parámetro debe ir inmediatamente después de “AT+RESTODELCOMANDO”; para cada consulta que se le quiere hacer al módulo acerca de su estado (si es que el comando lo permite), se debe ingresar como parámetro el carácter “?”. Ciertos comandos devolverán además la dirección web del fabricante, esto indica que el dispositivo ha sido reiniciado (por ejemplo ROLE, y obviamente también RENEW y RESET). Algunos comandos devuelven códigos de error como respuesta al ingresar un parámetro invalido, pero para indicar esto también podría ser que no haya respuesta alguna.

La conexión del módulo y configuración del IDE es igual que antes de haber actualizado, incluida también la carga al Arduino del sketch de comunicación serial ya mencionado.

Entonces una vez que tenemos en el módulo el firmware del HM-10, al menos versión 542, e independientemente del estado en el que se encuentre el módulo, realizamos los siguientes pasos para establecer una configuración inicial y que funcione del modo que necesitamos.

|  |  |  |
| --- | --- | --- |
| Comando | Respuesta vista en monitor serie | Parámetro:  **0: valor que debemos elegir (negrita).**  1: otro valor. |

Documentaremos estos comandos necesariosFALTA AGREGAR EL BAUDRATE según lo indicado en la hoja de datos del módulo, y además indicamos la opción que corresponde para nuestro uso, utilizando la siguiente estructura:

Comando para probar conexión:

|  |  |  |
| --- | --- | --- |
| AT | OK |  |

Restaurar el dispositivo a su estado “de fábrica”, borrando cualquier configuración anterior:

|  |  |  |
| --- | --- | --- |
| AT+RENEW | OK+RENEW |  |

Consultar/establecer si funciona modo maestro o esclavo:

|  |  |  |
| --- | --- | --- |
| AT+ROLE? | OK+GET:[P1] | [P1]:  0: Esclavo.  **1: Maestro.** |
| AT+ROLE[P1] | OK+SET:[P1] |

Consultar/establecer que hará el módulo al encenderlo (conectarlo):

|  |  |  |
| --- | --- | --- |
| AT+IMME? | OK+GET:[P1] | [P1]:  0: comenzará a trabajar de inmediato.  **1: no comenzará hasta que no se le indique (a través de comandos).** |
| AT+IMME[P1] | OK+SET:[P1] |

Consultar/establecer la duración de cada escaneo:

|  |  |  |
| --- | --- | --- |
| AT+SCAN? | OK+GET:[P1] | [P1]: cantidad de segundos del 1 al 9. **Elegir 1** |
| AT+SCAN[P1] | OK+GET:[P1] |

Reiniciar el dispositivo para asegurarnos que todas las opciones elegidas surtan efecto (algunos comandos lo requieren):

|  |  |  |
| --- | --- | --- |
| AT+RESET | OK+RESET |  |

A partir de aquí ya tendremos el módulo configurado para poder realizar escaneos de la forma deseada. Con el siguiente comando se realiza el escaneo de dispositivos iBeacon, y en nuestra aplicación son relevantes solamente los parámetros [P1] y [P4]:

|  |  |  |
| --- | --- | --- |
| AT+DISI? | OK+DISCS (comienza escaneo)  OK+DISC[P0:P1:P2:P3:P4]  OK+DISC[P0:P1:P2:P3:P4]  …  OK+DICE (fin del escaneo)  (cada OK+DISC representa a un dispositivo detectado, en este ejemplo serian dos dispositivos) | Siendo los parámetros:  [P0]: Factory ID (8).  [P1]: UUID del iBeacon(32).  [P2]: Compuesto por:  Major Value (4);  Minor Value (4);  Measured Power (2).  [P3]: MAC (12).  [P4]: RSSI (4). |

Si bien hemos probado todos estos comandos dentro de la terminal y corroborado su funcionamiento, aún no hemos conseguido hacer funcionar un sketch que automatice la tarea de enviar todos los comandos de configuración inicial y luego periódicamente envíe el comando de escaneo, junto con la lectura y envío de su respuesta al puerto serie. ESTO YA NO VA