

Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto Terminal

*Transmisión y registro de posiciones geográficas de dispositivos móviles,
usando el protocolo bluetooth*

Asesores:
Mario Alberto Lagos Acosta
José Ignacio Vega Luna

Alumnos:
Bernardo López Pérez
Raúl Hernández Jiménez

Trimestre 12 O
4 de enero de 2013

Objetivo general

Diseñar e implementar un sistema de rastreo de posiciones y tiempo basado en el protocolo **bluetooth**¹, para dispositivos móviles ubicados en un espacio abierto predeterminado.

Objetivo particulares

- Implementar la comunicación **bluetooth** entre los dispositivos móviles y un sistema digital basado en el micro controlador **PIC LV18FJ** [1].
- Diseñar un algoritmo para manipular la potencia de la señal **bluetooth** entre los puntos de acceso y dispositivos móviles.
- Diseñar un algoritmo para registrar la información de los dispositivos esclavos conectados al punto de acceso maestro.
- Identificar cuándo un dispositivo se encuentra fuera del área.
- Crear reglas de cruce para el sistema digital.
- Diseñar un algoritmo para monitorear y alertar cuándo algún dispositivo traspase una regla de cruce.

¹ **Bluetooth**: Tecnología inalámbrica que permite la conexión entre diferentes dispositivos empleando para ello un enlace de radio de baja frecuencia.

Introducción

El desarrollo tecnológico ha hecho posible la aparición de diversos dispositivos electrónicos. Estos dispositivos generalmente poseen una función específica, para así lograr satisfacer alguna necesidad básica del ser humano. A partir de esta necesidad comienzan a desarrollarse diversos métodos de interconexión de dispositivos, siendo la más primitiva el cable hasta así llegar a las más avanzadas como lo son las comunicaciones inalámbricas.

Probablemente la tecnología inalámbrica más innovadora e importante del último tiempo, ha sido el desarrollo de la tecnología bluetooth. Esta se basa en el uso de radiofrecuencias como principal medio de comunicación, lo cual la hace más robusta, eficiente, rápida, además de tener un mayor radio de funcionamiento en comparación con sus competidores directos, como el infrarrojo.

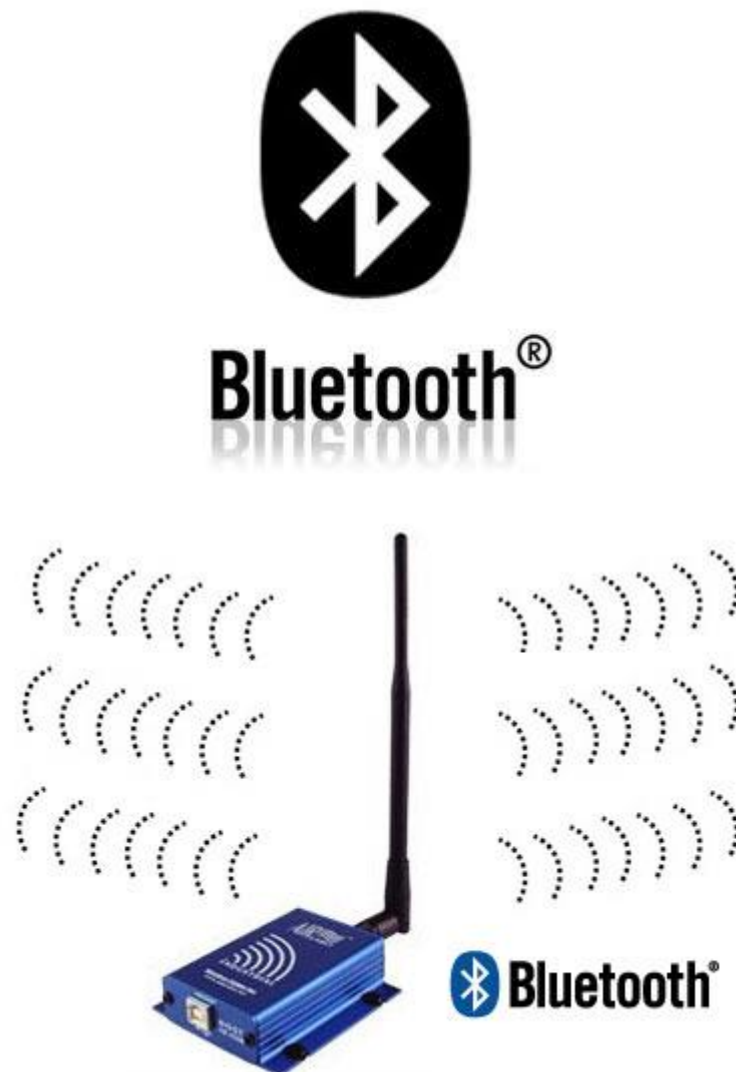


Figura 1: Dispositivos bluetooth.

Como nace Bluetooth.

Las ideas iniciales que llevaron al posterior desarrollo del dispositivo bluetooth fueron la idea de crear un medio de comunicación que permitiese conectar dos dispositivos electrónicos de manera inalámbrica y que además tuviera características tales como practicidad, comodidad, portabilidad, usabilidad, económico y que además consumiera poca potencia. Esto llevo a una serie de empresas a trabajar en diversos proyectos que finalmente decantaron en el desarrollo de esta tecnología.



Figura 2: Portabilidad de dispositivos Bluetooth.

Como Trabaja Bluetooth.

Bluetooth trabaja en base de radio frecuencias. La idea de usar esta tecnología radica en que utilizando este medio de comunicación se logra eliminar los cables y además es mucho más cómodo, rápido y eficiente en comparación con sus competidores directos como el infrarrojo. Además, las radio frecuencias son transmitidas por dispositivos que consumen una baja potencia lo cual lleva a que esta tenga un bajo costo de operación, así cumpliendo con los objetivos iniciales propuestos para esta tecnología. Por otro lado el hecho de trabajar con radiofrecuencia nos asegura (dependiendo de la frecuencia en la cual se trabaje y del medio por el cual se transmita) una mayor tolerancia a interferencia de obstáculos provenientes del medio por el cual se está transmitiendo.

Para poder trabajar con esta tecnología, primero que nada se necesita elegir una banda que fuese universal, para ello se necesitaba usar bandas que no fuesen licenciadas. Se utilizó la banda de 2.4 GHz hasta los 2.48GHz, implicando un ancho de banda de 79 [MHz]. Esta banda es universal y está plagada de interferencias. Por lo tanto para lograr trabajar en ella sin ser objeto de interferencias, se implementó el método de modulación de *Frequency Hopping* el cual nos asegura un trabajo sin interferencias de ruido.



Figura 3: Red entre dispositivos Bluetooth.

Conexiones Bluetooth

Bluetooth trabaja utilizando como base, un modelo jerárquico de 2 niveles: maestro y esclavo. Es bueno hacer notar que ningún dispositivo bluetooth posee un nivel predeterminado de jerarquía si no que es el dispositivo interesado en realizar la conexión con los demás quien asume el papel de maestro y los dispositivos que aceptan la conexión los que hacen el papel de esclavos. La idea base de bluetooth es que utilizando este sistema jerárquico, es posible formar redes de interconexión de dispositivos llamadas *piconet*, a través de las cuales es posible el traspaso de datos o información. Este sistema jerárquico que da vida finalmente a las *piconet*, posee ciertas restricciones las cuales veremos a continuación:

- Solo puede existir un maestro por piconet.
- Un esclavo no puede realizar una solicitud de conexión ni tampoco enviar datos, solo responder a una petición del maestro.
- El maestro es el que realiza la conexión.
- Una *piconet* puede contener hasta un máximo de 7 esclavos.
- El maestro define el canal a utilizar y la sub banda donde se sincronizaran posteriormente con los esclavos para el recibo de información. También es responsable de reconocer los sistemas bluetooth a su alrededor tanto como las restricciones de ellos, para luego poder usar os protocolos adecuados. También es responsable de administrar los canales entre los esclavos.

A continuación podemos ver más gráficamente a lo que nos referimos con un piconet:



Figura 4: una red de dispositivos a través de un maestro

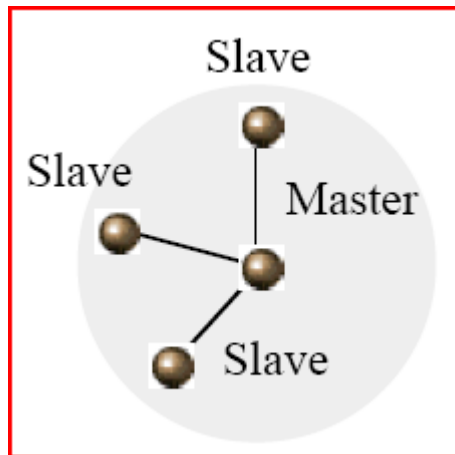


Figura 5: grafica de una piconet.

Cuando más de una *piconet* coexiste en el espacio y se interconectan entre sí, entonces forman una red de conexión mayor con propiedades distintas. Este tipo de conexión es posible cuando un maestro establece una *piconet* con un número finito de esclavos y aparece un segundo maestro solicitando una conexión con un esclavo ya presente en la piconet antes mencionada. Cuando este esclavo acepta la conexión del segundo maestro, entonces se genera esta red de *piconets* la cual se denomina *scatternet*. Una *scatternet* posee ventajas y desventajas en comparación con la piconet, entre las más relevantes se encuentran las siguientes:

- Puede poseer más de 1 maestro por red.
- Puede tener un sin número de dispositivos.
- En general posee un desempeño más pobre que la piconet.
- El desempeño va empeorando conforme aumentan los dispositivos asociados a la red.

Una muestra gráfica de una *scatternet* es lo que se ve a continuación:

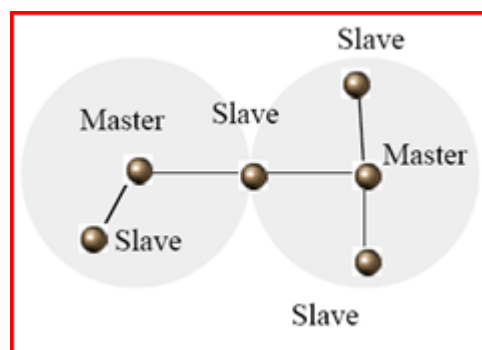


Figura 6: grafica de una scatternet



Figura 7: una *scatternet*

Características de un dispositivo bluetooth

Para que un dispositivo sea considerado un dispositivo bluetooth, debe contener al menos las siguientes características técnicas:

- Un número único llamado BD_ADDR o bluetooth device address de 48 bits el cual nos indica a que dispositivo corresponde, su función, marca, etc.
- Un reloj nativo interno de 28 bits el cual posee un periodo de 312.5 [µs]. Este reloj es el que permite que estos dispositivos envíen y reciban datos de manera sincronizada.

Pasos para crear una *Piconet*

Para lograr establecer una *piconet*, se siguen generalmente 2 procesos básicos:

- El primer paso, es detectar que efectivamente existe un dispositivo bluetooth en el área. Además se debe ver si efectivamente este está disponible para conexión. Esto se logra gracias a que el maestro envía en la banda de frecuencias utilizada, un paquete de datos de sincronización. Si algún dispositivo recibe este paquete de datos, le responde al maestro con su bluetooth device address o BD_ADDR (número único de dispositivo bluetooth) gracias al cual el maestro puede identificar el dispositivo y pasar al siguiente paso. A este proceso se le conoce como el proceso de Inquiry.
- El segundo paso a seguir, es el de invitar al dispositivo previamente detectado a formar parte de la red o piconet que el maestro está generando. Esto se logra enviándole a los esclavos detectados, el algoritmo o secuencia mediante el cual se irá eligiendo la banda de frecuencias a transmitir y además le envía la secuencia de su reloj interno. Gracias a estos 2 datos, los esclavos quedan totalmente sincronizados con el maestro y así pueden recibir los paquetes de datos de manera óptima. A este proceso se le conoce como el proceso de page.

Estructura de los paquetes bluetooth

Los paquetes bluetooth, poseen una estructura definida la cual se puede apreciar en la figura a continuación:

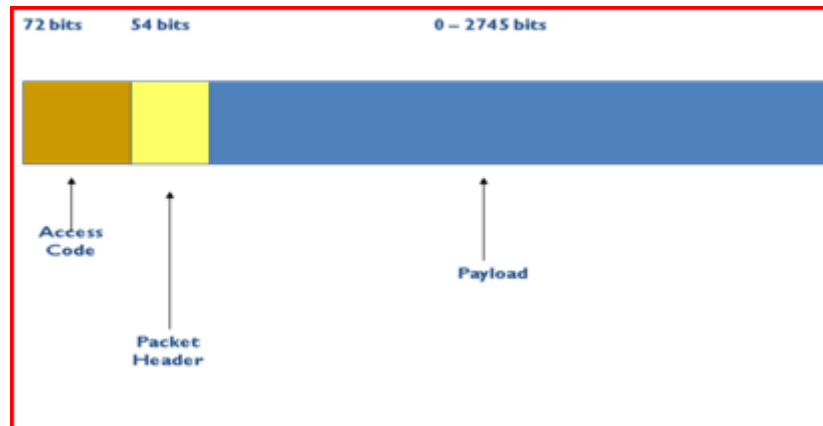


Figura 8: estructura de los paquetes bluetooth

En esta estructura vale la pena rescatar 3 partes principales:

- **Access Code:** Como lo indica su nombre, hace las veces de código de acceso. Si el paquete recibido no tiene el mismo Access Code, este es descartado.
- **Packet Header:** Contiene información relevante como el ACK, el número de secuencia, datos para verificar que el dato no está corrupto, datos para el control de flujo y la dirección que le asigna el maestro a cada esclavo.
- **Payload:** Acá es donde van los datos de información de los paquetes. Posee un largo variable dependiendo de los datos que quieran agregarse desde 0 a 2745 bits. Si es que se quiere mandar un archivo más grande que esto, el maestro puede elegir mandar datos en slots contiguos lo cual permite que luego se reciba el paquete correctamente.

Por lo tanto:

Se desarrollará e implementará un sistema de registro de posiciones para dispositivos móviles, usando el protocolo **bluetooth**. Los dispositivos móviles se encontrarán en un área abierta pero en un espacio delimitado.

La tecnología **bluetooth** es un protocolo que define un estándar global de comunicación inalámbrica, que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia en la banda de 2.4 GHz. Los principales objetivos que se pretende conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre nuestros equipos personales.

Cada dispositivo deberá tener su función encendida dentro de alguna red **Piconet**², para que cada lapso de tiempo se envíe un registro de posiciones por la red **WPAN**³ centralizada a través del PIC LV18FJ [2].

² **Piconet**: Red inalámbrica entre un dispositivo maestro y hasta 7 dispositivos esclavos utilizando el protocolo bluetooth.

³ **WPA** (Wireless Personal Area Networks, Red Inalámbrica de Área Personal o Red de área personal). Es una red de computadoras para la comunicación entre distintos dispositivos.

Justificación

En la actualidad el servicio de sistemas de localización **GPS**⁴ se proporciona de manera satelital mediante una conexión inalámbrica, ya sea **WAP**⁵ **SMS**⁶ o **WiFi**⁷. El uso de los servicios **GPS** que cuentan los dispositivos móviles en la actualidad genera un costo monetario. Dicho servicio se realiza a través de una aplicación que en la mayoría de los casos está desarrollada bajo una plataforma **J2ME** [7].

La idea principal de realizar un sistema de geolocalización por medio de **bluetooth** es satisfacer las necesidades de los usuarios usando bajos recursos.

Los sistemas de geolocalización o geo-posicionamiento ofrecen ventajas tanto a empresas como a usuarios. Éste tipo de sistemas son económicos y pueden acceder a él tanto empresas grandes como pequeñas. El beneficio para éstas últimas es aún mayor ya que se centra en comercios locales, ayudando a la continua lucha frente a grandes empresas. Por su parte, los usuarios pueden conocer las posibilidades que tienen a su alrededor evitando desplazamientos y grandes aglomeraciones. También es de gran beneficio conocer la ubicación de algún dispositivo por razones de seguridad o ayuda a la prevención o prueba de delitos.

El sistema de geolocalización para dispositivos móviles, en un ámbito profesional ofrecerá información sobre la ubicación de algún dispositivo en particular o de todos los dispositivos conectados en la red inalámbrica.

⁴ **GPS**: Sistema de Posicionamiento Global.

⁵ **WAP**: Wireless Application Protocol, Protocolo de Aplicaciones Inalámbricas.

⁶ **SMS**: Short Message Service, Sistema de Mensajes de Texto para Teléfonos Móviles.

⁷ **WiFi**: Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

Antecedentes

Existen ya diversos proyectos que implementan la comunicación **bluetooth** para diversas aplicaciones, incluso hay proyectos que realizan la localización de dispositivos móviles pero que no usan esta tecnología **bluetooth** para ese fin.

Entre estas referencias nosotros tomaremos como apoyo los proyectos que implementan una comunicación **WiFi** para la geolocalización, algunos proyectos utilizan dispositivos móviles para el uso de una aplicación en específica. Por nuestra parte integraremos estos conceptos para formar nuestro sistema de geo-posicionamiento usando el protocolo **bluetooth**.

Referencias Internas:

Transmisión y registro de las coordenadas geográficas de un dispositivo móvil [3]. Éste proyecto es muy similar al que se va a desarrollar, la diferencia es que el autor hace uso de la tecnología de comunicación **WiFi**, además de integrar la aplicación a los dispositivos móviles.

Por parte del proyecto actual, los autores usarán la tecnología **bluetooth** como plataforma de comunicación y sólo se abordará el tratamiento de los datos en el servidor.

Modelo de un sistema de información geográfica para la gestión catastral [4]. En este proyecto se implementó un sistema **GIS**⁸ usando comunicación **WiFi**, presenta ciertas semejanzas al proyecto que se va a desarrollar, nosotros tomaremos como apoyo el procesamiento o tratamiento de los datos.

Referencias externas:

Implementación del protocolo bluetooth para la conexión inalámbrica de dispositivos electrónicos programables [5]. En este trabajo de investigación se desarrolló la comunicación del protocolo **bluetooth** entre diversos dispositivos, tomaremos información para comprender la arquitectura del protocolo **bluetooth** y así formar nuestra red **scatternet**⁹.

Sistema de localización en interiores [6]. Este proyecto es mucho más parecido al que se quiere desarrollar, solo que el autor desarrolló la aplicación para la parte del cliente y está limitado a la localización en un lugar delimitado, por nuestra parte se desarrollará la aplicación en el servidor y tendremos una red más amplia ya que se implementará para espacios abiertos.

⁸ **GIS**: Aplicación para al tratamiento de información espacial.

⁹ **Scatternet**: Red que se forma de al menos una Piconet para obtener más dispositivos esclavos, utilizando varios puntos de acceso.

Descripción técnica

Para el desarrollo de este proyecto utilizaremos la metodología **RUP**¹⁰, cuya principal característica es iterar sobre las fases de diseño, implementación y pruebas.

Esta característica nos permitirá realizar prototipos ejecutables y detectar posibles inconsistencias entre los requerimientos y funcionalidades que se esperan obtener al finalizar el desarrollo de este proyecto.

Las partes que integran el desarrollo de este proyecto son:

- Implementar la comunicación **bluetooth** entre dispositivos móviles y un sistema digital basado en el micro controlador **PIC LV18FJ**.

Se armará el sistema digital integrado por el **PIC LV18FJ** y el módulo **bluetooth RN41** [8] y se programará a través del lenguaje **mikroC** [10] para que mediante probados puntos de acceso esclavos [9] se reciban las señales **bluetooth** emitidas por los dispositivos móviles que tengan encendido su radio en una área previamente establecida.

Las redes **bluetooth** de tipo **piconet** se encuentra limitadas por el número de dispositivos esclavos que pueden operar por lo que mejor se usará una red de tipo **scatternet** ya que contaremos con 2 antenas **bluetooth** que funcionarán como dispositivos maestros y esto nos dará mayor cobertura (figura 9).

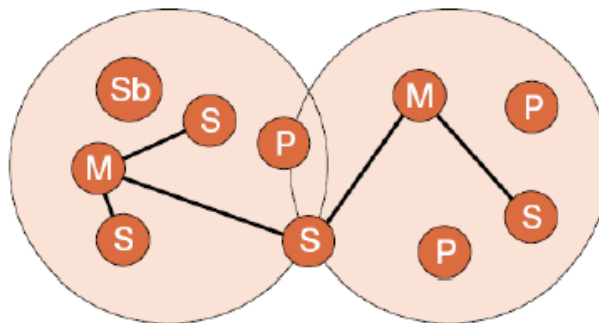


Figura 9. Red *scatternet* con varios dispositivos esclavos "S", maestros "M" y punto de acceso "P".

¹⁰ **RUP**: Rational Unified Process, Proceso Unificado de Rational. Divide el proceso de desarrollo en 4 etapas: Inicio, Elaboración, Construcción y Transmisión, cada una de ellas con fases de diseño implementación y pruebas.

- Diseñar un algoritmo para manipular la potencia de la señal entre puntos de acceso y dispositivos móviles.

Se diseñará desde cero un algoritmo, capaz de capturar señales de los puntos de acceso esclavo, ubicados dentro de un área que estará delimitada. Así mismo, este algoritmo permitirá seleccionar el nivel de potencia que será capaz de recibir por parte de los dispositivos móviles conectados al sistema.

Como parte importante del algoritmo, mantendremos comunicación constante entre los dispositivos móviles y los puntos de acceso, por lo que en el diseño del mismo se modelará un monitoreo dinámico capaz de detectar los cambios en las ubicaciones de los dispositivos y la detección de nuevos dispositivos que ingresen al sistema.

Para el desarrollo e implementación de este algoritmo, usaremos librerías propias del lenguaje **MikroC** y de los comandos de operación del módulo **RN41**.

- Diseñar un algoritmo para registro de información de dispositivos esclavos conectados al punto de acceso maestro.

Al tener comunicación estable entre los dispositivos móviles y el sistema, se diseñará un algoritmo que permita capturar y almacenar la posición de estos dispositivos, referenciados por los puntos de acceso maestro y esclavos que pertenecen al sistema.

Este registro se hará de forma dinámica y el administrador podrá manipular los tiempos de barrido para capturar la información.

Adicional a la administración de los tiempos de barrido, nuestro algoritmo podrá regular la potencia que es capaz de percibir de los dispositivos conectados. Así entonces el administrador tendrá la opción de establecer el área bajo la cual hay cobertura de nuestra red **scatternet**.

- Identificar cuándo un dispositivo se encuentre fuera de la área.

Se analizará la bitácora de los datos almacenados de un dispositivo móvil seleccionado para trazar los puntos por donde ha pasado este dispositivo, y así conocer su trayectoria y poder identificar en qué momento sale de la área de cobertura de la red **scatternet**.

También, el monitoreo será dinámico, para que el sistema detecte de forma automática cuándo un dispositivo deje el sistema o bien muestre alguna incongruencia a la hora de almacenar los datos en la bitácora.

Cada dispositivo usa un micro chip tranceiver que transmite y recibe en la frecuencia de 2.4 GHZ y tiene asignada una dirección única de 48 bits lo que indica que la conexión de los dispositivos son uno a uno.

La potencia de transmisión es de 1 mW para un enlace de 10 m y 100 mW para un enlace de hasta 100m.

Para tener una red **scatternet** debe haber un dispositivo maestro que ofrece la referencia de sincronización a partir del reloj interno.

- Crear reglas de cruce para el sistema digital.

Al implementar el sistema en un espacio abierto, los dispositivos móviles pueden desplazarse libremente dentro de un área delimitada dónde está disponible el sistema e incluso, fuera de él.

Estos traslados provocan incongruencias cada vez que se registra información de los dispositivos en el sistema, por lo que es necesario que el sistema advierta cuándo algún dispositivo pasó del punto A al punto C sin hacerlo por el punto B (figura 10).

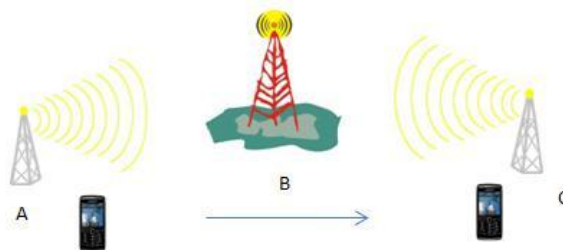


Figura 10. Puntos de acceso maestro y esclavos del sistema **Bluetooth**.

Así, es necesario establecer qué parámetros son correctos a la hora de registrar la información de los dispositivos móviles conectados al sistema.

- Diseñar un algoritmo para monitoreo y alerta que permita identificar cuándo algún dispositivo viole una regla de cruce.


Se diseñará un algoritmo que notifique al administrador del sistema cuando algún dispositivo móvil no respete las reglas de cruce que se establezcan en el punto anterior. Y podrá obtener la trayectoria de este dispositivo mediante el análisis de sus datos capturados por el sistema.

Manual de instalación de mikroC for PIC

Nosotros trabajaremos en un entorno Windows por lo que la instalación del software y el desarrollo del proyecto será a través de ventanas.

Dentro del Kit de desarrollo LV18FJ, encontramos el CD con el software para poder programar el **MCU** P18F87J60, al insertar el CD-ROM, en nuestra PC se desplegará un navegador web con el menú del disco.

Al navegar por el menú del disco, encontraremos la sección Compilers, y del cual seleccionaremos nuestro **IDE**, mikroC PRO for PIC, para nuestro hardware PIC Microcontrollers.



mikroElektronika
TOOLS | COMPILERS | BOOKS | MAGAZINE

Compilers | About us | Contact

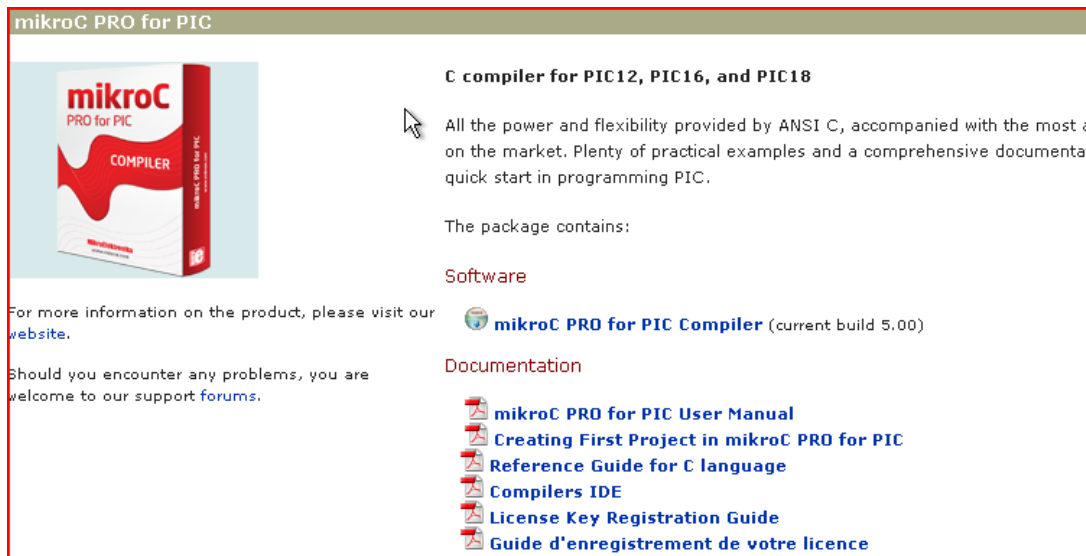
Compilers

PIC Microcontrollers mikroC PRO for PIC mikroBasic PRO for PIC mikroPascal PRO for PIC	dsPIC30/33 and PIC24 Microcontrollers mikroPascal PRO for dsPIC mikroBasic PRO for dsPIC mikroC PRO for dsPIC	PIC32 devices mikroC PRO for PIC32 mikroBasic PRO for PIC32 mikroPascal PRO for PIC32
8051 Microcontrollers mikroC PRO for 8051 mikroPascal PRO for 8051 mikroBasic PRO for 8051	AVR Microcontrollers mikroPascal PRO for AVR mikroBasic PRO for AVR mikroC PRO for AVR	Additional Software Visual GLCD Software Visual TFT Software GLCD Font Creator Software

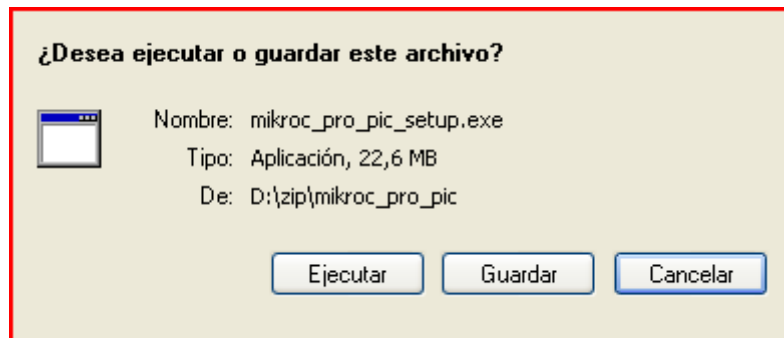
También podremos elegir algún otro IDE como mikroBasic o mikroPascal, pero para nuestra comodidad usaremos mikroC. También podremos instalar software adicional como los drivers para programar el GLCD que viene dentro del kit de desarrollo o para micro controladores 8051 o dsPIC30, etc.

Para nuestro proyecto será necesario instalar PIC32 drives donde se incluyen los drivers de comunicación USB para interconectar el kit de desarrollo y nuestra PC:

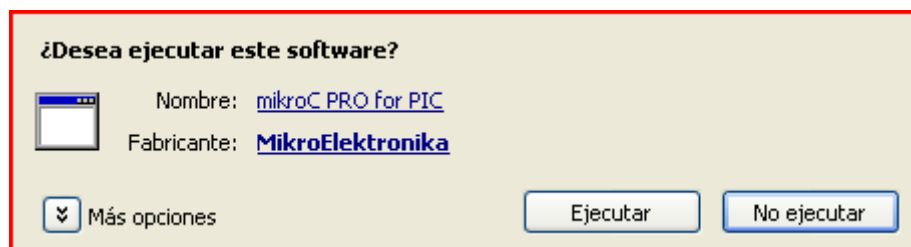
A continuación mostraremos el proceso de instalación de mikroC Pro for PIC.



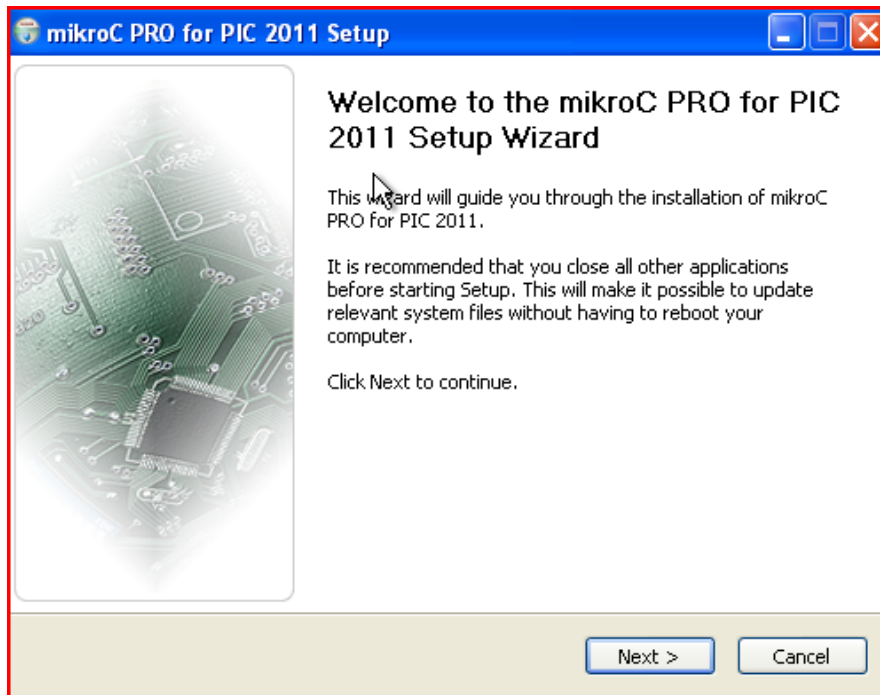
Al seleccionar mikroC for PIC en el menú Compilers del CD, nos mostrará la pantalla anterior, donde veremos el link para ejecutar el instalador y los manuales de operación de este IDE.



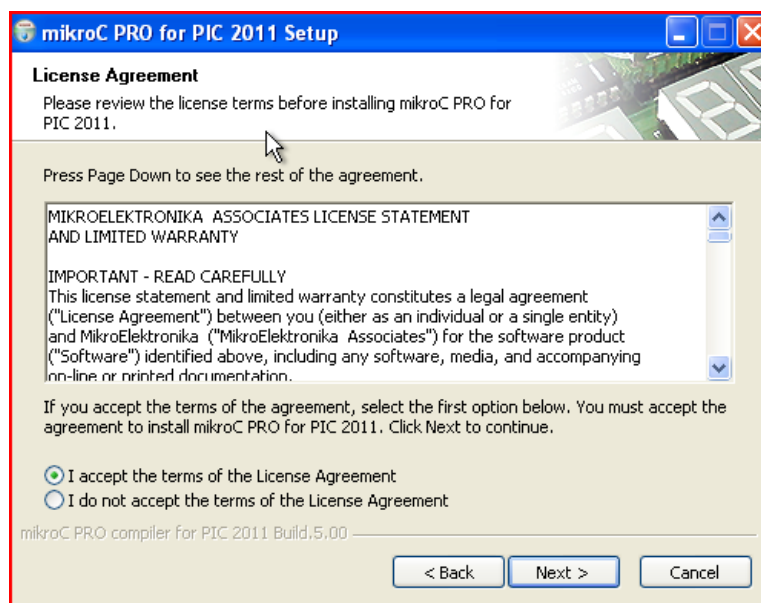
Al ejecutar el link del instalador nos mostrará la pantalla anterior que nos indica si queremos ejecutar la aplicación.

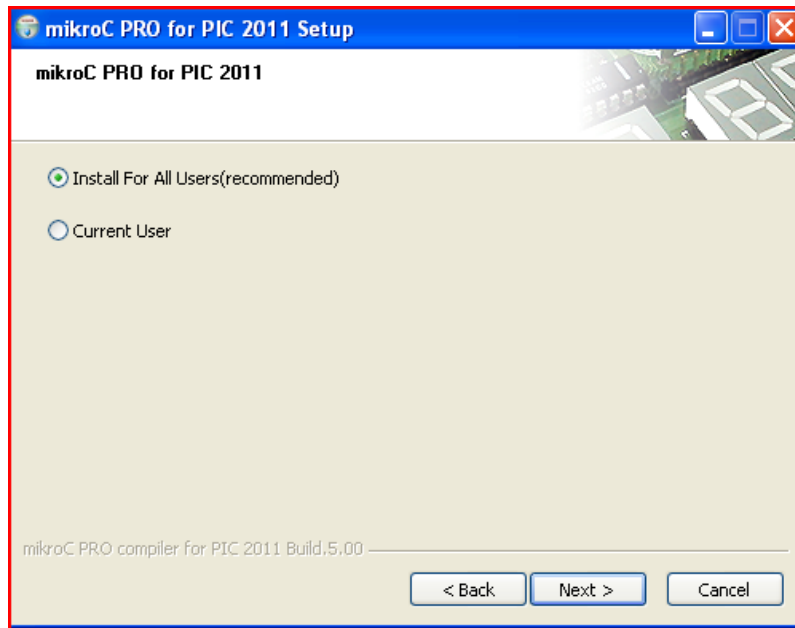


Al seleccionar Ejecutar ahora nos mostrará esta pantalla, donde es importante desplegar la pestaña Más opciones y elegir la opción Siempre instalar software de mikroelektronika, esto permitirá que durante la instalación se instalen todas la librerías que pertenecen al PIC

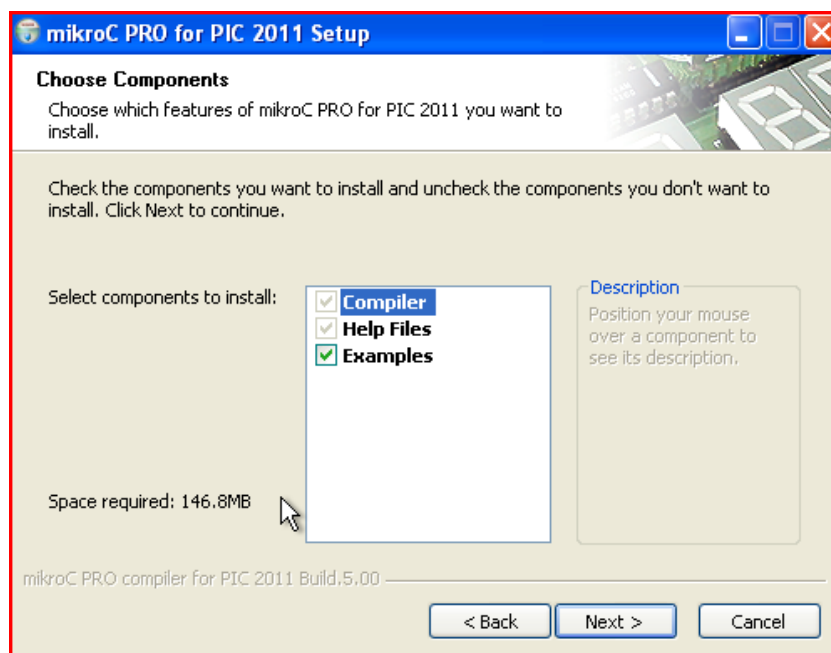


Después del paso anterior, entraremos al instalador como tal, de nuestro IDE, basta con dar click al botón Next para empezar al instalación.

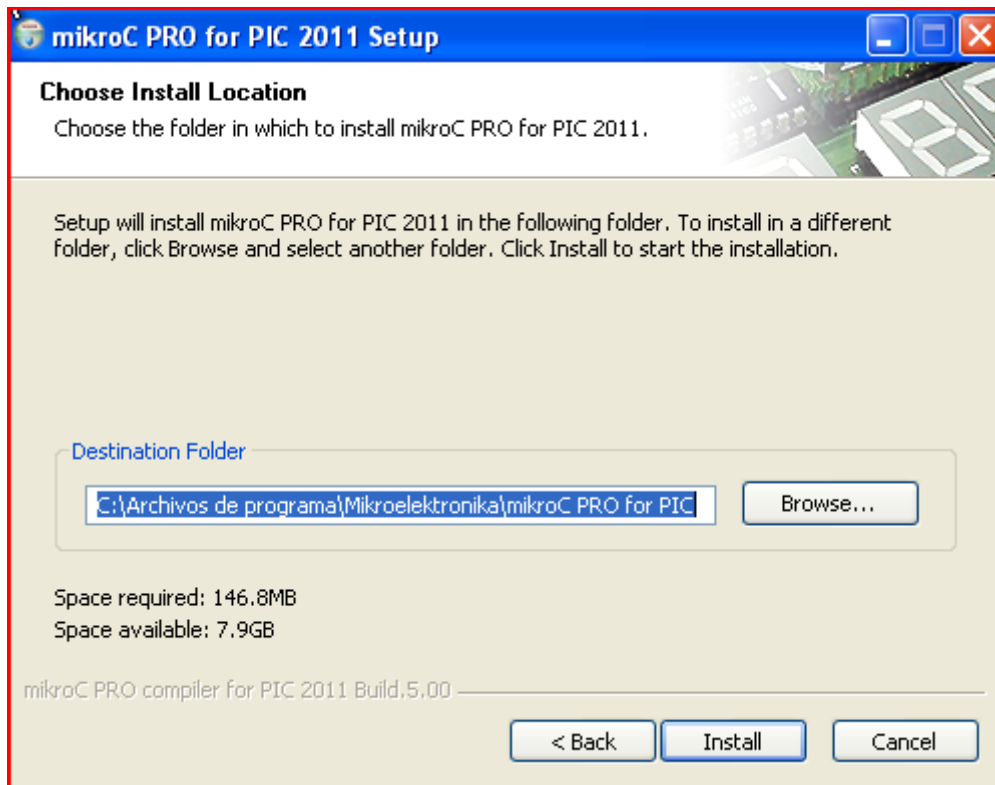




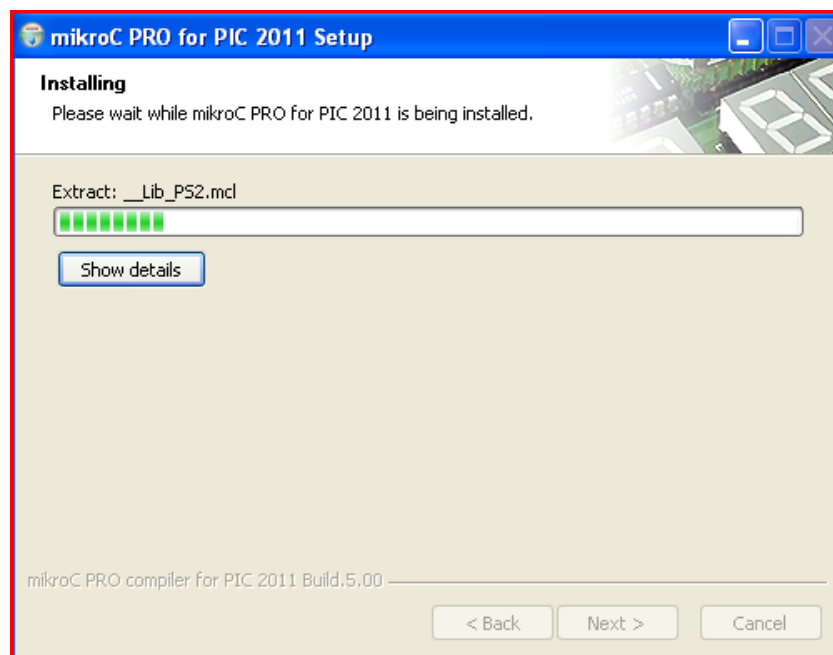
Nosotros dejaremos las opciones que marca por default, solo es importante seleccionar que aceptamos los términos de licencia y el software estará disponible para todos los usuarios de la PC.

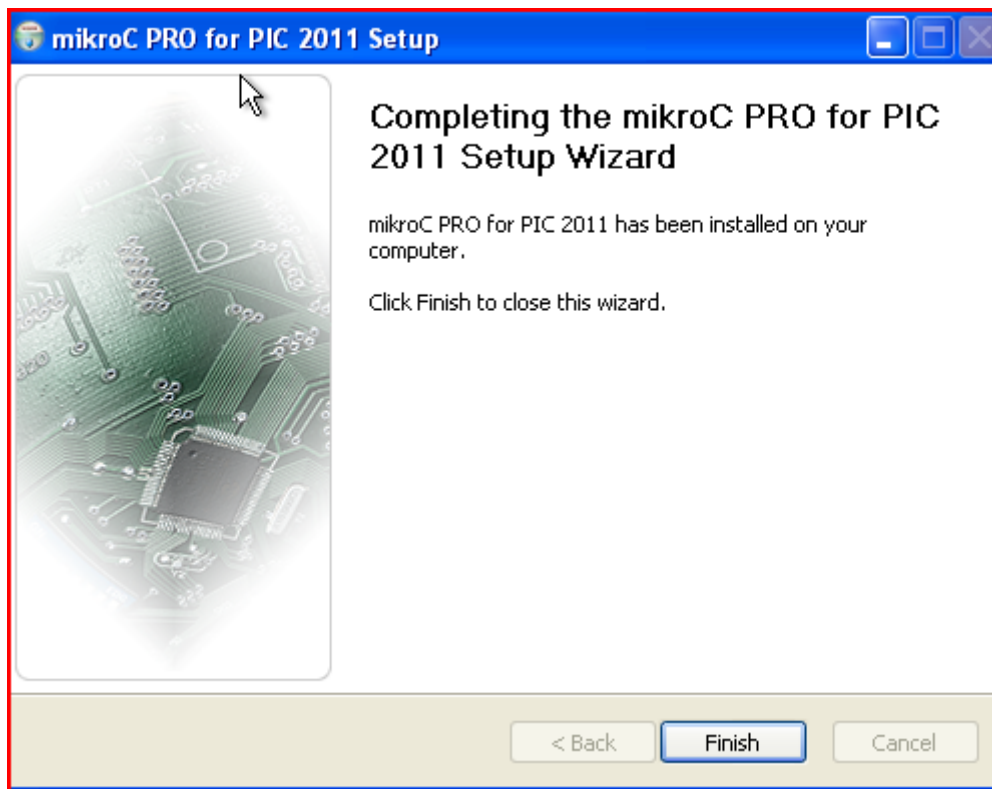


Es importante que en la pantalla anterior notemos que estén seleccionados las opciones de Compiler y HelpFile, la opción de Examples se puede descartar ya que posterior a la instalación del IDE, podremos instalar los códigos de ejemplos para el kit de desarrollo.



Durante este proceso de instalación, veremos en la pantalla anterior la ruta donde se instalará el software y si en el punto anterior seleccionamos que se instalarán los códigos de ejemplo, estos se instalarán en esta misma ruta para su posterior consulta.

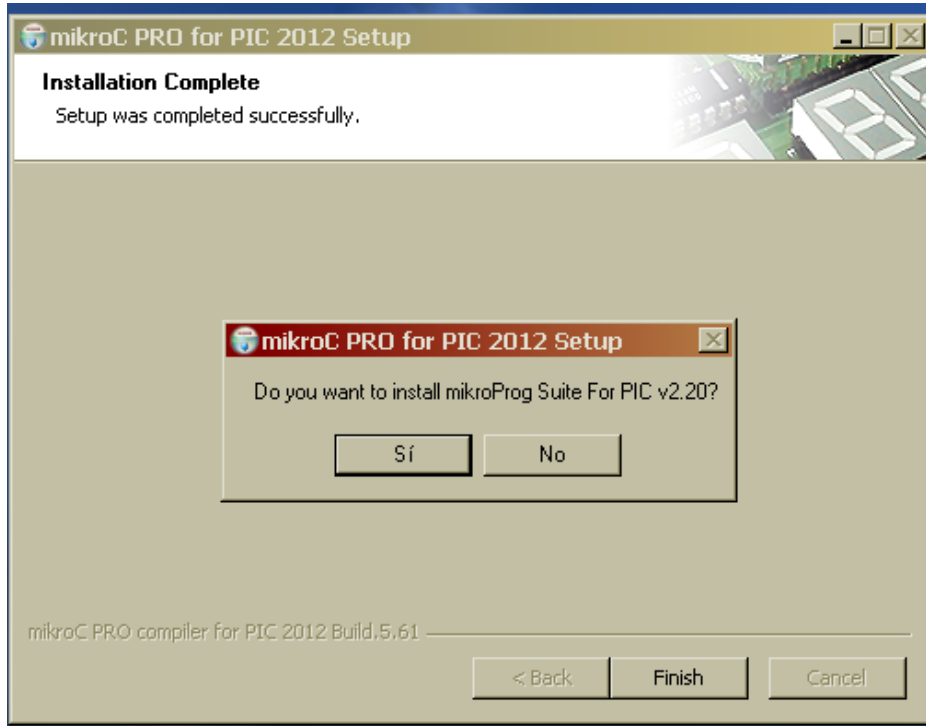




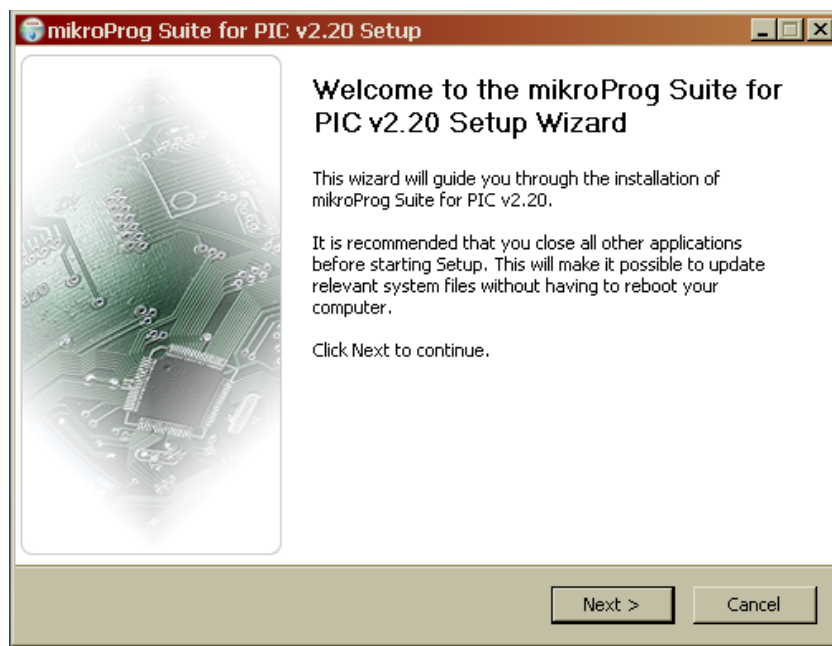
Para finalizar el proceso de instalación, basta con esperar a que se llene la barra de progreso y dar click al botón Finish de la pantalla anterior.

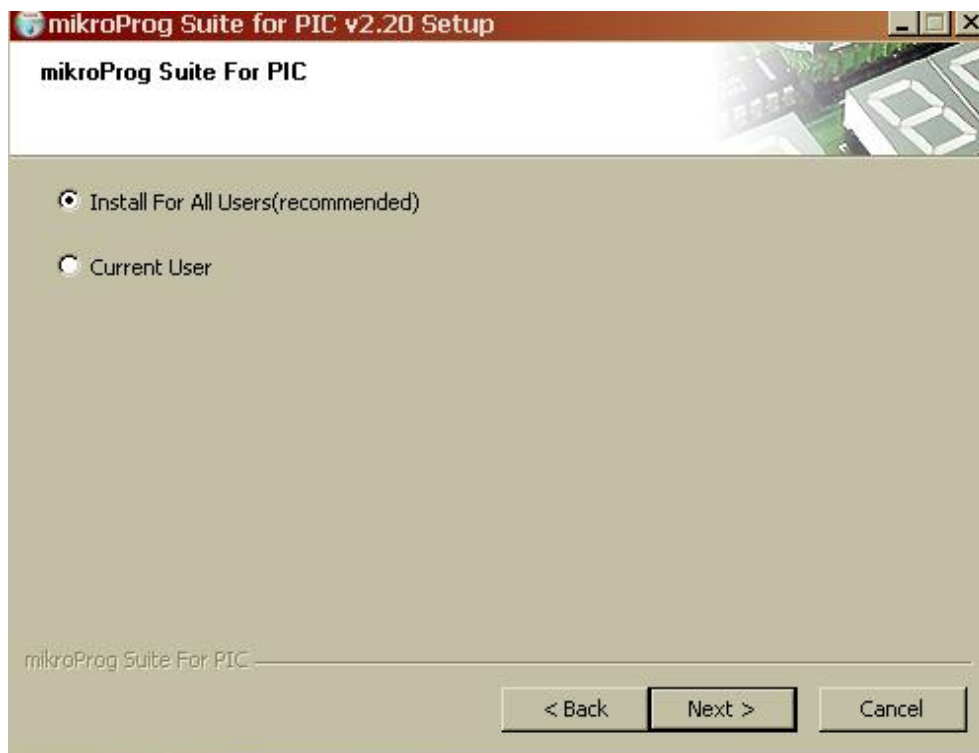
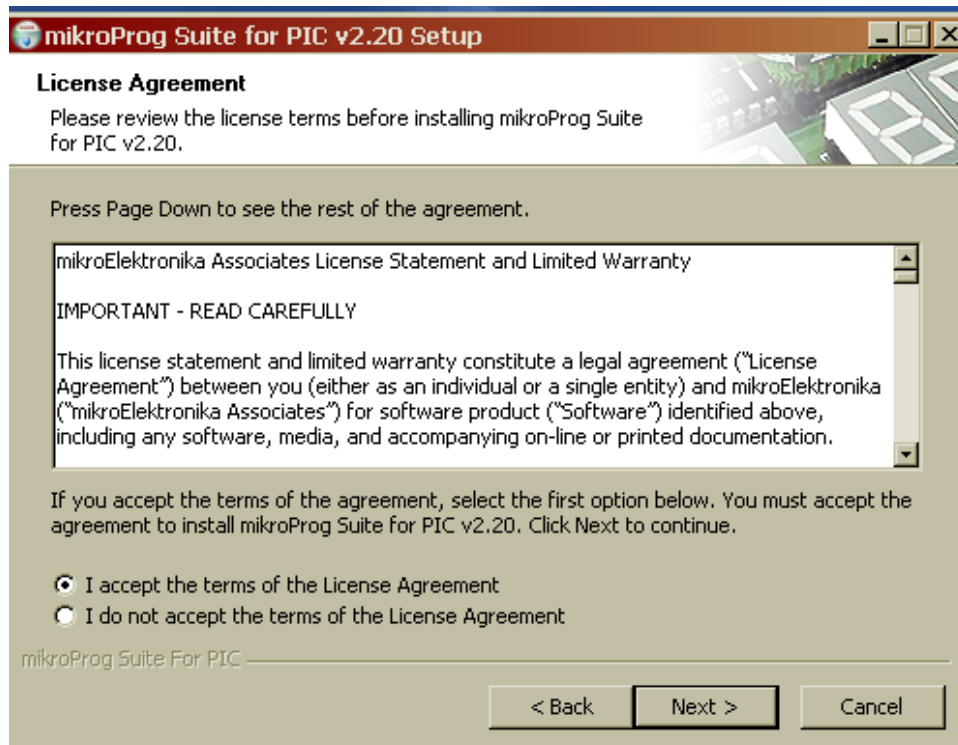
Instalación del software mikroProg Suite for PIC

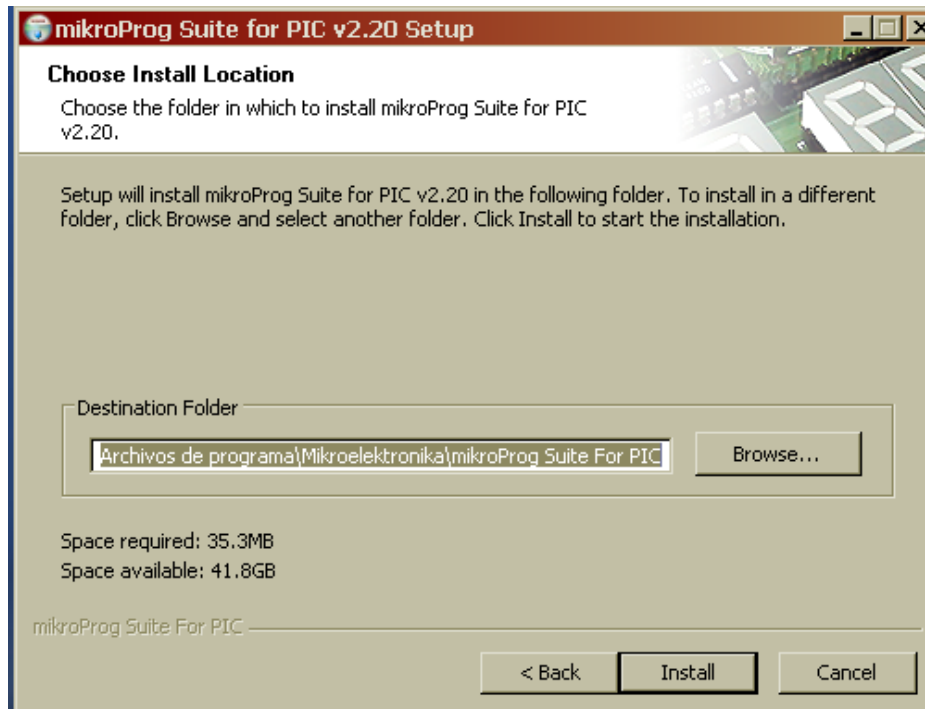
Después de instalar nuestro IDE, de manera automática, nos preguntará si queremos instalar el software para poder manipular nuestro PIC LV18FJ, por lo elegiremos la opción Si en la siguiente pantalla:



Al igual que en la instalación del IDE, dejaremos las opciones por default, por lo que solo será necesario dar clic al botón *Next* en todas las pantallas que nos vayan apareciendo







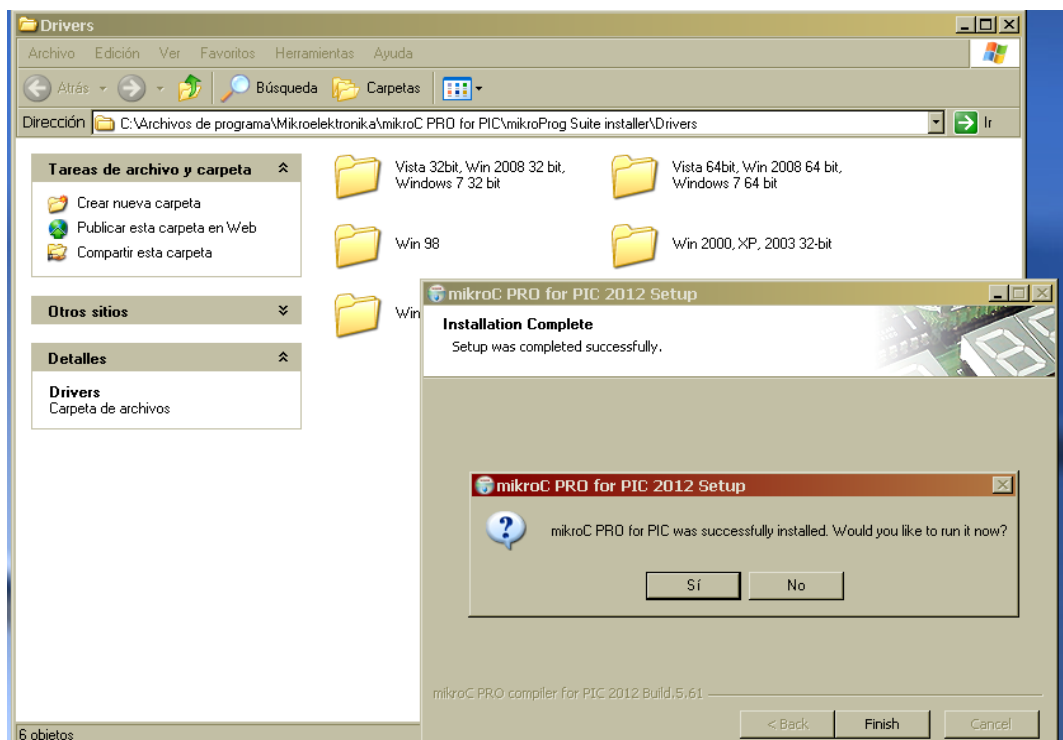
Para finalizar la instalación daremos clic al botón *Finish* de la pantalla anterior, este software nos permitirá programar el MCU P18F87J60 que se encuentra en el kit de desarrollo LV18FJ

Instalación de los drivers para el PIC LV18FJ

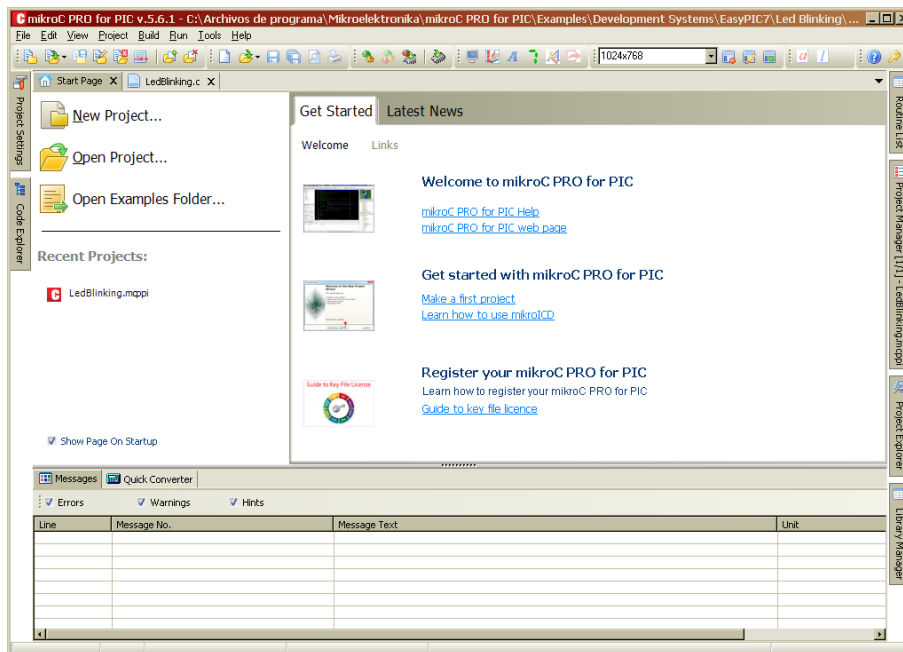
El siguiente software instalaremos serán los drivers USB para comunicar nuestro programa hecho en mikroC con el PIC LV18FJ



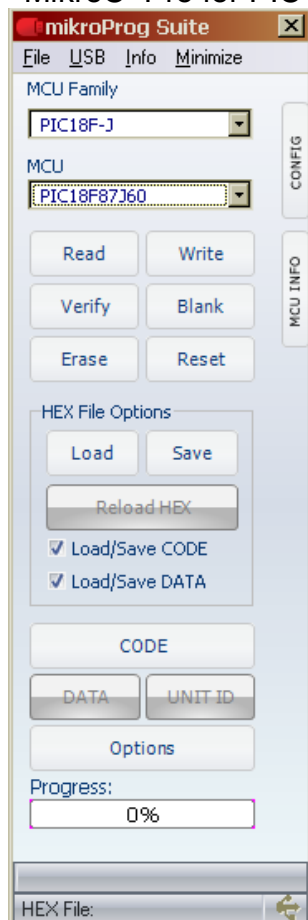
Al seleccionar la opción Si en la pantalla anterior, nos abrirá una nueva ventana donde es importante recordar la dirección donde se almacenan los drivers para las diferentes versiones de sistema operativo.



Por ultimo al presionar el botón Si acabará el proceso de instalación y tendremos listos nuestro centro de desarrollo:



MikroC Pro for PIC



MikroProg Suite para el MCU P18F87J60

Especificaciones técnicas

El proyecto será desarrollado bajo el entorno propio del **PIC LV18FJ** por lo que será necesario trabajar con plataformas y herramientas de desarrollo ya antes mencionadas.

Para realizar la comunicación entre los dispositivos móviles en una red **scatternet** es necesario primero formar la red **piconet**. Al unir el **PIC LV18FJ** con el módulo **RN41** se tendrá la red **WPAN** la cual podrá tener una cobertura de un área abierta pero delimitada entre los dispositivos móviles donde la conexión entre las antenas receptoras incrementará el rango de cobertura de nuestra red.

Para manipular el sistema digital formado, se usará el puerto en serie del **PIC** conectando a nuestra **PC**⁹ que fungirá como servidor central. Una vez que se tenga todo conectado, se usará una consola **telnet**¹⁰ para poder manipular el módulo **bluetooth RN41**.

Será necesario configurar el módulo **RN41** que funcionará en modo maestro, esto se realizará bajo línea de comandos descritos en el módulo **bluetooth RN41** [8].

Una vez que se tenga la red **scatternet** con base en la red **WPAN** y se establezca comunicación entre el servidor, se realizará un testeo de los dispositivos móviles detectados dentro de la red.

Con los datos adquiridos por acción de testear, se modelará una base de datos para ser procesados. Con la ayuda del diagrama entidad relación del sistema y se modelará e implementará en **MySQL**⁸ para obtener información relevante por dichos informes de concurrencia, puntos de acceso, límites de alcance y monitoreo constante de localización.

Dentro de las pruebas a realizar para la continua conexión entre los dispositivos en la red **scatternet** con base en la red **WPAN**, los móviles estarán constantemente emitiendo señal de encendido o activo. Estas señales emitidas por los dispositivos serán canalizadas en el punto de acceso el cual tendrá conexión a un servidor.

Este proceso será constante durante un tiempo determinado en la cual el administrador podrá controlar manualmente dicho tiempo ya que podrá ser en lapsos de 10, 15, 20 ó 30 minutos.

⁸ **MySQL**: Es un manejador de base de datos que soporta el lenguaje SQL y la conexión de varios usuarios cuya base de datos es gratuita.

⁹ **PC**: Computadora personal.

¹⁰ **Telnet**: protocolo de red a otra máquina para manejarla remotamente.

Como primer análisis, el registro de información contendrá un identificador y nombre del dispositivo móvil así como su punto de acceso más cercano al cual se conectó, usando como parámetro de referencia la señal **bluetooth** que fue percibida. Más adelante y usando el testeado ya antes descrito, definiremos que otros datos serán necesarios almacenar en la base de datos.

Para obtener más información de los dispositivos se pueden usar ciertos comandos del módulo **bluetooth RN41** descritos en su manual de operación.

Así mismo, la aplicación desplegará una lista de los dispositivos conectados. El administrador podrá seleccionar alguno de éstos y se filtrará cada uno de los registros que se tengan de ese dispositivo a manera de bitácora.

Mediante una gráfica tiempo contra ubicación, visualizaremos las localidades visitadas por los dispositivos móviles en nuestra área determinada. En este análisis gráfico, el administrador podrá detectar cuándo un dispositivo no se encuentra en función o fuera de rango de la red **scatternet**.

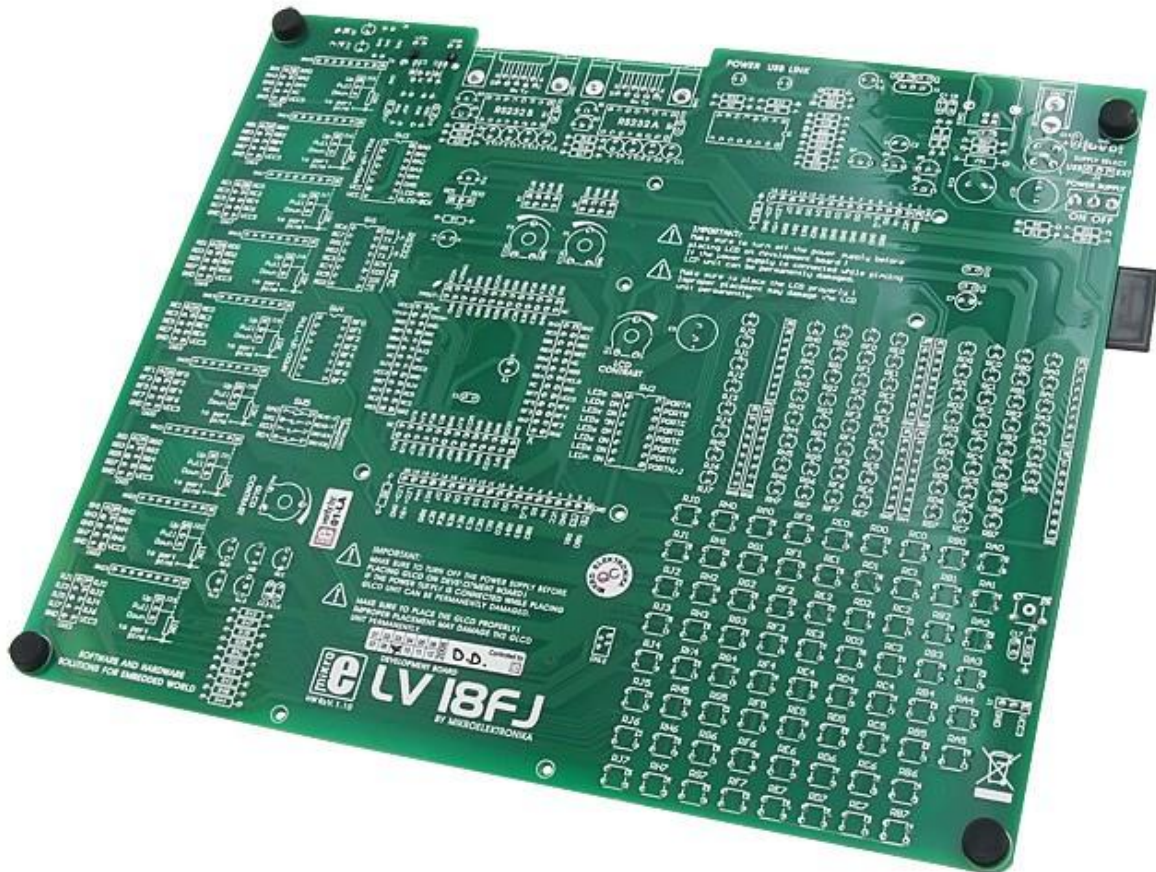
Para evitar la redundancia en datos capturados por el sistema, se realizará un ambiente de prueba de cruce entre los dispositivos móviles. Planteando todos los posibles escenarios de concurrencia, trayectoria, empalme de dispositivo o fallo de hardware. Con las conclusiones obtenidas gracias al análisis se crearán reglas de cruce que se usaremos para optimizar el sistema.

Para dar por terminado este proyecto se realizará un informe clasificado de los puntos de geolocalización de los dispositivos móviles en un tiempo real.

Se entregará el reporte final en PDF junto con un CD que contendrá el código fuente de la aplicación.

Kit de Desarrollo LV18FJ

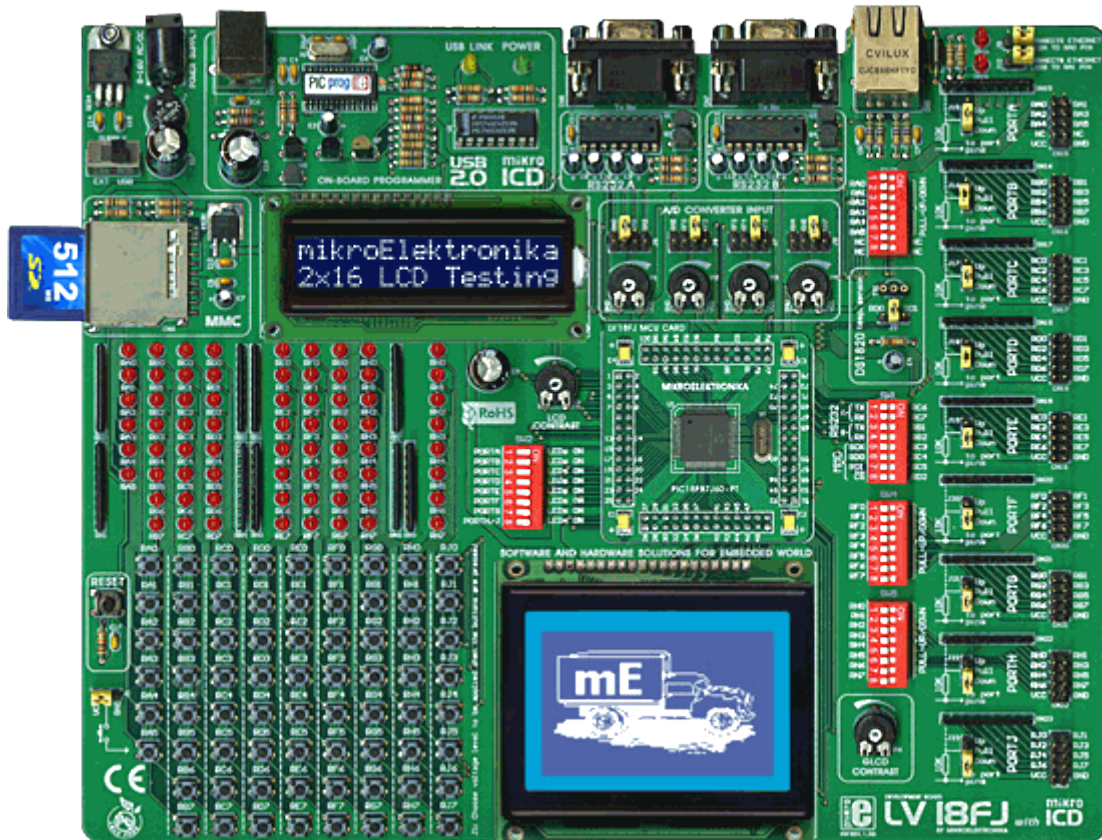
El Kit de LV18FJ se concibe para el desarrollo en microcontroladores PIC18FxxJxx dotado de Ethernet 10 Mbps. El kit de Desarrollo LV18FJ viene con numerosos periféricos, destacándose el programador con conexión de USB que le permitirá cargar sus programas desarrollados en Assembler o con cualquier tipo de compilador para PICs en un microcontrolador virgen PIC











El PIC suministrado con la placa es PIC18F87J60 con 80 pines, qué es pre-soldado encima de un soporte para un manejo fácil. Este PIC sin BOOT Loader es programable con el programador USB integrado en la placa.










La placa también tiene un de bugger (depurador) ICD on board permitiendo (si usted desarrollara sus aplicaciones con la ayuda de los compiladores *MikroBASIC*, *MikroC* o *MikroPASCAL*) para verificar en la pantalla de su ordenadores valores de sus variables, valores de los registros especiales (SFRs) o el estado de la memoria de *EEPROM* durante la ejecución de su programa, para hacerle beneficiar de una herramienta apropiada que es sumamente poderosa y eficaz.



Componentes:

<p>➤ Doble Puerto RS232.</p>	
<p>➤ Dip-switches permiten configurar parámetros.</p>	
<p>➤ Conexión de Tarjeta SD™.</p>	
<p>➤ PIC incluso en soporte.</p>	
<p>➤ Potenciómetros sobre 4 entradas A/D.</p>	
<p>➤ Alimentación a través de USB o externa.</p>	
<p>➤ Salidas de E/S a través de IDC.</p>	
<p>➤ 70 Leds integrados en la placa</p>	

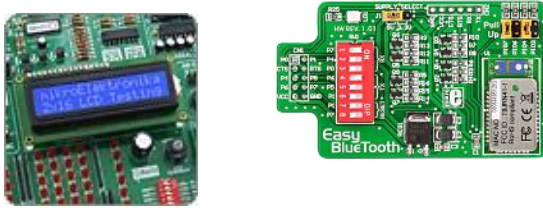
<ul style="list-style-type: none"> ➤ Resistencias de Salida "Alto/Bajo". 	
<ul style="list-style-type: none"> ➤ 70 botones de presiones integradas en la placa. 	
<ul style="list-style-type: none"> ➤ Programador USB integrado. 	
<ul style="list-style-type: none"> ➤ Modo Debug integrado. 	
<ul style="list-style-type: none"> ➤ Alimentación AC o DC. 	
<ul style="list-style-type: none"> ➤ Conexión para DS18S20. 	
<ul style="list-style-type: none"> ➤ Potenciómetro de regulación de contraste. 	

<p>➤ Seri gráfica con indicaciones.</p>	
<p>➤ Selección de niveles de los botones.</p>	
<p>➤ LCD Gráfico.</p>	
<p>➤ Botón de RESET.</p>	

Desarrollo

Después de instalar el software procedemos a ensamblar el hardware, es decir, el kit de desarrollo LV18FJ.

Nosotros solo usaremos algunos componentes, estos son graphic LCD 128x64, modulo bluetooth RN-41 ubicado en el Easy bluetooth board y usaremos el USB Dongle para compilar nuestro códigos sin problemas.



El graphic LCD no es muy importante, así que bastará con ensamblar, el modulo bluetooth a nuestro kit de desarrollo LV18FJ, para ello colocaremos el Easy bluetooth board en el puerto E/S C, y activaremos solo los pines Rx y TX.



Después de ensamblar el modulo bluetooth, y veamos que sea reconocido en nuestra computadora, procedemos a codificar nuestro programa de operación.

Primero empezaremos con el código para entablar comunicación bluetooth entre el modulo RN-41 del kit de desarrollo y algún dispositivo móvil que tenga función de bluetooth.

Implementación

La programación del micro controlador PIC18FJ60 es a través de código hexadecimal, para facilitarnos la creación de este código, usamos la herramienta mikroC para pic, la cual nos permite con una interfaz sencilla, alterar el comportamiento del micro controlador de acuerdo a las intrusiones que nosotros definamos.

Dentro de los componentes del micro controlador tenemos un puerto serial en el cual podemos conectar dispositivos externos que aumentan potencialmente el uso del micro controlador.

En nuestro proyecto usaremos el modulo bluetooth RN41 y para poder enviarles instrucciones de operación, es necesario usar el puerto serial del micro controlador, al usar este puerto es necesario definir algunos parámetros ya que nuestro equipo de cómputo lo renacerá como un dispositivo conector a un puerto COM.

Los parámetros que usaremos serán: la velocidad de comunicación será de 115200 baudios, el tamaño del string que se envía y recibe es de 8 bits, de los cuales se usara 1 para definir el límite del string y no usaremos ninguno para definir la paridad.

Cabe resaltar que la definición de estos parámetros se debe hacer al configurar el puerto COM de la computadora y únicamente especificamos la velocidad de transmisión dentro del código de operación del micro controlador.

También es importante mencionar que de acuerdo a las especificaciones del micro controlador, la forma de programar el modulo bluetooth RN41 es necesario usar las instrucciones USART, y al ser un dispositivo extremo a nuestra kit de desarrollo es necesario usar interrupciones para el acceso a él.

Ahora bien, tenemos dos tipos de comandos que le enviaremos al módulo bluetooth RN41, primero tenemos comando SET los cuales definirán su comportamiento y segundo los comandos que permitirán modificar este comportamiento en tiempo real.

Los comando SET que usaremos son el de definir el nombre del dispositivo, si existirá autenticación entre los dispositivos bluetooth y el módulo RN41, su modo de operación ya sea maestro o esclavo y los comando que nos permitirán escanear y recuperar información de los dispositivos bluetooth que se encuentren dentro del rango de cobertura del módulo bluetooth RN41.

Después de haber entablado la comunicación estable entre el modulo bluetooth RN41 y los dispositivos con su radio bluetooth encendida, modificaremos el código de funcionamiento del RN41 para que realice las acciones antes descritas en los objetivos.

Primero modificaremos el código para que se pueda variar la potencia de la señal emitida por el modulo bluetooth, para realizar esto usaremos el comando SET sy, hex donde hex es un código en hexadecimal que nos indica la cantidad de decibelios que usará el dispositivo bluetooth.

La cantidad de decibelios nos permitirá indicarle al módulo bluetooth su rango de cobertura permitiendo abarcar 1, 10 y 100 metros dentro campo abierto, pero habrá que contemplar que los muros y otras radiofrecuencias harán que el rango de cobertura baje alrededor de un 20%.

Al usar un comando SET, es necesario que después de modificar el código, reiniciemos el módulo RN41, esto lo haremos a través del comando R, 1 que hará que se reinicie nuestro modulo bluetooth sin necesidad de apagar y encender la tarjeta de desarrollo.

El código es el siguiente:

```
// Respuestas tras ejecutar un comando
const BT_CMD = 1;
const BT_AOK = 2;
const BT_CONN = 3;
const BT_Done = 4;

// Variables para el control de estado de la maquina
char BT_state = 0;
char response_rcvd = 0;
char responseID = 0, response = 0;

// Manejo de interrupciones
void interrupt(){
char tmp;
if (PIR1.RCIF == 1) {
tmp = UART1_Read(); //Aqui se recibe el byte
//Proceso del estado de la maquina
switch (BT_state) {
case 0: {
response = 0;
if (tmp == 'C')
BT_state = 1;
if (tmp == 'A')
BT_state = 11;
if (tmp == 'D')
BT_state = 31;
break;
}

case 1: {
if (tmp == 'M')
BT_state = 2;
else if (tmp == 'O')
BT_state = 22;
else
BT_state = 0;
break;
}

case 2: {
if (tmp == 'D') {
response = BT_CMD;
BT_state = 40;
}
}
}
}
```

```
    else
    BT_state = 0;
    break;
}
```

```
case 11: {
    if (tmp == 'O')
        BT_state = 12;
    else
        BT_state = 0;
    break;
}
```

```
case 12: {
    if (tmp == 'K'){
        response = BT_AOK;
        BT_state = 40;
    }else
        BT_state = 0;
    break;
}
```

```
case 22: {
    if (tmp == 'N')
        BT_state = 23;
    else
        BT_state = 0;
    break;
}
```

```
case 23: {
    if (tmp == 'N') {
        response = BT_CONN;
        response_rcvd = 1;
        responseID = response;
    }
    BT_state = 0;
    break;
}
```

```
case 31: {
    if (tmp == 'o')
        BT_state = 32;
    else
        BT_state = 0;
}
```

```

        break;
    }

    case 32: {
        if (tmp == 'n')
            BT_state = 33;
        else
            BT_state = 0;
        break;
    }

    case 33: {
        if (tmp == 'e') {
            response = BT_Done;    // "Inquiry Done"
            response_rcvd = 1;
            responseID = response;
        }
        BT_state = 0;
        break;
    }

    case 40: {
        if (tmp == 13)
            BT_state = 41;
        else
            BT_state = 0;
        break;
    }

    case 41: {
        if (tmp == 10){
            response_rcvd = 1;
            responseID = response;
        }
        BT_state = 0;
        break;
    }

    default: {
        BT_state = 0;
        break;
    }
}
}
}

```



```

//Obtener la respuesta del módulo bluetooth
char BT_Get_Response() {
    if (response_rcvd) {
        response_rcvd = 0;
        return responseID;
    }
    else
        return 0;
}

//Programa principal
void main() {
    int i;
    ADCON1 |= 0xFF; // Configuro pins como digitales
    CMCON |= 7; // Deshabilitar comparadores

    TRISB = 0; //El pto B es salida
    //Activo los pines del puerto C donde esta el RN41 para usar las interrupciones RX y TX
    TRISC3_bit = 0;
    RC3_bit = 0;

    //Habilito la interrupcion Rx
    PIE1.RCIE = 1;
    INTCON.PEIE = 1;
    INTCON.GIE = 1;

    Delay_ms(500);
    UART1_init(9600); //Inicializo modulos UART1 a 115200 bps
    Delay_ms(1000);

    PORTB=0x00;
    do {
        for(i=0; i<3; i++) { //envía "$$$"
            UART1_Write(0x24);
        }
        Delay_ms(500);
    } while (BT_Get_Response() != BT_CMD);

    do {
        UART1_Write_Text("SN,Bluetooth-UAM");
        UART1_Write(13);
        UART1_Write(10);
        Delay_ms(500);
    } while (BT_Get_Response() != BT_AOK);

    do {
        UART1_Write_Text("SO,Bluetooth UAM Azc");

```

```

    UART1_Write(13);
    UART1_Write(10);
    Delay_ms(500);
} while (BT_Get_Response() != BT_AOK);

do {
    UART1_Write_Text("SM,1");
    UART1_Write(13);
    UART1_Write(10);
    Delay_ms(500);
} while (BT_Get_Response() != BT_AOK);

do {
    UART1_Write_Text("SA,1");
    UART1_Write(13);
    UART1_Write(10);
    Delay_ms(500);
} while (BT_Get_Response() != BT_AOK);

do {
    UART1_Write_Text("SP,1111");
    UART1_Write(13);
    UART1_Write(10);
    Delay_ms(500);
} while (BT_Get_Response() != BT_AOK);

UART1_Write_Text("I,30");           //Mirar los dispositivos bluetooth
UART1_Write(13);
UART1_Write(10);
while (BT_Get_Response() != BT_Done);

do {
    //UART1_Write_Text("SR,0017E879785A");           // Store the address just found
if there is one only device found
    UART1_Write_Text("SR,9C4A7B545645");           // Store the address just found if
there is one only device found
    // (if there is more than one device, you need to select one
by its exact address)
    UART1_Write(13);           // CR
    Delay_ms(500);
    portj.b7 = 1;
} while (BT_Get_Response() != BT_AOK);

if(UART1_Tx_Idle()==1){
    UART1_Write_Text("C");
    UART1_Write(13);           // CR
    while (BT_Get_Response() != BT_CONN);

```

```

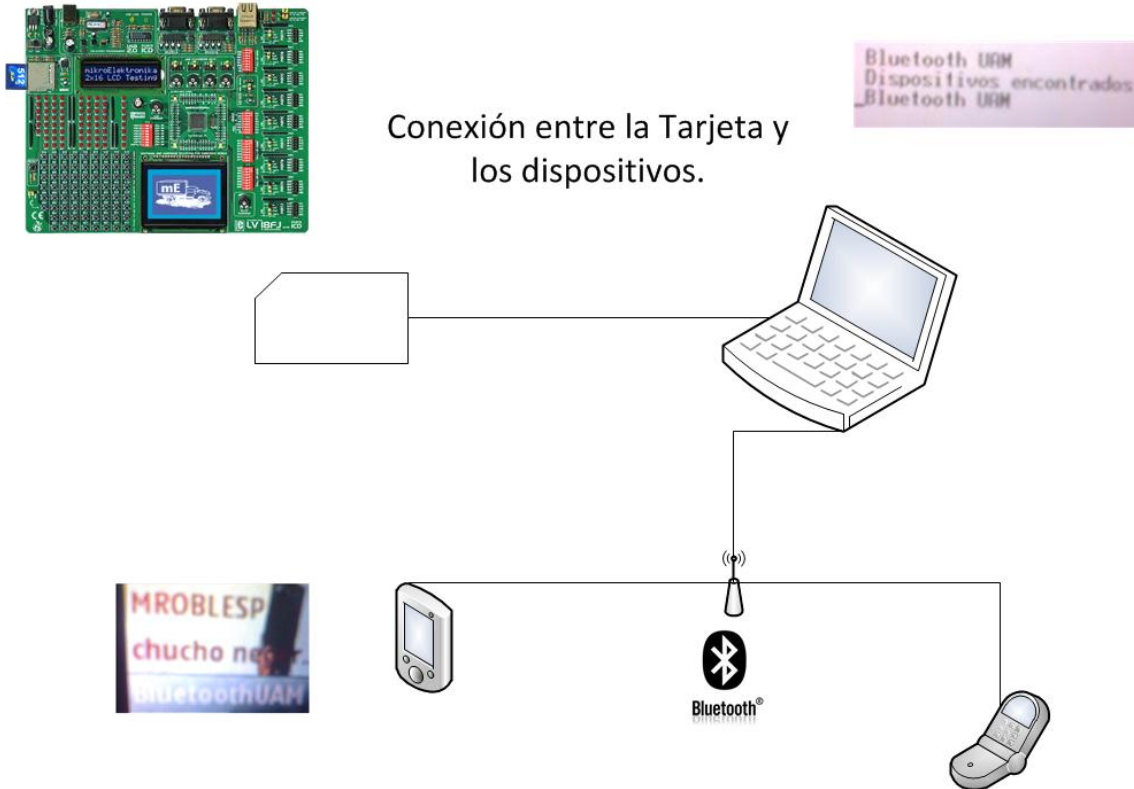
    portb.b3 = 0x01;
    Delay_ms(500);
}else{
    portb.b6 = 0x01;
}
    UART1_Write_Text("\n Dispositivo conectado!! ");
    UART1_Write(13);
    Delay_ms(1000);
}

```

Con el código anterior entablamos comunicación entre el modulo bluetooth RN-41 y un teléfono móvil.



Diagrama de Conexión entre los dispositivos



The screenshot shows two software windows. The left window is mikroC PRO for PIC v5.6.1, displaying a C program with UART-related code. The right window is mikroProg Suite, showing COM port settings and a terminal window with a message: 'Communication port 'COM5' could not be opened. Connected to COM5'. The terminal also shows a list of pins and their status (Connected, RI, TxD, etc.).

```

//elija con que potencia quiere
do {
    UART1_Write_Text("SY,0001");
    UART1_Write(13);
    UART1_Write(10);
    Delay_ms(300);
} while (BT_Get_Response() != BT

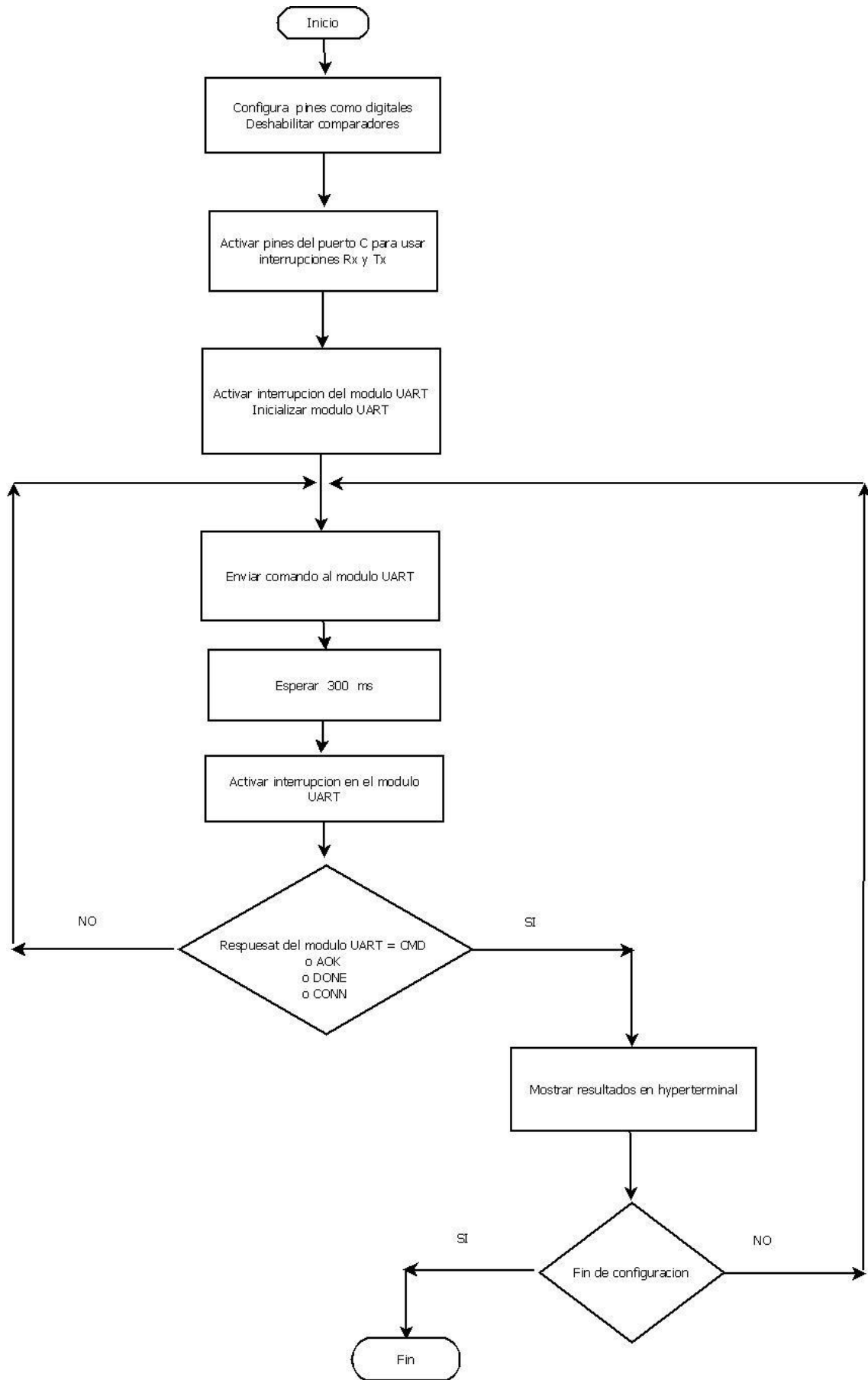
do {
    UART1_Write_Text("I,30");
    UART1_Write(13);
    UART1_Write(10);
    Delay_ms(300);
} while (BT_Get_Response() != BT

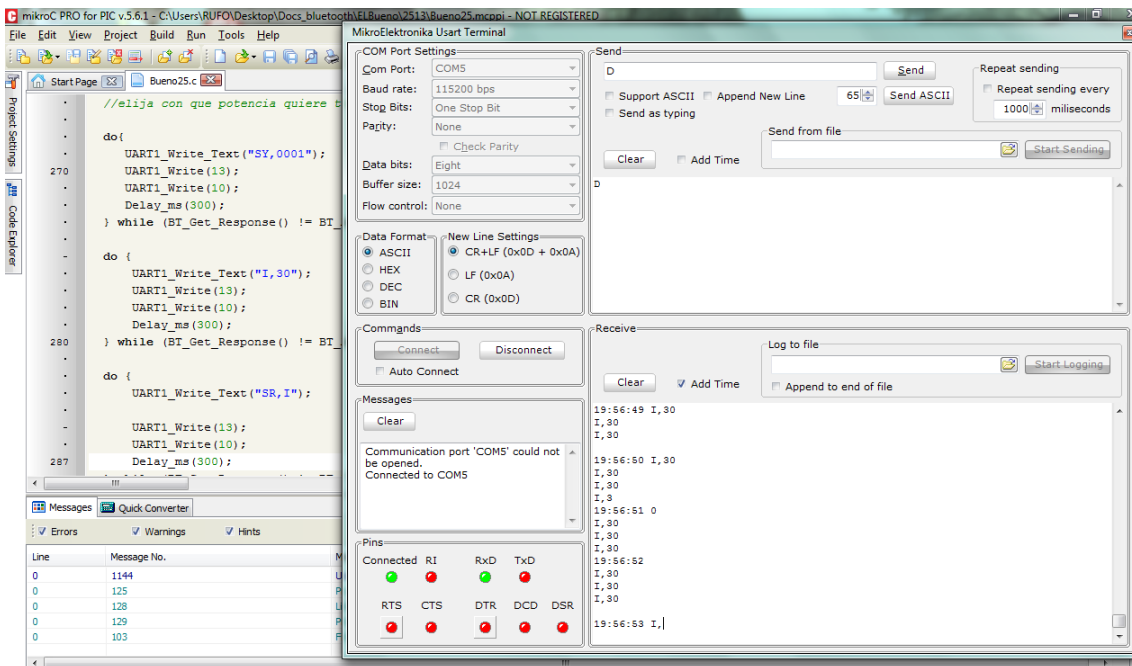
do {
    UART1_Write_Text("SR,I");

    UART1_Write(13);
    UART1_Write(10);
    Delay_ms(300);
} while (BT_Get_Response() != BT
    
```

COM Port Settings:
 Com Port: COM5
 Baud rate: 115200 bps
 Stop Bits: One Stop Bit
 Parity: None
 Data bits: Eight
 Buffer size: 1024
 Flow control: None
 Data Format: ASCII
 New Line Settings: CR+LF (0x0D + 0x0A)
 Commands: Connect, Disconnect, Auto Connect
 Messages: Clear
 Pins: Connected, RI, TxD, RTS, CTS, DTR, DCD, DSR

mikroProg Suite:
 MCU Family: PIC18F3
 MCU: PIC18F87360
 Read, Write, Verify, Blank, Erase, Reset
 HEX File Options: Load, Save, Reload HEX, Load/Save CODE, Load/Save DATA
 CODE, DATA, UNIT ID, Options
 Progress: 0%
 Operation: None
 HEX File: Loaded





Funcionalidad de los comandos que están el código visto desde la consola, realizando el enlace con tres dispositivos:

- Sony Ericsson Neo V
- Motorola Red King
- Nokia N8-00

1. CMD	93. 00066608CD14
2. Inquiry, COD=0	94. 0
3. Found 1	95. TRYING
4. 90CF155E7CE9,,5A020C	96. CONNECT failed
5. Inquiry Done	97. TRYING
6. Inquiry, COD=0	98. CONNECT failed
7. Found 3	99. TRYING
8. 0017E879785A,,5A2204	100. CONNECT failed
9. 3039263D124E,Xperia neo V,58020C	101. TRYING
10. 90CF155E7CE9,Nokia N8-00,5A020C	102. CONNECT failed
11. Inquiry Done	103. TRYING
12. CMD	104. CONNECT failed
13. Inquiry, COD=0	105. CONNECT failed
14. Found 3	106. CONNECT failed
15. 90CF155E7CE9,Nokia N8-00,5A020C	107. CONNECT failed
16. 0017E879785A,Red king,5A2204	108. Inquiry, COD=0
17. 3039263D124E,Xperia neo V,58020C	109. Found 2
18. Inquiry Done	110. 90CF155E7CE9,Nokia N8-00,5A020C
19. CMD	111. 70D4F240253D,BlackBerry 8520 BOP,7A020C
20. ***ADVANCED Settings***	112. Inquiry Done
21. SrvName= SPP	113. TRYING
22. SrvClass=0000	114. CONNECT failed
23. DevClass=1F00	115. *** SET COMMANDS ***
24. InqWindw=0100	116. SA,<1,0> - Authentication
25. PagWindw=0100	117. SB,<num> - Send Break
26. CfgTimer=60	118. SC,<hex> - Service Class
27. StatuStr=Master	119. SD,<hex> - Device Class
28. ***Settings***	120. SE,<1,0> - Encryption
29. BTA=00066608CD14	121. SF,1 - Factory Defaults
30. BTName=Bluetooth-Master	122. SI,<hex> - Inquiry Scan Window
31. Baudrt(SW4)=9600	123. SJ,<hex> - Page Scan Window
32. Parity=None	124. SL,<E,O,N> - Parity
33. Mode =Mstr	125. SM,<0-5> - Mode (0=slav,1=mstr,2=trig,3=auto,4=DTR,5=Any)
34. Authen=1	126. SN,<name> - Name
35. Encryp=0	127. SO,<text> - conn/discon Status
36. PinCod=mikroe	128. SP,<text> - Pin Code
37. Bonded=0	129. SR,<adr> - Remote Address
38. Rem=NONE SET	130. SS,<text> - Service Name
39. AOK	131. ST,<num> - Config Timer
40. ***Settings***	132. SU,<rate> - Baudrate
41. BTA=00066608CD14	133. SW,<hex> - Sniff Rate
42. BTName=Bluetooth-Master	134. SX,<1,0> - Bonding
43. Baudrt(SW4)=9600	135. SY,<hex> - TX power
44. Parity=None	136. SZ,<num> - Raw Baudrate
45. Mode =Mstr	137. S7,<0-1> - 7bit data
46. Authen=1	
47. Encryp=0	
48. PinCod=mikroe	
49. Bonded=0	

50. Rem=NONE SET	138. S~,<0-3> - Profile (0=SPP,1=DCE,2=DTE,3=MDM,4=D&S
51. ***ADVANCED Settings***	139. S?,<0-1> - role switch
52. SrvName= wtc	140. S\$,<char> - CMD mode char
53. SrvClass=0000	141. S@,<hex> - io port dir
54. DevClass=1F00	142. S&,<hex> - io port val
55. InqWindw=0100	143. S%,<hex> - io boot dir
56. PagWindw=0100	144. S^,<hex> - io boot val
57. CfgTimer=60	145. S*,<hex> - pio(8-11) set
58. StatuStr=Master	146. S ,<hex> - low power timers
59. AOK	147. *** DISPLAY ***
60. ***ADVANCED Settings***	148. D - Basic Settings
61. SrvName= wtc	149. E - Extended Settings
62. SrvClass=0000	150. G<X> - Stored setting
63. DevClass=1F00	151. GB - BT Address
64. InqWindw=0100	152. GK - Connect Status
65. PagWindw=0100	153. G& - I/O Ports
66. CfgTimer=60	154. *** OTHER ***
67. StatuStr=Master	155. CONNECT failed
68. ***Settings***	156. AOK
69. BTA=00066608CD14	157. CONNECT failed
70. BTName=Bluetooth-Master	158. TRYING
71. Baudrt(SW4)=9600	159. CONNECT failed
72. Parity=None	160. AOK
73. Mode =Mstr	161. CMD
74. Authen=1	162. Inquiry, COD=0
75. Encryp=0	163. Found 1
76. PinCod=mikroe	164. 90CF155E7CE9,Nokia N8- 00,5A020C
77. Bonded=0	165. Inquiry Done
78. Rem=NONE SET	166. ?
79. CMD	167. Inquiry, COD=0
80. AOK	168. Found 1
81. ***Settings***	169. 90CF155E7CE9,Nokia N8- 00,5A020C
82. BTA=00066608CD14	170. Inquiry Done
83. BTName=Bluetooth-Master	171. Inquiry, COD=0
84. Baudrt(SW4)=9600	172. Found 1
85. Parity=None	173. 0017E879785A,Red king,5A2204
86. Mode =Mstr	174. Inquiry Done
87. Authen=1	175. Inquiry, COD=0
88. Encryp=0	176. Found 2
89. PinCod=1234	177. 0017E879785A,Red king,5A2204
90. Bonded=0	178. 3039263D124E,Xperia neo V,58020C
91. Rem=NONE SET	179. Inquiry Done
92. AOK	

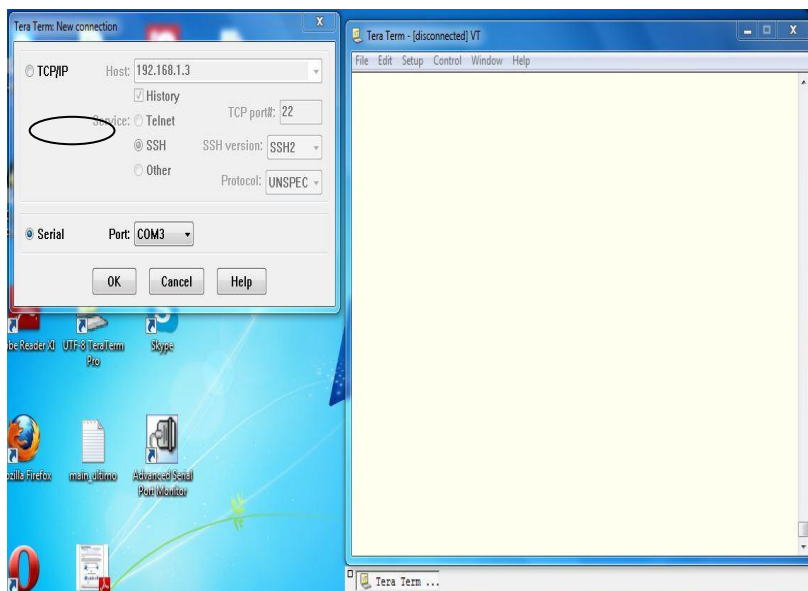
Visto de otra manera...

1. CMD	92.
2. Inquiry, COD=0	93. AOK
3. Found 1	94. 00066608CD14
4. 90CF155E7CE9,,5A020C	95. 0
5. Inquiry Done	96. TRYING
6. Inquiry, COD=0	97. CONNECT failed
7. Found 3	98. TRYING
8. 0017E879785A,,5A2204	99. CONNECT failed
9. 3039263D124E,Xperia neo V,58020C	100. TRYING
10. 90CF155E7CE9,Nokia N8-00,5A020C	101. CONNECT failed
11. Inquiry Done	102. TRYING
12. CMD	103. CONNECT failed
13. Inquiry, COD=0	104. TRYING
14. Found 3	105. CONNECT failed
15. 90CF155E7CE9,Nokia N8-00,5A020C	106. CONNECT failed
16. 0017E879785A,Red king,5A2204	107. CONNECT failed
17. 3039263D124E,Xperia neo V,58020C	108. CONNECT failed
18. Inquiry Done	109. Inquiry, COD=0
19. CMD	110. Found 2
20. ***ADVANCED Settings***	111. 90CF155E7CE9,Nokia N8- 00,5A020C
21. SrvName= SPP	112. 70D4F240253D,BlackBerry 8520 BOP,7A020C
22. SrvClass=0000	113. Inquiry Done
23. DevClass=1F00	114. TRYING
24. InqWindw=0100	115. CONNECT failed
25. PagWindw=0100	116. *** SET COMMANDS ***
26. CfgTimer=60	117. SA,<1,0> - Authentication
27. StatuStr=Master	118. SB,<num> - Send Break
28. ***Settings***	119. SC,<hex> - Service Class
29. BTA=00066608CD14	120. SD,<hex> - Device Class
30. BTName=Bluetooth-Master	121. SE,<1,0> - Encryption
31. Baudrt(SW4)=9600	122. SF,1 - Factory Defaults
32. Parity=None	123. SI,<hex> - Inquiry Scan Window
33. Mode =Mstr	124. SJ,<hex> - Page Scan Window
34. Authen=1	125. SL,<E,O,N> - Parity
35. Encryp=0	126. SM,<0-5> - Mode (0=slav,1=mstr,2=trig,3=auto,4=DTR,5= Any)
36. PinCod=mikroe	127. SN,<name> - Name
37. Bonded=0	128. SO,<text> - conn/discon Status
38. Rem=NONE SET	129. SP,<text> - Pin Code
39. AOK	130. SR,<adr> - Remote Address
40. ***Settings***	131. SS,<text> - Service Name
41. BTA=00066608CD14	132. ST,<num> - Config Timer
42. BTName=Bluetooth-Master	133. SU,<rate> - Baudrate
43. Baudrt(SW4)=9600	134. SW,<hex> - Sniff Rate
44. Parity=None	135. SX,<1,0> - Bonding
45. Mode =Mstr	
46. Authen=1	
47. Encryp=0	
48. PinCod=mikroe	

49. Bonded=0	136. SY,<hex> - TX power
50. Rem=NONE SET	137. SZ,<num> - Raw Baudrate
51. ***ADVANCED Settings***	138. S7,<0-1> - 7bit data
52. SrvName= wtc	139. S~,<0-3> - Profile
53. SrvClass=0000	(0=SPP,1=DCE,2=DTE,3=MDM,4=D&S
54. DevClass=1F00	140. S?,<0-1> - role switch
55. InqWindw=0100	141. S\$,<char> - CMD mode char
56. PagWindw=0100	142. S@,<hex> - io port dir
57. CfgTimer=60	143. S&,<hex> - io port val
58. StatuStr=Master	144. S%,<hex> - io boot dir
59. AOK	145. S^,<hex> - io boot val
60. ***ADVANCED Settings***	146. S*,<hex> - pio(8-11) set
61. SrvName= wtc	147. S ,<hex> - low power timers
62. SrvClass=0000	148. *** DISPLAY ***
63. DevClass=1F00	149. D - Basic Settings
64. InqWindw=0100	150. E - Extended Settings
65. PagWindw=0100	151. G<X> - Stored setting
66. CfgTimer=60	152. GB - BT Address
67. StatuStr=Master	153. GK - Connect Status
68. ***Settings***	154. G& - I/O Ports
69. BTA=00066608CD14	155. *** OTHER ***
70. BTName=Bluetooth-Master	156. CONNECT failed
71. Baudrt(SW4)=9600	157. AOK
72. Parity=None	158. CONNECT failed
73. Mode =Mstr	159. TRYING
74. Authen=1	160. CONNECT failed
75. Encryp=0	161. AOK
76. PinCod=mikroe	162. CMD
77. Bonded=0	163. Inquiry, COD=0
78. Rem=NONE SET	164. Found 1
79. CMD	165. 90CF155E7CE9,Nokia N8-
80. AOK	00,5A020C
81. ***Settings***	166. Inquiry Done
82. BTA=00066608CD14	167. ?
83. BTName=Bluetooth-Master	168. Inquiry, COD=0
84. Baudrt(SW4)=9600	169. Found 1
85. Parity=None	170. 90CF155E7CE9,Nokia N8-
86. Mode =Mstr	00,5A020C
87. Authen=1	171. Inquiry Done
88. Encryp=0	172. Inquiry, COD=0
89. PinCod=1234	173. Found 1
90. Bonded=0	174. 0017E879785A,Red
91. Rem=NONE SET	king,5A2204
	175. Inquiry Done
	176. Inquiry, COD=0
	177. Found 2
	178. 0017E879785A,Red
	king,5A2204
	179. 3039263D124E,Xperia neo
	V,58020C
	180. Inquiry Done

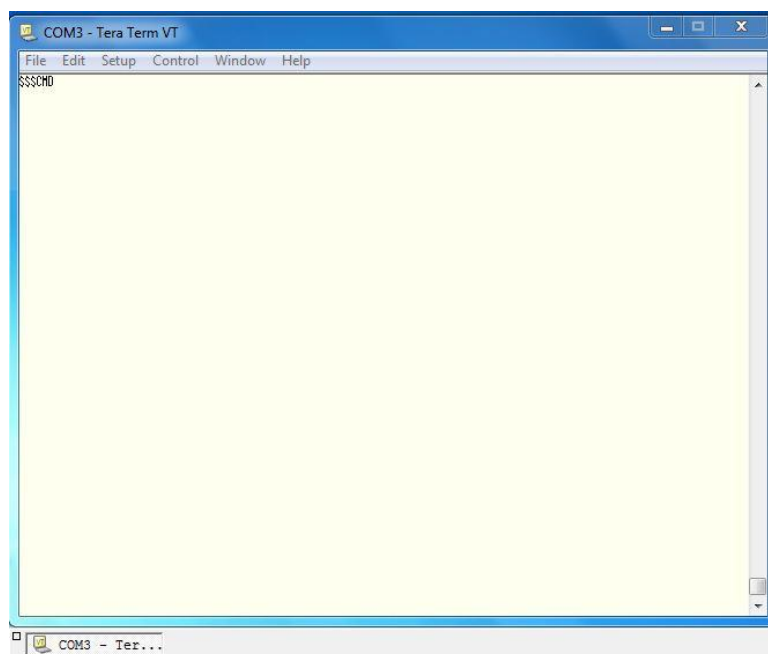
Empatar los dispositivos

Para empatar los dispositivos RN41 con otro dispositivo utilizaremos una terminal llamada “**Tera Term**”, en la cual se configurara el puerto que se utilizara.



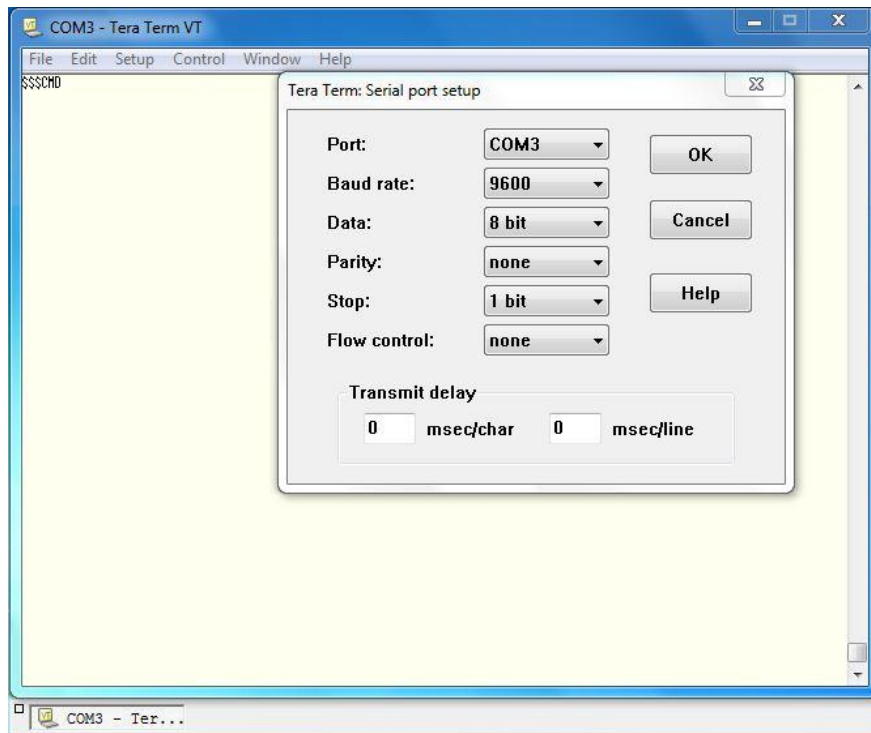
Configuración del puerto a utilizar.

Después de utilizará en modo comando, lo cual es por medio de \$\$\$.



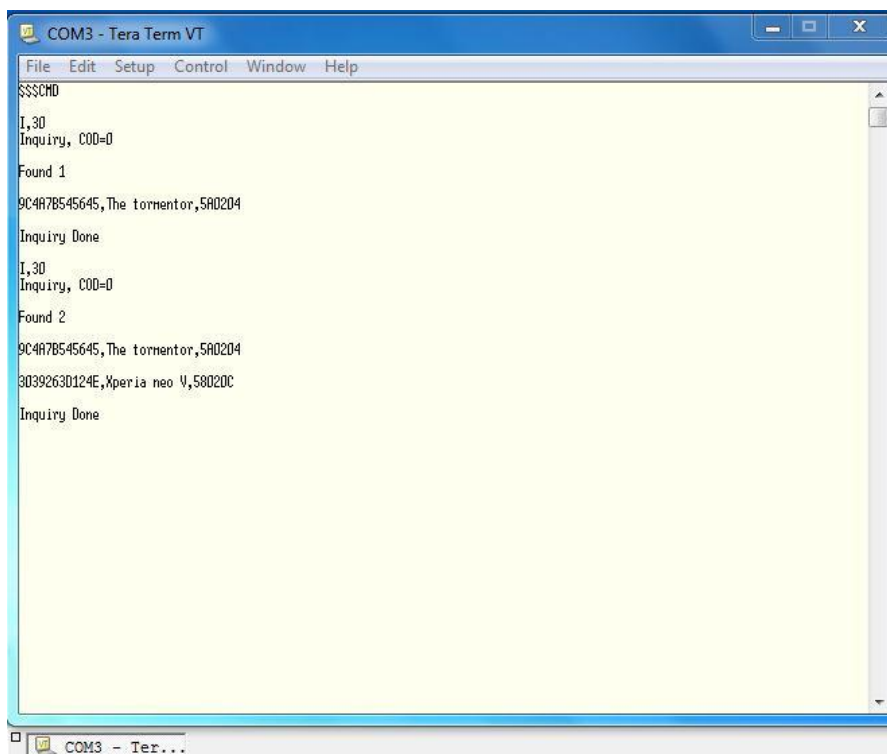
Modo comando.

Enseguida verificaremos que esté operando modo comando CMD y verificando que la velocidad sea la correcta.



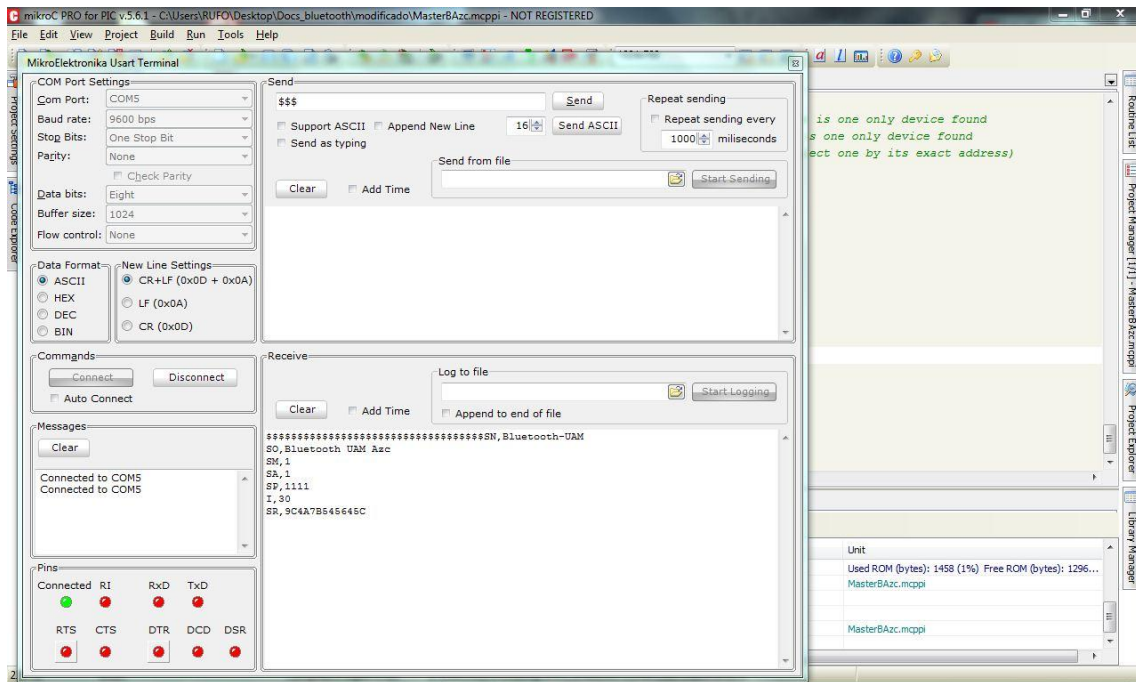
Verificación de modo de las propiedades.

Realización del escaneo de los dispositivos por medio del comando I, 30;

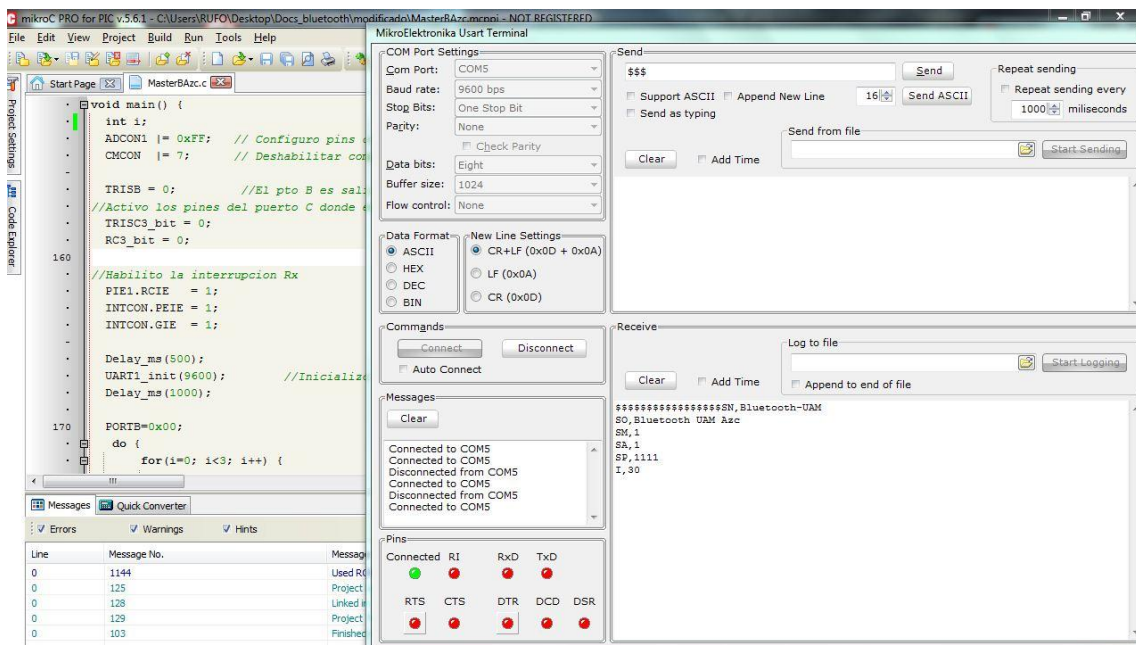


Escaneo de dispositivos por 30 seg.

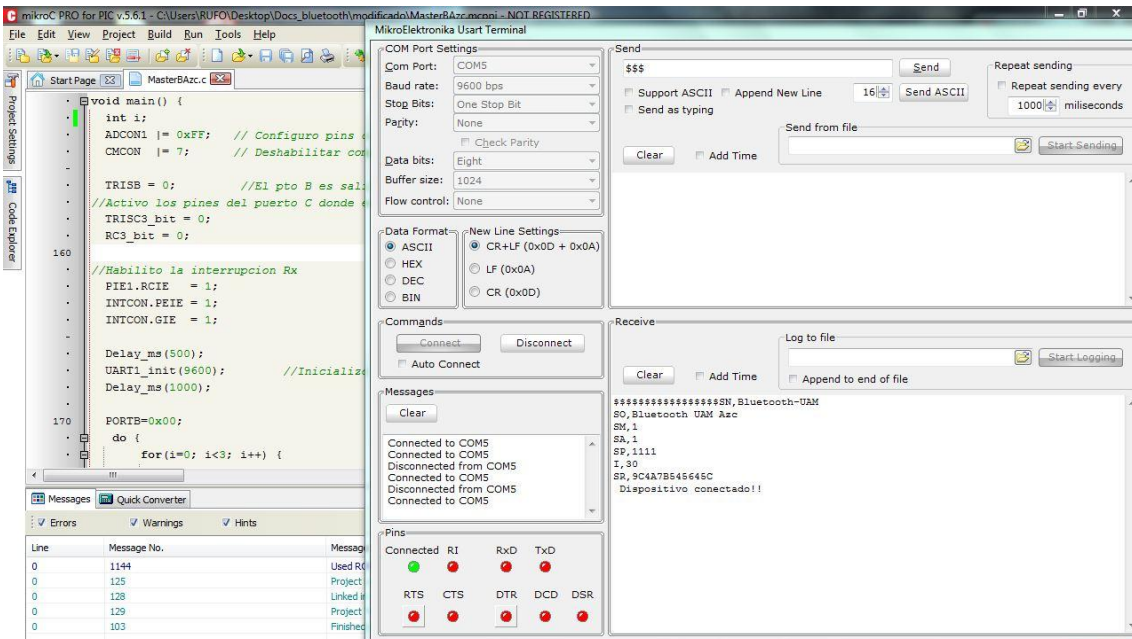
Por medio de la programación en el lenguaje de Mikro C, realizaremos la conexión con los dispositivos móviles.



Búsqueda del dispositivo.

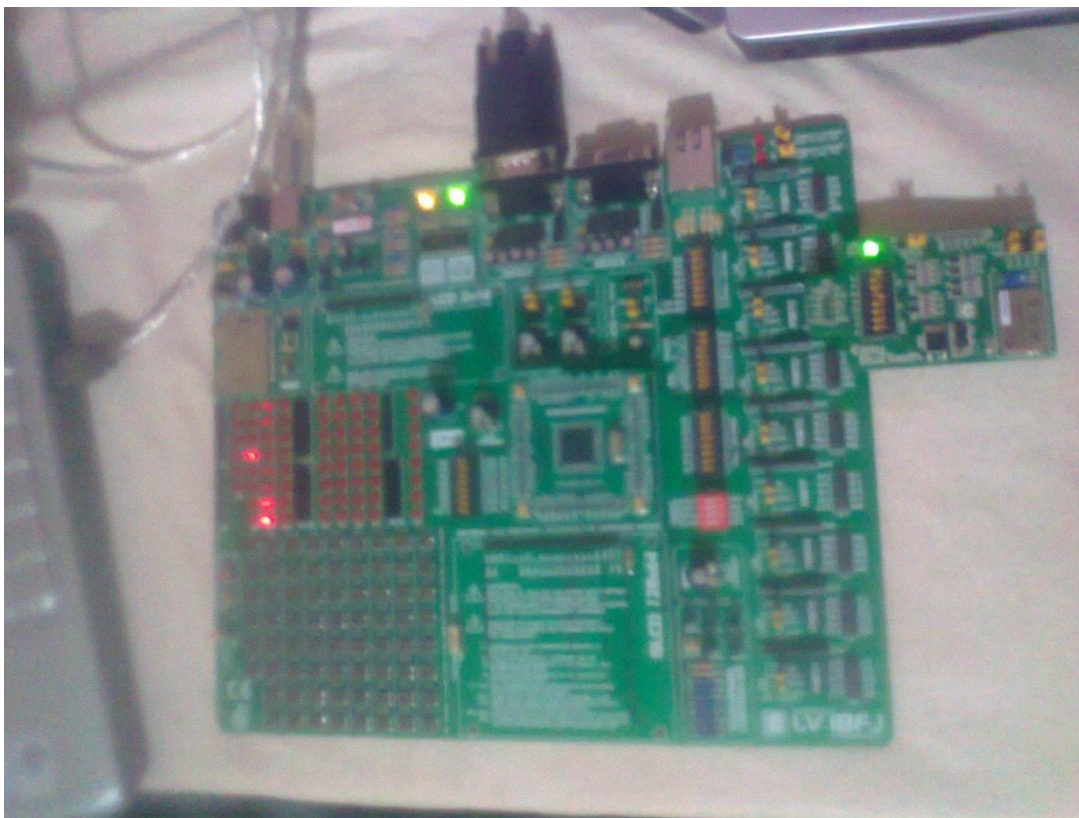


Muestra parte del código para la búsqueda de un dispositivo Bluetooth.



Se realiza el emparejamiento entre la tarjeta y el dispositivo móvil
Por medio de la dirección MAC.

La tarjeta y el Bluetooth ensamblada, para programar y validar la conexión y monitoreo del dispositivo



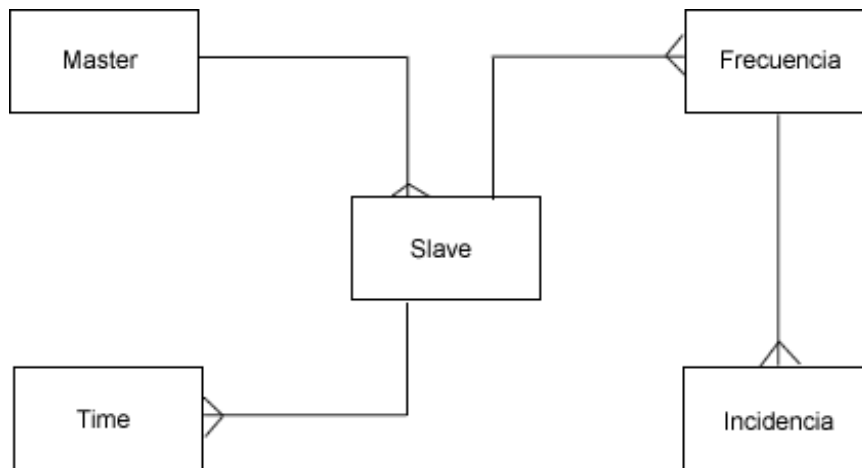
El led ubicado en la posición b3 se encenderá cuando detecte un dispositivo un dispositivo móvil.

Tarjeta y Bluetooth.

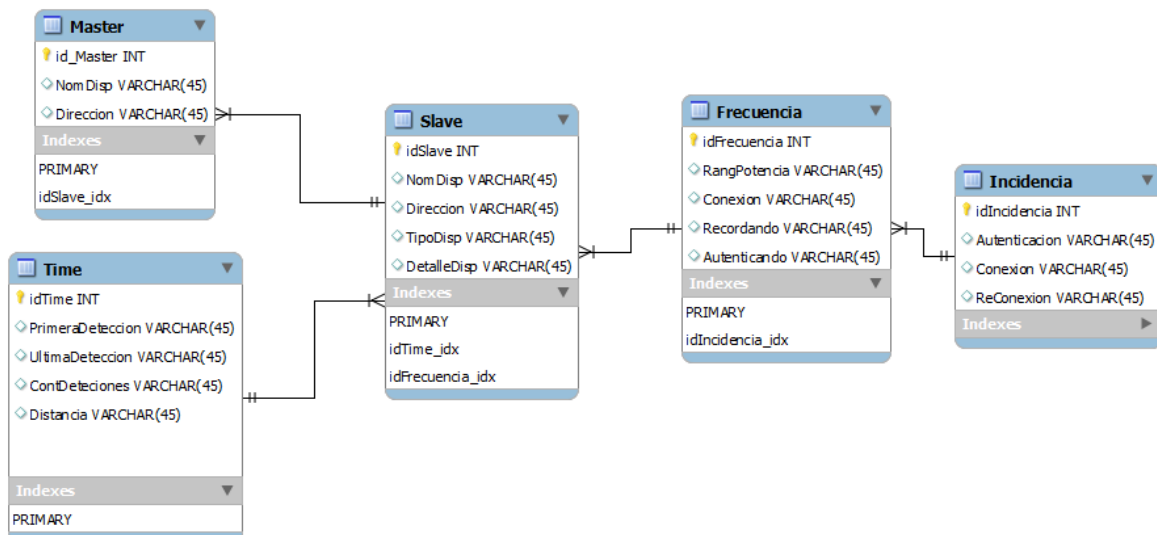


Diseño del Diagrama Entidad Relación

- Modelo E/R









- Modelado de Entidad/Relación









Diccionario de Datos

Columna	Tipo	Unique	Not null	Llave	Descripción
Tabla	Master				
id_Master	Serial	unique	Not null	primary key	Identificador de numero de tienda
NomDisp	varchar(40)		Not null		Nombre del Dispositivo
Dirección	varchar(17)		Not null		Direccion del dispositivo MAC Adres
Tabla	Slave				
Id_Slave	serial	unique	Not null	primary key	Identificador de Esclavo
NomDisp	varchar(40)		Not null		Nombre de Dispositivo
Direccion	varchar(17)		Not null		Apellido Paterno del empleado
TipoDisp	varchar(40)		Not null		Tipo de Dispositivo
DetalleDisp	varchar(40)		Not null		Detalle del Dispositivo
Tabla	Time				
Id_Time	serial	unique	Not null	primary key	
PrimeraDeteccion	varchar(40)		Not null		Primera Detección del Dispositivo
UltimaDeteccion	varchar(40)		Not null		Última Detección del Dispositivo
ContDetecciones	varchar(40)		Not null		Conteo de la conexiones
Distancia	varchar(40)		Not null		Alcance del dispositivo
Tabla	Frecuencia				
Id_Frecuencia	serial	unique	Not null	primary key	Identificador de Frecuencia
RangPotencia	varchar(40)				Área de conexión
Conexión	varchar(40)				Si el dispositivo está activo
Autenticando	varchar(40)				Autenticación por primera vez del dispositivo
Tabla	Incidencia				
Id_incidencia	serial	unique	Not null	primary key	
Autenticacion	varchar(40)		Not null		Nombre del video juego
Conexión	varchar(40)		Not null		El tipo de permiso
Reconexion	varchar(40)		Not null		Si el dispositivo se activa de nuevo

Se muestra en esta tabla los dispositivos que se encuentran en radio permitido, en donde se puede ver el nombre del dispositivo, descripción, dirección MAC, tipo de dispositivo, detalle de la clase

Nombre de dispositivo	Descrip...	Dirección	Tipo de dispositivo	Detalle de clase
		a0:75:91:fe:ee:a5	Teléfono	Teléfono móvil
 Bluetooth-UAM		00:06:66:06:60:96	Miscelanea	
 iPhone de MLagos		44:d8:84:2b:71:e7	Teléfono	Smart
 Red king		00:17:e8:79:78:5a	Teléfono	Teléfono móvil
 SNPS		00:21:4f:bd:72:74	Ordenador	Escritorio
 CHESTER-PC		00:24:2c:b3:53:50	Ordenador	Ordenador portátil

Primera detección	Última detección	Contador de detecc...	No Detection Counter	% Detection
04/01/2013 11:28:34 a.m.	04/01/2013 11:28:34 a.m.	1	151	0.7%
04/01/2013 11:30:44 a.m.	04/01/2013 11:44:21 a.m.	21	124	14.5%
04/01/2013 11:35:33 a.m.	04/01/2013 11:42:27 a.m.	20	110	15.4%
04/01/2013 11:28:03 a.m.	04/01/2013 11:45:38 a.m.	36	118	23.4%
04/01/2013 11:28:03 a.m.	04/01/2013 12:14:40 p.m.	144	10	93.5%
04/01/2013 11:28:03 a.m.	04/01/2013 12:14:40 p.m.	154	0	100.0%

Nombre de dispositivo	Descrip...	Dirección	Tipo de dispositivo	Detalle de clase	Primera detección	Última detección	Contador de detecc...	No Detection Counter	% Detection
		a0:75:91:fe:ee:a5	Teléfono	Teléfono móvil	04/01/2013 11:28:34 a.m.	04/01/2013 11:28:34 a.m.	1	145	0.7%
 Bluetooth-UAM		00:06:66:06:60:96	Miscelanea		04/01/2013 11:30:44 a.m.	04/01/2013 11:44:21 a.m.	21	118	15.1%
 iPhone de MLagos		44:d8:84:2b:71:e7	Teléfono	Smart	04/01/2013 11:35:33 a.m.	04/01/2013 11:42:27 a.m.	20	104	16.1%
 Red king		00:17:e8:79:78:5a	Teléfono	Teléfono móvil	04/01/2013 11:28:03 a.m.	04/01/2013 11:45:38 a.m.	36	112	24.3%
 SNPS		00:21:4f:bd:72:74	Ordenador	Escritorio	04/01/2013 11:28:03 a.m.	04/01/2013 12:12:50 p.m.	141	7	95.3%
 CHESTER-PC		00:24:2c:b3:53:50	Ordenador	Ordenador portátil	04/01/2013 11:28:03 a.m.	04/01/2013 12:12:50 p.m.	148	0	100.0%

Bibliografía

- [1] MikroElektronika [05-07-2012]. "Software and Hardware Solutions for the Embedded World. [En línea]
Disponible:http://www.mikroe.com/eng/downloads/get/58/lv18fj_manual_v100.pdf Consultado: 25 de diciembre de 2012.
- [2] Carballar, José A. "Wi-Fi. Instalación Seguridad y Aplicaciones". Primera Edición. Alfaomega Grupo Editor, S.A. de C.V., México ISBN: 978-970-15-1292-0.
- [3] J. R. Castillo Velázquez, Universidad Autónoma Metropolitana unidad Azcapotzalco, Proyecto terminal, Ingeniería en Computación. "Transmisión y registro de las coordenadas geográficas de un dispositivo móvil", 2010.
- [4] J. A. González Romero, M. A. Cruz Salas y A. A. Mellado Fernández, Universidad Autónoma Metropolitana unidad Azcapotzalco, Proyecto terminal, Ingeniería en Computación. "Modelo de un sistema de información geográfica para la gestión catastral", 2010.
- [5] R. Linares Ruiz, J. A. Quijano Vázquez y G. A. Holguín Londoño, Universidad Tecnológica de Pereira, Trabajo de investigación, "Implementación del protocolo bluetooth para la conexión inalámbrica de dispositivos electrónicos programables", 2004.
- [6] L. D. Anbrona Tabernilla, Universidad Politécnica de Madrid, Trabajo fin de carrera, Facultad de informática, "Sistema de localización en interiores", 2008.
- [7] J2ME [30-12-2010]. Java Micro Edition [En línea]. Disponible en: http://es.wikipedia.org/wiki/Java_Micro_Edition Consultado: 25 de diciembre 2012.
- [8] Microingenia Electronics. [13-09-2012] Módulo *Bluetooth* basado en RN41 de Network [En Línea] Disponible: http://www.microingenia.com/electronics/upload/docs/ModBluetooth/Manual_ModBluetooth_V1.0.pdf Consultado: 13 de febrero del 2012
- [9] Zaapa. [28-06-2006]. Antena Receptora GPS con tecnología *Bluetooth*, [En Línea] Disponible en: http://www.zaapa.co.uk/drv/Connectivity/Bluetooth/Antena%20GPS%20Bluetooth/GpsBT_UserManualV1.5MR_Spa.pdf, Consultado: 7 de marzo del 2012.
- [10] MikroC. [19-08-2011]. Diseño y Simulación de Sistemas Microcontrolados en Lenguaje C, [En Línea]. Disponible: http://www.mikroe.com/download/eng/documents/publications/other-books/libro_simulacion_mikroc.pdf Consultado: 25 de diciembre de 2012.

[11] Microcontroladores PIC diseño práctico de aplicaciones”, Angulo Usategui Jose M. Mc Graw Hill, 2003

[12] “Serie de telecomunicaciones tecnología bluetooth”, Nathan J. Muller, Mc Graw Hill, 2002.

[13] MikroC. [19-08-2011]. USB UART Board, [En Línea]. Disponible <http://www.mikroe.com/add-on-boards/communication/usb-uart/> Consultado: 4 de enero de 2013.

