# 

**Descripción del Sistema**

## Introducción

En el capítulo anterior se han estudiado algunos de los kits de desarrollo existentes en el mercado, así como las posibilidades de comunicación de éstos.

Dentro de las alternativas propuestas, se ha seleccionado proponer una arquitectura basada en el módulo IOIO en conjunto con la placa Arduino UNO.

Las ventajas de utilizar dichos elementos son que el *Gadget* se puede comunicar inalámbricamente mediante *Bluetooth* con el dispositivo móvil inteligente y que en caso de tener que conectar un sensor con una interfaz no compatible con el módulo IOIO, éste se podría conectar fácilmente con Arduino.

Mencionar que el Kit de Desarrollo IOIO OTG fue lanzado al mercado después de la realización de este Proyecto Fin de Carrera.

## IOIO

En esta sección se va a describir tanto el *hardware* como el *software* de la placa IOIO, así como el entorno de desarrollo necesario para la realización de la aplicación Android.

### Hardware

#### Vista Conjunta

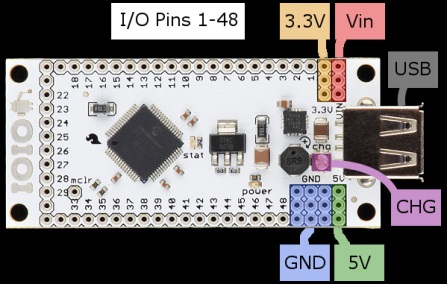


Ilustración 3‑1: Vista Conjunta Placa IOIO

La placa IOIO contiene los siguientes componentes:

* Conector USB (tipo A): Se utiliza para conectar IOIO con el dispositivo Android en caso de comunicación cableada o con el módulo *Bluetooth* en caso de comunicación inalámbrica.
* 9 Pines GND: Conexión a tierra.
* 3 Pines VIN: Se utiliza para la alimentación de la placa. El voltaje suministrado debe ser entre 5 V – 15 V.
* 3 Pines 5 V: Se utiliza normalmente como salida de 5 V, cuando se alimenta la placa a través de los pines VIN, pero se puede utilizar como entrada de 5 V para alimentar a la placa en el caso en el que no esté alimentada a través de VIN.
* 3 Pines de 3,3 V: Salida de voltaje regulada a 3,3 V.
* 48 Pines E/S (numerado del 1-48): Pines de E/S de propósito general (más adelante se detallaran las funciones especiales que tienen algunos).
* LED de alimentación: Se ilumina cuando IOIO está correctamente alimentada.
* Pin MCLR: Se utiliza para la programación de un nuevo *firmware bootloader* en la placa IOIO.
* CHG o *Charge Current Trimmer*: Se utiliza para ajustar la cantidad de corriente suministrada en la línea VBUS del USB para el dispositivo Android o para el dispositivo inalámbrico. Girando en dirección “+” se incrementa la intensidad.

#### Pines E/S

Los Pines de E/S son la esencia de IOIO. Se utilizan para conectar los circuitos externos utilizando diferentes interfaces. El denominador común de todos los pines es que pueden servir tanto para entradas como salidas digitales a 3,3 V.

Para mayor comodidad, las funciones comunes de los pines se representan gráficamente en la parte inferior de la placa, con una leyenda:

* Pines rodeados por un cuadrado se pueden utilizar como entradas analógicas (3,3 V).
* Pines rodeados por un círculo son tolerantes a 5 V, es decir, se puede utilizar como entradas de lógica a 5 V o salida de 5 V lógica, cuando se utiliza el modo de apertura con la ayuda de una resistencia de pull-up.
* Pines marcados con “P” se utilizan como entradas y salidas periféricas. Principalmente, esto incluye PWM, UART y SPI.
* Pines marcados con “Pi” se utilizan como entrada periférica (pero no de salida).
* Pines marcados con DAx y CLx se utilizan para TWI.

#### Tabla Funciones Pines

Leyenda:

* A/D: Pin que puede ser utilizado como entrada analógica.
* I²C: Pin que puede ser utilizado como I²C, DAx (Pin Data) y CLx (Pin Reloj), siendo “x” el número del módulo del I²C.
* PPSi: Pin que puede ser utilizado como entrada para periféricos reasignables (captura de entrada, UART, SPI).
* PPSo: Pin que puede ser utilizado como salida para periféricos reasignables (salida del comparador, UART, SPI).
* 5 V: Pin tolerante a 5 V, que se puede utilizar como entrada o salida a 5 V en el modo de apertura.
* Comp: Pin que puede ser utilizado como número de entrada del comparador.
* Prog: Pin que puede ser utilizado para ICSP, usa Vpp y/o C1/D1, C2/D2 o C3/D3, que son reloj y datos, respectivamente.
* Todos los pines pueden ser utilizados como entrada/salida digital (GPIO), además de soportar interrupciones.

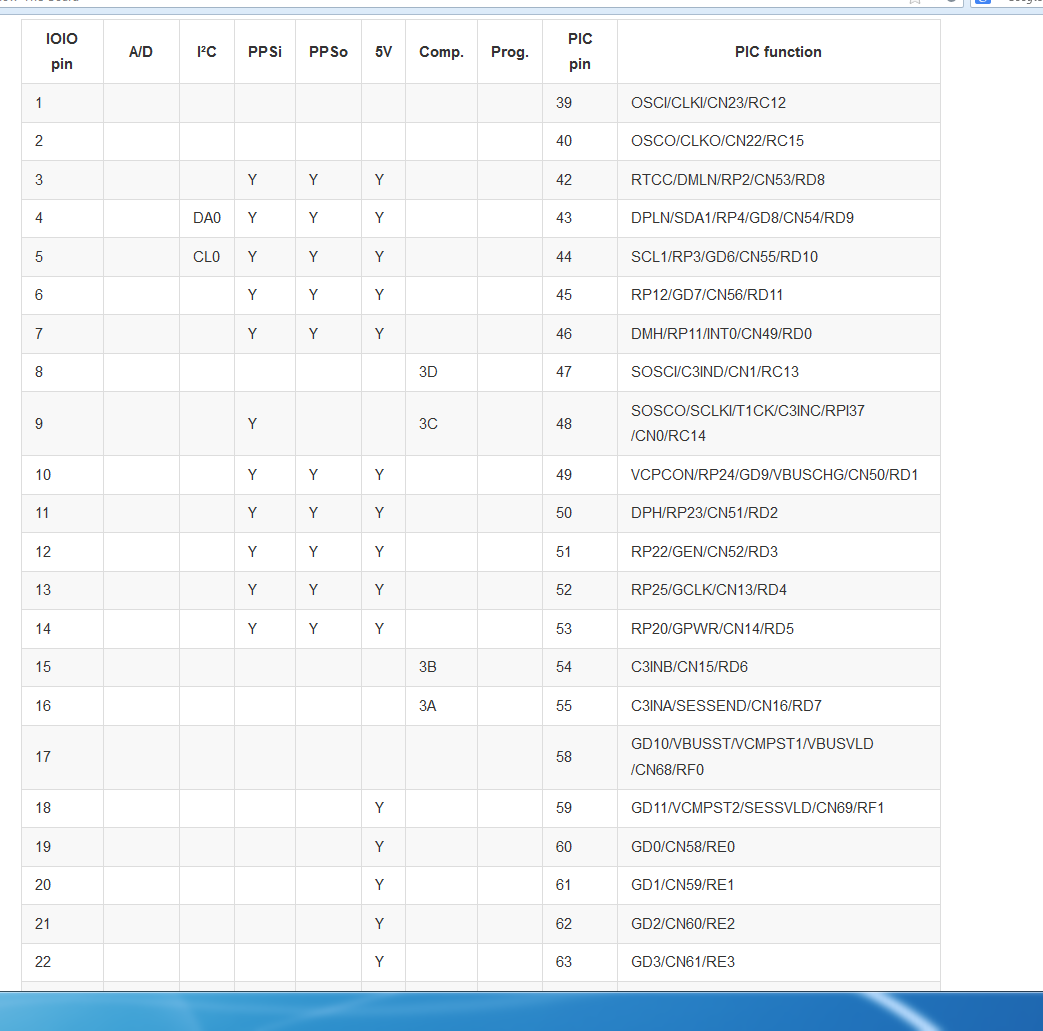


Tabla 3‑1: Tabla Funciones Pines

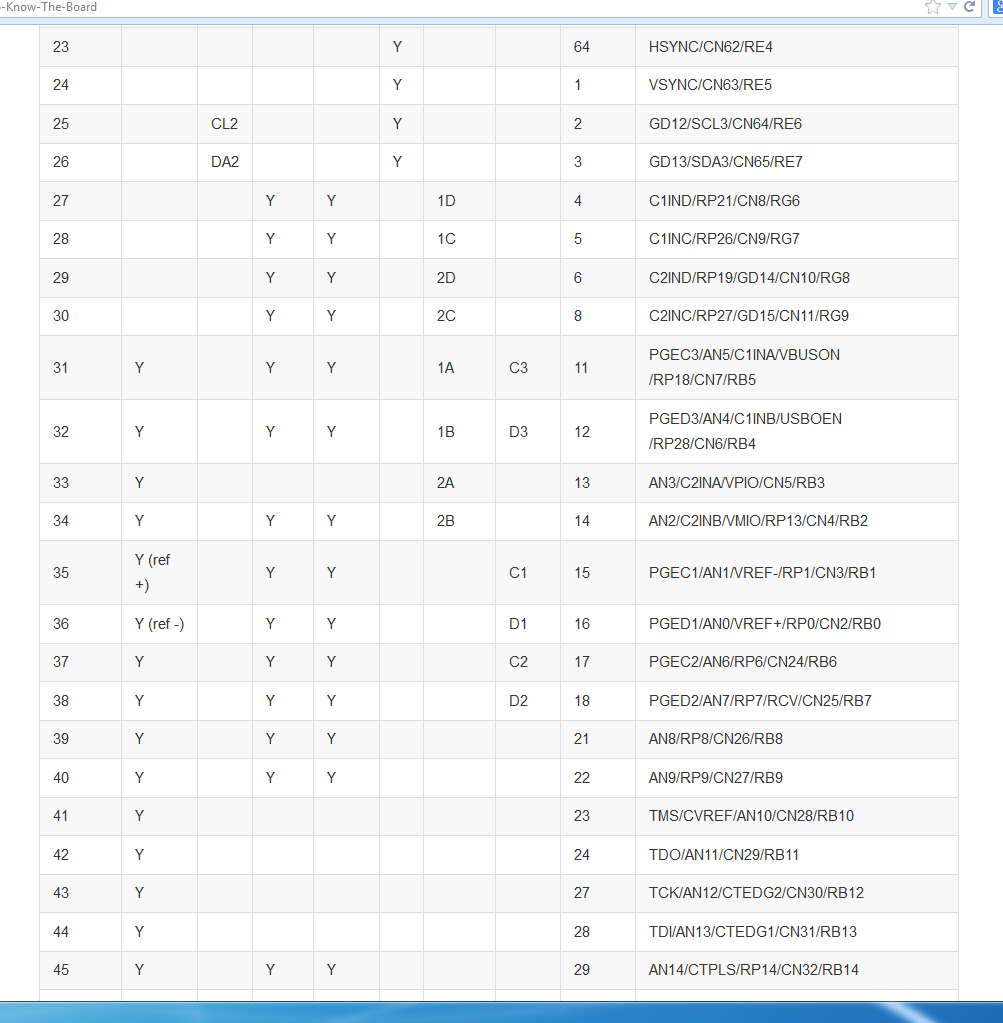


Tabla 3‑2: Tabla Funciones Pines (continuación)

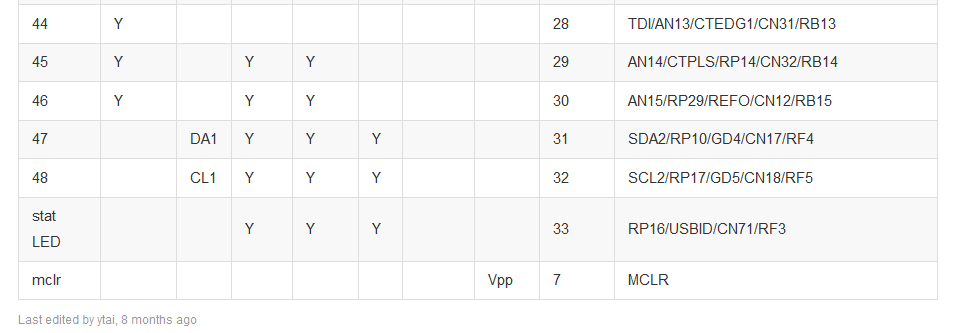


Tabla 3‑3: Tabla Funciones Pines (continuación)

### Fuente de Alimentación

#### Conexión Básica

La forma más común para alimentar a la placa IOIO es mediante la conexión de una fuente DC, en cualquier lugar entre 5 V – 15 V, con el lado “+” a uno de los tres pines marcados como “VIN” y el lado “+” a uno de los 9 pines marcados como “GND”.

Tan pronto como se hace esto, el LED de encendido rojo debe encenderse. Otra forma más robusta es soldar un conector JST de montaje en superficie en la parte inferior de la placa, en los *pads* derecha debajo del conector USB.

#### Carga del Dispositivo Android desde IOIO

IOIO suministra 5 V para el dispositivo Android a través de la conexión USB (en la línea VBUS). Este suministro de 5 V permite la carga de la batería del dispositivo Android además de permitirle saber al dispositivo Android que un host USB está conectado. Con el potenciómetro se regula la corriente en la carga.

#### Limitación de Corriente de Carga

En aquellas aplicaciones en las que no se desee cargar el dispositivo, la corriente en la carga puede ser limitada. Una limitación de la corriente de carga, supone una disminución de la tensión en la línea VBUS del USB. De acuerdo con las especificaciones USB, la disminución de VBUS por debajo del umbral de 5 V es un voltaje erróneo. Esto no va a dañar el dispositivo Android, pero puede hacer que el dispositivo Android deje de detectar la conexión con la placa IOIO.

El punto en el que el dispositivo móvil inteligente deja de detectar la conexión con la placa IOIO varía según el dispositivo. La forma de ajustar la corriente en la carga necesaria para un dispositivo específico es ir regulándola con el potenciómetro.

#### Descripción de las Características de Potencia

IOIO tiene 2 reguladores de tensión en la placa:

* Un regulador de conmutación para entradas de 5 V – 15 V y para salidas de 5 V estables y que puede suministrar hasta 1,5 A.
* Un regulador lineal que se alimenta de la línea de 5 V, con salidas de 3,3 V estables y que puede suministrar hasta 800 mA.

Los 3 Pines de 5 V proporcionan 1 A de intensidad a los dispositivos externos, mientras que los pines de 3,3 V proporcionan 700 mA a los periféricos externos.

### Bluetooth

#### Introducción

Como se mencionó en el capítulo anterior, la placa de desarrollo IOIO no posee un módulo integrado de *Bluetooth*, pero permite este tipo de comunicación conectando un módulo *Bluetooth*.

#### Módulos Bluetooth

A continuación se muestran los diferentes módulos *Bluetooth* que han sido probados y que funcionan con la placa IOIO.

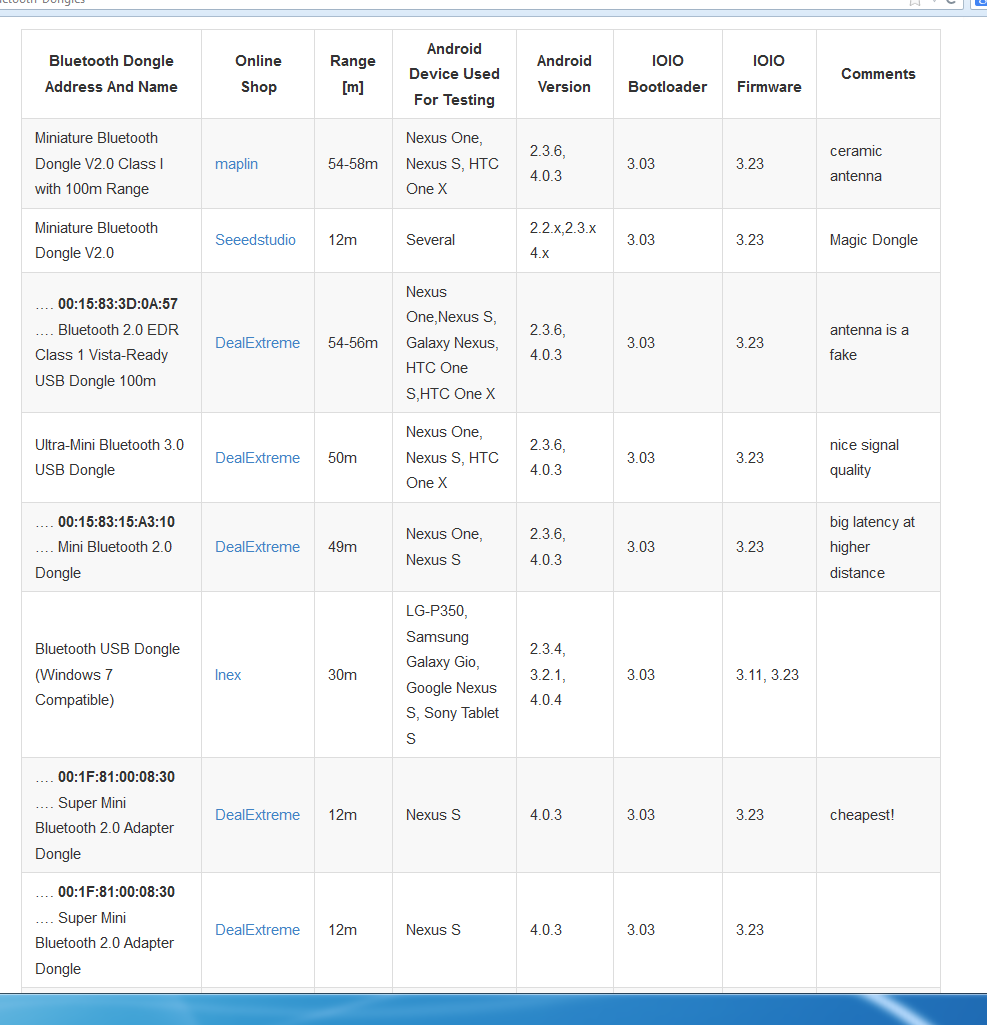


Tabla 3‑4: Tabla Módulos *Bluetooth* Compatibles



Tabla 3‑5: Tabla Módulos *Bluetooth* Compatibles (continuación)

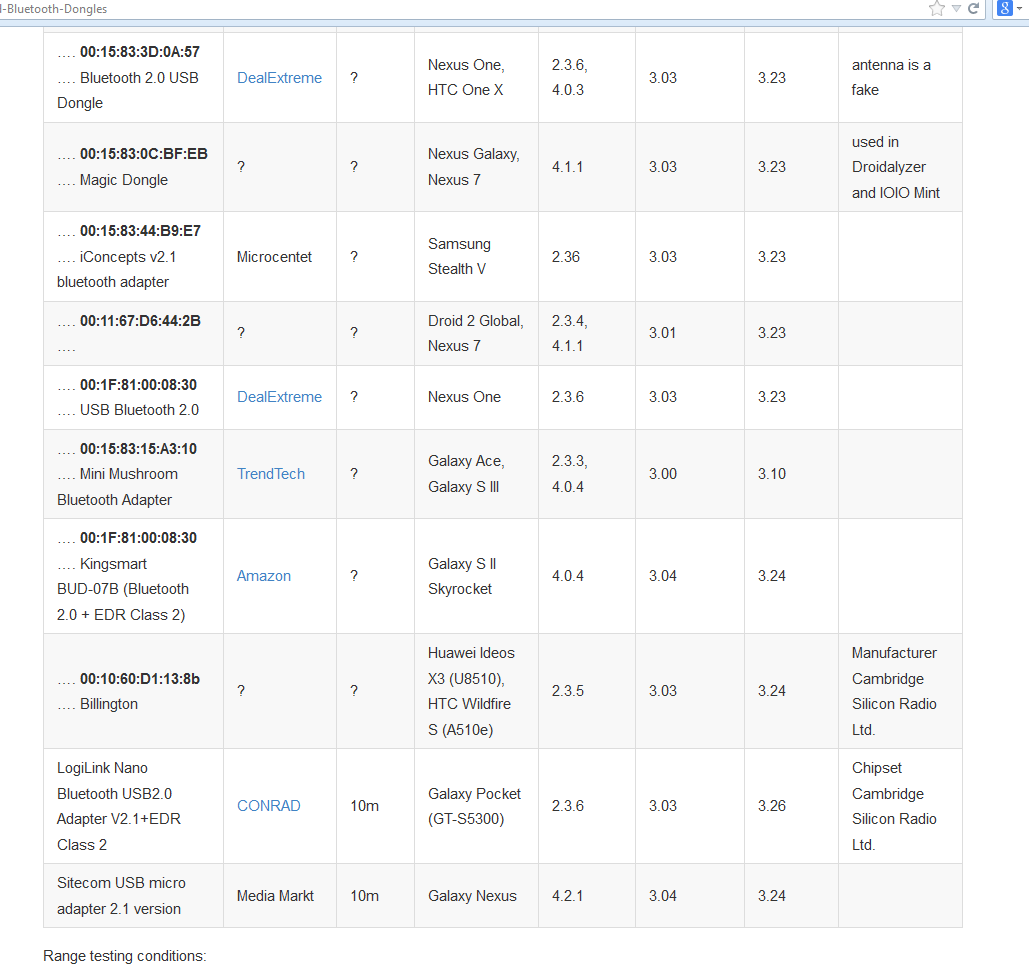


Tabla 3‑6: Tabla Módulos *Bluetooth* Compatibles (continuación)

### Software

#### Introducción

La placa IOIO está basada en un micro-controlador de Microchip. Principalmente, dicha placa incluye una interfaz USB y tiene externalizados los pines del micro-controlador. En relación al desarrollo del *software*, se puede optar por dos alternativas.

La primera de ellas sería un diseño tradicional usando una herramienta de carga y depuración de Microchip.

La otra alternativa es posible gracias al desarrollo realizado por la comunidad IOIO. En dicha comunidad se puede descargar un *firmware* para el micro-controlador de la placa IOIO. Este *firmware* permite comunicar la placa IOIO a través de la interfaz USB (de forma cableada, así como inalámbrica conectando un módulo *Bluetooth*) con el dispositivo móvil inteligente. Además, proporcionan un conjunto de componentes *software* compatible con Android, los cuales permiten un control de bajo nivel de dicha placa. Así por ejemplo, desde Android se puede configurar un determinado pin digital de salida y poner dicho pin a valor alto o bajo. Naturalmente, esta segunda alternativa de diseño es la que permite diseñar prototipos de *Gadgets* de una forma más rápida y será la utilizada para implementar la arquitectura de diseño de accesorios electrónicos propuesta en este Proyecto Fin de Carrera.

#### Conceptos Previos

El corazón de la placa IOIO es un micro-controlador. Este micro-controlador ejecuta el código (“*firmware*”) que da a IOIO su funcionalidad: el establecimiento de una conexión de datos con un dispositivo Android, el control de los pines, etc. Este código se almacena en la memoria Flash que existe en el interior del micro-controlador. El código *firmware* de IOIO consta de dos partes principales: gestor de arranque (*bootloader*) y la aplicación.

El gestor de arranque (*bootloader*) es el primer código que ejecuta IOIO cada vez que se reinicia. Lo que hace es establecer una conexión de datos con el dispositivo Android, y a continuación, utiliza esa conexión de datos para comprobar la existencia de nuevos códigos de aplicación en el dispositivo Android. Si lo encuentra, lo instala y lo ejecuta. Si no, se ejecuta cualquier aplicación que ya haya sido instalada.

El *firmware* de la aplicación proporciona a la placa IOIO su funcionalidad principal, se comunica con el IOIOLib, que se ejecuta en el dispositivo Android, ejecutándose posteriormente los comandos para el control de algún periférico, pines digitales, módulo UART etc.

IOIO viene pre-programado con un gestor de arranque y una aplicación.

#### IOIO Manager

IOIO Manager es una aplicación para Android que permite instalar un nuevo *firmware* en la placa IOIO. La aplicación es de código abierto y está disponible para descargarla de forma gratuita desde el *Play Store*.

Dicha aplicación permite gestionar los paquetes de imágenes en 2 bibliotecas: las imágenes de aplicaciones y las imágenes del gestor de arranque.

Al iniciar la aplicación IOIO Manager, la primera pantalla que aparece es la biblioteca de imágenes de las aplicaciones, tal como se muestra en la Ilustración 3-2.



Ilustración 3‑2: Pantalla Inicial IOIO Manager

Hay varias maneras de agregar elementos en esta biblioteca. Una de ellas consiste en escanear un código QR (previa instalación de la aplicación Barcode Scanner), seleccionando la tecla menú y pulsando en “Scan QR” para descargar las imágenes a través del código QR en la página de descargas oficiales de IOIO (<https://github.com/ytai/ioio/wiki/Downloads>).

En la Ilustración 3-3 se muestra un ejemplo de varios elementos agregados en la biblioteca de imágenes de aplicaciones.

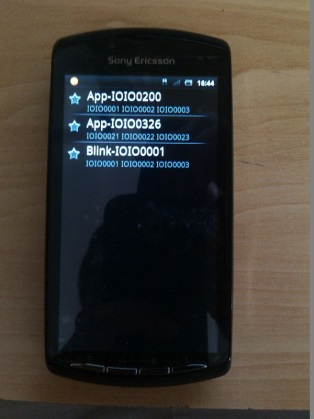


Ilustración 3‑3: Biblioteca de Imágenes de Aplicaciones

En la Ilustración 3-4 se puede observar la pantalla de programación, que se accede a ella, seleccionando la tecla menú (en la pantalla principal) y posteriormente pulsando en “*programmer*”. Para añadir elementos se ha de acceder previamente a la biblioteca de imágenes del gestor de arranque, a través de “*Select*…” (Situado en la esquina superior derecha de nuestro dispositivo Android) o seleccionando la tecla menú y pulsando en “Library”.

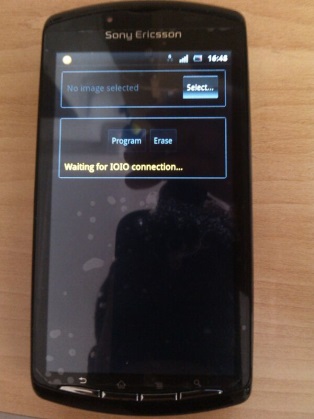


Ilustración 3‑4: Pantalla de Programación IOIO Manager

Al igual que en la biblioteca anteriormente comentada, una de las maneras de añadir elementos a la biblioteca de imágenes del gestor de arranque es mediante el escaneo de un código QR, seleccionado la tecla menú y pulsando en “Scan QR”.

En la Ilustración 3-5 se observa el modo de conexión para programar/actualizar el *firmware* de la placa IOIO. La imagen seleccionada (para actualizar el *firmware*) dentro de la biblioteca de imágenes del gestor de arranque, ha de ser compatible con nuestro *hardware*, por ejemplo, el micro-controlador PIC24FJ128DA206 requiere la instalación de la imagen SPRK0015. Sin embargo, el micro-controlador PIC24FJ256DA206 requiere la imagen SPRK0016.

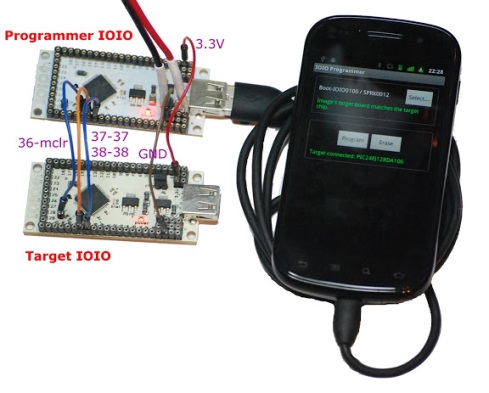


Ilustración 3‑5: Conexión Programador

En caso de seleccionar una imagen errónea para el dispositivo móvil inteligente, aparecerá un error tal como se muestra en la Ilustración 3-6.



Ilustración 3‑6: *Firmware* Incompatible *Hardware*

En la Ilustración 3-7 se puede observar la pantalla del programador con la selección de una imagen correcta. A continuación, es necesario pulsar sobre “*program*” para comenzar la actualización del *firmware*.



Ilustración 3‑7: *Firmware* Compatible *Hardware*

En la Tabla 3-7 se muestra la conexión de los pines necesarios para programar la placa IOIO.

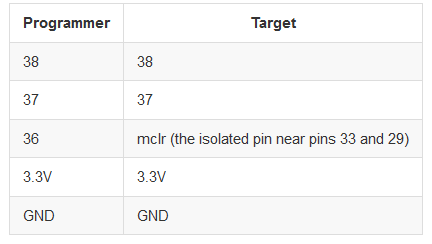


Tabla 3‑7: Conexión Pines Programar IOIO

#### IOIO Hardware Tester

IOIO *Hardware Tester* (ver Ilustración 3-8) es una aplicación para Android. La aplicación está disponible para descargarla de forma gratuita desde el *Play Store*.

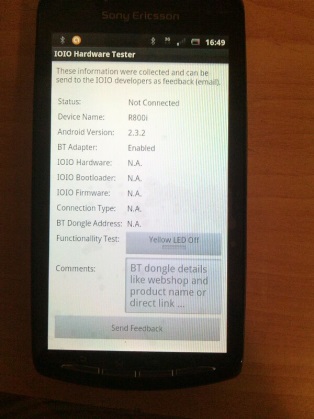


Ilustración 3‑8: Aplicación IOIO *Hardware Tester*

Su funcionalidad es informar si el dispositivo móvil inteligente funciona correctamente con la placa IOIO, proporcionando la siguiente información:

* Estado de Conexión.
* Nombre Dispositivo Móvil Inteligente.
* Versión Android de dicho dispositivo.
* Existencia de un módulo *Bluetooth*.
* Versión IOIO *Hardware*.
* Versión IOIO *Bootloader*.
* Versión IOIO *Firmware*.
* Tipo de Conexión.
* Dirección del módulo *Bluetooth*.
* Test de Funcionalidad.

### Entorno de Desarrollo

El desarrollo de programas para Android se hace habitualmente con el lenguaje de programación Java y el conjunto de herramientas de desarrollo (SDK, *Software Development Kit*).

El SDK de Android incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales.

La plataforma integral de desarrollo (IDE, *Integrated Development Environment*) soportada oficialmente es Eclipse junto con el complemento ADT (*Android Development Tools Plugin*). Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama “Aplicaciones de Cliente Enriquecido”.

Las actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato “.apk”.

#### Instalación del Entorno de Desarrollo

Descargar el paquete ADT en la página oficial de Android *Developer* (http://developer.android.com/sdk/index.html), el paquete incluye todo lo necesario para comenzar a desarrollar aplicaciones para Android.

Descomprimir el archivo en el lugar donde se quiera tener instalado.

Ejecutar el archivo “*eclipse.exe*” para lanzar la aplicación. La primera vez que arranque, solicitará que se le indique un directorio de trabajo donde se guardaran los proyectos (marcando en “*Use this as the default and do not ask again*” no preguntará de nuevo por el directorio de trabajo).

#### Instalación del Plugin ADT

Una vez que aparece en la pantalla principal de Eclipse, se selecciona en “*Help*” > “*Install New Software*”, tal como se muestra en la Ilustración 3-9.

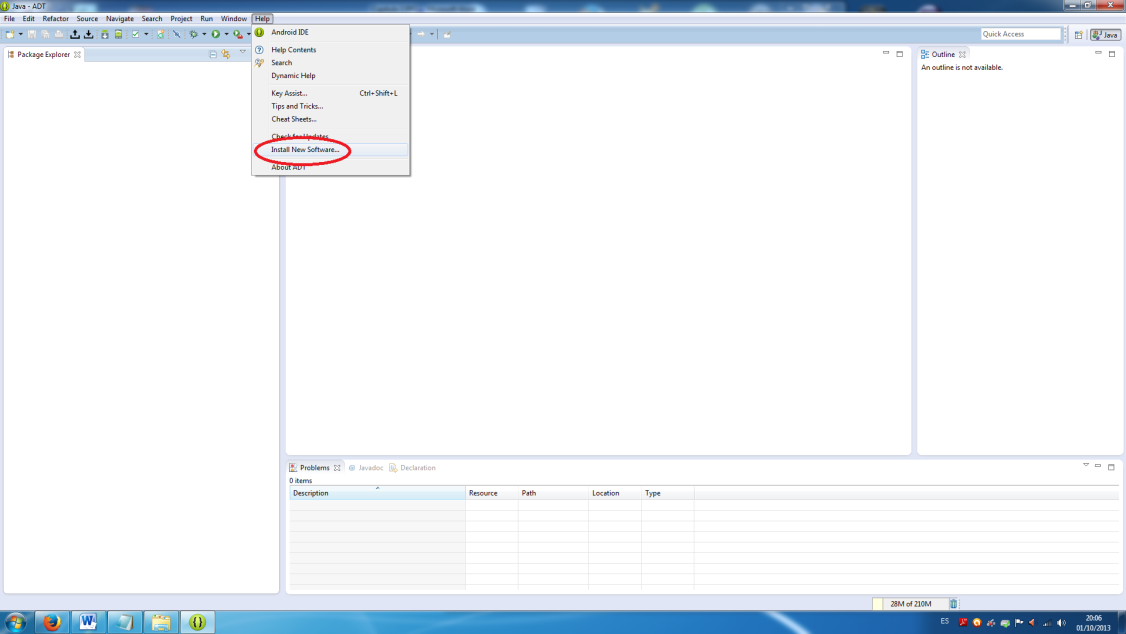


Ilustración 3‑9: Instalación Plugin ADT

En esta pantalla, es necesario hacer click en “*Add*” (situado en la esquina superior derecha), y rellenamos los campos con los datos “ADT Plugin” para el nombre y la URL de la localización “https://dl-ssl.google.com/android/eclipse/”, tal como se muestra en la Ilustración 3-10.

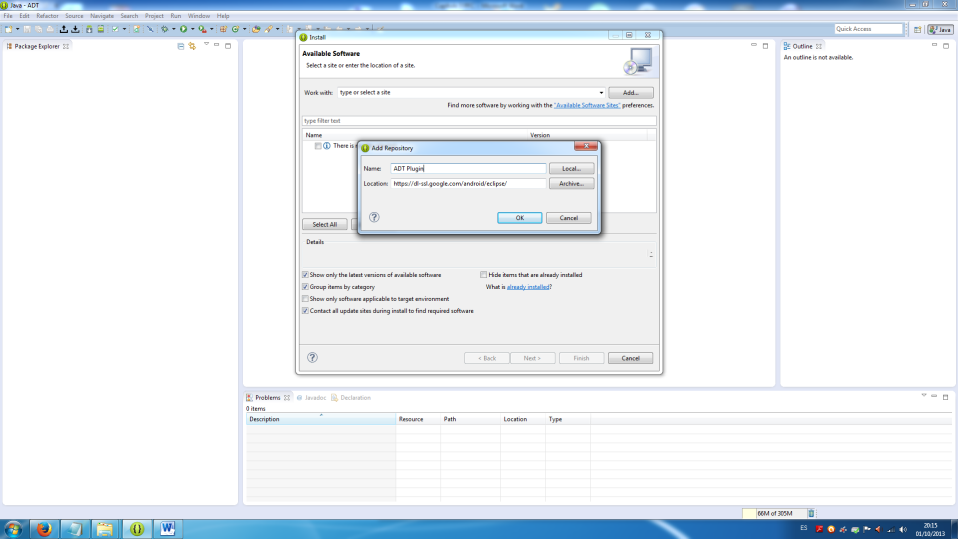


Ilustración 3‑10: Instalación Plugin ADT (continuación)

Se seleccionan las herramientas de desarrollo, se aceptan los acuerdos de licencias, etc. Finalmente, una vez terminada la instalación deberá reiniciarse Eclipse.

En las últimas versiones la descarga del paquete ADT, ya incluye la instalación del plugin ADT en Eclipse.

#### Instalación del SDK

En la pantalla principal de Eclipse, se selecciona “*Window*” > “*Preferences*”, tal como se muestra en la Ilustración 3-11.

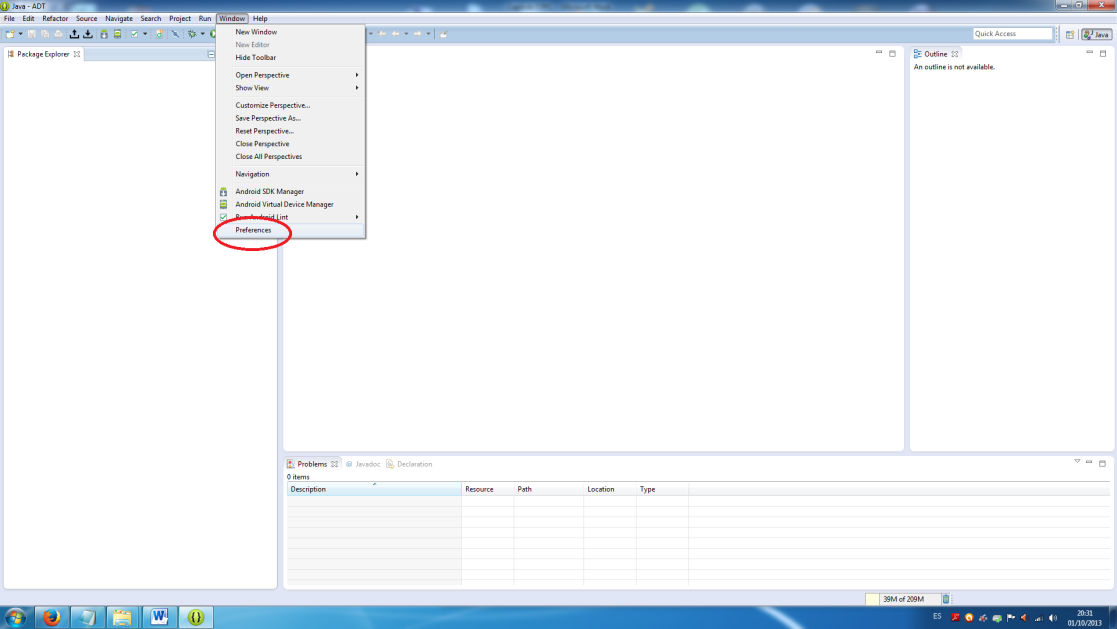


Ilustración 3‑11: Instalación SDK

A continuación se selecciona la sección de Android y se busca la localización del SDK, tal como se muestra en la Ilustración 3-12.

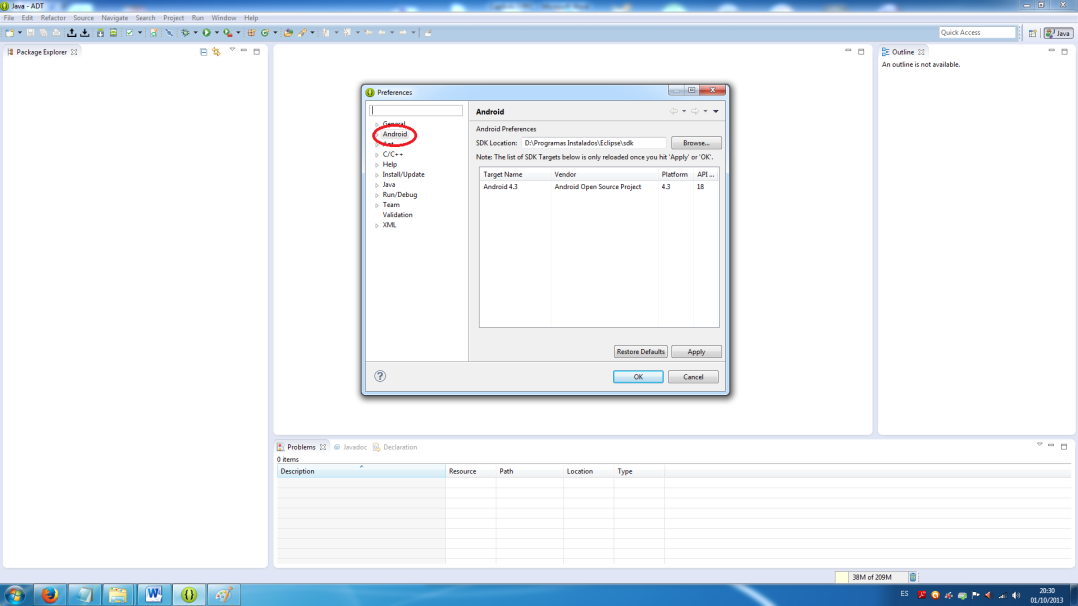


Ilustración 3‑12: Instalación SDK (continuación)

Al igual que lo ocurrido en la instalación del plugin ADT, el paquete ADT oficial ya incluye el proceso descrito en este apartado, además de incluir la instalación mínima del SDK necesaria para comenzar a desarrollar aplicaciones en Android.

En la Ilustración 3-13, se muestra una de las formas de acceder al administrador del SDK (“*Window*” > “*Android SDK Manager*”).

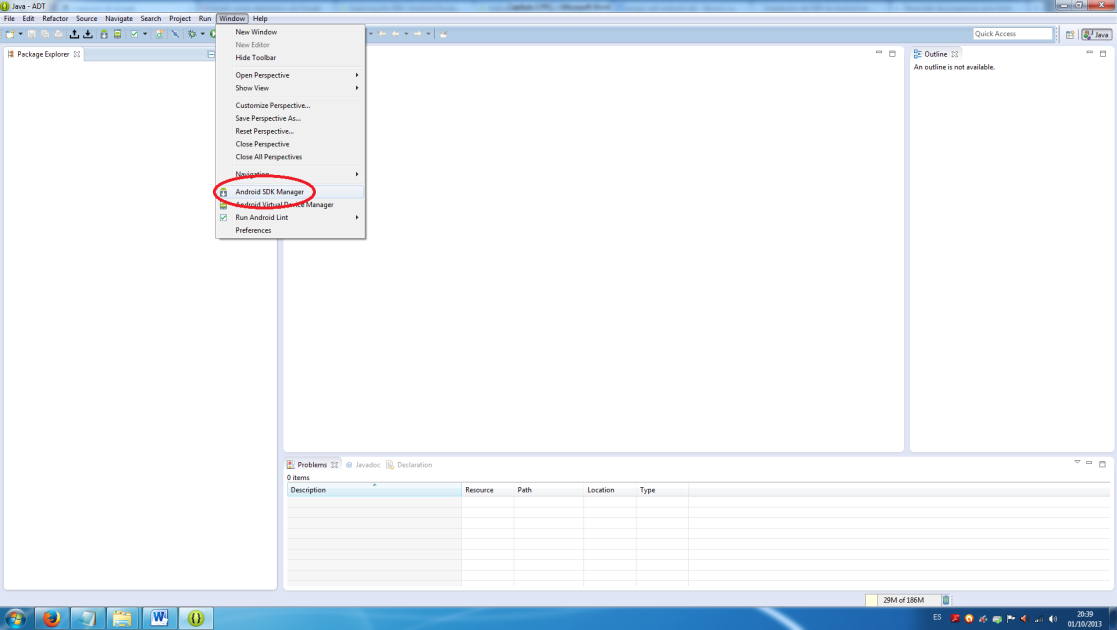


Ilustración 3‑13: Acceso Android SDK Manager

El SDK de Android separa herramientas, plataformas y otros componentes en paquetes que se pueden descargar a través del administrador del SDK, tal como se muestra en la Ilustración 3-14.

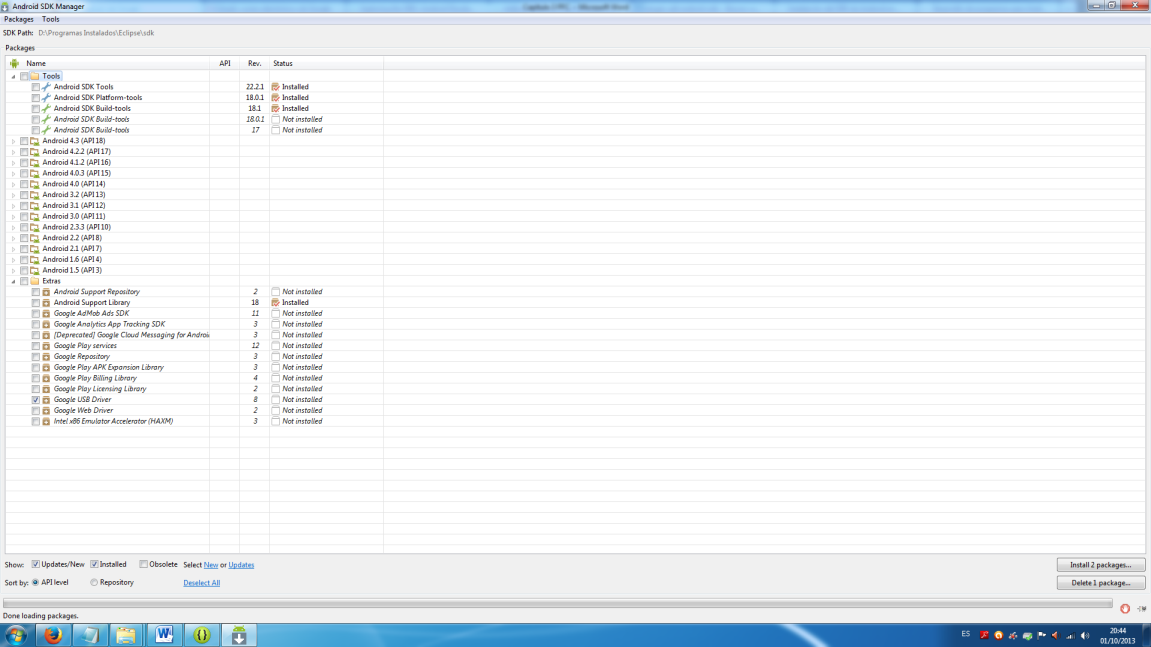


Ilustración 3‑14: Android SDK Manager

#### Creación Proyecto IOIO

La comunidad IOIO proporciona una serie de bibliotecas (IOIOLib, IOIOLibBT y IOIOLibAccessory) que permiten el control de la placa IOIO por parte de nuestra aplicación Android. Las bibliotecas exponen un conjunto de interfaces Java, que abarcan las diversas características de la placa IOIO. Al crear la aplicación, IOIOLib se empaqueta en nuestro archivo “.apk”, dando independencia a nuestra aplicación que no requiere ninguna instalación adicional.

IOIOLib es la biblioteca principal que se utiliza para la creación de la interfaz entre nuestra aplicación Android y la placa IOIO. IOIOLibBT y IOIOLibAccessory son bibliotecas adicionales para la comunicación inalámbrica Bluetooth y el protocolo de “*Open Accessory*” de Android, respectivamente.

Dichas bibliotecas se encuentran separadas ya que cada una de ellas requiere una versión de Android distinta, IOIOLib funciona en cualquier versión de Android (v1.5 en adelante), IOIOLibBT se ha introducido para v2.x de Android en adelante y el protocolo “*Open Accessory*” está disponible en determinados dispositivos con versión de Android 2.3.4 y dispositivos posteriores.

Además del *software*, la descarga incluye alguna aplicación a modo de ejemplo.

Las bibliotecas de IOIO han de ser importadas a nuestro proyecto para poder utilizarlas en nuestra aplicación Android. Para ello, una vez descargadas (https://github.com/ytai/ioio/wiki/Downloads), seleccionamos en nuestro entorno de desarrollo Eclipse en “*File*” > “*Import*”, tal como se muestra en la Ilustración 3-15.

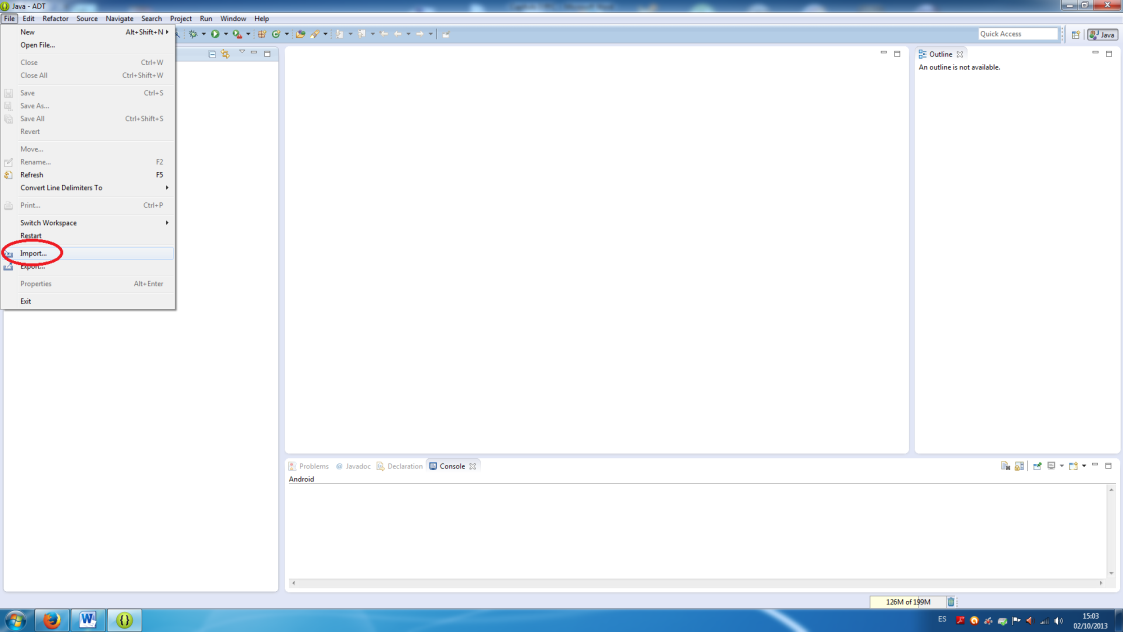


Ilustración 3‑15: 1º Parte Importación Bibliotecas IOIO

A continuación, seleccionamos en “*General*” > “*Existing Projects into Workspace*” > “*Next*”, tal como se muestra en la Ilustración 3-16.

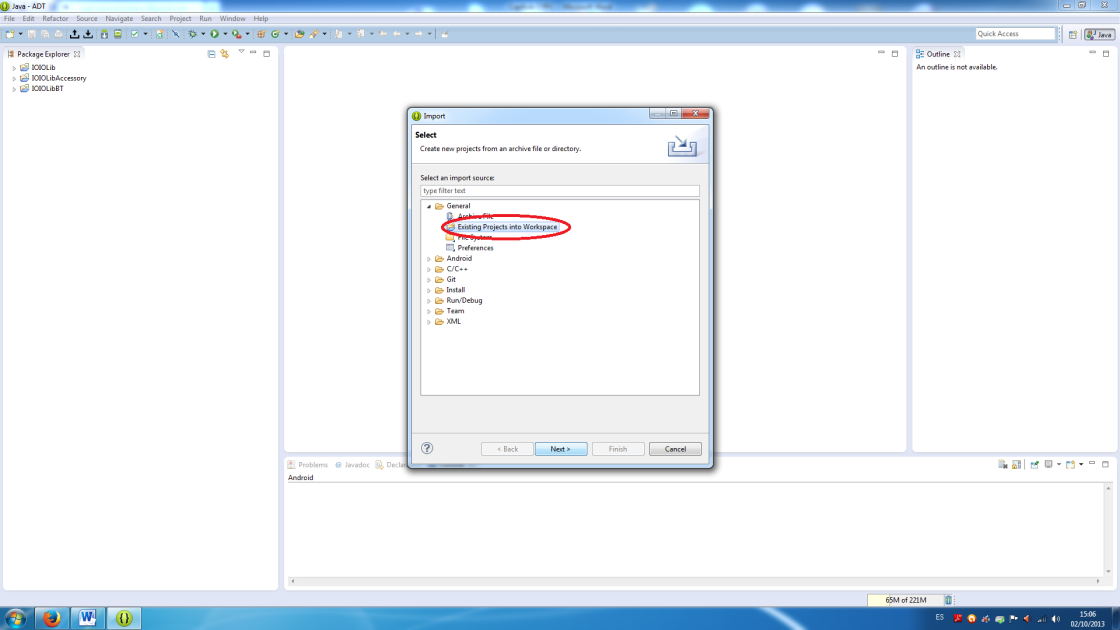


Ilustración 3‑16: 2º Parte Importación Bibliotecas IOIO

Seguidamente, seleccionamos el directorio donde tenemos descargadas las librerías de IOIO y los proyectos (que en este caso son las librerías) que deseamos incluir, y por último pulsamos en “*Finish*”.

Cabe mencionar que a la hora de importar las librerías de IOIO a Eclipse, se pueden producir diversos errores, dichos errores suelen estar descritos en la Wiki de IOIO (<https://github.com/ytai/ioio/wiki/Eclipse-Troubleshooting>), así como la solución a los mismos.

Una vez importadas las librerías IOIO y creado nuestro proyecto de aplicaciones Android (en blanco), podemos comenzar con la programación de nuestra aplicación. El punto de partida es crear una instancia con la interfaz IOIO, para ello hay 2 caminos posibles, una configuración “manual” con las funciones que se proporcionan en la Wiki de IOIO o utilizar un marco de trabajo proporcionado por la comunidad IOIO.

La recomendación por parte de la comunidad IOIO es utilizar el marco de trabajo, ya que en él, se preveen ciertas complejidades que en la configuración “manual” no se tienen en cuenta, favoreciendo el correcto funcionamiento entre la aplicación Android y la placa IOIO.

A continuación se muestra el marco de trabajo recomendado para la realización de nuestra aplicación Android:

**public** **class** MainActivity **extends** IOIOActivity

{

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

}

**class** Looper **extends** BaseIOIOLooper

{

@Override

**protected** **void** setup() **throws** ConnectionLostException

{

}

@Override

**public** **void** loop() **throws** ConnectionLostException

{

}

}

@Override

**protected** IOIOLooper createIOIOLooper()

{

**return** **new** Looper();

}

}

Además de esto, como mínimo en nuestra aplicación Android debe declararse el permiso “*android.permission.INTERNET*”. Esto se configura abriendo el archivo “*AndroidManifest.xml*”, pestaña “*Permissions*” > “*Add*” > “*Uses Permission*” > y seleccionar en nombre “*android.permission.INTERNET*”, tal como se muestra en la Ilustración 3-17.

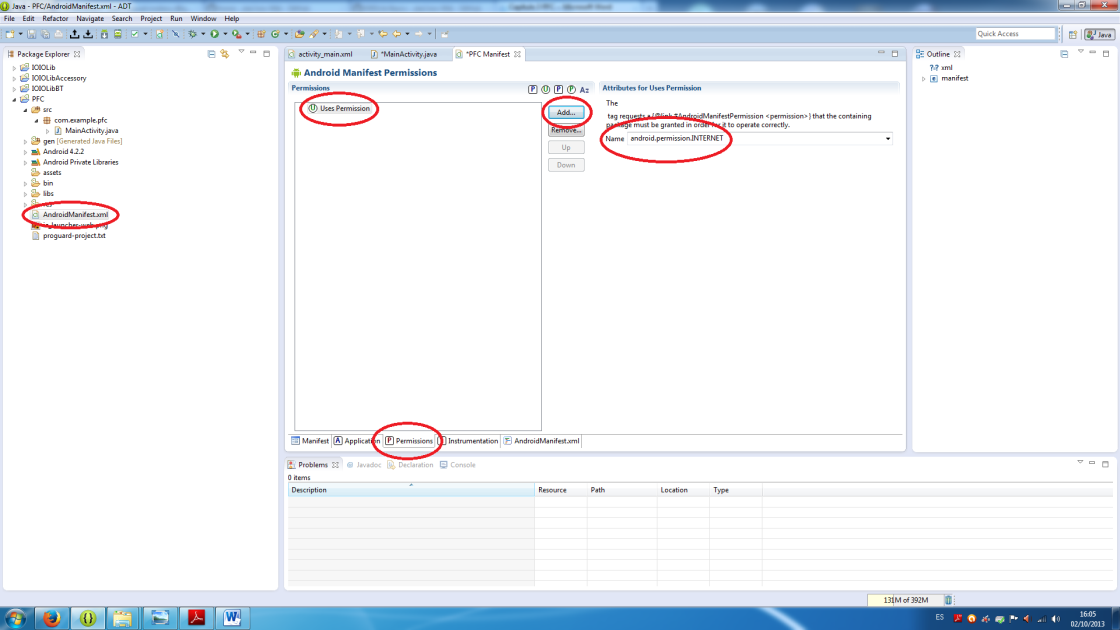


Ilustración 3‑17: Declaración Permisos Android

## Arduino

En esta sección se va a describir tanto el *hardware* como el *software* de Arduino UNO [10], así como el entorno de desarrollo necesario para la realización de un proyecto con Arduino.

Arduino es una plataforma de desarrollo de computación física (*physical computing*) de código abierto, basada en una placa con un sencillo micro-controlador y un entorno de desarrollo para crear *software* (programas) para la placa.

Arduino se usa para crear objetos interactivos, leyendo datos de una gran variedad de interruptores y sensores y controlar multitud de luces, motores y otros actuadores físicos. Los proyectos de Arduino pueden ser autónomos o comunicarse con un programa (*software*) que se ejecuta en un ordenador.

El lenguaje de programación de Arduino es una implementación de *Wiring*, una plataforma de computación física parecida, que a su vez se basa en *Processing*, un entorno de programación multimedia.

Algunas de las ventajas de Arduino respecto a otros sistemas son las siguientes:

* Asequible: Las placas Arduino son más asequibles comparadas con otras plataformas de micro-controladores.
* Multi-Plataforma: El software de Arduino funciona en los sistemas operativos Windows, Macintosh OSX y Linux.
* Entorno de programación simple y directo: El entorno de programación de Arduino es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados.
* *Software* ampliable y de código abierto: El *software* Arduino está publicado bajo una licencia libre y preparado para ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías de C++.
* *Hardware* ampliable y de código abierto: Arduino está basado en los micro-controladores ATMEGA168, ATMEGA328 y ATMEGA1280. Los planos de los módulos están publicados bajo licencia *Creative Commons*, por lo que los diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliándolo u optimizándolo.

### Hardware Arduino UNO

#### Vista Conjunta



Ilustración 3‑18: Vista Conjunta Arduino UNO

La placa de Arduino UNO está basada en el micro-controlador ATMEGA328. Tiene 14 pines digitales de entrada/salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, una conexión USB utilizada para subir programas a la placa y para la comunicación serie entre la placa y el ordenador (puede utilizarse como alimentación a la placa), un conector Jack de alimentación externa, un programador serie en circuito “*In-circuit Serial Programmer*” o “ICSP” y un botón de reinicio.

Incluye 2 pines (SDA y SCL) localizados cerca del pin AREF (terminal de referencia analógica), internamente un selector de alimentación entre el USB y el conector de alimentación externa. El pin IOREF permite que los *Shields* se adapten al voltaje empleado por la placa y el pin reset utilizado para añadir un botón de reset a los *Shields* que bloquean el de la placa principal.

#### Resumen

Micro-controlador ATMEGA328

Voltaje Funcionamiento 5 V

Voltaje Entrada (Recomendado) 7 – 12 V

Voltaje Entrada (Límites) 6 – 20 V

Pines E/S Digitales 14 (6 proporcionan salida PWM)

Pines Entrada Analógica 6

Corriente Pines E/S 40 mA

Corriente Pin 3.3 V 50 mA

Memoria Flash 32 KB (0.5 KB para el *bootloader*)

SRAM 2 KB

EEPROM 1 KB

Velocidad Reloj 16 MHz

#### Memoria

El ATMEGA328 tiene 32 KB de Memoria Flash para almacenar código (de los cuales 0.5 KB se usan para el *bootloader*). Tiene 2 KB de SRAM y 1 KB de EEPROM (que puede ser leída y escrita con la librería EEPROM).

#### Funciones Pines E/S

Cada uno de los 14 pines digitales de Arduino UNO puede ser utilizados como entrada o salida, utilizando las funciones “*digitalRead()*”, “*digitalWrite()*” y “*pinMode()*”. Funcionan a 5 V. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia pull-up interna (desconectada por defecto) de 20 – 50 KOhm. Además, algunos pines tienen funciones especiales:

Serie: Pines 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y de transmisión (TX) de datos en serie TTL.

Interrupciones Externas: Pines 2 y 3. Estos pines pueden ser configurados para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio de valor.

PWM: Pines 3, 5, 6, 9, 10 y 11. Proporcionan salida PWM de 8 bits.

SPI: Pines 10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK). Estos pines soportan la comunicación SPI usando la librería SPI.

Arduino UNO tiene 6 entradas analógicas, numeradas A0 – A5, cada una de las cuales proporcionan 10 bits de resolución, es decir, 1024 valores diferentes.

TWI: Pines A4 o SDA y pines A5 o SCL. Soporta comunicación TWI usando la librería Wire.

AREF: Voltaje de referencia para entradas analógicas.

Reset: Pin para resetear el micro-controlador, normalmente se utiliza para agregar un botón de reinicio para los *Shields* que bloquean la que está en la placa principal.

#### Comunicación

El Arduino UNO tiene un número de infraestructuras para comunicarse con un ordenador, otro Arduino, u otros micro-controladores. El ATMEGA328 provee comunicación serie UART TTL (5 V), la cual está disponible en los pines digitales 0 (Rx) y 1 (Tx). Un FTDI FT232RL en la placa canaliza esta comunicación serie al USB y los drivers FTDI (incluidos con el *software* Arduino) proporcionan un puerto de comunicación virtual al *software* del ordenador. El *software* Arduino incluye un monitor serie que permite a datos de texto simple ser enviados a y desde la placa Arduino.

El ATMEGA328 tambien soporta comunicación I2C (TWI) y SPI.

#### Programación

El Arduino UNO puede ser programado con el *software* Arduino vía USB.

El ATMEGA328 del Arduino UNO viene con un *bootloader* pregrabado que te permite subirle nuevo código sin usar un programador *hardware* externo. Se comunica usando el protocolo original STK500. Es algo no habitual en otros tipos de micro-controladores que generalmente requieren un programador externo.

Tambien puedes saltar el *bootloader* y programar el ATMEGA328 a través de la cabecera ICSP (*In-Circuit Serial Programming*).

#### Protección de Sobrecarga del USB

El Arduino UNO tiene un fusible reseteable que protege los puertos USB del ordenador de cortes y sobrecargas. Aunque la mayoría de los ordenadores proporcionan su propia protección interna, el fusible proporciona una capa de protección extra. Si se aplican más de 500 mA al puerto USB, el fusible automáticamente romperá la conexión hasta que el corte o la sobrecargar sean eliminados.

### Alimentación

El Arduino UNO puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa, la selección se hace de manera automática internamente.

La placa puede funcionar con un suministro externo de 6 - 20 V. Si se proporcionan menos de 7 V, el pin de 5 V puede suministrar menos de 5 V y la placa puede ser inestable. Si se alimenta con más de 12 V, el regulador de voltaje se puede sobrecalentar y dañar la placa. El rango recomendado es de 7 V – 12 V.

Vin: Se utiliza para la alimentación de la placa. El voltaje suministrado debe ser entre 7 V – 12 V. En caso de alimentar a la placa mediante el conector de alimentación externo se puede utilizar este pin como salida.

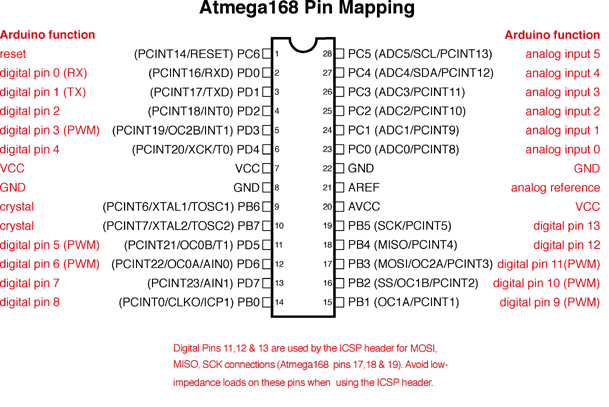
5 V: Salida a 5 V. No se recomienda alimentar la placa a través de este pin, ya que no pasa por el regulador y puede dañar la placa.

3.3 V: Salida a 3.3 V, 50 mA de corriente máxima.

GND: Pines de tierra.

IOREF: Este pin de la placa Arduino UNO proporciona la referencia de tensión con la que opera el micro-controlador a los *Shields*.

### Mapeado entre ATMEGA168/328 y Arduino



### Software

#### Introducción

El entorno de código abierto Arduino hace fácil escribir código y cargarlo a la placa. Es un entorno multiplataforma, pudiendo ser utilizado en Windows, Macintosh y Linux.

El entorno de desarrollo (IDE, *Integrated Development Environment*) está escrito en Java y basado en *Processing*, *avr-gcc* y otros programas también de código abierto.

Es un entorno fácil de usar, es simple y directo, dando flexibilidad tanto a usuarios principiantes como avanzados.

La programación de la placa de Arduino se hace vía USB, contando con una amplia y activa comunidad de usuarios.

Se puede descargar libremente desde la página oficial de Arduino (<http://arduino.cc/en/Main/Software>) y no requiere instalación.

#### Bibliotecas

Arduino ofrece una serie de bibliotecas “estándar” basadas en C/C++ que se pueden importar al Sketck, las bibliotecas “estándar” son las siguientes:

* EEPROM: Para leer y escribir en memorias “permanentes”.
* Ethernet: Para conectar a internet usando el Ethernet Shield.
* Firmata: Para comunicarse con aplicaciones en la computadora usando un protocolo estándar Serial.
* LiquidCrystal: Para controlar Displays de cristal líquido (LCD).
* Servo: Para controlar servomotores.
* SoftwareSerial: Para la comunicación serial de cualquier pin digital.
* Stepper: Para controlar motores paso a paso.
* Wire: Interfaz de dos cables (TWI/I2C), para enviar y recibir datos a través de una red de dispositivos y sensores.

#### Diseño Sketch

El diseño de un Sketch para Arduino no requiere de avanzados conocimientos en electrónica y en programación. Así, por ejemplo, para configurar una línea digital como salida se puede utilizar la siguiente función “pinMode(Pin13, OUTPUT)”. De mismo modo, para escribir un valor alto o bajo en dicho pin se utiliza la siguiente función “digitalWrite(Pin13, HIGH o LOW)”.Ademas, para realizar tareas más complejas, tales como gestionar una comunicación por UART, se pueden utilizar las siguientes funciones “Serial.begin(speed)”, “Serial.read()”, “Serial.available()”, etc…

La estructura básica del lenguaje de programación Arduino es bastante simple y se organiza en al menos dos partes o funciones que encierran bloques de declaraciones.

void setup()

{

Statements;

}

void loop()

{

Statements;

}

Ambas funciones son requeridas para que el programa funcione.

* setup(): La función “setup” debería contener la declaración de cualquier variable al comienzo del programa. Es la primera función a ejecutar en el programa, es ejecutada una vez y usada por ejemplo para inicializar las comunicaciones serie.
* loop(): La función “loop” se ejecuta a continuación e incluye el código que se ejecuta continuamente leyendo entradas, activando salidas, etc. Esta función es el núcleo de todos los programas Arduino y hace la mayor parte del trabajo.

### Entorno de Desarrollo

Como ya se ha mencionado anteriormente, el entorno de desarrollo de Arduino es gratis y no requiere instalación (<http://arduino.cc/en/Main/Software>).

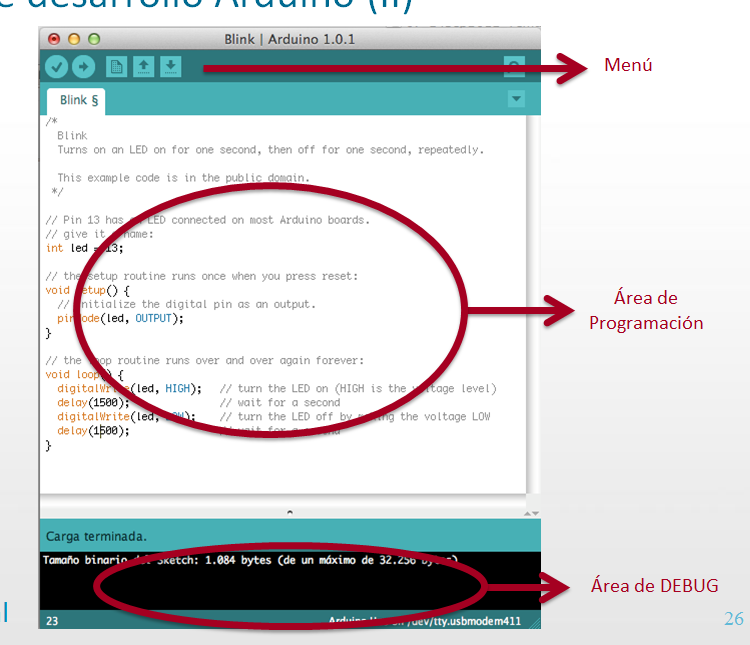


Ilustración 3‑19: Entorno de Desarrollo Arduino

En el siguiente enlace se encuentra toda la información necesaria para la instalación del software en la plataforma Windows, Mac o Linux (<http://arduino.cc/es/Guide/HomePage>).



Ilustración 3‑20: Menú Entorno de Desarrollo

El software desarrollado con Arduino se conoce como sketches. Los sketches se escriben con un editor de texto y son guardados con la extensión “.ino”.

En la Ilustración 3-21, se muestra como seleccionar el puerto.

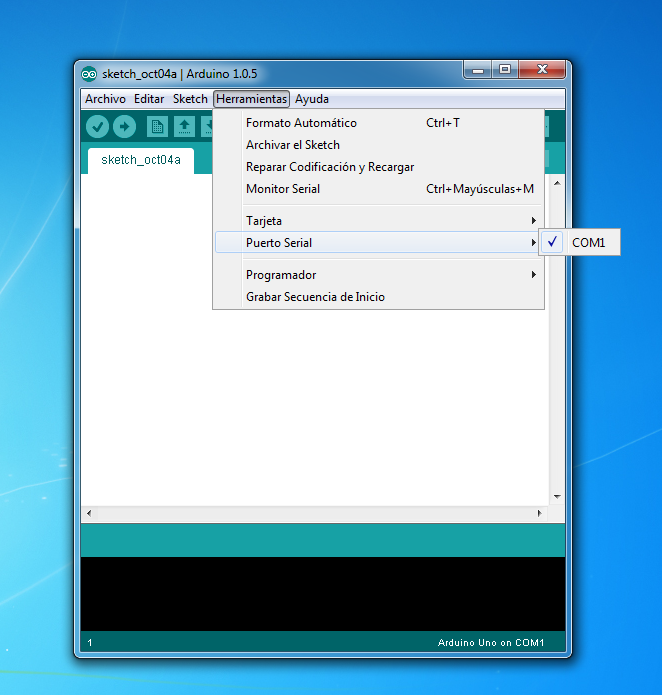


Ilustración 3‑21: Menú de selección de puerto del Entorno Arduino

En la Ilustración 3-22, se muestra como seleccionar la placa de Arduino con la que estemos trabajando.

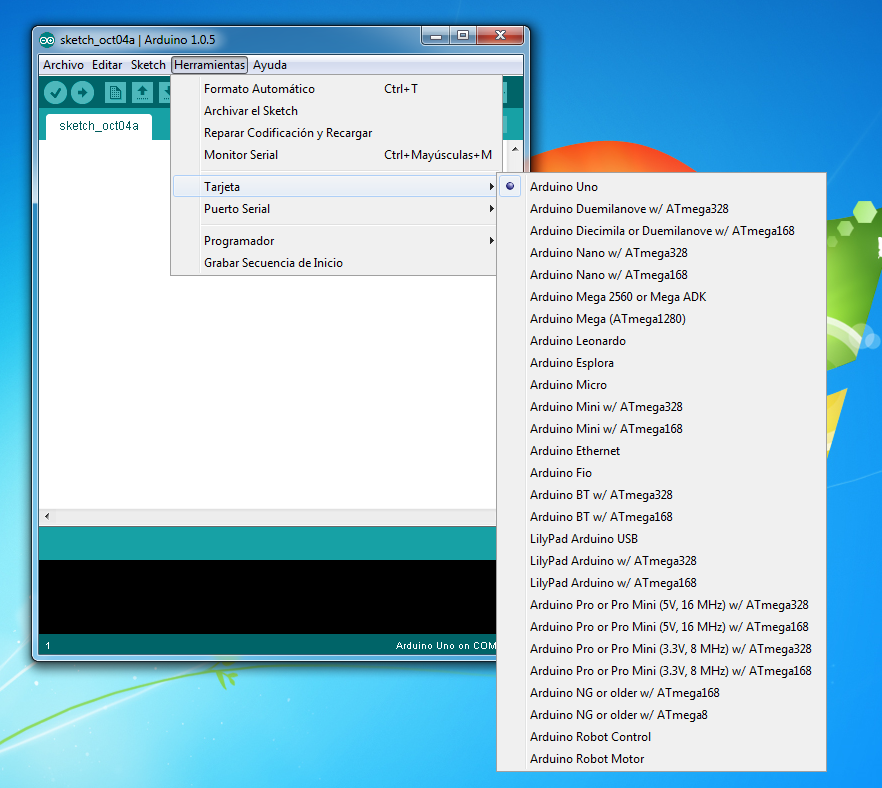


Ilustración 3‑22: Menú de selección de placa del Entorno Arduino

### Shields

Las Shields son placas que pueden ser conectadas encima de la placa Arduino extendiendo sus capacidades. Para ello los pines de sus puertos guardan una disposición de compatibilidad.

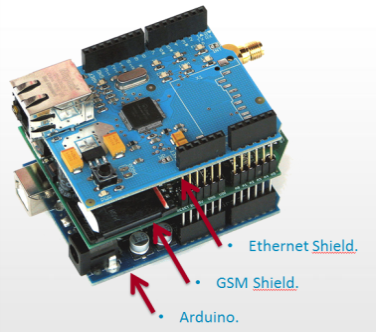


Ilustración 3‑23: Arduino & Shields

Las diferentes Shields siguen la misma filosofía que el conjunto original: son fáciles de montar y baratas de producir.

Existe una gran variedad de Shields con diversa funcionalidad: control de motores, comunicaciones, prototipado rápido, etc…Ademas, de incluir una biblioteca software con objeto de poder desarrollar un prototipo con poco tiempo de diseño invertido.