

**Práctica Profesional Supervisada**  
**Características software de la placa Intel Galileo GEN 1**

**Marón, Gastón Ariel**

**Contacto: [gastonmaron@gmail.com](mailto:gastonmaron@gmail.com)**



# Tabla de contenido

## Práctica Profesional Supervisada

### Introducción

### Detalles de arquitectura

### Sistemas Operativos

### Puesta en marcha de la placa

#### Paso 1: Actualización del firmware

#### Paso 2: Instalación de Linux Yocto en tarjeta SD

#### Paso 3: Encienda la placa Intel Galileo

#### Paso 4: Instalación de controladores

#### Paso 5: Comunicación

#### Paso 6: Instalación del entorno

#### Eclipse

#### Arduino

#### Entorno de desarrollo Intel ® XDK IoT Edition

### Windows Embedded for IoT en Intel Galileo

#### Instalación

#### Prueba de funcionamiento correcto

#### Desarrollo (FALTA LA PARTE DE LA SIMULACION)

### Node.js

#### Paralelismo

#### V8

#### Desarrollo homogéneo entre cliente y servidor

[Lazo de eventos](#)

[Módulos](#)

[NPM \(Node Package Manager\)](#)

[Crear una aplicación con npm](#)

[Poner un paso a paso de lo que el gestor te va diciendo](#)

[Instalación de una aplicación](#)

[Instalar un paquete con NPM para usar en el proyecto](#)

[Iniciar una aplicación](#)

[Aplicación en Intel ® XDK IoT Edition](#)

[Explicación de la aplicación](#)

[Módulos instalados y su funcionamiento](#)

[Express](#)

[Ejs](#)

[Sequence](#)

[Módulos nativos utilizados y su funcionamiento](#)

[Es](#)

[Path](#)

[childProcess](#)

[Instrucciones para importar y ejecutar](#)

[Árbol de archivos](#)

[Módulos probados](#)

[Bibliografía](#)





## Introducción

Galileo es una placa microcontroladora basada en el procesador Intel® Quark SoC X1000 con una arquitectura 32-bit. Es la primera placa basada en la arquitectura Intel® diseñada para ser pin-compatible tanto en hardware como software con los shields Arduino UNO. Posee 12 pines digitales de entrada/salida (numerados del 2 al 13), los pines digitales AREF y GND; 6 entradas analógicas, una tira de pins hembra para tomar tensión (5V – 3.3V – GND), pines ICSP, y los pines 0 y 1 del puerto UART. Todos los pines se encuentran ubicados de la misma manera que el Arduino UNO R3.

Galileo está diseñada para soportar shields que operan tanto con 3.3V o 5V. La tensión principal en la que opera Galileo es 3.3V. Sin embargo, un jumper en la placa habilita la tensión 5V en los pines de entrada/salida. Esto provee soporte para los shields UNO 5V, siendo éste es su comportamiento por defecto.

La placa Intel Galileo, es también software compatible con el IDE de Arduino. Además, la placa Galileo tiene muchos puertos de entrada/salida para el uso industrial y características para expandir el uso nativo y capacidad más allá del ecosistema Arduino.

Galileo contiene sobre su placa: un slot mini-PCI Express, un puerto Ethernet de 100Mb, un slot para tarjeta Micro-SD, un puerto serie RS232, un puerto Host USB, un puerto Cliente USB, y una memoria flash NOR de 8Mbyte.





- o Entre 256 y 512 KB son dedicados al almacenamiento del *sketch*.
- o 512KB SRAM embebida, habilitada por defecto por el firmware.
- o 256MB DRAM, habilitada por defecto por el firmware.
- o Micro SD hasta 32gb de almacenamiento. (Preferentemente clase 10).
- o Almacenamiento USB que permite cualquier tipo de dispositivo USB 2.0.
- o 11 KB memoria EEPROM programable a través de librería EEPROM.

## **Sistemas Operativos**

Galileo está diseñada para correr dos sistemas operativos separados. El primero es una imagen de tan solo 8 Mb que está situado en la memoria flash (también referenciada como la versión SPI-flash), que contiene las cosas básicas para correr los programas de Arduino. Es sorprendente que un Sketch de Arduino es un programa de Linux que es subido por el IDE de Arduino. Si se corre un Sketch, no se podrá bootear desde la tarjeta SD.

Por otra parte, la imagen completa en la tarjeta SD, está basada en una comunidad de desarrolladores que construyen una distribución de Linux denominada Yocto. La imagen por defecto está disponible desde el sitio de Intel, cuyo nombre es LSB, (Linux Standard Build). Hasta el momento, no ha habido una gran cantidad de apoyo "oficial" de la placa en cuanto a correcciones de errores y gestión de paquetes, pero la comunidad ha tomado el relevo de actualizar el software como Node.JS y habilitar nuevos controladores para periféricos USB.

Sin embargo, hay otros sistemas operativos, como por ejemplo el proyecto “Windows Embebed for IoT” el cual tiene soporte para la placa Galileo. Lamentablemente, el soporte otorgado por Microsoft caducará el 30 de Noviembre de 2015. Se explicará más adelante como se puede instalar esta versión de sistema operativo y correr el ejemplo básico de un led parpadeante (*Blinking Led*).

## Puesta en marcha de la placa

### Paso 1: Actualización del firmware

En este apartado se explicará cómo actualizar el firmware de la placa Intel Galileo Gen 1.

#### Paso 1.1: Descargar el software

Primero hay que descargar la herramienta *Intel® Galileo Firmware Updater*,<sup>1</sup> y como se está trabajando con Windows, se descarga además en el mismo .zip el archivo *linux-cdc-acm.inf*, a continuación coloque los dos archivos en una carpeta de su preferencia. Además, en caso de ser necesario, instalar *Java Runtime Enviroment (JRE) v6*, *Java Enterprise Edition 6 SDK* o alguna versión más reciente.

#### Paso 1.2: Preparar placa Intel Galileo

Antes de empezar, se debe realizar lo siguiente:

- Remover la fuente de alimentación.
- Remover cualquier cable USB conectado a la placa.
- Remover la tarjeta SD, en caso de estar colocada en la ranura.

**Nota:** Es necesario remover la tarjeta SD ya que no es posible actualizar el *firmware* cuando la placa bootea desde una imagen guardada en la tarjeta SD o si hay algún programa corriendo.

#### Paso 1.3: Conectar la placa Intel Galileo

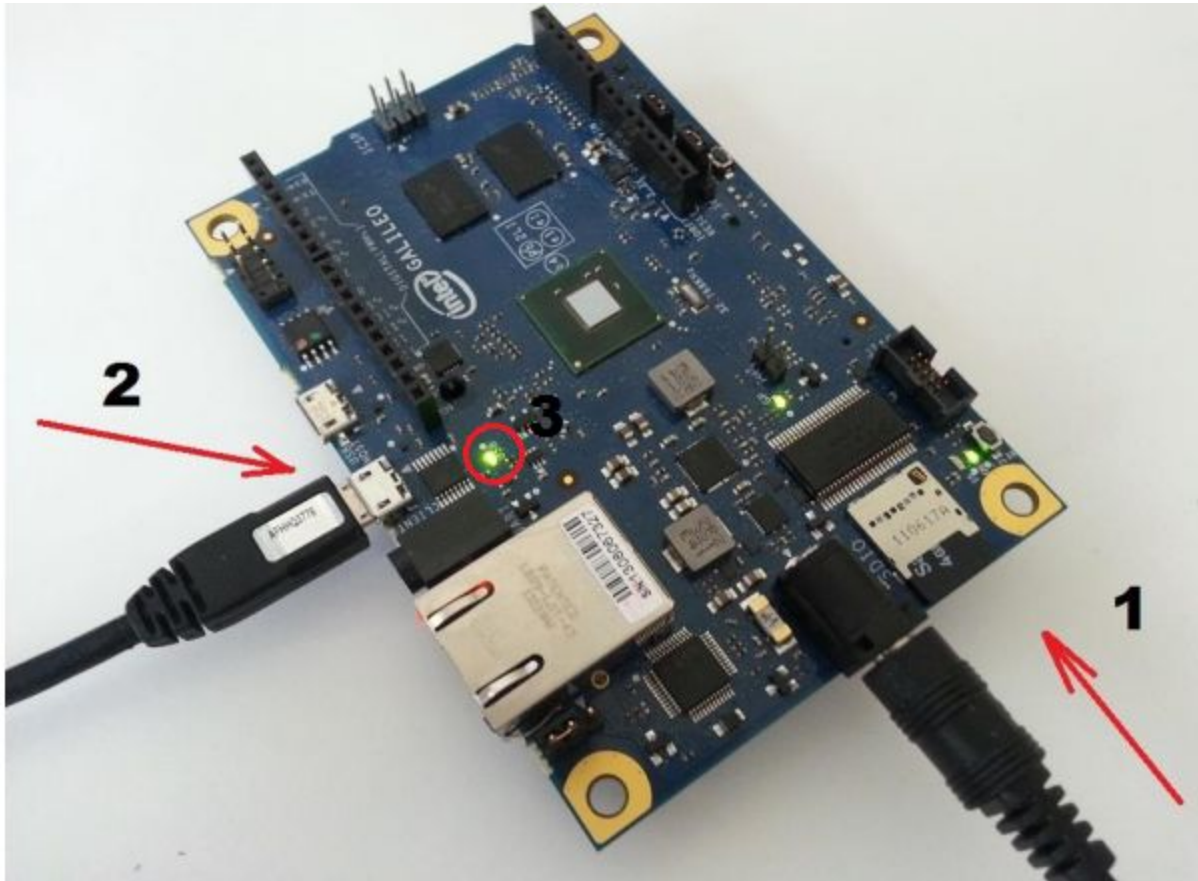
Conectar la placa Intel Galileo a la computadora con el siguiente procedimiento:

- Siempre conectar a la fuente de alimentación (1) antes del cable USB en el puerto de *USB Client* (2) como se muestra en la Figura 1 para prevenir daños en el hardware. El puerto *USB Client* es un conector micro USB cerca del puerto Ethernet. Se debe conectar el micro USB una vez que el led de USB (3) esté encendido.

---

<sup>1</sup> <http://downloadcenter.intel.com/download/24748>

- Siempre mantener la fuente de alimentación conectada la placa Intel Galileo cuando se transfieren programas o se está actualizando el *firmware* de la placa.



**Figura 1.** Conectar la fuente de alimentación antes del cable de datos de USB en la Intel Galileo.

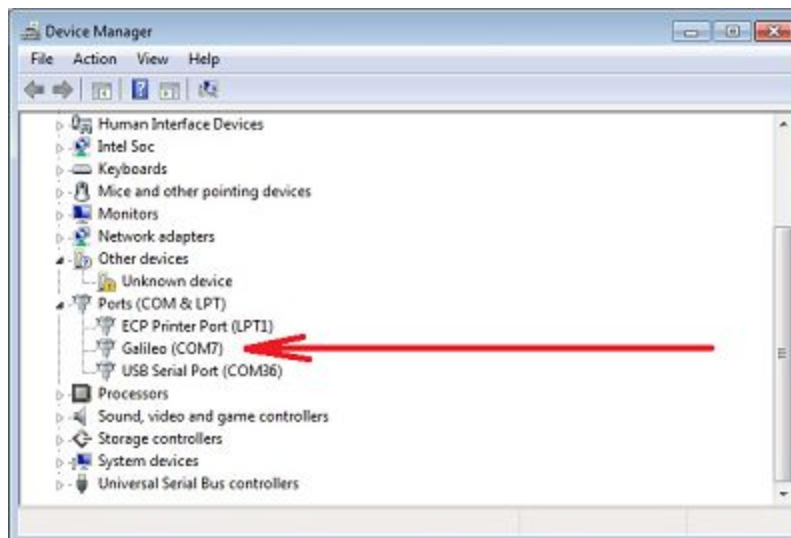
#### **Paso 1.4: Instalar los drivers.**

Antes de poder usar la herramienta *Intel® Galileo Firmware Updater*, hay que asegurarse de que la computadora contenga los drivers del puerto serie así la herramienta Intel Galileo Firmware Updater para comunicarse con la placa Intel Galileo.

##### **Paso 1.4.1: Instalar drivers en Windows**

Para instalar los drivers en Windows, haga lo siguiente:

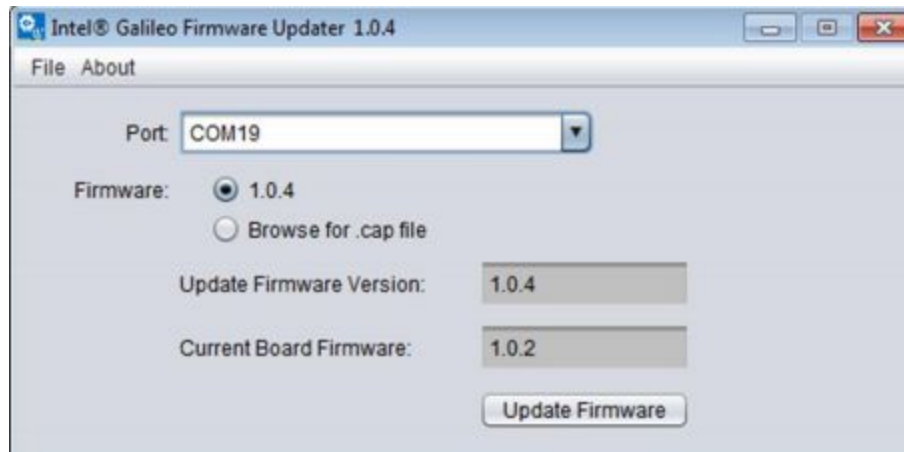
1. Conectar la fuente de alimentación y los cables USB, esperar 10 segundos, entonces abrir el Administrador de dispositivos (desde el Menú de Inicio, abrir el Panel de Control y seleccionar Sistema y seguridad en Windows).
2. En Puertos (COM & LPT), se debe ver un puerto llamado *Gadget Serial V2.4*, si la placa contiene una vieja versión, como V0.7.5 o simplemente Galileo.
3. Click derecho sobre puerto *Gadget Serial V2.4* o *Galileo* y elegir la opción *Actualizar software de controlador*.
4. Seleccionar la opción *Buscar software de controlador en el equipo*.
5. Navegar hacia la locación donde se descargó el archivo `linux-cdc-acm.inf` y seleccionarlo.
6. Una vez que el driver es satisfactoriamente instalado, el *Administrador de tareas* mostrará un dispositivo *Galileo (COMx)* en *Puertos (COM & LPT)*, como se muestra en la Figura 2.



**Figura 2.** Puerto serie de la placa Intel Galileo en el Administrador de tareas.

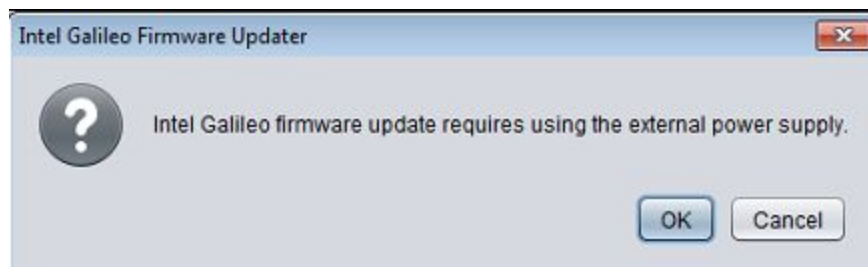
### Paso 1.5: Actualizar el firmware

Al correr la herramienta Intel® Galileo Firmware Updater descargada en la sección 3.1 y seleccione el puerto serie que es conectado desde la placa Intel Galileo usando el menú desplegable llamado Port (Figura 3). Si el puerto seleccionado es el correcto, *Current Board Firmware* mostrará la versión actual que la placa tiene instalada.



**Figura 3.** La ventana de Intel Galileo Firmware Updater.

Un mensaje de dialogo (Figura 4) preguntando si se desea confirmar que la fuente de alimentación está conectada a la placa Intel Galileo. Si es así, click en OK.



**Figura 4.** Diálogo para confirmar si la fuente de alimentación está conectado.

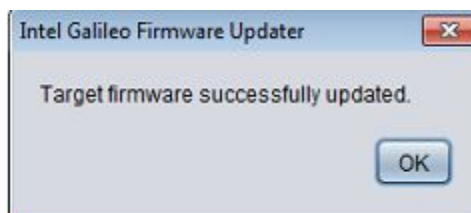
Un segundo mensaje muestra cual es la versión actual del firmware de la placa y el archivo de la versión candidata para actualizar la placa (Figura 5). Si se está de acuerdo, click en *YES*.



**Figura 5.** Diálogo preguntando si la actualización debe proceder.

**Precaución:** En este punto, el proceso de actualización de firmware comenzará. Durante este proceso, NO remueva el cable USB o la fuente de alimentación. La actualización durará aproximadamente unos 5 para completarse.

Si la actualización es satisfactoria, el mensaje que se despliega es el de la figura 6:



**Figura 6.** Actualización del Firmware satisfactoria.

## **Paso 2: Instalación de Linux Yocto en tarjeta SD**

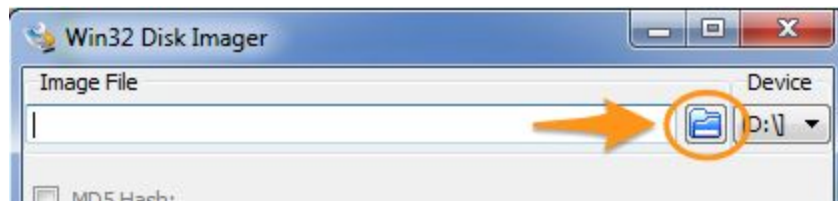
La preparación de la tarjeta microSD permite cargar un sistema operativo como puede ser Linux para su ejecución en la placa de desarrollo. Esto da mayor flexibilidad (herramientas como SSH y drivers para shields WIFI) que utilizar únicamente el firmware de la placa.

### **Requisitos**

- Tarjeta microSD desde 2GB - 32GB.
- Lector de tarjeta microSD para la computadora.

### **En Windows**

- Descargar y descomprimir la última versión de la imagen<sup>2</sup>.
- Una vez descomprimida la imagen, se mostrará el archivo *iot-devkit-version-mmcbllkp0.direct*.
- Escribir imagen en la tarjeta microSD
  1. Descargar la última versión del programa *win32diskimager*<sup>3</sup>.
  2. Insertar tarjeta microSD en la computadora.
  3. Abrir el archivo *Win32DiskImager.exe* con permisos de administrador.
  4. Hacer click en el icono de carpeta mostrado en la siguiente imagen:



Buscar en el directorio donde fue extraído el archivo *iot-devkit-version-mmcbllkp0.direct*.

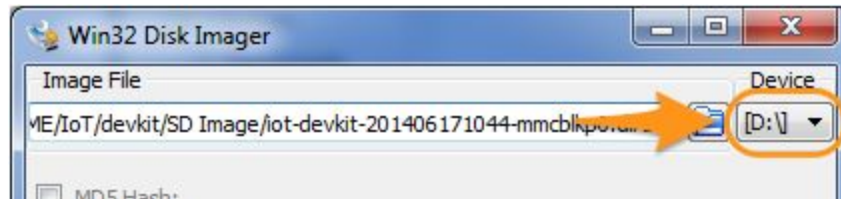
1. Seleccionar la opción \*.\* para visualizar cualquier archivo sin importar la extensión.



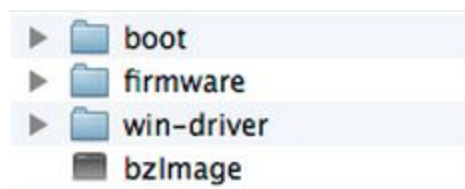
1. Seleccionar el archivo *iot-devkit-version-mmcbllkp0.direct* y hacer click en open.
1. Seleccionar de la lista mostrada a continuación, el punto de montaje de la tarjeta microSD.

<sup>2</sup> <http://iotdk.intel.com/images/iot-devkit-latest-mmcbllkp0.direct.bz2>

<sup>3</sup> <http://sourceforge.net/projects/win32diskimager>



1. Clickear en el botón write y esperar que termine la escritura de la imagen.
1. Luego de completarse, realizar click en exit y luego opcionalmente verificar el contenido de la tarjeta microSD la cual debe tener lo siguiente:



**Nota:** La tarjeta estará lista para su uso, inserte la tarjeta microSD en la placa de desarrollo intel galileo.

### **Paso 3: Encienda la placa Intel Galileo**

1. Verifique que no se encuentre conectado el cable microUSB.
2. Conectar el cable de corriente.
3. Si se utiliza el cable USB (En general con Arduino)
  - 3.1. Esperar que se encienda la luz verde de USB
  - 3.2. Conectar el cable USB
1. En este caso no se utilizó el cable USB sino que se conecta el cable serie UART

### **Paso 4: Instalación de controladores**

Para la placa Intel Galileo gen 1, no se necesita la instalación de controladores adicionales. En caso de que no se pueda reconocer el puerto de comunicaciones serie, se deben seguir los siguientes pasos:



1. Descargar el driver FTDI<sup>4</sup> para Windows.
2. Click derecho sobre el archivo descargado, el cual tiene como nombre “CDM...” y selecciona *Ejecutar como administrador*.



- 2.1. Click en *Extract*
- 2.2. Click en *Next*
- 2.3. Click en *Finish*
- 2.4. Ahora debería aparecer una nueva entrada **USB Serial Port** en “*Ports (COM&LPT)*” en la sección de *Administrador de dispositivos*.

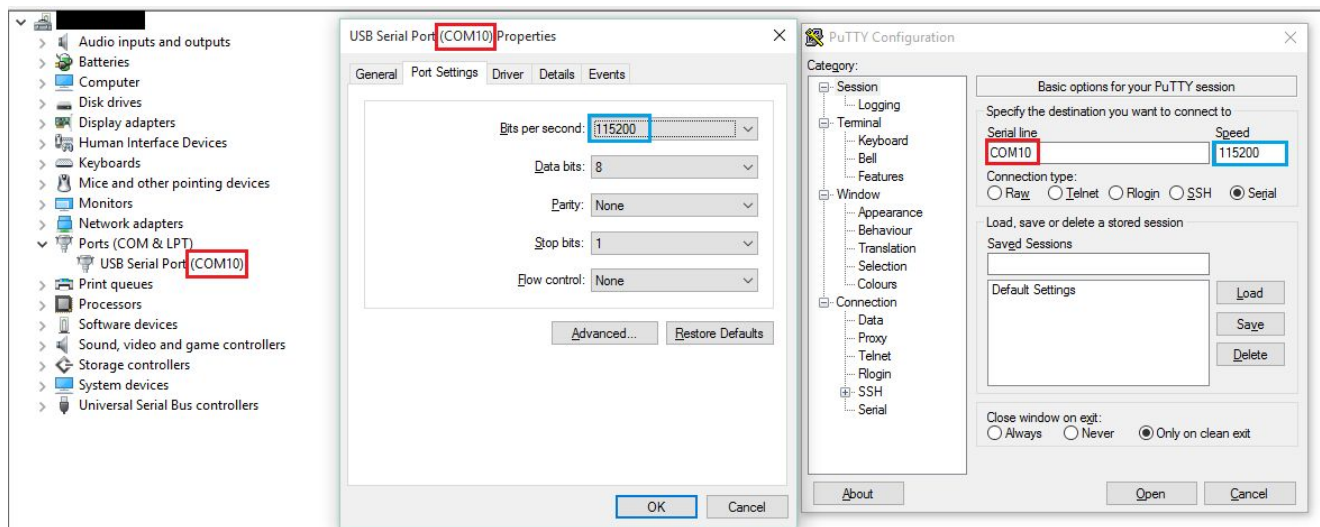
## Paso 5: Comunicación

### Windows

1. Descargar e instalar el software PuTTY<sup>5</sup>.
2. Realizar las siguientes configuraciones
  - 2.1. Tipo de conexión *Serial*
  - 2.2. En el campo *Serial Line*, escribir el puerto COM# correspondiente al conector UART de la placa.
  - 2.3. En el campo *Speed* colocar el valor 115200.

<sup>4</sup> [https://mega.nz/#!ddphEQJZ!yNLQjt\\_RnTOpR0M7d4qDNBxR-acrT3E6F-6w\\_aUhEEw](https://mega.nz/#!ddphEQJZ!yNLQjt_RnTOpR0M7d4qDNBxR-acrT3E6F-6w_aUhEEw)

<sup>5</sup> <http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>



3. Hacer click en *open* del PuTTY, luego se abrirá una nueva ventana en la cual se mostrará el inicio del firmware o sistema operativo en caso de que estuviera disponible.
4. Una vez iniciado el sistema operativo, iniciar sesión con el usuario *root* y sin contraseña (En caso de mostrarse una pantalla en blanco, apretar Enter dos veces).

## Paso 6: Instalación del entorno

Los entornos de desarrollo que nos permiten programar la placa Intel Galileo son Eclipse, con el cual la podemos programar en lenguaje C, con el entorno ECLIPSE; Arduino IDE, para utilizar el lenguaje Arduino y toda las ventajas que éste provee en el uso de librerías; Intel XDK, utilizado para programar en lenguaje Node.js. Este último será el utilizado para esta investigación.

A continuación, se explicará la instalación de cada uno de los entornos, como conectarse con la placa y realizar el básico ejemplo de un parpadeo de un led (*Blinking LED*).

### *Eclipse*

Este apartado contiene los pasos para trabajar con proyectos en lenguaje C/C++ con *Internet Of Things (IoT)*, usando el entorno de desarrollo Eclipse.

Se explicará cómo instalar el Entorno de desarrollo Eclipse sobre un sistema operativo Windows. Eclipse permite crear y testear aplicaciones las plataformas basadas en IoT de Intel. Para la utilización de los sensores, se proveen dos librerías especialmente diseñadas para las placas de Intel:

- MRAA (o libmraa) es una librería de bajo nivel que ofrece una traslación desde la interfaz de entrada/salida a los pines disponibles de la placa Intel Galileo. Así que en lugar de leer la información del nivel del módulo GPIO disponible en el kernel de Linux, un desarrollador puede seleccionar fácilmente un número de pin y trabajar directamente con él. MRAA se encarga de los detalles subyacentes.
- UPM es una librería de sensores con soporte en múltiples lenguajes, utiliza MRAA y que está escrita en C++. Permite usar o crear representaciones de los sensores en los proyectos que se deseen desarrollar. UPM y MRAA tienen la traducción de C++ a JavaScript. Con el Intel Iot XDK Edition, se puede usar Node.js para comunicar todos los pines GPIO, librerías y paquetes.

### *Requerimientos*

- Java\* SE Development Kit 8: se debe tener la versión 64 bits de Java SE Development Kit 8 (también llamado JDK versión 1.8). Eclipse también soporta OpenJDK 1.8 o mayor.

Se puede descargar el JDK desde Oracle<sup>6</sup>. Se debe instalar el JDK 64 bits (indicado por el nombre x64), no la versión 32-bits (x86).

Si no se instaló la versión correcta del JDK, puede llegar a aparecer el mensaje de advertencia similar al siguiente:

---

<sup>6</sup> <http://www.oracle.com/technetwork/java/javase/downloads/index.html>



For steps to create and work with Java projects, see the Eclipse\* User Guide for Java\*. Fijar si lo pongo o no

### *Instalar el IDE Eclipse para Kit de desarrollo de Intel*

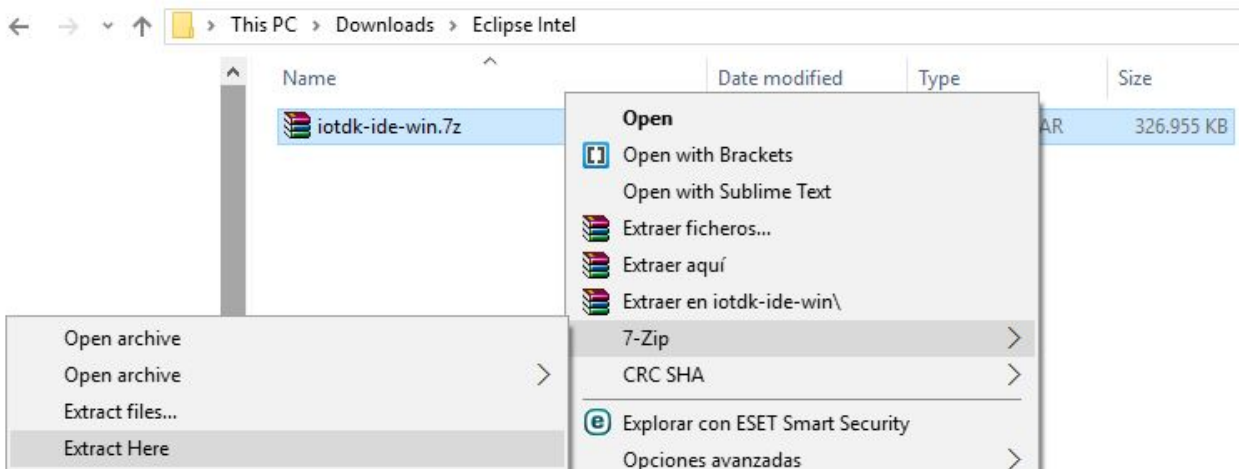
Para poder realizar la instalación adecuada del entorno, se debe tener el programa 7-Zip para poder extraer archivos que están comprimidos. Para la instalación se deben seguir los siguientes pasos:

1. Instalar 7-Zip:
  - 1.1. Descargar el programa 7-Zip<sup>7</sup>.
  - 1.2. Hacer click-derecho en el archive descargado y seleccionar “Ejecutar como administrador”.
  - 1.3. Hacer click en “Siguiente” y seguir las instrucciones en el programa de instalación.
2. Descargar el instalador del IDE Eclipse<sup>8</sup>.
3. Usando 7-Zip, extraer el archivo instalador. Hacer click derecho y seleccionar “Extract Here”.

---

<sup>7</sup> <http://www.7-zip.org/download.html>

<sup>8</sup> [LINK DE MEGA DEL ECLIPSE](#)



### *Ejecutar Eclipse*

Una vez que los archivos son extraídos, navegar hasta el directorio creado en el paso anterior: `iot-ide-win`. Hacer doble click en el archivo “**devkit-launcher.bat**” para ejecutar el Eclipse.

**Nota:** usando el archivo “.bat” (en vez del ejecutable de Eclipse) se ejecutará el Eclipse con todas las configuraciones del entorno de desarrollo necesarios. Usar siempre “devkit-launcher.bat” para ejecutar el Eclipse.

La Figura X muestra la ventana para la selección del espacio de trabajo, lugar donde se guardarán todos los proyectos creados desde el entorno Eclipse. Se puede seleccionar cualquier carpeta del sistema de archivos.

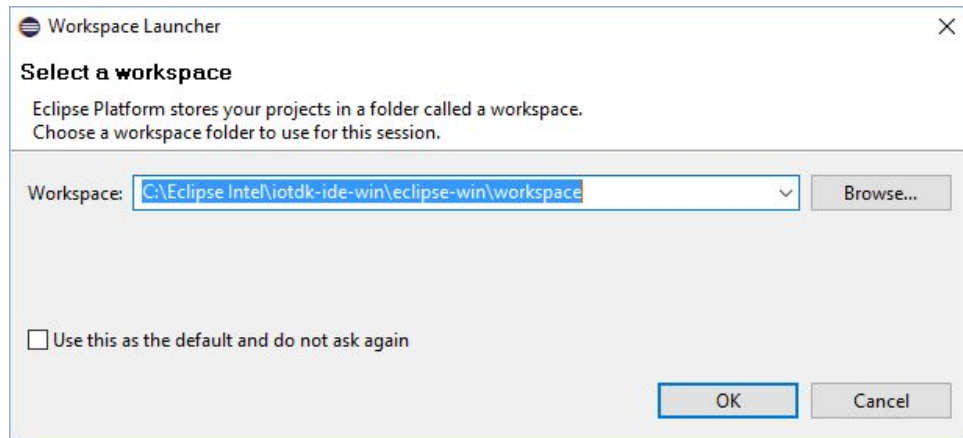
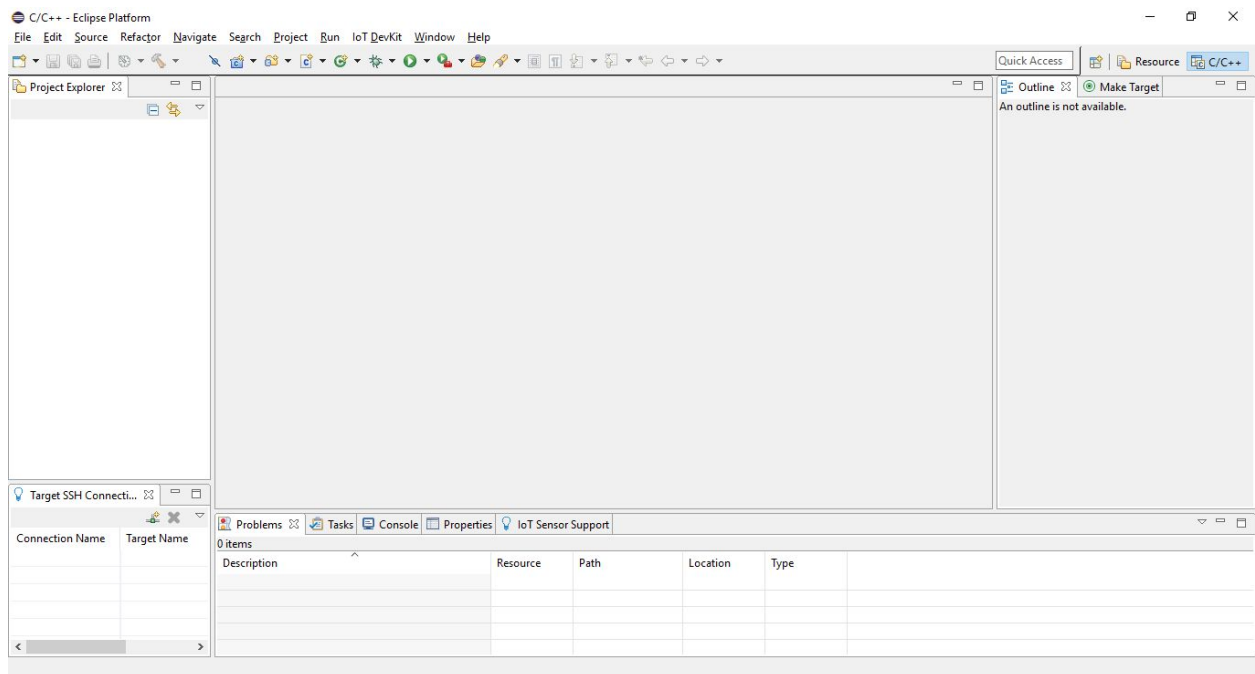


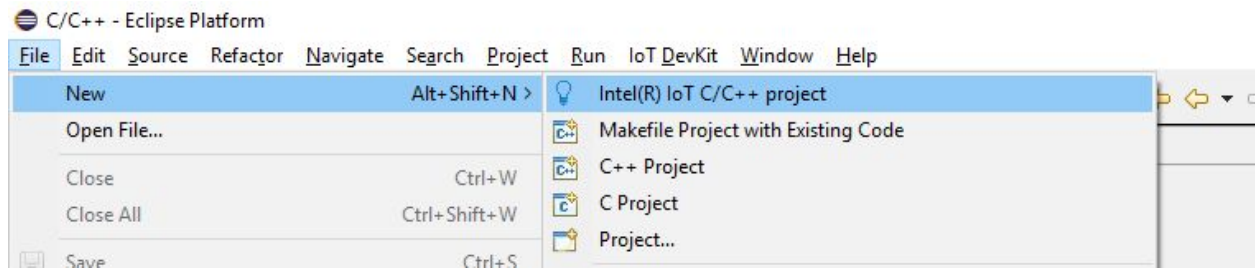
Figura X: Selección del espacio de trabajo.

### *Ejemplo del LED parpadeante (Blinking LED)*

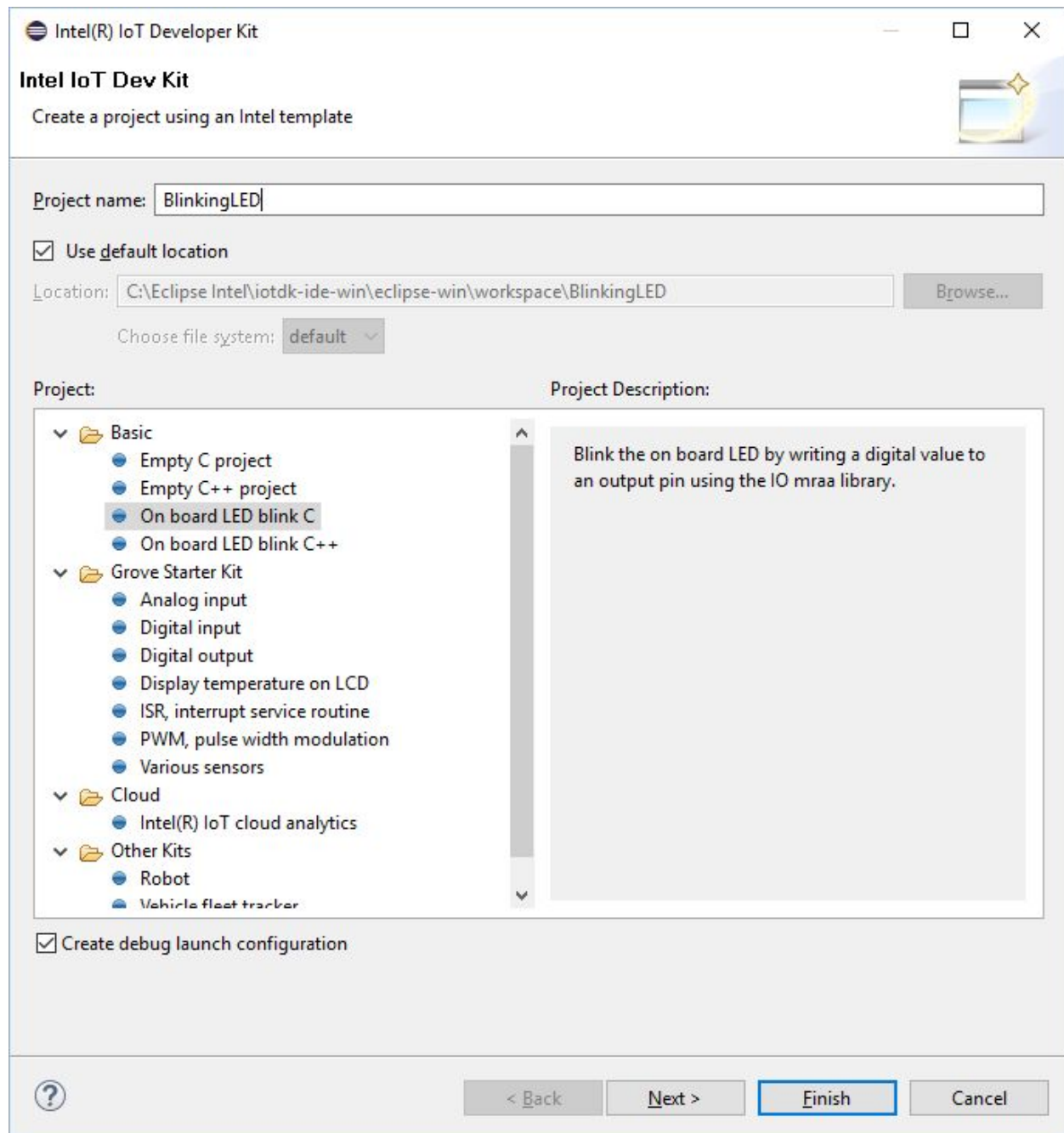
Ejecutar el archivo “devkit-launcher.bat” y se abrirá el entorno de Eclipse.



En el menu “File”, hacer click en “New” y en la opción que dice “Intel(R) IoT C/C++ project”.



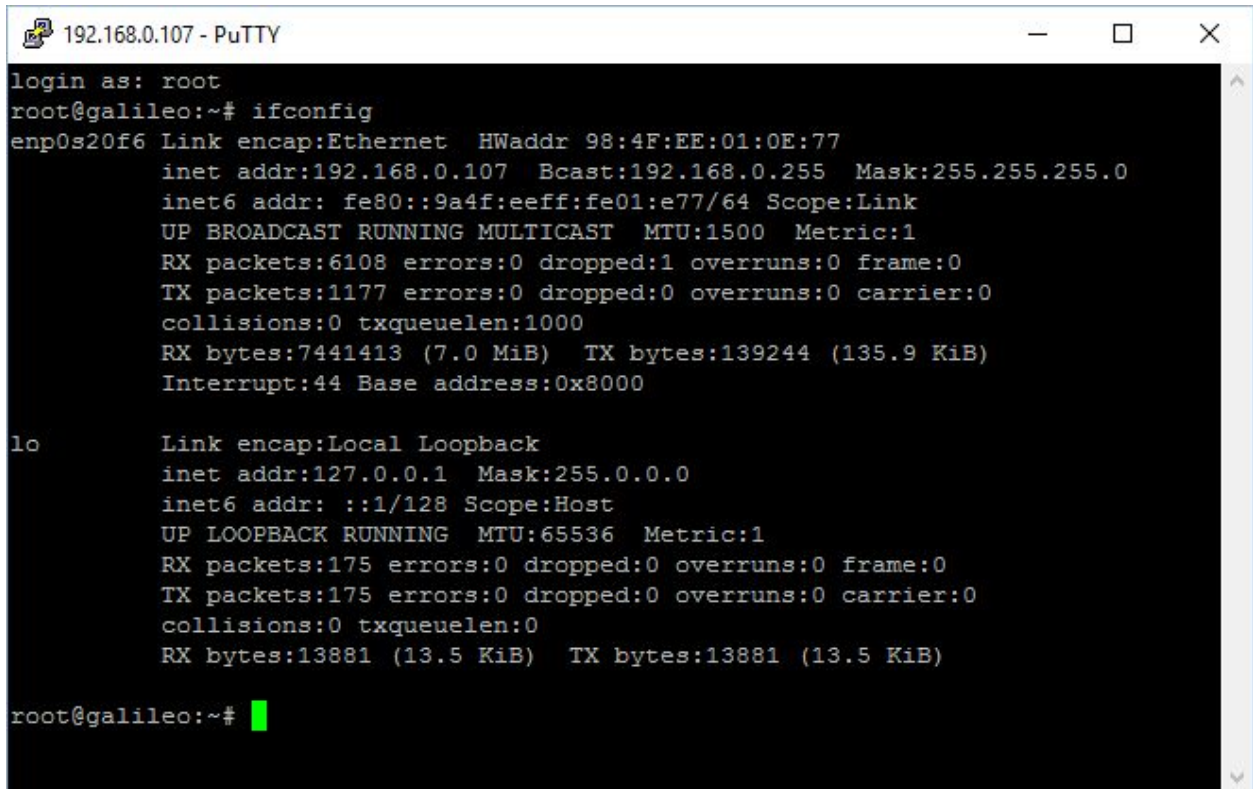
Luego, se abrirá una ventana que le pedirá un nombre para el proyecto y seleccionar un proyecto ejemplo. En este caso, seleccionar “On board LED blink C”. Hacer click en “Next”.



Ahora hay que configurar la conexión con la placa. Para ello, se debe saber cuál es la dirección IP que posee la placa. Hay que asegurarse que la placa esté conectada a internet a través de Ethernet y esté dentro de la misma red que la PC donde se está desarrollando.



Para obtener la dirección IP, se debe conectar el cable “Serie Jack(3.5mm) a USB” y realizar una conexión serie con la placa. Se puede utilizar el programa PuTTY y realizar los pasos en el Paso 5 “Comunicación” de este mismo apartado. Una vez que se conectó, ingresar a la placa con el usuario y contraseña y ejecutar el comando “ifconfig”, el cual muestra la información de las interfaces de la placa. La que interesa para este ejemplo, y en general, es la “inet addr” de la interfaz “enp0s20f6”. En este caso, la dirección IP es: 192.168.0.107.



```
192.168.0.107 - PuTTY
login as: root
root@galileo:~# ifconfig
enp0s20f6 Link encap:Ethernet  HWaddr 98:4F:EE:01:0E:77
        inet addr:192.168.0.107  Bcast:192.168.0.255  Mask:255.255.255.0
        inet6 addr: fe80::9a4f:eeff:fe01:e77/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:6108 errors:0 dropped:1 overruns:0 frame:0
        TX packets:1177 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:7441413 (7.0 MiB)  TX bytes:139244 (135.9 KiB)
        Interrupt:44 Base address:0x8000

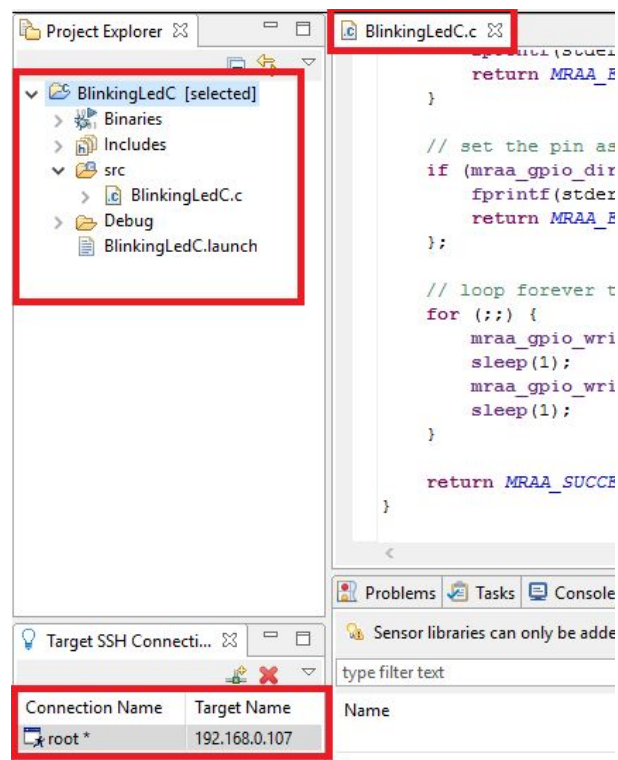
lo        Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:175 errors:0 dropped:0 overruns:0 frame:0
        TX packets:175 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:13881 (13.5 KiB)  TX bytes:13881 (13.5 KiB)


root@galileo:~#
```

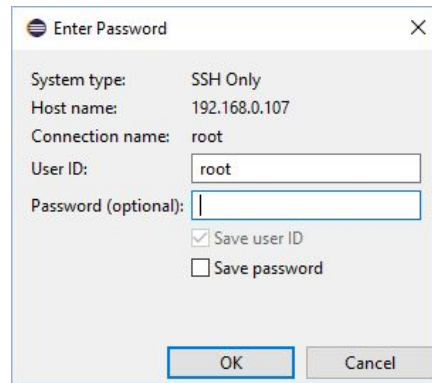
En donde dice “Connection Name” poner “root” y en la solapa “Target Name” poner la dirección IP encontrada. Hacer click en “Finish”.



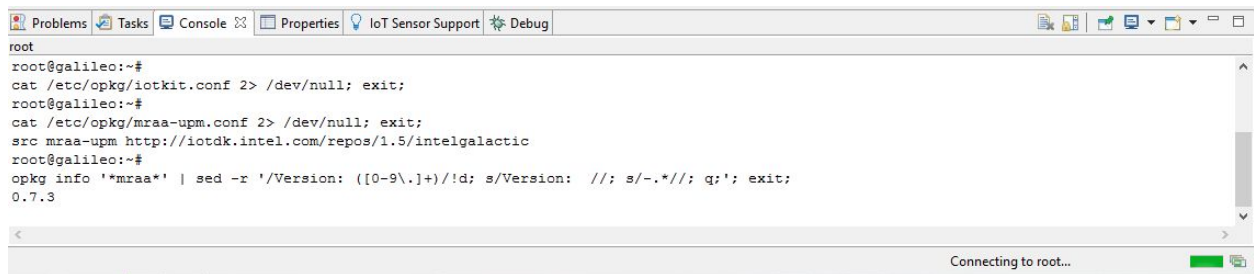
Finalmente, aparece en la parte de “Project Explorer” el proyecto completo con sus respectivas carpetas, debajo de éste aparece los parámetros de la conexión. En el centro de la pantalla aparecerá el editor para poder programar.



Finalmente, para ejecutarlo hay que hacer click en el ícono  que representa el “debug” de la aplicación. Aparecerá la siguiente pantalla con los datos de la conexión y con un usuario y contraseña (por defecto, Usuario: root, Password no posee). Hacer click en “Ok”.

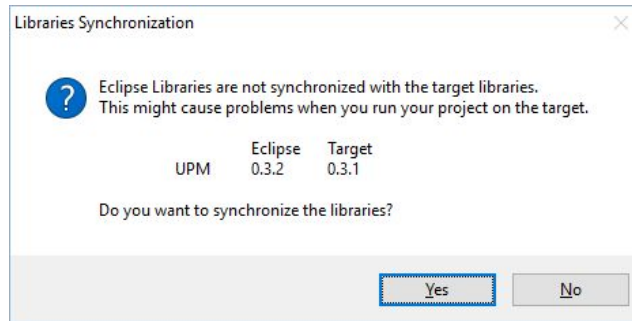


Luego en la parte de la consola, se visualizará los comandos que va ejecutando el Eclipse sobre la placa.

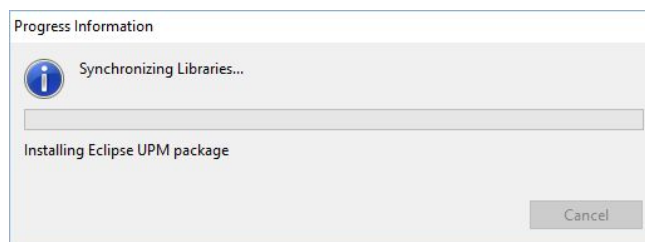


```
root
root@galileo:~#
cat /etc/opkg/iotkit.conf 2> /dev/null; exit;
root@galileo:~#
cat /etc/opkg/mraa-upm.conf 2> /dev/null; exit;
src mraa-upm http://iotdk.intel.com/repos/1.5/intelgalactic
root@galileo:~#
opkg info '*mraa*' | sed -x '/Version: ([0-9\\.]+)!!d; s/Version: //; s/-.*//; q; exit;
0.7.3
```

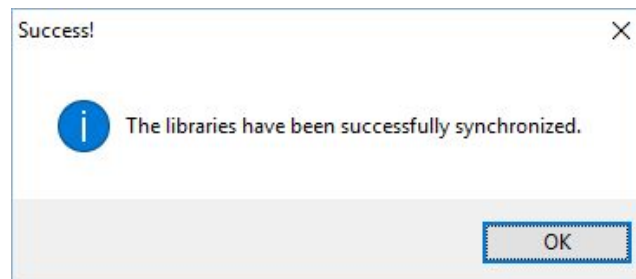
En el caso de que aparezca cuadros como la figura siguiente, quiere decir que es necesario sincronizar las librerías “UPM” y “MRAA” que posee el Eclipse y la placa. Para un correcto funcionamiento, se recomienda hacer click en “Yes”.



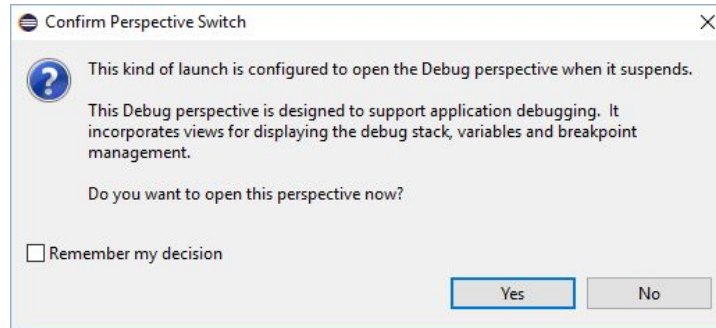
Este proceso mostrará una ventana como la figura que sigue y actualizará las librerías en la placa.



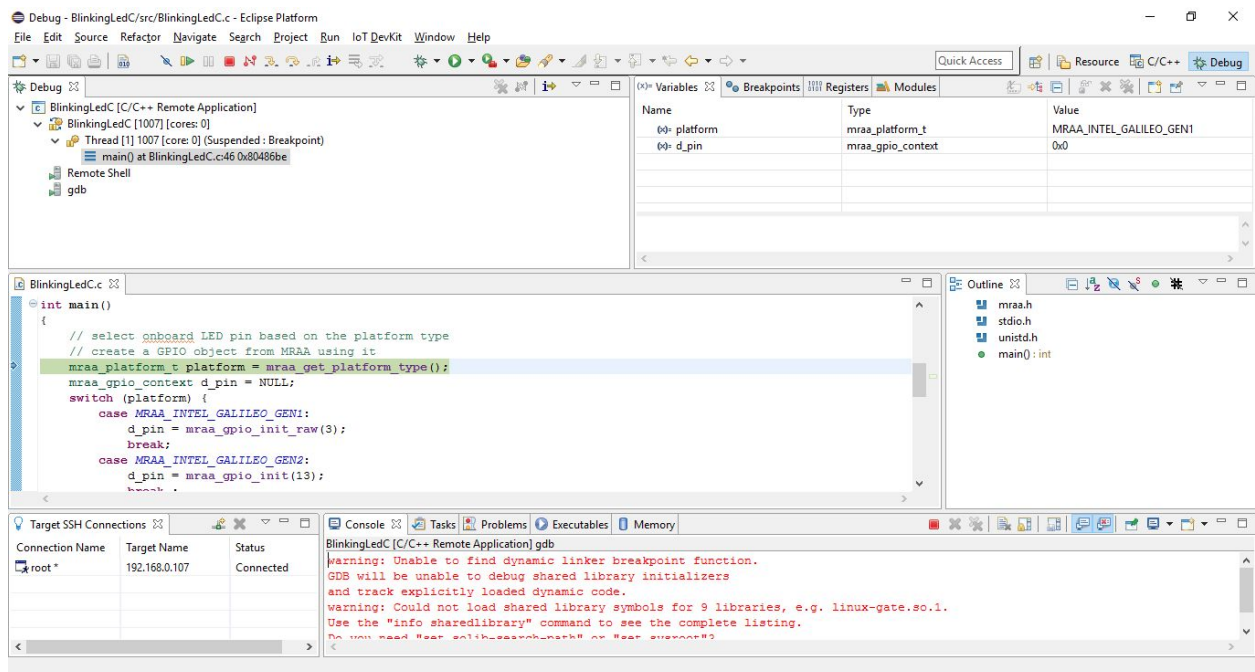
Una vez terminado, si todo se realizó de manera correcta, aparece un mensaje de “éxito”.



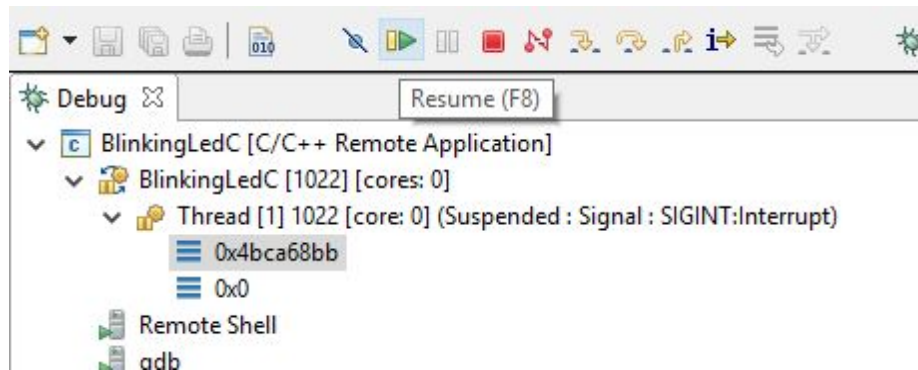
Cuando se haga click en OK, el Eclipse mostrará un cuadro de diálogo que pregunta si el usuario quiere cambiar la perspectiva del Eclipse para hacer el “Debug”. Hacer click en “Yes”



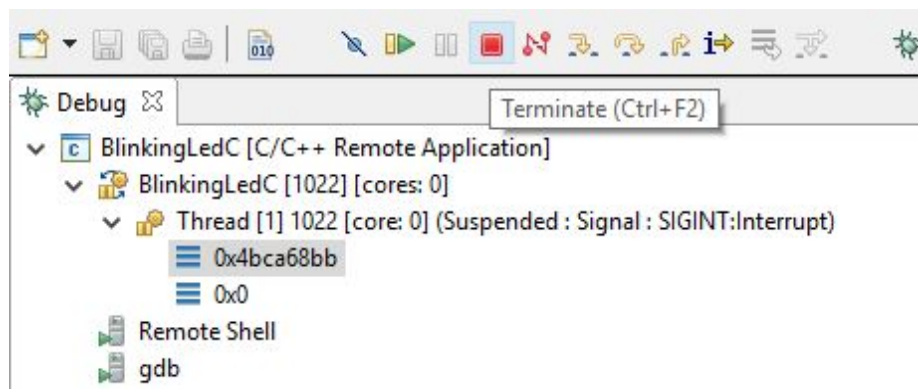
Ahora, el Eclipse tiene una perspectiva con la información del programa, los registros, las variables, los breakpoints y demás herramientas para poder “debuguear” el programa escrito.



Para ejecutar el programa, hacer click en el botón “Resume (F8)”. A partir de ese momento, verá en la placa como parpadea el LED.



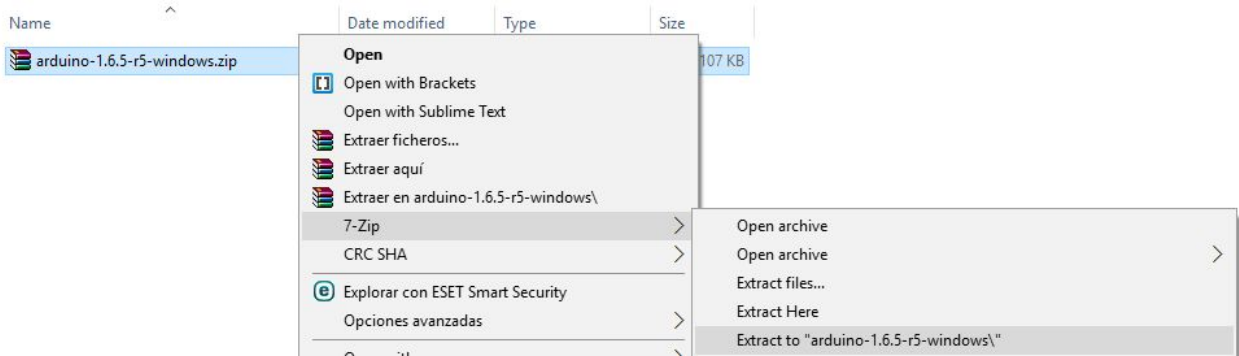
Para finalizar la ejecución del programa, hacer click en el botón “Terminate (Ctrl+F2)”.



## Arduino

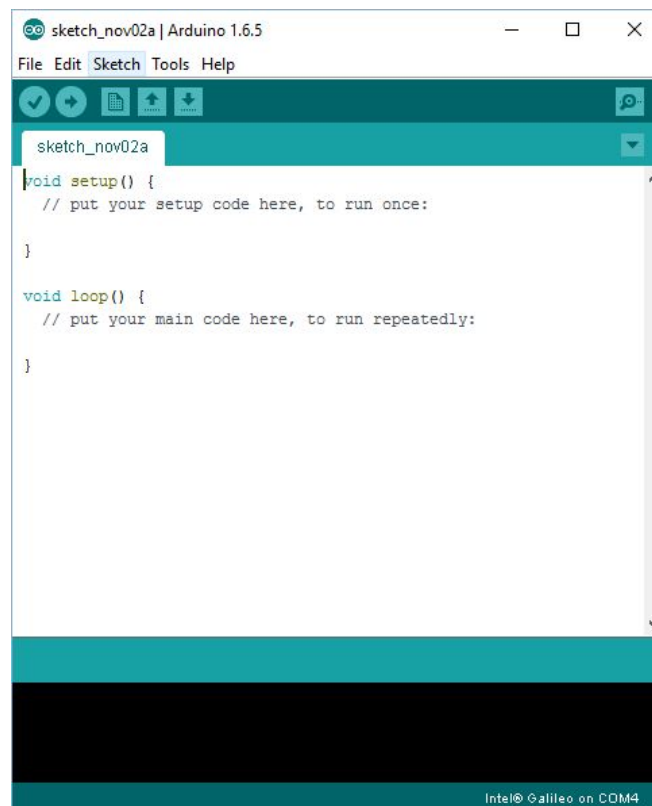
Para instalar el entorno de Arduino, primero hay que descargarlo<sup>9</sup>. Navegar en las carpetas hasta donde se descargó el IDE de Arduino en el archivo “.zip”. Hacer click derecho en el archivo .zip, apuntar “7-Zip” y selección “Extract to Arduino-...”.

<sup>9</sup> PONER LINK DEL INSTALADOR DE ARDUINO



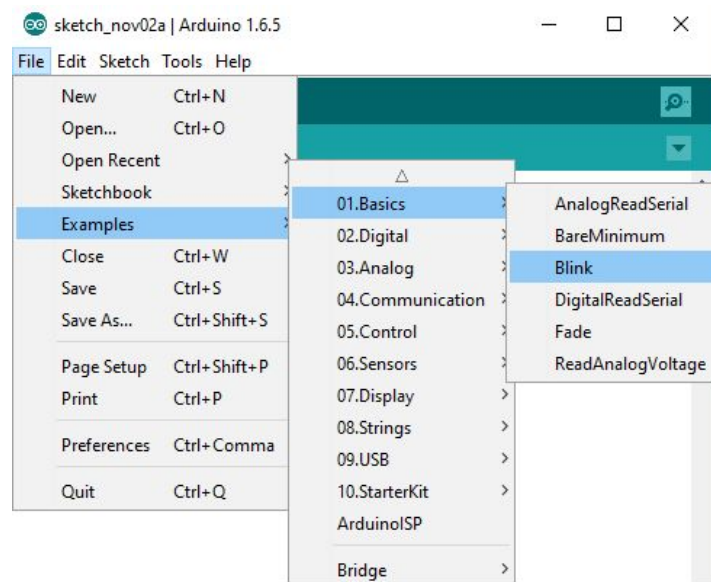
Abrir la carpeta extraída. Si se desea, se puede mover la carpeta a un directorio diferente. Un lugar común es creando una carpeta en “C:\Arduino”.

Para ejecutar el entorno, hacer doble click en el archivo “Arduino.exe” y abrirá la siguiente ventana:



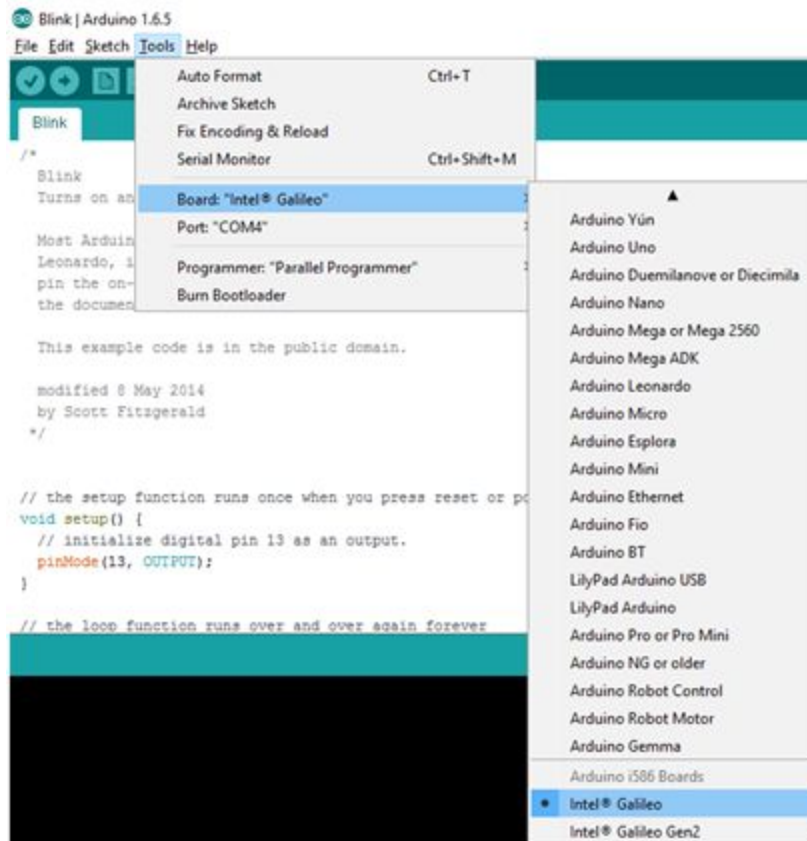
### *Ejemplo del LED parpadeante (Blinking LED)*

Para ejecutar el ejemplo del Blinkin LED, se debe hacer click en “File”, seguido de “Examples”, “01-Basics” y, finalmente, hacer click en “Blink”.

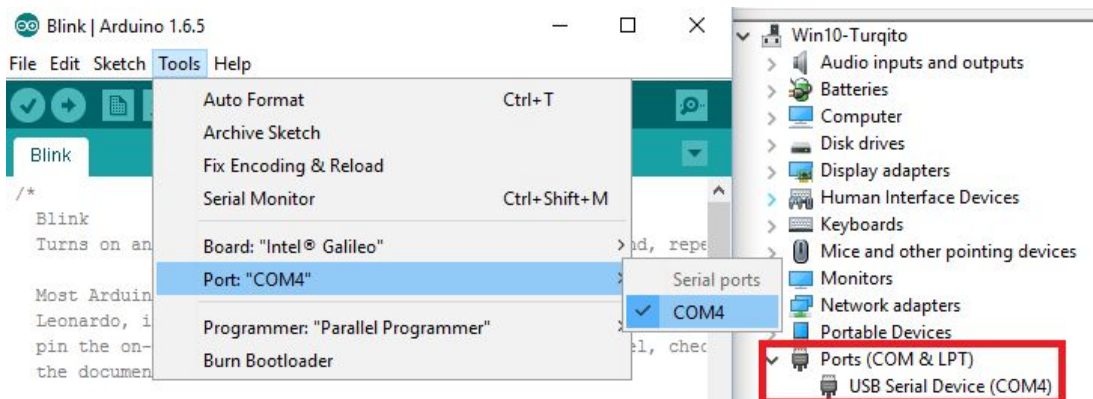


Para poder transferir el programa ejemplo a la placa hay que conectar la placa mediante el cable “mini USB – USB” desde el puerto “USB-Client” de la Galileo. Una vez conectada, hacer click en “Tools” y después en la solapa “Board”. Buscar hasta encontrar la placa “Intel Galileo”.





La transferencia se realiza a través del puerto de comunicaciones y, por lo tanto, hay que ver cuál es el puerto COM que pertenece a la placa. Para ello, se debe ingresar al “Administrador de dispositivos” y, en la solapa de “Ports (COM & LPT)”, verificar que se encuentre el siguiente puerto:



Finalmente para ejecutar el programa en la placa, primero se debe hacer click en el ícono



que verifica que el código esté correctamente escrito. Como paso

final, hacer click en el ícono  para subir el ejemplo a la placa.



Ingresar y descargar el instalador<sup>10</sup> del software correspondiente al sistema operativo.

En ese momento, en la consola, comenzará la transferencia y la ejecución de los comandos para que la placa corra el Sketch de Arduino.

### ***Entorno de desarrollo Intel® XDK IoT Edition***

El entorno de desarrollo Intel XDK IoT es una solución completa para crear y probar aplicaciones en las plataformas de Intel IoT. Este entorno provee componentes hardware y software para que los desarrolladores puedan crear proyectos usando la placa y el Grove – Starter Kit (conjunto de herramientas diseñadas para minimizar la dificultad de utilizar componentes electrónicos, ver **Figura X**).

---

<sup>10</sup> <https://software.intel.com/en-us/iot/software/ide>



crear las aplicaciones en su lenguaje de programación favorito. Esta versión incluye compilador GCC, Python, Node.js y OpenCV, entre otros.

- El sistema Yocto instalado en la placa Intel Galileo provee un conjunto de librerías específicas especialmente diseñadas para el “Intel IoT Developer Kit” y el entorno de desarrollo Intel XDK IoT Edition. Estas librerías son las mencionadas MRAA y UPM.

### Instalación de entorno de desarrollo Intel® XDK IoT Edition

1. Ingresar y descargar el instalador del Intel IoT XDK Edition<sup>11</sup> correspondiente al sistema operativo.
2. Abrir el instalador con permisos de administrador, si aparece un mensaje de confirmación hacer click en *yes*.
3. Instalar *Bonjour Print Services* (Solo Windows).
  - 3.1. Este programa habilita al *XDK IoT Edition* detectar automáticamente los dispositivos instalados en la red.
  - 3.2. Descargar *Bonjour*.<sup>12</sup>
  - 3.3. Click en *Download*.
  - 3.4. Click derecho sobre *BonjourPSSetup.exe* y luego seleccionar **Ejecutar como administrador**. Si un mensaje de configuración aparece, hacer click en **YES** para continuar.
  - 3.5. Seguir las instrucciones del asistente de instalación para instalar *Bonjour*.

---

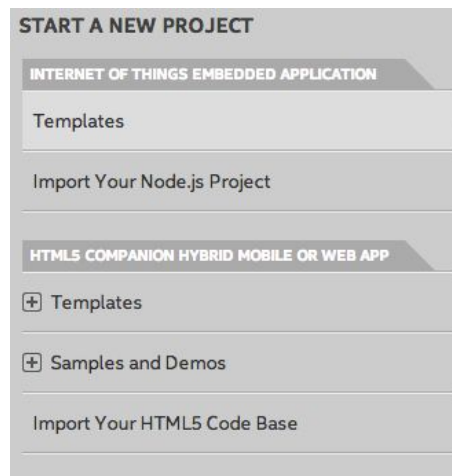
<sup>11</sup> <https://software.intel.com/en-us/iot/software/ide>

<sup>12</sup> <http://support.apple.com/kb/DL999>

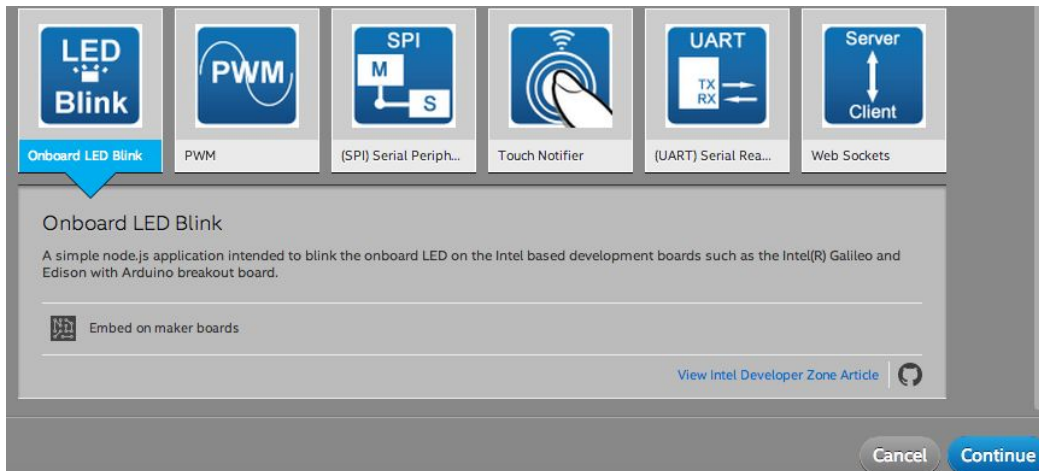


*Ejemplo del destello de un led*

Dentro de la pestaña *Start a new project* se selecciona el ítem *Templates* como muestra la siguiente figura:



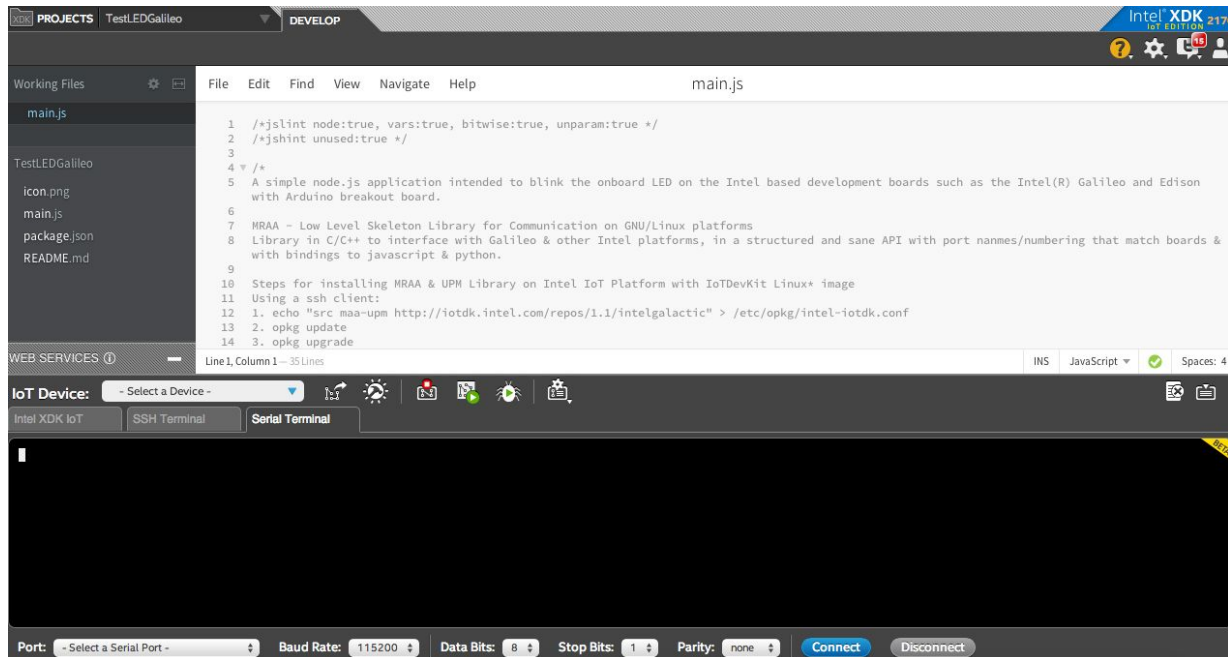
Luego se elige la opción *Onboard led blink* y se hace click click en *Continue*.



Elegir el nombre del proyecto y la ruta donde queremos guardarlo.

A screenshot of the 'New Project Name and Location' dialog box. The dialog has a blue header with the title 'New Project Name and Location'. Below the header, there are two input fields. The first is labeled 'Project Directory' and contains the text '/Users/Bezzi'. The second is labeled 'Project Name' and contains the placeholder text 'Enter a working name for your project.'. At the bottom right of the dialog, there are two buttons: 'Cancel' and 'Create'.

Luego se mostrará el entorno listo para programar:



En esta etapa se conecta la placa por puerto serie y ethernet.

### *Conexión Serial:*

Luego de conectar el cable serie a la placa se elige en el menú contenedor *Port* en la esquina inferior izquierda de la pantalla, el dispositivo serie que representa la placa, además, se deberá configurar el *Baud Rate* en 115200, *Data Bits* en 8, *Stop Bits* en 1 y *Parity* en 0.

Finalmente se presiona el botón *Connect* (se recomienda presionar el botón *Reboot* de la placa para que inicie el sistema operativo).



En la terminal se verá el *boot* del sistema operativo y la posibilidad de iniciar sesión con el usuario *root*.

### *Conexión Ethernet*

Para este paso simplemente se conecta un cable Ethernet (Rj-45) a la placa y el otro extremo a un *access point* disponible sin la necesidad de una configuración adicional.

Una vez conectado el cable, para ver la dirección IP asignada a la placa deberá ingresar el comando de *ifconfig* para luego realizar la conexión IoT posteriormente explicada.

```
root@galileo:~# ifconfig
enp0s20f6 Link encap:Ethernet HWaddr 98:4F:EE:01:0E:77
          inet addr:192.168.0.117 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::984f:eeff:fe01:e77/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:46801 errors:0 dropped:11 overruns:0 frame:0
          TX packets:14646 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:52940637 (50.4 MiB) TX bytes:1268945 (1.2 MiB)
          Interrupt:43
```

### Configuración Intel XDK IoT

Para cargar los programas que deba ejecutar necesitará establecer una conexión con la placa a través de Ethernet.

Para ello en el menú contenedor *IoT Device* se elige la opción *Add manual connection* que desplegará una nueva ventana como indica la siguiente figura:

Connect to IoT Device (must be running the Intel XDK app daemon)

Address: 192.168.2.15 (ex: 192.168.1.104)

Port: 58888 (ex: 58888) Default Intel XDK app daemon port is 58888

☐ Use ssh keys

User Name: root ?

Password: ?

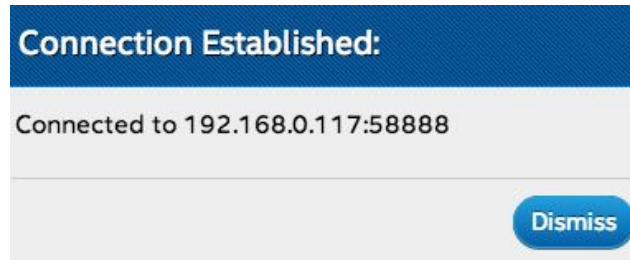
[Why is my device not auto-detected?](#)

Cancel Connect



Dentro del campo *Address* se coloca la IP previamente obtenida a través del comando *ifconfig*, luego en los campos *User* y *Password* se deberá ingresar el usuario y contraseña siendo el usuario *default root* y sin contraseña.

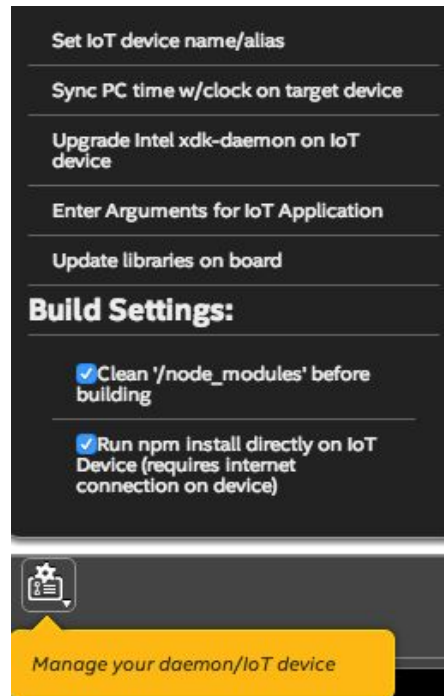
Al botón *Connect* aparecerá un cartel indicando que la conexión fue establecida:



#### *Compilación, carga y ejecución del programa*

Para esta etapa se deberán seguir los siguientes pasos:

En botón *Manage your daemon/IoT device* se tildan las opciones de *Clean* y *Run* de ambos *checkbox* contenidos en *Build Settings*. También deberá hacer click sobre la opción *Sync PC time w/clock on target device* y esperar unos segundos.



Luego dentro de la terminal serie ejecutar los siguientes comandos para la instalación de las librerías MRAA & UPM que luego se utilizarán.

1. `echo "src maa-upm http://iotdk.intel.com/repos/1.1/intelgalactic" > /etc/opkg/intel-iotdk.conf`
2. `opkg update`
3. `opkg upgrade`

Para el funcionamiento del led *onboard* deberá comentar la línea de código 22 y descomentar la línea de código 23, resultando de la siguiente manera:

```
var myOnboardLed = new mraa.Gpio(3, false, true);  
//var myOnboardLed = new mraa.Gpio(13);
```

El código está listo para ser compilado y cargado en la placa.

Para compilar se debe presionar el botón *install/Build* y esperar unos instantes hasta que se muestre el resultado en consola.



The screenshot shows the Intel IoT Device Manager interface. At the top, there's a dropdown menu for 'IoT Device:' set to 'Manual Connection (192.168.0.117:58888)'. Below it are tabs for 'Intel XDK IoT', 'SSH Terminal', and 'Serial Terminal'. The main terminal area displays the following text:

```
MRAA Version: v0.7.2
UPLOADING: Uploading project bundle to IoT device.
[ Upload Complete ]
Intel XDK - Message Received: install

=====
Intel (R) IoT - NPM Install - (may take several minutes)
=====
npm WARN package.json OnboardLEDBlink@0.0.0 No repository field.
=====
NPM INSTALL COMPLETE! [ 0 ] [ 0 ]
=====
```

Una vez realizado esto, se presiona el botón *Upload* para cargar el programa en la placa y finalmente se presiona el botón *Run* para la ejecutar el programa en la placa.

## Windows Embbebed for IoT en Intel Galileo

### Instalación

Para poder instalar “Windows Embbebed for IoT” se deben seguir los pasos siguientes:

1. Descargar la carpeta “Galileo Win” que posee los siguientes archivos<sup>13</sup>
  - “9600.16384.x86fre.winblue\_rtm\_iotbuild.150309-0310\_galileo\_v2.wim”: es el sistema operativo “Windows Embbebed for IoT”
  - “apply-BootMedia.cmd”: es el programa encargado de pasar los archivos a la tarjea SD.
  - “WindowsDeveloperProgramforIoT.msi”: este programa realiza la conexión entre la PC y la placa Intel Galileo.
2. Poner la tarjea SD dentro de la ranura para tarjetas externas. Formatearla con formato FAT32 y ver cuál es la ruta a esa tarjeta de memoria. Por ejemplo, F:\.
3. Abrir una consola de comandos como Administrador y dirigirse a donde se descargó la carpeta en el punto 1.

<sup>13</sup> [https://mega.nz/#F!vpAFDKYI!HzIHBwmnJ4RrJpFLBG\\_Qbw](https://mega.nz/#F!vpAFDKYI!HzIHBwmnJ4RrJpFLBG_Qbw)

```

C:\>cd Users\turco\Desktop\PPS\INFORME\GalileoWin
C:\Users\turco\Desktop\PPS\INFORME\GalileoWin>dir
Volume in drive C has no label.
Volume Serial Number is 9C68-BFB2

Directory of C:\Users\turco\Desktop\PPS\INFORME\GalileoWin

02/10/2015  11:04    <DIR>          .
02/10/2015  11:04    <DIR>          ..
04/09/2015  16:29         174.867.749 9600.16384.x86fre.winblue_rtm_iotbuild.150309-0310_galileo_v2.wim
02/10/2015  14:57             20.825 apply-BootMedia.cmd
                2 File(s)      174.888.574 bytes
                2 Dir(s)      24.511.029.248 bytes free

C:\Users\turco\Desktop\PPS\INFORME\GalileoWin>

```

4. Una vez en el directorio, ejecutar el programa “apply-BootMedia.cmd” de la siguiente manera:

- apply-BootMedia.cmd –destination f: –image  
9600.16384.x86fre.winblue\_rtm\_iotbuild.150309-0310\_galileo\_v2.wim
- Esto puede tardar varios minutos. Al finalizar se observarán los datos importantes como el usuario, la contraseña para ingresar a la placa.

```

Applying image
[=====100.0%=====]
The operation completed successfully.
**** Mounting f:\Windows\System32\config\SYSTEM
**** to HKEY_USERS\Galileo-10254-SYSTEM
**** Setting hostname to GALILEOPC-10254
**** Restoring time zone to 'Argentina Standard Time'
****
**** Successfully applied C:\Users\turco\Desktop\PPS\INFORME\GalileoWin\9600.16384.x86fre.winblue_rtm_iotbuild.150309-0310_galileo_v2.wim
**** to f:\
****
**** hostname: GALILEOPC-10254
**** timezone: Pacific Standard Time
**** Username: Administrator
**** Password: p27238
****
**** Done.

```

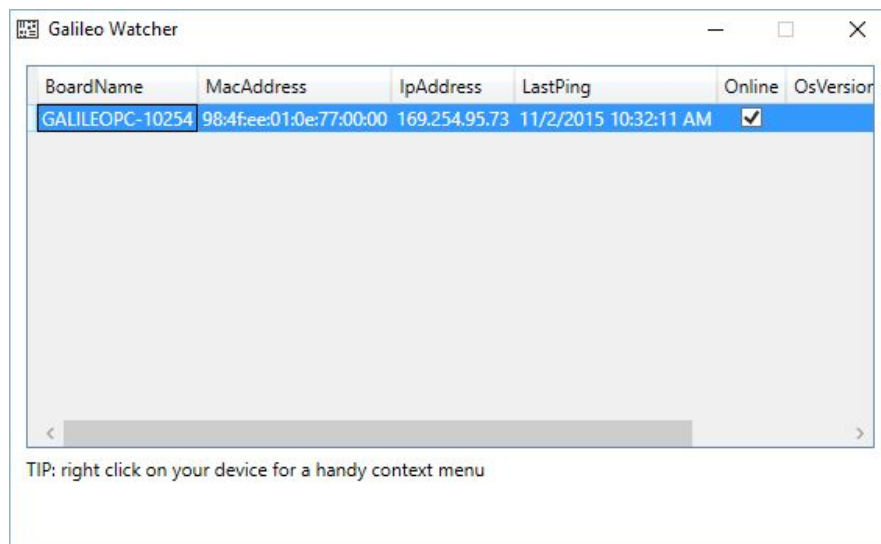
5. Ahora solo resta poner la tarjeta SD en la placa y encender para que pueda bootear con Windows Embbebed.

### Prueba de funcionamiento correcto

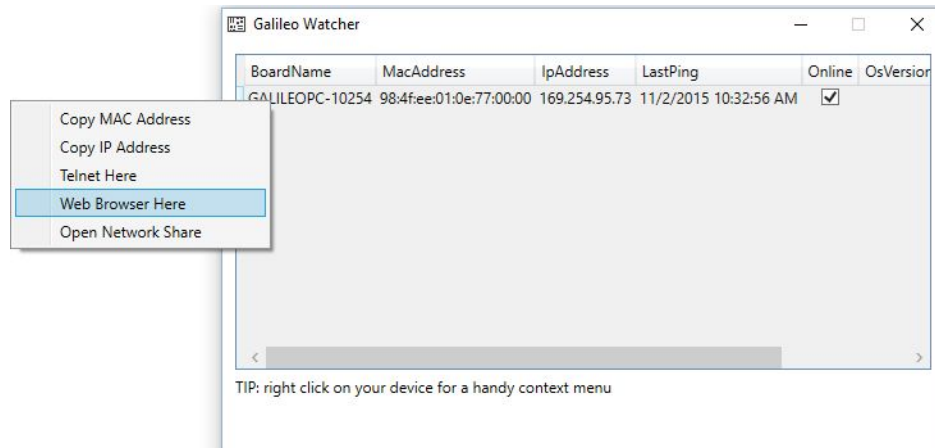
Para poder conectarse en la placa Intel Galileo con Windows Embbebed for IoT, se necesita el programa WindowsDeveloperProgramforIoT.msi: que ya se incluye en la carpeta “Galileo Win”.

Instalar el “WindowsDeveloperProgramforIoT.msi”. Cuando se haya completado la instalación, la herramienta “Galileo Watcher” será la encargada de buscar en la red a través de Ethernet para saber si se encuentra Intel Galileo.

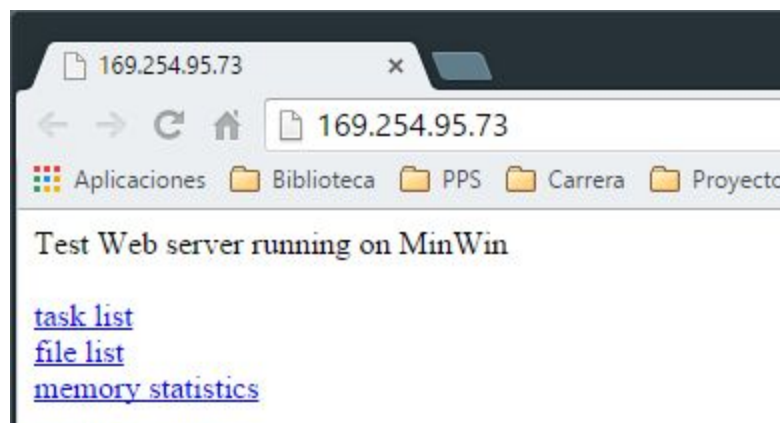
Una vez que la tarjeta SD haya sido puesta, conectar un cable Ethernet en un extremo en la placa Intel Galileo y en el otro extremo en la PC que se va a desarrollar y se encienda la placa. Luego, encender la placa y abrir el programa “Galileo Watcher”. En algunos segundos, aparecerán los datos de la placa como el nombre de la placa, su dirección MAC e IP, entre otras cosas.



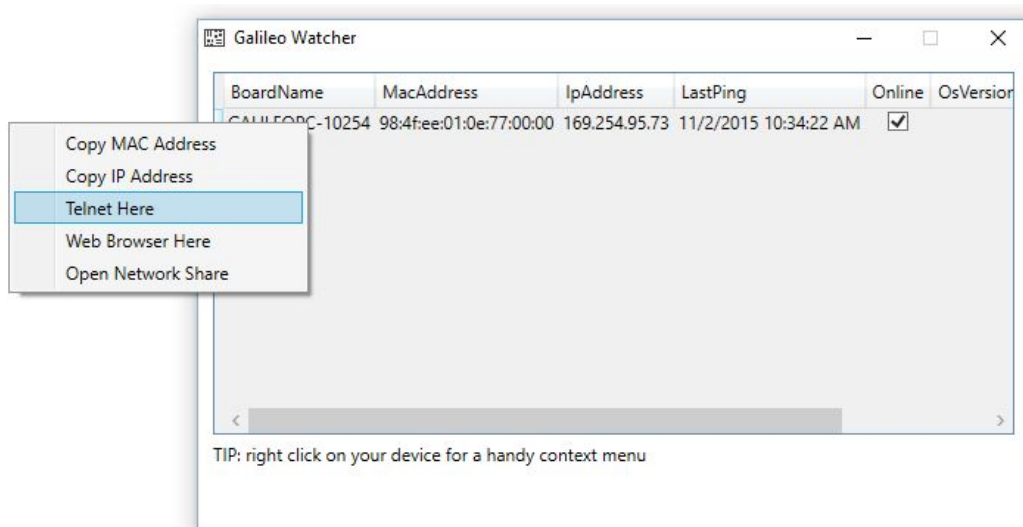
Para comprobar que la placa está funcionando de manera correcta, hacer click-derecho sobre la placa y presionar sobre “Web Browser Here”.



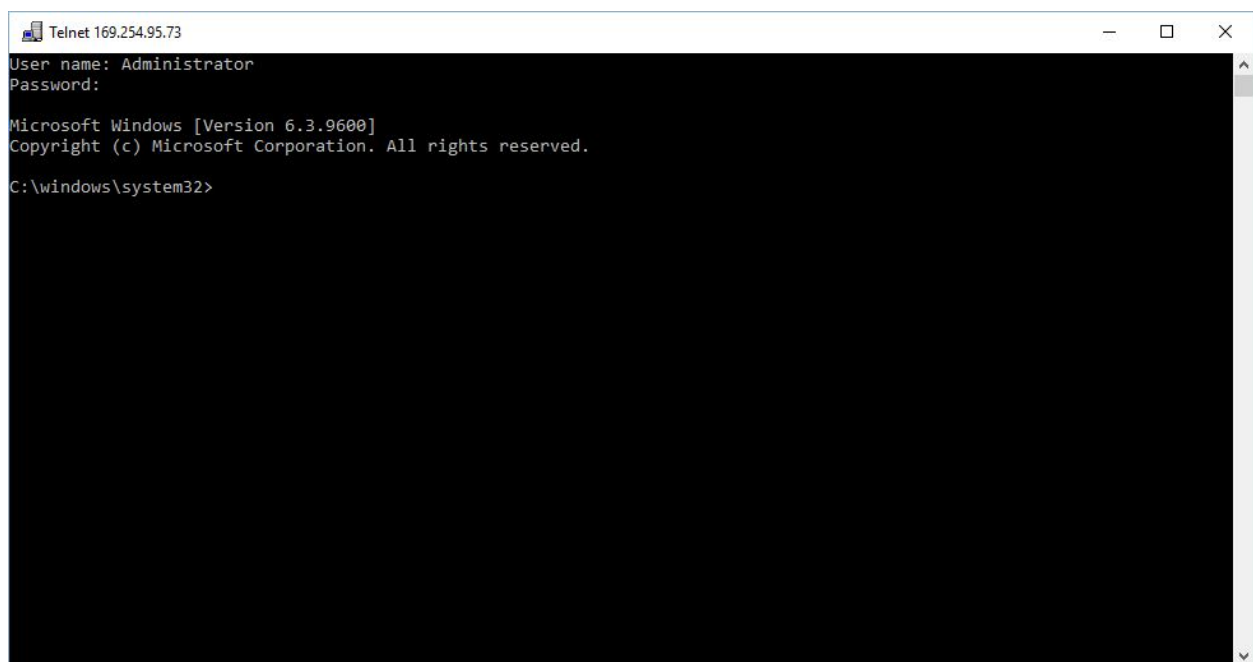
En este momento, se abrirá el navegador por defecto con información de la placa.



Si se quiere ingresar al sistema de archivos de la placa, hay que hacer click-derecho sobre la placa en el “Galileo Watcher” y presionar sobre “Telnet Here”.



Luego, hay que poner los datos que el “apply-bootmedia.cmd” otorgó para poder iniciar sesión.



Finalmente se puede ejecutar comandos Windows dentro de la placa. Los archivos dentro del directorio raíz son:

```

C:\windows\system32>cd C:\

C:\>dir
Volume in drive C is GALILEO
Volume Serial Number is 7241-726D

Directory of C:\

2015-03-11  08:16:36    <DIR>          efi
2015-03-09  10:47:10    <DIR>          Program Files
2015-03-11  08:17:20    <DIR>          Tools
2015-03-09  08:28:20    <DIR>          Users
2015-03-09  10:52:24    <DIR>          Windows
2015-03-09  06:07:18             3,310 BOOTEX.LOG
                1 File(s)              3,310 bytes
                5 Dir(s)  13,496,524,800 bytes free

C:\>

```

## Desarrollo (FALTA LA PARTE DE LA SIMULACION)

El entorno para desarrollar con la placa Intel Galileo con Windows Embbebed for IoT es:

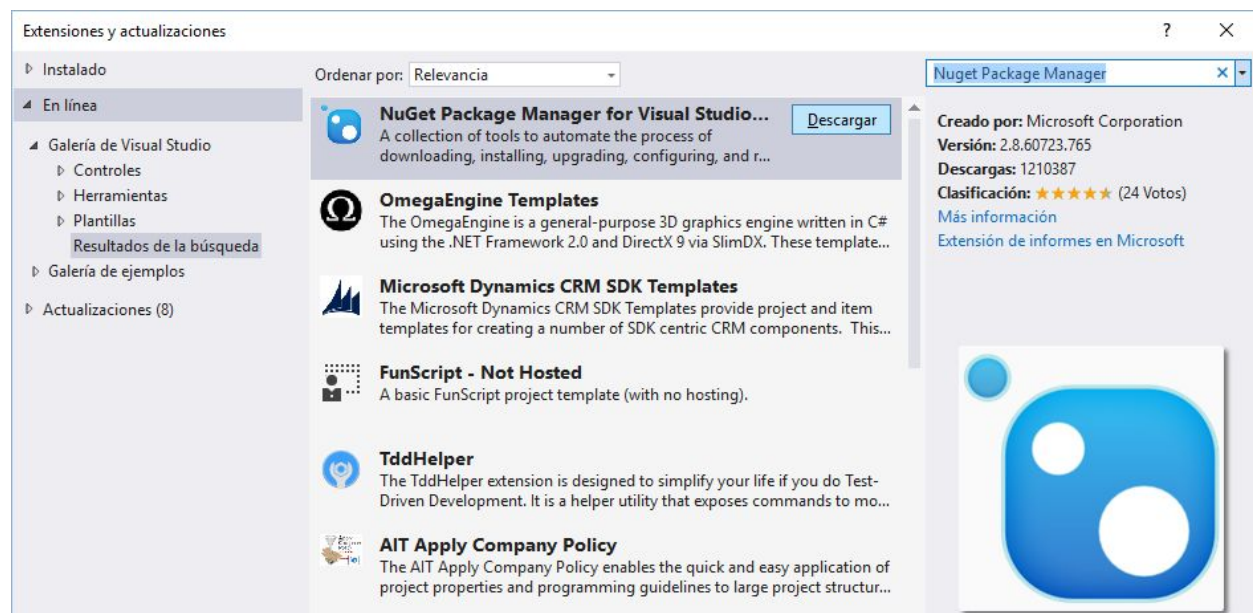
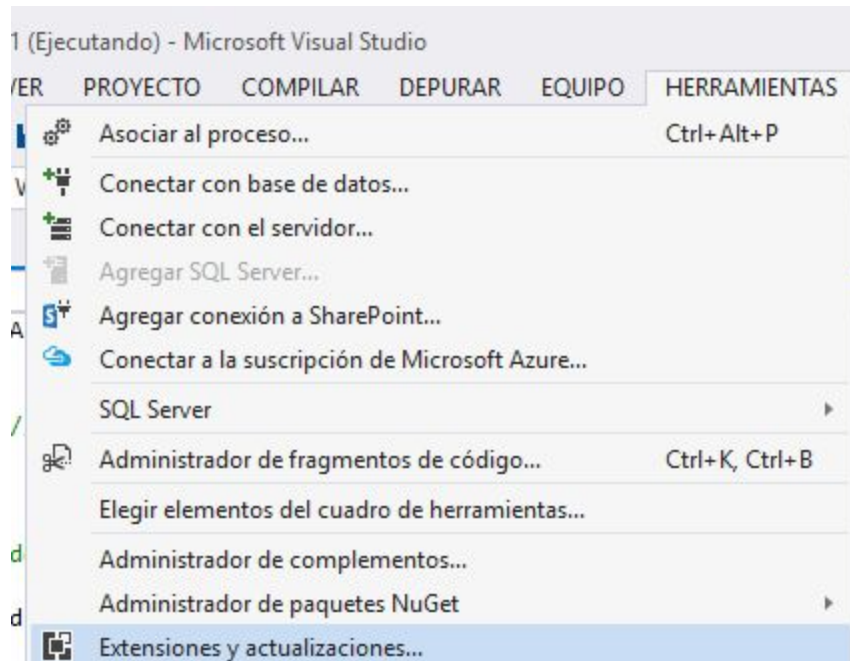
- Visual Studio Professional 2013: es un entorno de programación más para poder programar la placa Intel Galileo. En primera instancia se debería programar en C++. Igualmente, la idea que tiene este entorno hacer una traducción de C++ a Arduino. Es decir, cuando se comienza a programar se puede observar las dos funciones principales de Arduino “setup()” y “loop()” y el estilo de programación es en Arduino.

ATENCIÓN: ES EL ÚNICO ENTORNO DE DESARROLLO QUE SOPORTA A INTEL GALILEO CON WINDOWS EMBBEBED FOR IOT. NO DESCARGAR OTRA VERSIÓN QUE NO SEA CUALQUIERA DE LAS DISTRIBUCIONES PERO SI O SI QUE SEA LA 2013.

Descargar el Visual Studio 2013 Professional<sup>14</sup> e instalar. Este proceso suele tardar varios minutos ya que el entorno contiene todas las herramientas para poder desarrollar en Windows, Android, entre otras cosas.

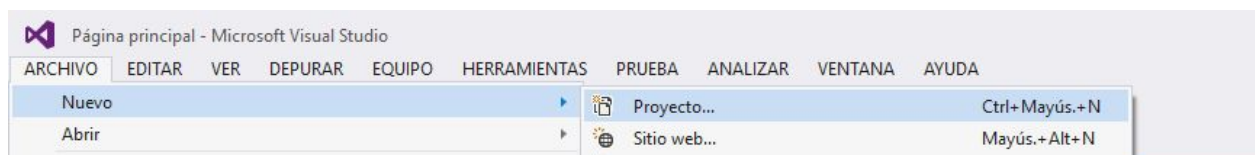
<sup>14</sup> <https://go.microsoft.com/fwlink/?LinkId=532509&clcid=0xc0a>



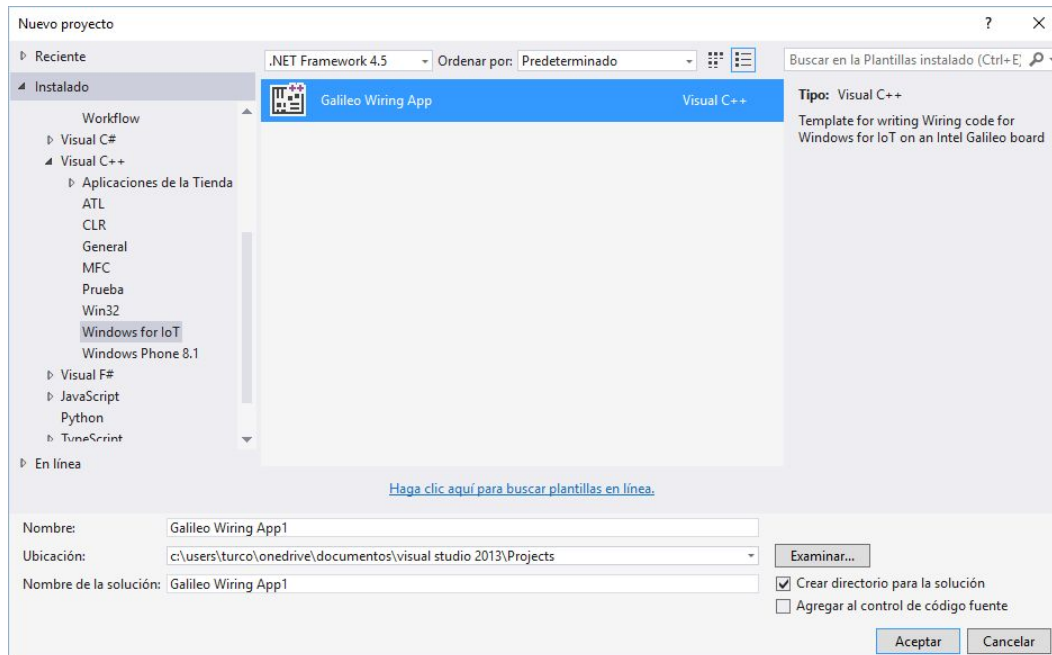




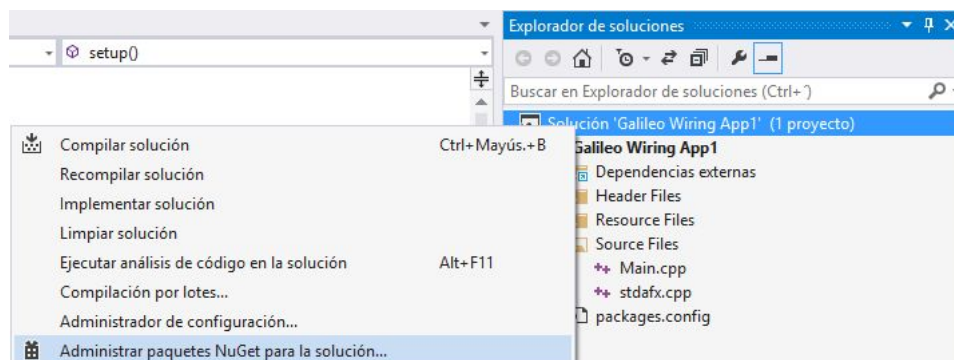
Cuando el programa hay sido instalado, para poder comenzar a programar, hay que hacer click sobre “Archivo” -> “Nuevo” -> “Proyecto”.

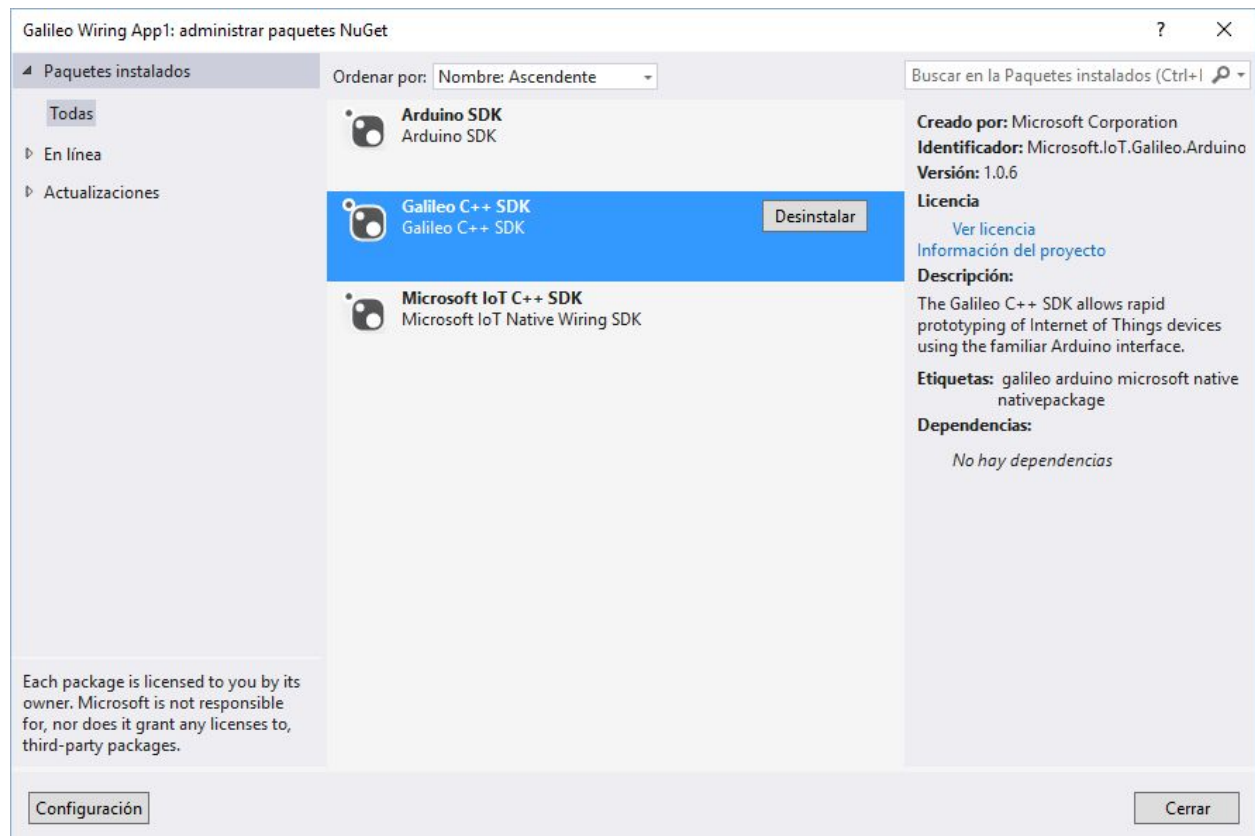


Dentro del apartado de “Instalado”, abrir la pestaña de “Visual C++” y hacer click donde dice “Windows for IoT”. Allí aparecerá una opción con el nombre de “Galileo Wiring App” y en el panel de abajo, habrá opciones para poner el nombre del proyecto, la ubicación y, lo que Windows denomina a los programas, el nombre de la solución.

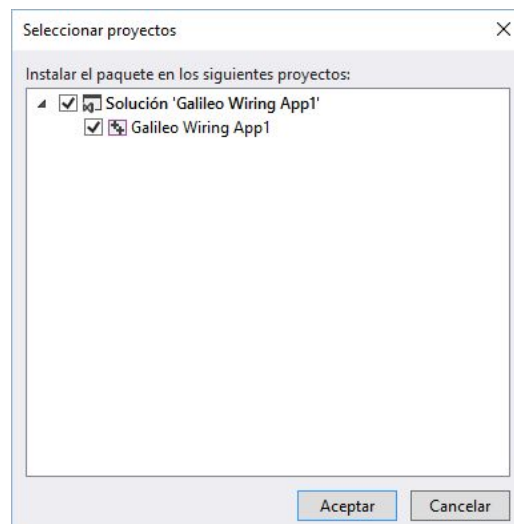


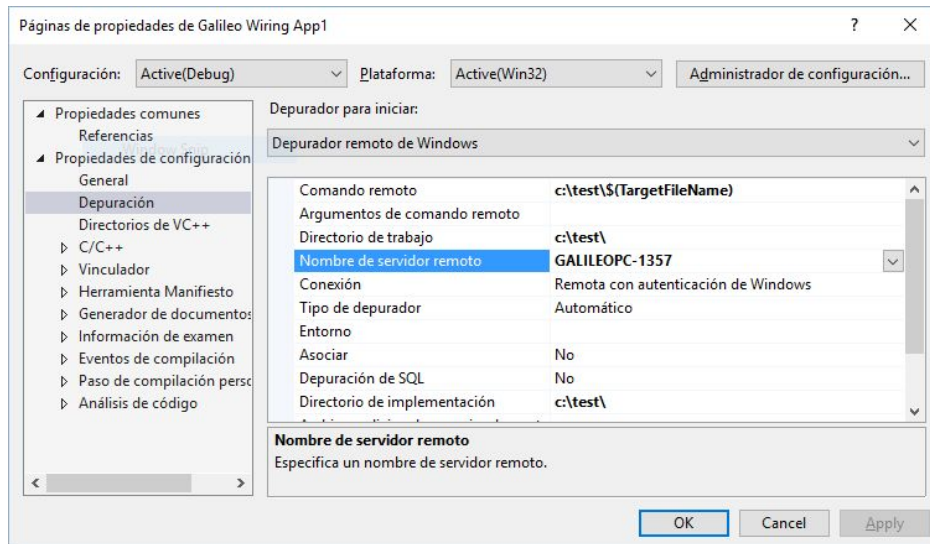
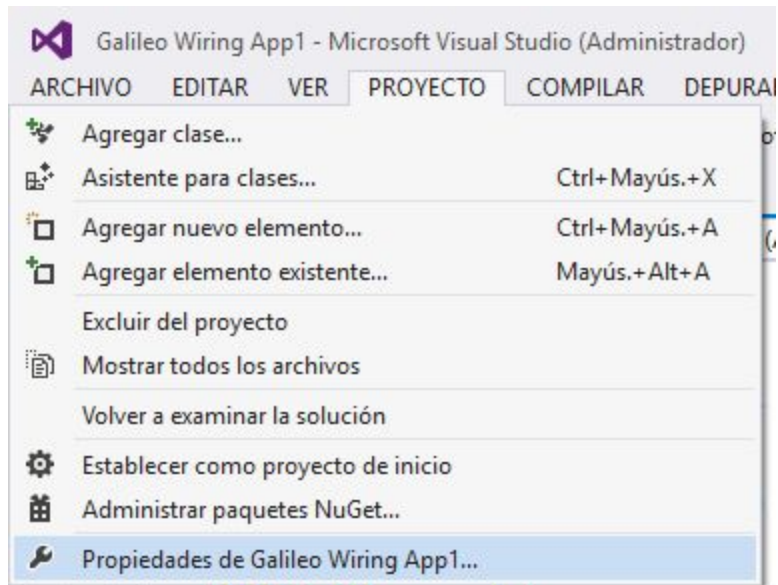
Luego, aparecerá el entorno con el proyecto ya creado y, por defecto, con el ejemplo de un destello de un LED.

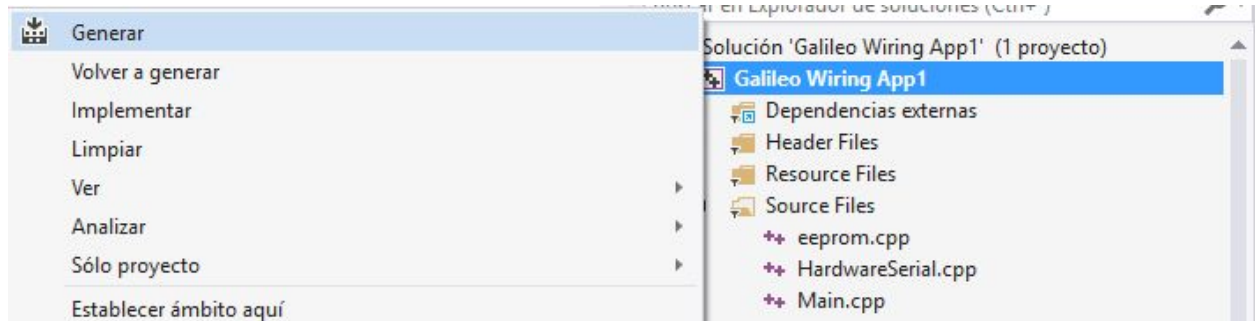




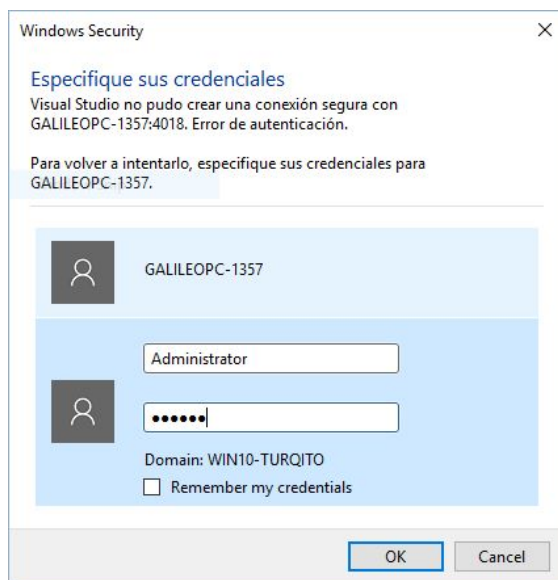
Tiene que desinstalar el que dice galileo y dejar los otros dos.



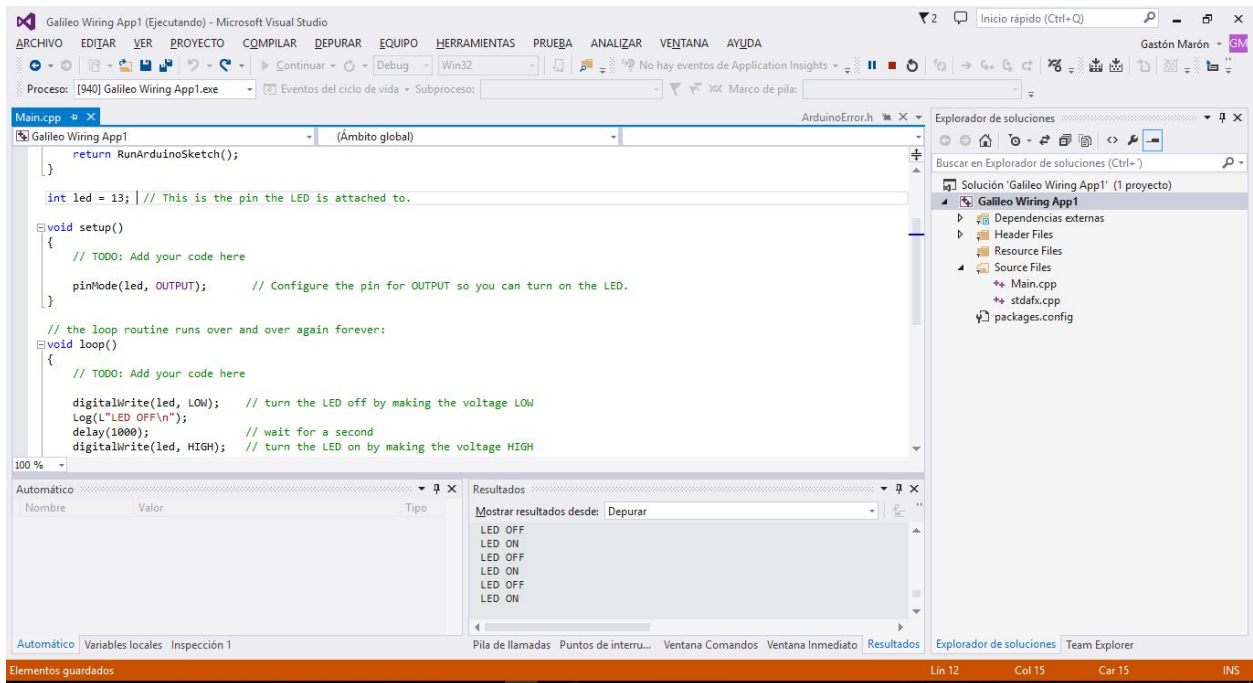




```
1>----- Operación Recompilar todo iniciada: proyecto: Galileo Wiring App1, configuración: Debug Win32 -----
1> Restoring NuGet packages...
1> To prevent NuGet from downloading packages during build, open the Visual Studio Options dialog, click on the Package Manager node a
1> All packages listed in packages.config are already installed.
1> WInterrupt.cpp
```







## Node.js

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor. Node.js implementa algunas especificaciones de CommonJS. Node.js incluye un entorno REPL para depuración interactiva.

## Paralelismo

Node.js funciona con un modelo de evaluación de un único hilo de ejecución, usando entradas y salidas asíncronas las cuales pueden ejecutarse concurrentemente en un número de hasta cientos de miles sin incurrir en costos asociados al cambio de contexto. Este diseño de compartir un único hilo de ejecución entre todas las solicitudes atiende a necesidades de aplicaciones

altamente concurrentes, en el que toda operación que realice entradas y salidas debe tener una función callback. Un inconveniente de este enfoque de único hilo de ejecución es que Node.js requiere de módulos adicionales como **clúster** para escalar la aplicación con el número de núcleos de procesamiento de la máquina en la que se ejecuta.

## **V8**

V8 es el ambiente de ejecución para JavaScript creado para Google Chrome. Es software libre desde 2008, está escrito en C++ y compila el código fuente JavaScript en código de máquina en lugar de interpretarlo en tiempo real.

Node.js contiene libuv para manejar eventos asíncronos. Libuv es una capa de abstracción de funcionalidades de redes y sistemas de archivo en sistemas Windows y sistemas basados en POSIX como Linux, Mac OS X y Unix.

El cuerpo de operaciones de base de Node.js está escrito en JavaScript con métodos de soporte escritos en C++.

## **Desarrollo homogéneo entre cliente y servidor**

Node.js puede ser combinado con una base de datos documental (por ejemplo, MongoDB o CouchDB) y JSON lo que permite desarrollar en un ambiente de desarrollo JavaScript unificado. Con la adaptación de los patrones para desarrollo del lado del servidor tales como MVC y sus variantes MVP, MVVM, etc. Node.js facilita la reutilización de código del mismo modelo de interfaz entre el lado del cliente y el lado del servidor.

## **Lazo de eventos**

Node.js se registra con el sistema operativo y cada vez que un cliente establece una conexión se ejecuta un callback. Dentro del ambiente de ejecución de Node.js, cada conexión recibe una pequeña asignación de espacio de memoria dinámico, sin tener que generar un hilo de trabajo. A diferencia de otros servidores dirigidos por eventos, el lazo de manejo de eventos de Node.js no



es llamado explícitamente sino que se activa al final de cada ejecución de una función de callback. El lazo de manejo de eventos se termina cuando ya no quedan eventos por atender.

## **Módulos**

Node.js incorpora varios "módulos básicos" compilados en el propio binario, como por ejemplo el módulo de red, que proporciona una capa para programación de red asíncrona y otros módulos fundamentales, como por ejemplo Path, FileSystem, Buffer, Timers y el de propósito más general Stream. Es posible utilizar módulos desarrollados por terceros, ya sea como archivos ".node" precompilados, o como archivos en javascript plano. Los módulos Javascript se implementan siguiendo la especificación CommonJS para módulos, utilizando una variable de exportación para dar a estos scripts acceso a funciones y variables implementadas por los módulos.

Los módulos de terceros pueden extender node.js o añadir un nivel de abstracción, implementando varias utilidades middleware para utilizar en aplicaciones web, como por ejemplo los frameworks connect y express. Pese a que los módulos pueden instalarse como archivos simples, normalmente se instalan utilizando el **Node Package Manager (npm)** que nos facilitará la compilación, instalación y actualización de módulos así como la gestión de las dependencias. Además, los módulos que no se instalen el directorio por defecto de módulos de Node necesitarán la utilización de una ruta relativa para poder encontrarlos. El wiki Node.js proporciona una lista de varios de los módulos de terceros disponibles.

## **NPM (Node Package Manager)**

Npm es un gestor de paquetes para javascript, predeterminado para node.js, cuando instalas node también se instala npm. Esto quiere decir que a través de npm podremos instalar y gestionar los paquetes para nuestras aplicaciones.

### ***Crear una aplicación con npm***

Para crear una aplicación desde 0, tenemos el comando init

- npm init

Cuando se ejecuta este comando npm nos hace varias preguntas para realizar la configuración inicial del proyecto a realizar, como el nombre del proyecto, si tiene un repositorio git, descripción del proyecto, la versión actual, etc. Una vez que todas las preguntas sean contestadas, npm creará un archivo package.json con los datos que se acaban de introducir.

El archivo package.json lleva la configuración del paquete, donde se guardarán las dependencias de paquetes del proyecto y la configuración básica de este.

### ***Poner un paso a paso de lo que el gestor te va diciendo***

#### ***Instalación de una aplicación***

Sí se descarga una aplicación o paquete de node.js desde github o de cualquier control de versiones y se quiere ejecutar lo primero que se tiene que hacer es resolver las dependencias que tenga. Para ello tienes que ejecutar el comando install desde la ruta donde esté el código.

- npm install

Este comando irá al fichero package.json y revisará las dependencias del proyecto y las instalará.

#### ***Instalar un paquete con NPM para usar en el proyecto***

Para instalar un paquete con NPM se debe ejecutar este comando.

- npm install [nombre\\_paquete]

Esto instalará el paquete indicado en la ruta actual. Esto nos permitirá ya utilizar el paquete en la aplicación, pero que pasa si pasa el código a alguien e intenta utilizar la aplicación, que fallara ya que no se instalaran las dependencias que hemos agregado al código ya que no hemos indicado en ningún sitio que estamos usando este nuevo paquete. Para ello se tiene que ejecutar el comando install con el parametro --save que guardara esta nueva dependencia en el package.json.

- npm install [nombre\\_paquete]--save

También se nos puede dar el caso de que el paquete que se quiera guardar solo lo se utilizara cuando se está desarrollando y que cuando se despliegue el proyecto en producción no se necesitará, pues existe también una forma de decirle a npm que la nueva dependencia es solo para desarrollo y esto se consigue pasandole el parametro `--save-dev`.

- `npm install [nombre\_paquete]--save-dev``

### ***Iniciar una aplicación***

Sí se quiere ejecutar una aplicación, aparte de poder ejecutarla con el comando `node` podemos ejecutarla con:

- `npm start`

Para que este comando funcione tenemos que tener bien configurado el `package.json` con el archivo inicial que queremos que se ejecute al realizar el `start`.

## **Aplicación en Intel ® XDK IoT Edition**

La mejor manera para poder ver toda la parte teórica anterior, se propone hacer una aplicación sencilla utilizando el entorno Intel XDK IoT Edition, por lo tanto, se programará en Node.js y se explicará cómo hacer el proyecto desde cero y como correrlo. Se debe destacar que la placa tendrá que tener su firmware actualizado (Puesta en Marcha, Paso 1) y en la tarjeta SD tiene que estar instalado el sistema operativo Linux Yocto (Puesta en Marcha, Paso 2).

### **Explicación de la aplicación**

La aplicación tiene como objetivo explicar cómo hacer un servidor con el módulo “Express”, como “renderizar” una página a través del módulo “ejs” y como secuenciar porciones de código con el módulo “Sequence”. Además, en la aplicación se realizará la ejecución de comandos en el Linux a través del módulo “ChildProcess” y, por un lado, se le dará la posibilidad al usuario de ejecutar comandos Linux (como por ejemplo, “ls”, “pwd”, entre otros) y se le mostrará el resultado en pantalla. En la Figura X, se puede observar como son los pasos desde el requerimiento hasta la respuesta del Node.js a la página web.

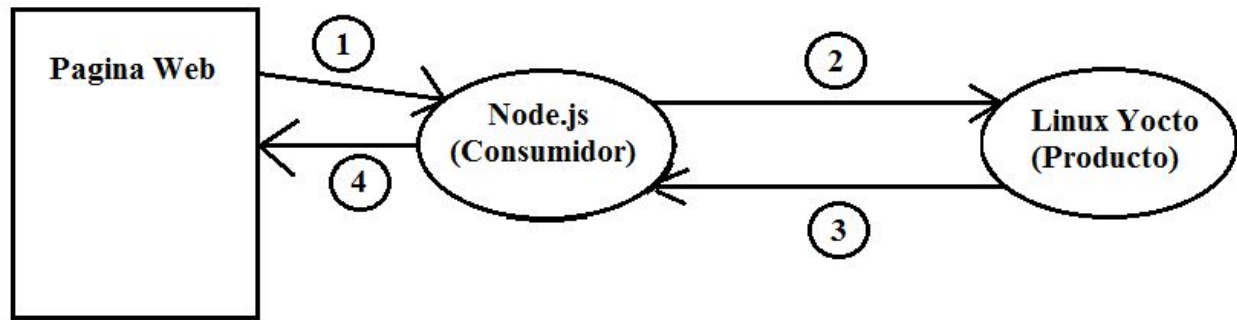


Figura X: esquema desde el requerimiento hasta la respuesta.

Los pasos son:

1. Requerimiento http al Node.js.
2. El Node.js captura los parámetros y los ejecuta en el Linux.
3. El Linux Yocto responde con la salida del comando.
4. El Node.js envía la información obtenida a la página web.

Por otro lado, se establecerá una comunicación entre el Linux y el Node.js mediante la lectura y escritura de archivos. Es un modelo de productor-consumidor bajo demanda donde el Node.js (consumidor) ejecuta un comando (“top”, “free” o “who”) en el Linux (productor) y, este último, escribe en un archivo el resultado de la salida para que el consumidor lea esos datos y los muestre en la página web. Se eligió estos tres comandos para poder monitorizar la placa, en tiempo real, desde una página web. El comando “top” muestra información referida a los procesos que se están ejecutando en ese momento; el comando “free” muestra información de la memoria RAM, cache y Swap; y el comando “Who” muestra los usuarios que están dentro del sistema operativo.

En la Figura X, se muestra un esquema de cómo es el circuito de comunicación entre el Node.js y el Linux Yocto.

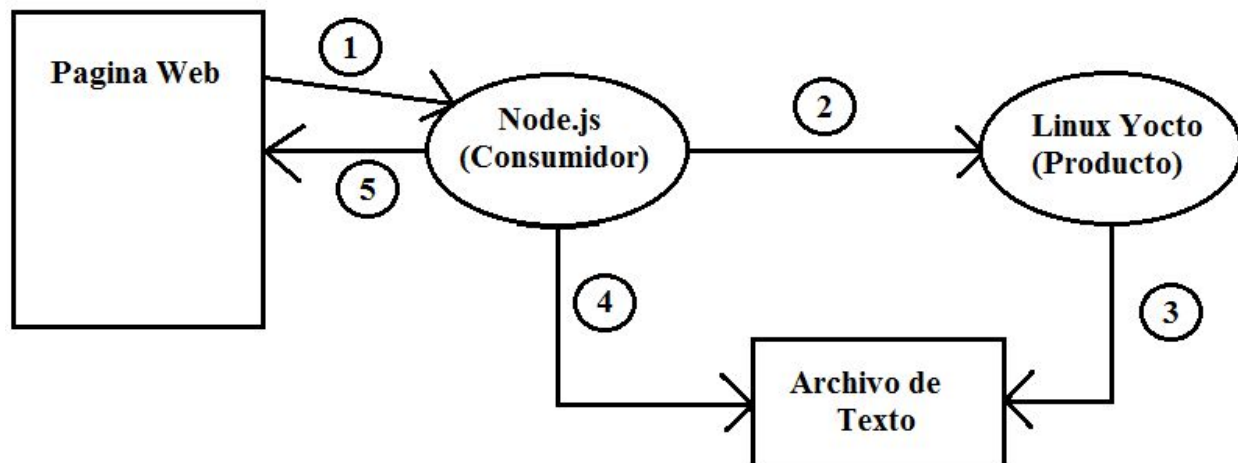


Figura X: Esquema de la ejecución de los programas “top”, “free” y “who”.

1. Requerimiento http, cada 500ms, de uno de los tres comandos al Node.js.
2. El Node.js los ejecuta en el Linux cada 500ms.
3. El Linux Yocto guardará en un archivo la salida del comando. Esto se repetirá cada 500ms.
4. El Node.js leerá el archivo siempre que el requerimiento http sea capturado
5. Responde a la página web con la información pedida.

### **Módulos instalados y su funcionamiento**

Los módulos instalados son “express”, “ejs” y “sequence” y, además, se utilizan los módulos “fs” (para manejar el file system), “path” para obtener la ruta actual de trabajo y, el antes mencionado, “childProcess” para ejecutar comandos en el Linux Yocto.

#### ***Express***

Es un módulo para Node.js que permite crear aplicaciones. La función `express()` es una función exportada por el módulo Express. En la Figura X, se muestra como crear un servidor con este módulo, particularmente, en el puerto 5000.

```
var express = require("express");
var app = express();

var port = process.env.PORT || 5000;
app.listen(port, function() {
  console.log("Listening on " + port);
});
```

Figura X: creación de un servidor con “express”

Para poder obtener los requerimientos, simplemente se debe especificar por cual método se desea que lleguen (“POST” o “GET”) y que función realizar a partir de allí.

- app.get(dirección,function(req,res){})
  - dirección: la cual se hace el requerimiento. Ejemplo (“/”)
  - function(req,res){}
    - req: se pueden recuperar los parámetros como si fuera un “query\_string” que están en la página web.
    - res: sirve para responder al requerimiento que llegó,

Con respecto al método “POST” para obtener los parámetros, se debe incorporar un módulo denominado “**body-parser**” que es capaz de poder poner los parámetros en formato JSON (clave:valor) y poder convertirlos para ser utilizados. La configuración para que el módulo Express conozca esto, se muestra en la Figura X:

```
var bodyParser = require('body-parser');
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extend:true
}));
```

Figura X: Configuración del body-parser

Luego cuando se desee mandar los datos a través de un método “POST” solamente basta con poner los comandos de la Figura X:

```
app.post("/registro", function (req, res){
  var regEmail = req.body.email;
  res.end("hola: "+regEmail);
})
```

Figura X: obtención de parámetros en el método POST y respuesta al requerimiento “/registro”.

## Ejs

Este módulo sirve para poder “renderizar” una página web. Para poder usarlo con Express se debe configurar al motor de vistas (view-engine) de la aplicación express con la siguiente línea:

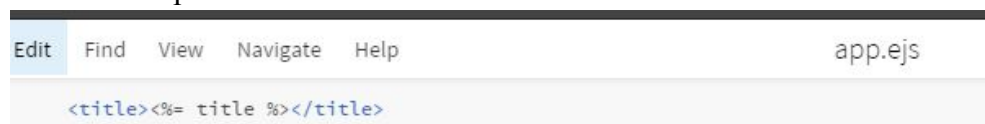
- `app.set('view engine', 'ejs');`

Para hacer una página web que permita poder enviar la información desde el Node.js hacia la página web se debe poner los tags (etiquetas) que provee este módulo:

- `<%` : tag de script, permite realizar flujo de control, no tiene salida
- `<%=` pone en la salida el valor enviado desde el Node.js dentro de la página web.
- `%>` Es para cerrar el bloque de código del ejs.

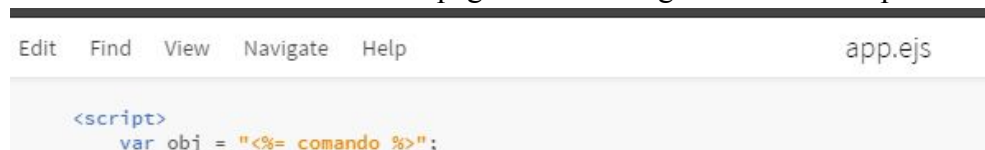
## Ejemplo del código

- Node.js
  - `res.render(appDir + '/app.ejs'`
    - Dirección de donde se encuentra la página web.
  - `, {title:"Galileo / Comando Top",`
    - Al valor `<%= title %>` en la página web le asignara el mensaje “Galileo / Comando Top”



The screenshot shows a code editor window titled 'app.ejs'. The menu bar includes 'Edit', 'Find', 'View', 'Navigate', and 'Help'. The code content is: `<title><%= title %></title>`.

- `comando:"top",`
  - Al valor `<%= comando %>` en la página web le asignara el valor “top”



The screenshot shows a code editor window titled 'app.ejs'. The menu bar includes 'Edit', 'Find', 'View', 'Navigate', and 'Help'. The code content is: `<script>var obj = "<%= comando %>";`.

- `commandMessage:"Monitoreo de procesos activos (top -i)"});`

- Al valor `<%= commandMessage %>` en la página web le asignada el valor “Monitoreo de procesos activos (top -i)”



```

Edit Find View Navigate Help app.ejs
<br>
<div class="container" align="center" >
  <div id="panel" class="panel panel-primary" >
    <div class="panel-heading"><%= commandMessage %></div>

```

## Sequence

Este módulo ejecuta código asíncronico en un orden cronológico. Para poder usarlo se debe crear el objeto de secuencia:

- `var Sequence = require('sequence').Sequence;`
- `var sequence = Sequence.create();`

Luego, para ejecutar la porción de código en orden cronológico se necesita las siguientes consideraciones:

- Se deben poner `sequence.then(function(next){}).then(function(next){})` y así siguiendo para poder ejecutar código en orden cronológico.
  - El parámetro “next” debe estar ya que es el indicador de cuándo debe continuar al siguiente “.then”. En el caso del último, termina la ejecución del “sequence”.
- Dentro de la función, el código que se quiere ejecutar debe estar dentro de la función “`setTimeout(function(),tiempoEnMilisegundos)`” ya que al ser en orden cronológico, debe terminar la ejecución del “sequence”. Además, dentro de esta función, debe estar la función “next()” que es la que viene por parámetro.

En la Figura X, se ve un ejemplo del uso del “sequence”.



```

sequence
  .then(function(next){
    setTimeout(function(){
      child = exec(command, function (error, stdout, stderr) {
        if (error !== null) {
          console.log('exec error: ' + error);
          result += error;
        }else{
          result += stdout;
        }
      });
      next();
    },100);
  })
  .then(function(next){
    setTimeout(function(){
      res.end(result);
      next();
    },100);
  });

```

Figura X: ejemplo del uso del “sequence”

## Módulos nativos utilizados y su funcionamiento

*Fs*

*Path*

*childProcess*

**EXPLICAR LA PARTE WEB (REQ CON AJAX (hacer un grafico) Y como se ACTUALIZA EL PANEL)**

## Instrucciones para importar y ejecutar

Para poder ejecutar la aplicación se debe importar el proyecto desde un repositorio Github<sup>15</sup>. Se debe hacer click en el panel derecho, como muestra en la Figura X, donde dice “Download ZIP”.

<sup>15</sup> <https://github.com/turqito/PracticaProfesionalSupervisada>

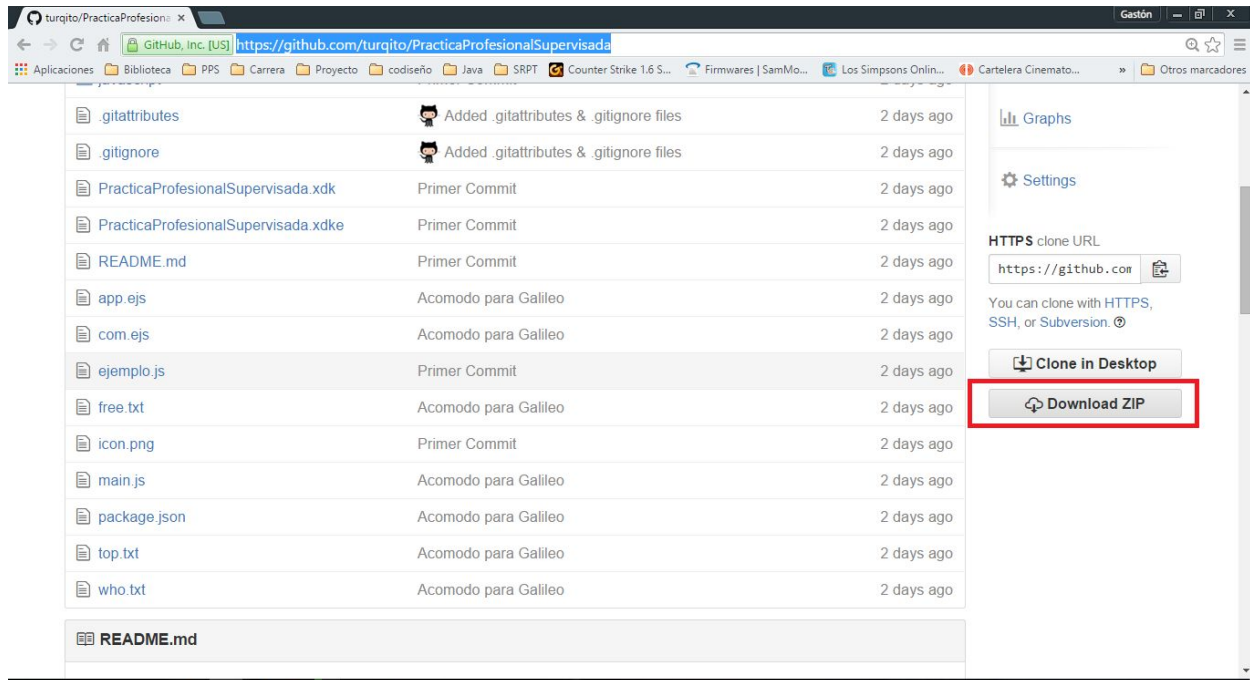


Figura X: repositorio donde se encuentra la aplicación

Luego, se debe extraer con la herramienta “7-zip”, instalada anteriormente. Una vez extraído debe aparecer una carpeta con los archivos del proyecto, como muestra la Figura X.

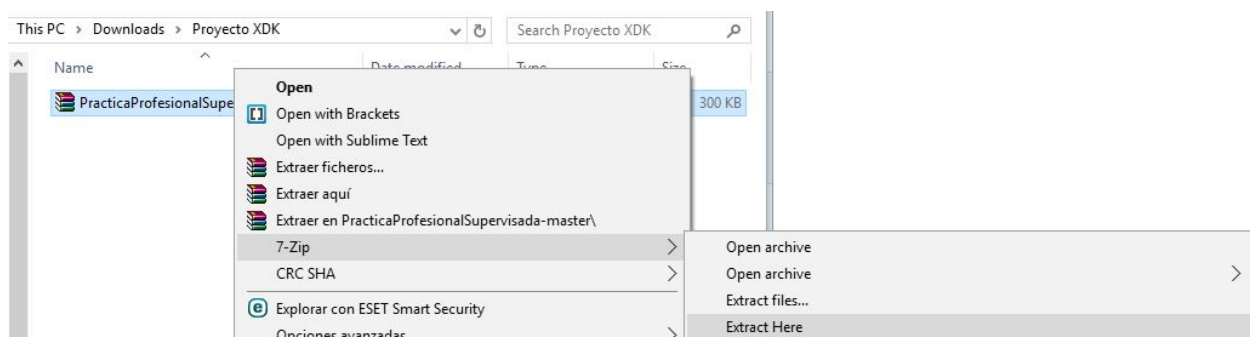


Figura X: extracción del proyecto

Para poder importar el proyecto, hay que abrir el Intel XDK IoT Edition y hacer click en “OPEN AN INTEL XDK PROJECT” y navegar hasta la carpeta donde se descomprimió el proyecto.

Finalmente, seleccionar el proyecto “PracticaProfesionalSupervisada.xdk” y hacer click en “Open”. A partir de allí, como se observa en la Figura X, ya se tiene el proyecto listo para ejecutar en la placa Galileo.

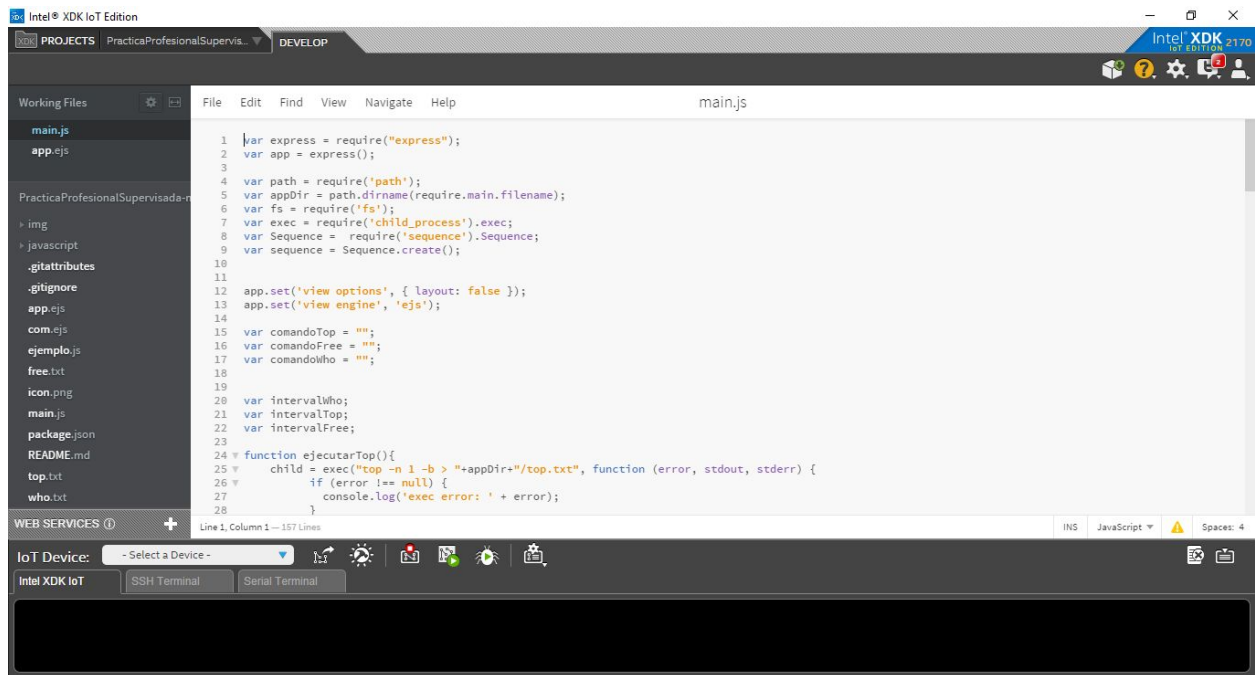

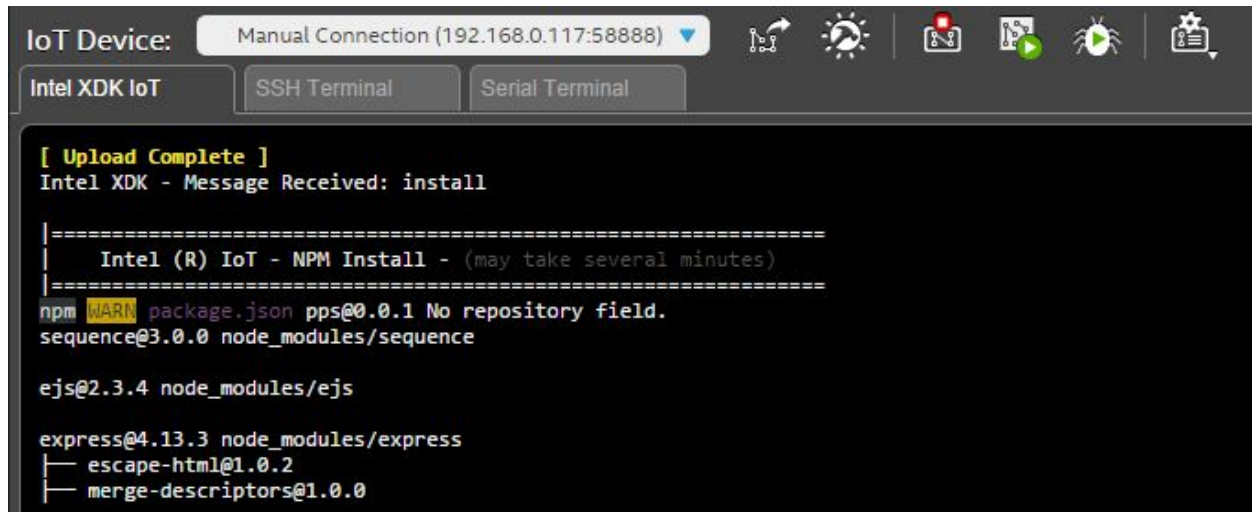


Figura X: entorno con el proyecto cargado.

Para ejecutar la aplicación, hay que conectarse con la placa como se realizó en el apartado “Puesta en Marcha” -> “Paso 6” -> “Entorno de desarrollo” -> “Conexión Ethernet” y



“Configuración Intel XDK”. Finalmente, hay que hacer click en el ícono  para que se construya el proyecto, se instalen los módulos externos y se ejecute el programa.

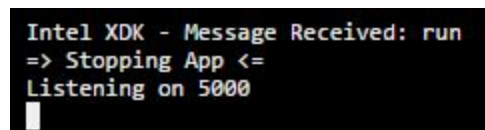


```
[ Upload Complete ]
Intel XDK - Message Received: install

=====
| Intel (R) IoT - NPM Install - (may take several minutes)
|=====
npm WARN package.json pps@0.0.1 No repository field.
sequence@3.0.0 node_modules/sequence

ejs@2.3.4 node_modules/ejs
express@4.13.3 node_modules/express
├─ escape-html@1.0.2
└─ merge-descriptors@1.0.0
```

Para asegurarse que la aplicación comience a correr, hacer click en el ícono de “Stop”  y luego hacer click en el botón “Run” . Si se ejecutó con éxito deberá aparecer el mensaje de la Figura X en la consola de salida.



```
Intel XDK - Message Received: run
=> Stopping App <=
Listening on 5000
```

Figura X: ejecución correcta de la aplicación.

Como se puede observar en la Figura X, aparece un mensaje que dice “Listening on 5000” que lo que quiere decir que la placa tiene abierto el puerto 5000 para acceder a la aplicación. Entonces, abrir un navegador y poner en la barra de direcciones la siguiente información:

- direcciónIpDeLaPlaca:puertoQueEstaEscuchando
  - o Ej: 192.168.0.117:5000

Cuando se presione la tecla Enter, aparecerá una

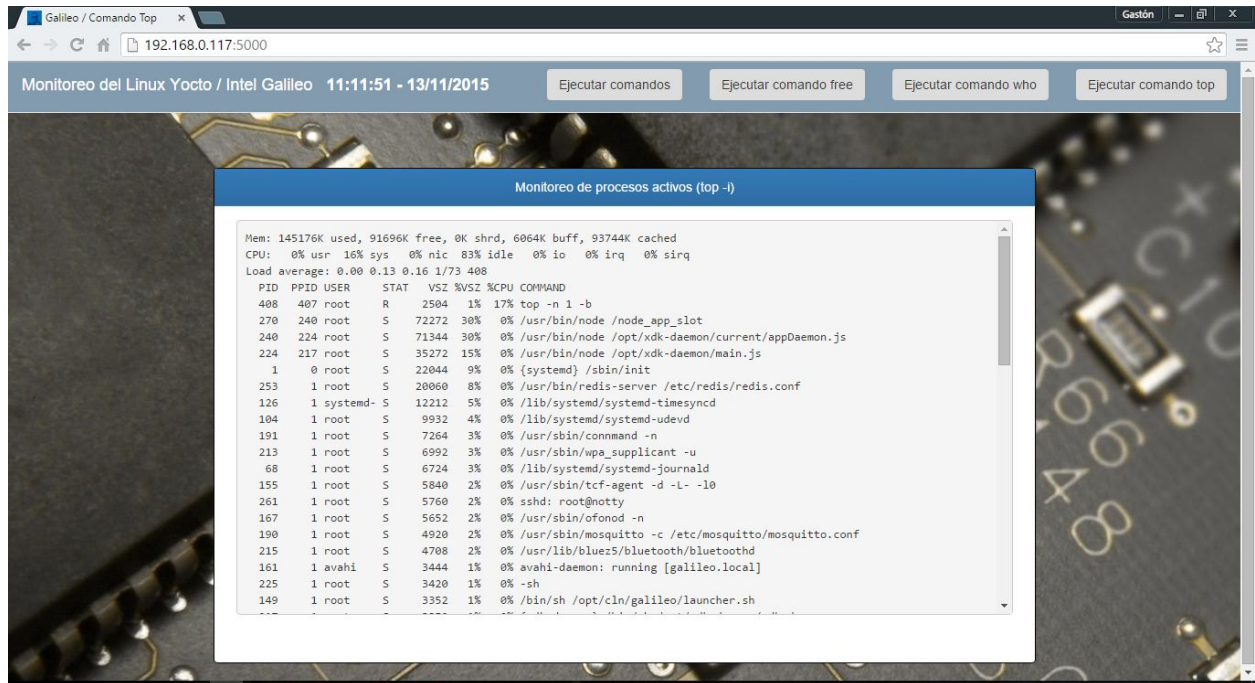


Figura X: Página principal

## Árbol de archivos

- Img
  - o favicon.png
  - o fondo.jpg
-

## **Módulos probados**

### *Cluster*

Una sola instancia de Node.js se ejecuta en un solo hilo. Para poder tener la ventaja de los sistemas con muchos núcleos, se desarrolla, a veces, con un cluster de procesos de Node.js para atender las peticiones.

El módulo clúster permite crear procesos hijos que comparten los puertos del servidor.

```

var cluster = require('cluster');
var http = require('http');
var numCPUs = require('os').cpus().length;

if (cluster.isMaster) {
  // Fork workers.
  for (var i = 0; i < numCPUs; i++) {
    cluster.fork();
  }

  cluster.on('exit', function(worker, code, signal) {
    console.log('worker ' + worker.process.pid + ' died');
  });
} else {
  // Workers can share any TCP connection
  // In this case it is an HTTP server
  http.createServer(function(req, res) {
    res.writeHead(200);
    res.end("hello world\n");
  }).listen(8000);
}

```

Corriendo el ejemplo de arriba, el Puerto 8000 es compartido por los “workers” (trabajadores):

```

% NODE_DEBUG=cluster node server.js
23521,Master Worker 23524 online
23521,Master Worker 23526 online
23521,Master Worker 23523 online
23521,Master Worker 23528 online

```

Como funciona

Los procesos trabajadores se generan utilizando el método `child_process.fork`, para que puedan comunicarse con los padres a través de IPC y pasar servidor maneja de ida y vuelta.

El módulo de clúster admite dos métodos de distribución de conexiones entrantes. El primero (y el que viene por defecto en todas las plataformas excepto Windows), es el enfoque de round-robin, donde el proceso maestro escucha en un puerto, acepta nuevas conexiones y los distribuye a los trabajadores en un round-robin, con algunas de ellas construidas con inteligencia

para evitar la sobrecarga de un proceso de trabajo. El segundo enfoque es donde el proceso maestro crea el socket de escucha y la envía a los trabajadores interesados. Los trabajadores entonces aceptan conexiones entrantes directamente.

### *Express*

Es un módulo para Node.js que permite crear aplicaciones. La función `express()` es una función exportada por el módulo Express.

```
var express = require('express');  
var app = express();
```

### *Emailjs*

### *Mysql*

### *Request*

**Que es lo que hace cada uno. Como instalar individualmente y como se agrega el package.json**



## **Bibliografia**

<https://software.intel.com/en-us/iot/library/galileo-getting-started>

<http://blog.nodejitsu.com/package-dependencies-done-right/>

<https://nodejs.org/api/>

<http://browsenpm.org/help>

<https://docs.npmjs.com/files/package.json>

<https://docs.npmjs.com/misc/semver>