# Capítulo 7 – Comunicación NodeJS con Arduino

Este capítulo introduce sobre un framework denominado Johnny-five el cual es utilizado para las comunicaciones entre la aplicación web y las placas Arduinos que componen al SAR. Además, se explica en detalle el protocolo subyacente a esta librería, conocido como Firmata, que en esta tesina es implementado en cada uno de los Arduinos.

## **7.1 Johnny-five**

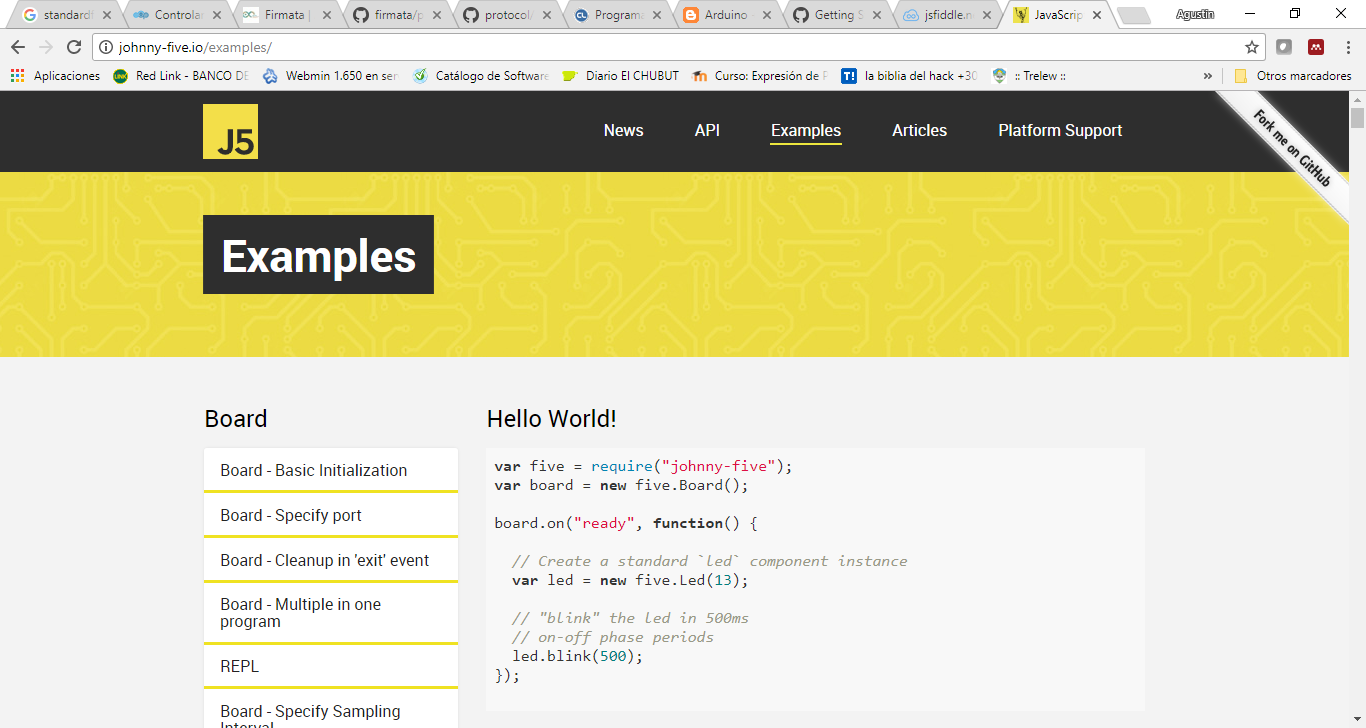
Johnny-five, más conocido como J5, es un framework de programación robótica basado en JavaScript lanzado por la compañía Bocoup en el 2012 bajo licencia abierta y que ha logrado ser adoptado tanto por desarrolladores como por ingenieros que cómo comunidad no solo son usuarios, sin que ptambién provee soporte, mejoras y nuevas características al Framework. Una captura de pantalla de su sitio se encuentra en (Ilustración 42 - Página oficial de Johnny-Five).

Ilustración – Sitio web oficial de Johnny-Five (<http://johnny-five.io/>)

Es compatible con una gran mayoría de microcontroladores y SBC más populares, incluyendo los utilizados en el desarrollo de esta tesina: Arduino y Raspberry Pi (ambos en todas sus versiones). Dentre de la variedad de dispositivos soportados se encuentran las placas como BeagleBone, SparkFun, ChipKit, Intel-Galileo, entre otras.

Su librería posee compatibilidad a una gran cantidad de sensores y actuadores, así como ejemplos de utilización. Éstos se encuentran disponibles en su página oficial, complementado con esquemas de conexión de los componentes, para cada plataformas soportada.

## **7.2 Instalación**

Para poder utilizar Johnny-Five, se debe contar con NodeJS (visto en el apartado **¡Error! No se encuentra el origen de la referencia.**). Se realiza a través de el gestor de paquetes de NodeJS, llamado npm, que permite la gestión de proyectos y administración de sofware de terceros.

Dentro del directorio del proyecto de debe ejecutar la sigueinte orden:

npm install Johnny-five

## **7.3 Arduino Firmata**

Firmata se trata de un protocolo serial, genérico orientado a la comunicación entre microcontroladores y una computadora. Al ser genérico, se puede implementar en cualquier arquitectura de microcontroladores asi como también las bibliotecas para utilizarlo desde la computadora se pueden implementar en cualquier lenguaje.

Ilustración – Firmata como interfaz

Su objetivo es permitir controlar completamente un microcontrolador de forma remota (Ilustración 43 - Remote Wiring de Windows), eliminando la necesidad de la escritura de código específico para cada microncontrolador[[1]](#endnote-1).

Podemos enumerar las siguientes ventajas:

* El programa no está limitado por la memoria RAM ni Flash de Arduino.
* El software de control se puede programar en cualquier lenguaje que tenga soporte para Firmata. Algunos jenguajes son por ejemplo: Firmata: Processing, Visual Basic, Perl, C#, PHP, Java, **JavaScript** (a través de JhonnyFive u otras bibliotecas), Ruby, Python, etc.

Sin embargo, podemos enumerar también una serie de Desventajas:

* Programas más restringidos, operaciones específicas como manejo pormenorizado de interrupciones no es posible.
* El microcontrolador deja de ser autónomo, es decir, depende de la conexión al computador para poder recibir comandos.

## **7.4 Surgimiento y funcionamiento de Firmata**

El protocolo surge con el objetivo de hacer del microcontrolador una extensión del entorno de desarrollo, quitando del foco la programación embebida y permiteiendo el uso de lenguajes de mayor nivel de abstracción o familiaridad con el desarrollador.

Su creación data del año 2006 como una demostración para Arduino por Hans-Chistoph Steiner, mientras trabajaba en un proyecto musical con dispositivos MIDI interconectados a varios Arduinos que contaban con varios sensores. La problmática de Hans era que debía replicar en cada microcontrolador cada cambio realizado.

Actualmente la implementación de referencia se encuentra en la biblioteca de Arduino IDE, incluida a partir de la versión 0012 y está disponible para cualquier placa compatible.

Firmata expone la API de Arduino en la computadora conectada al microcontrolador, permitnendo la operación remota. A través del puerto serie se codifican los mensajes utilizando la codificación del protocolo MIDI (Musical Instrument Digital Interface), un protodolo orientado a la comunicación de dispositivos musicales con computadoras.

Estos mensajes se conforman de bytes de comando seguidos de bytes de datos. Los bytes de comando son 8 bits y los bytes de datos son 7 bits. Por ejemplo, el mensaje MIDI Channel Pressure (Comando: 0xD0) tiene 2 bytes de longitud, en Firmata se utiliza para habilitar informes para un puerto digital (colección de 8 pines). Si bien Firmata respeta la catnidad de bytes de cada comando MIDI que reemplaza, hace uso de los mensajes de Midi System Exclusive (Sysex) para comunicaciones dónde se necesite un mensaje de longitud arbitraria.

Firmata permite operar tanto entradas y salidas analógicas y digitales. Soporta más de 16 pines analógicos con una resolución de 14 bits y más de 128 pines digitales.

Por ser de código abierto, se han implementado diferentes versiones con características específicas para dar soporte a gran variedad de funcionalidades más allá de la operación de I/O analógica y digital. La versión original, Standard\_Firmata, se incluye dentro de las versiones del entorno oficial de Arduino y Wiring e incluye soporte para las siguientes características[[2]](#endnote-2):

* Entradas y salidas analógicas
* Salidas PWM
* Conmutación entre entradas y salidas analógicas
* Control de Servomotores
* Matrices de LEDs
* Comunicación I2C

## **7.5 Métodos de librería Firmata en Arduino**

La librería Firmata de Arduino cuenta con un conjunto de métodos, relacionados con la API de Arduino (como se describió en el apartado anterior). A continuación, se detallan los más relevantes:

### 7.5.1 Métodos de propósito general

* **begin**(long): Comienza la librería, es posible utilizar otra velocidad diferente a la velocidad por defecto que es 57600 baudios. También es posible iniciar el protocolo Firmata desde otro Stream que no sea el que viene por defecto que es Serial.
* **printVersion()**: Envía la versión del protocolo al ordenador.
* **blinkVersion()**: Parpadea la versión de protocolo en el “build in LED”, generalmente el pin 13.
* **printFirmwareVersion():** Envía la versión de firmware y su versión al ordenador.
* **setFirmwareVersion(**byte major, byte minor**)**: Configura la versión del firmware.
* **setFirmwareNameAndVersion(**const char \*name, byte major, byte minor**)**: Configura nombre y versión del firmware.

### 7.5.2 Métodos para el envío de mensajes

* **sendAnalog(**byte pin, int value**)**: Envía el valor del pin analógico.
* **sendDigitalPort(**byte portNumber, int portData) : Envía el valor de un puerto digital de 8 bits.
* **sendString(**const char\* string**)**: Envía un string a una computadora.
* **sendString(**byte command, byte bytec, byte \*bytev**)**: Envía un string a la computadora usando un tipo de comando.
* **sendSysex(**byte command, byte bytec, byte\* bytev**)**: Envía un comando un con array de bytes
* **write(**byte c**)** – Envía un byte al stream de datos.

### 7.5.3 Métodos para la recepción de mensajes

* **available()**: Comprueba si hay algún mensaje entrante en el buffer.
* **processInput()**: Procesar los mensajes entrantes que hay en el buffer, mandado los datos a cualquiera de las funciones de callback registradas.
* **attach(**byte command, callbackFunction myFunction**)**: Registrar una función a un tipo de mensaje entrante.
* **detach(**byte command**)** : Suprimir la función del tipo de mensaje

### 7.5.4 Otros métodos

* **sendValueAsTwo7bitBytes(**int value) : Escribe el valor como 2 bytes.
* **startSysex(**void**)**: Comenzar mensaje sysex.
* **endSysex(**void**)**: Finalizar mensaje sysex[[3]](#endnote-3).

## **7.6 Instalación de Firmata en Arduino**

Con la instalación del IDE Arduino en una computadora se incluyen librerías y ejemplos que permiten manipular diversos componentes que se conecten a la plataforma. Dentro de estos ejemplos de códigos se encuentran los del protocolo Firmata. Para instalar Firmata se necesita tener conectada la placa Arduino a la computadora a través de un puerto USB. Luego de ello se debe seleccionar desde el IDE de Arduino (Ilustración 44 - IDE de Arduino) el código Firmata, dentro de su respectiva librería, según los dispositivos que se requieran comunicar con el mismo. Para ello se debe ir a Archivo 🡪 Ejemplos 🡪 Firmata.

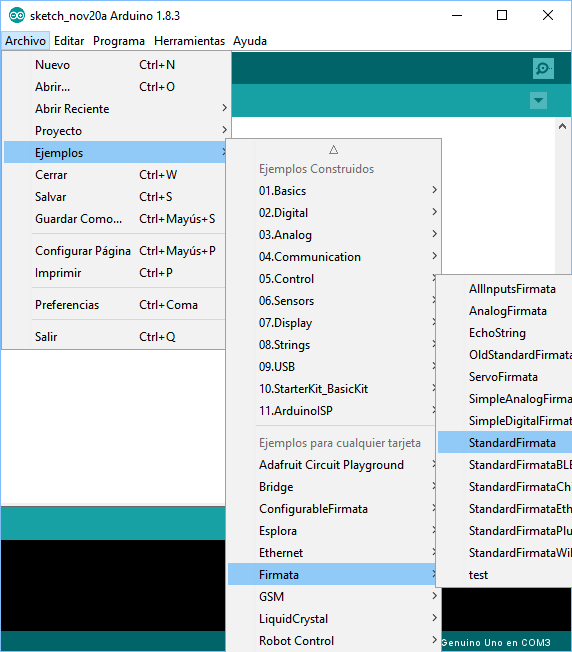


Ilustración - IDE de Arduino

En nuestro caso se utilizaron dos códigos Firmata:

**StandardFirmata**: Es, como su nombre lo indica, el estándar del protocolo que permite la comunicación con la mayoría de los componentes compatibles con Arduino. En nuestro caso, es el utilizado para cargarlo dentro del Arduino Mega para manipular la mayoría de sensores y actuadores del SAR. (Anexo de código)

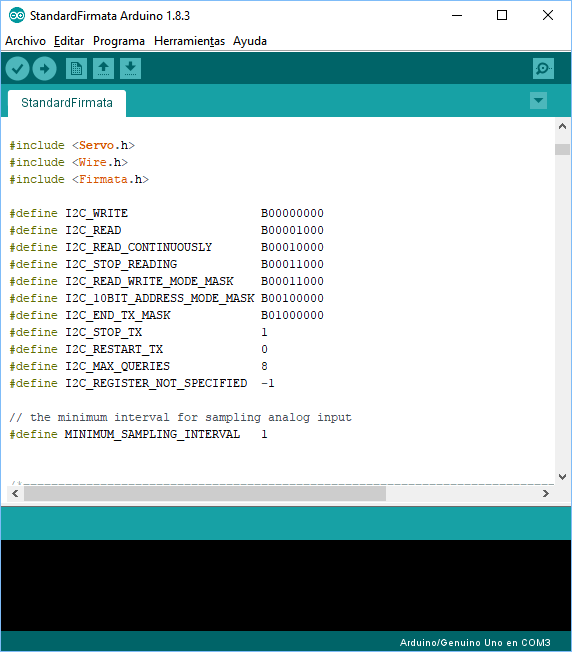


Ilustración - Código StandardFirmata

**ConfigurableFirmata**: Esta versión personalizada del protocolo[[4]](#footnote-2), es la utilizada para captar la temperatura mediante el sensor para dicho fin conectado a un Arduino Nano. Permite separar las características del protocolo en clases individuales, haciendo más sencillo mezclar las características estándar del protocolo con otras personalizadas. (Anexo de código)

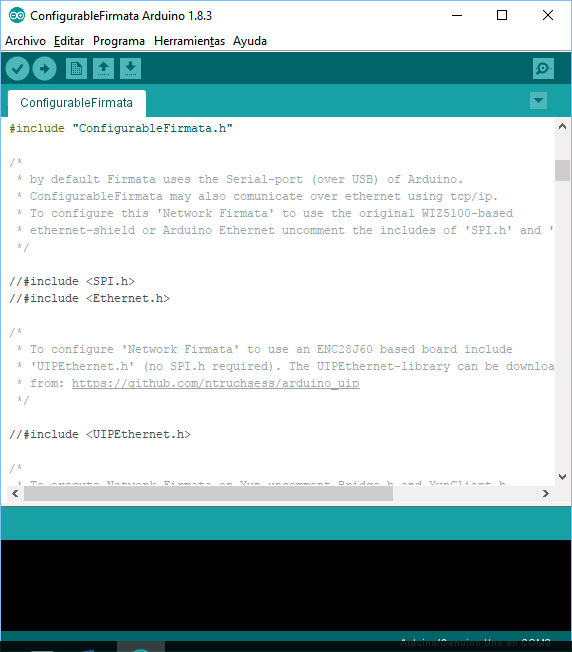


Ilustración - Código ConfigurableFirmata

## **Resumen**

Como se pudo ver, en este capítulo, se abordo la utilidad en el SAR de un framework denominado Johnny-five, el cual es utilizado para la programación robótica y está basado en JavaScript utilizando Firmata.

A continuación éste protocolo mencionando sus ventajas y desventajas en su utilización. Se explicó su surgimiento y su funcionamiento en general, como es el formato de sus mensajes. Finalmente se vio como instalar y/o cargar este protocolo en la familia de placas Arduino y los dos códigos utilizados en el desarrollo de esta tesina (StandardFirmata y ConfigurableFirmata)

Agregar los códigos StandardFirmata y ConfigurableFirmata (en Anexos de códigos)

1. <https://github.com/firmata/arduino> [↑](#endnote-ref-1)
2. <https://programarfacil.com/podcast/arduino-day-protocolo-de-comunicaciones-firmata/> [↑](#endnote-ref-2)
3. <https://www.arduino.cc/en/Reference/Firmata> [↑](#endnote-ref-3)
4. Se puede obtener del siguiente sitio web <https://github.com/firmata/ConfigurableFirmata> [↑](#footnote-ref-2)