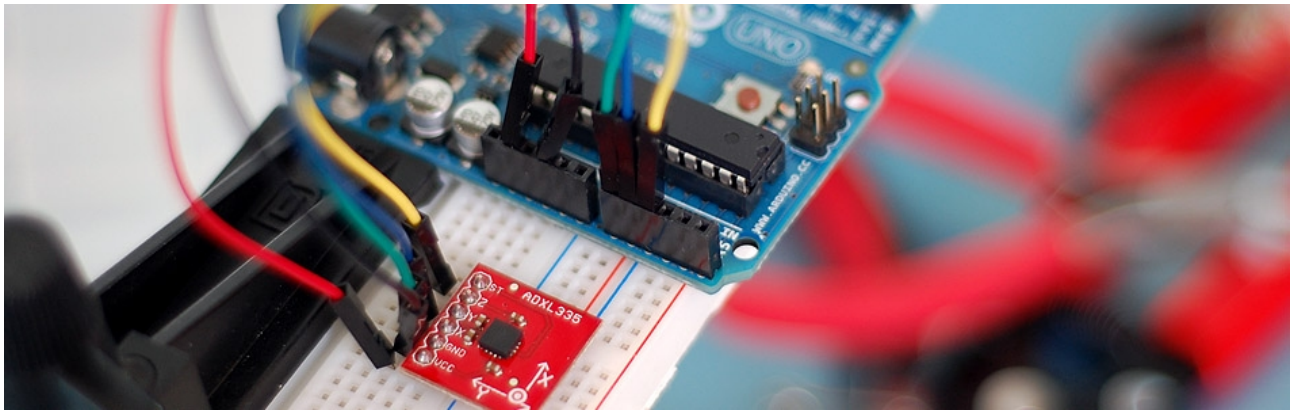
Seminario
Programación de Microcontroladores con Arduino

Manipulación de objetos en 3D usando Arduino



Cálcena Eugenio - Charkiewicz Gastón

SOY UN ÍNDICE

Qué quisimos hacer?

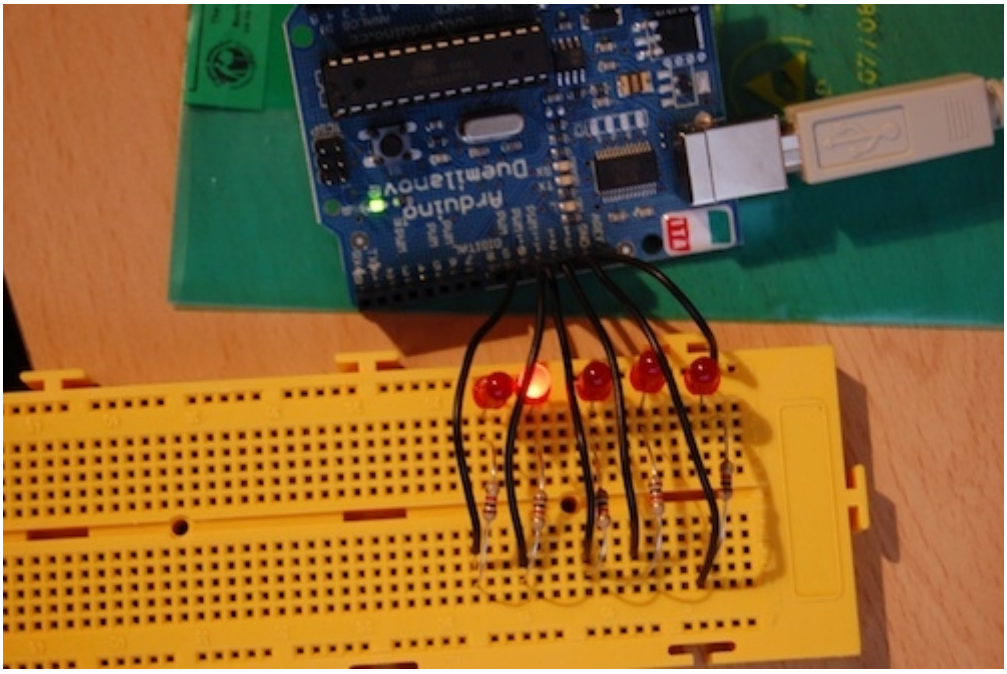
Todo comenzó con la unión de dos ideas: por un lado, la utilización de un protocolo de transmisión a través de Bluetooth; por el otro, la manipulación de objetos 3D utilizando un Arduino y hardware que lo haga posible. Es así como, mezclando ambas ideas, decidimos implementar una figura en 3D, para manipularla a través de un Arduino, emparejado vía Bluetooth.

Para poder entender qué fue lo que quisimos hacer, primero es necesario entender todas las tecnologías que están involucradas en el proceso.

Qué tecnologías están involucradas en el proceso?

- Arduino
- Lenguaje de Programación Arduino
- Bluetooth
- Bluetooth, modelo específico: JY-MCU (Chip CSR BC417143)
- Acelerómetro de 3 ejes
- Acelerómetro, modelo específico: ADXL335
- Puerto Serial
- Java
- Eclipse
- Applets
- RXTX, librería para manejo de Puerto Serial a través de Java

- Arduino



Arduino [1] es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.
Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).
Las placas pueden ser hechas a mano o compradas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta, así pues eres libre de adaptarlos a tus necesidades.
Arduino recibió una Mención Honorífica en la sección *Digital Communities* de la edición del 2006 del *Ars Electronica Prix*. El equipo Arduino (Arduino team) es: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, y David Mellis.

Para el desarrollo y pruebas, utilizamos dos Arduino Duemilanove, con microcontroladores ATmega168 y ATmega328 respectivamente.

- Lenguaje de Programación Arduino

La plataforma Arduino se programa mediante el uso de un lenguaje propio [2] basado en el popular lenguaje de programación de alto nivel Processing. Arduino está basado en C y soporta todas las funciones del estándar C y algunas de C++.

- Bluetooth [3]

Es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Los principales objetivos que se pretenden conseguir con esta norma son:

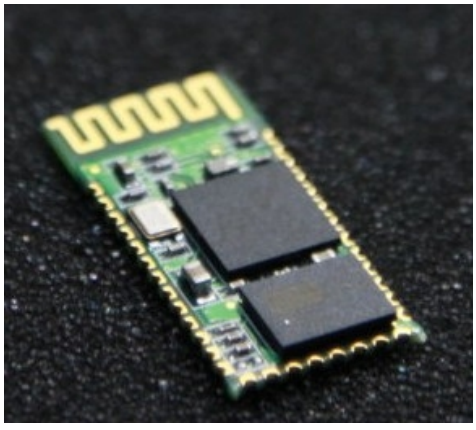
- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Para utilizar una cierta tecnología Bluetooth un dispositivo deberá soportar ciertos perfiles. Un perfil Bluetooth es la especificación de una interfaz de alto nivel para su uso entre dispositivos Bluetooth. Los perfiles son descripciones de comportamientos generales que los dispositivos pueden utilizar para comunicarse, formalizados para favorecer un uso unificado. La forma de utilizar las capacidades de Bluetooth se basa, por tanto, en los perfiles que soporta cada dispositivo. Los perfiles permiten la manufactura de dispositivos que se adapten a sus necesidades.

Como mínimo, una especificación de perfil debe cubrir:

- Dependencias con otros perfiles.
- Formatos recomendados para la interfaz con el usuario.
- Partes concretas de la pila Bluetooth que se utilizan (opciones particulares, parámetros). Puede incluir una descripción del tipo de servicio requerido.

- Bluetooth, modelo específico: JY-MCU (Chip CSR BC417143) [4]



Se trata de un módulo Bluetooth que implementa comunicaciones a través del protocolo serie RS232, como una especie de adaptador Bluetooth - serie. Es muy pequeño (4,4 x 1,6 x 0,7 cm) y liviano (7 g). Se usa para enviar y recibir datos serie a través una conexión inalámbrica Bluetooth y permite velocidades serie desde 1200 a 115200 bps.

Posee el chip CSR BlueCore4 Ext que implementa Bluetooth V2.0 Clase II y perfil HCI (I2C, UART, PCM y USB), además posee 8 Mb flash.

- Acelerómetro de 3 ejes

Se denomina **acelerómetro** [5] a cualquier instrumento destinado a medir aceleraciones. La aceleración se define como el cambio de la velocidad en el tiempo. La aceleración medida por este instrumento se conoce como aceleración propia y no es igual a la aceleración de coordenadas (que depende del sistema de coordenadas y del observador -sistema de referencia-) elegidos. Sino que es el tipo de aceleración asociada con el peso de una masa de prueba que se encuentra en el marco de referencia del dispositivo. El acelerómetro mide entonces todas las aceleraciones excepto aquellas debidas a la gravedad, mide peso por unidad de masa o comunmente llamado fuerza G o fuerza específica (si bien no es una fuerza).
Actualmente es posible construir acelerómetros de tres ejes (X,Y,Z) en un sólo chip de silicio, incluyendo en el mismo la parte electrónica que se encarga de procesar las señales. Sus protocolos de comunicación pueden variar (USART, I2C, SPI entre otros).
Para lo propuesto en este documento, utilizaremos uno de los mencionados anteriormente.

- Acelerómetro, modelo específico: ADXL335

El ADXL335 [6] es un acelerómetro de 3 ejes pequeños y de baja potencia.
Para poder ver la información completa, pueden inspeccionar su Data Sheet correspondiente, presente en las referencias de este mismo documento.

- Puerto Serial *(el viejo y nunca bien ponderado Puerto Serial)*

Un puerto serial [7] es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente.

- Java

Java [8] es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems en 1995. El lenguaje en sí mismo toma mucha de su sintaxis de C, Cobol y Visual Basic, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La memoria es gestionada mediante un recolector de basura. Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.

- Eclipse

Eclipse [9] es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.
Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.
Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

Líneas de código fuente	2.063.083
Esfuerzo estimado de desarrollo (persona-año/persona-mes)	604,33 / 7.251,93
Estimación de tiempo (años-meses)	6,11 / 73,27
Estimación del nº de desarrolladores en paralelo	98,98
Estimación de coste	$ 81.636.459

- Applet

Un *applet* [10] es un componente de una *aplicación* que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El *applet* debe ejecutarse en un *contenedor*, que lo proporciona un programa anfitrión, mediante un *plugin,* o en aplicaciones como teléfonos móviles que soportan el modelo de programación por "applets".

A diferencia de un *programa*, un *applet* no puede ejecutarse de manera independiente, ofrece información gráfica y a veces interactúa con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un *applet* normalmente lleva a cabo una función muy específica que carece de uso independiente. El término fue introducido en AppleScript en 1993.

La única consideración tomada para elegir el uso de applets en este proyecto, fue su simpleza.

-RXTX

RXTX es un library para comunicación con el Puerto Serial para Java.

Luego de toda esta información aburrida pero necesaria, vamos con la parte divertida :)

Cómo lo hicimos?

Primero, modelamos un cubo 3D en Java, y lo utilizamos dentro de un Applet. Luego de asegurarnos de que podíamos rotarlo utilizando el mouse, decidimos comenzar a adaptarlo.
Luego de programar el Arduino para que sense las mediciones del acelerómetro de 3 ejes, adaptamos el código Java que teníamos para que, utilizando RXTX, reciba del Puerto Serial los datos sensados.
Finalmente, hicimos la integración total, logrando que el cubo en el Applet se moviera acorde a los movimientos aplicados al acelerómetro.
Por la parte de la comunicación inalámbrica, se comenzó a trabajar en la conexión entre el módulo bluetooth (ligado al arduino) y un módulo genérico externo conectado a la computadora. Por último se utilizó un módulo integrado para realizar esta conexión (ya que no se logró establecer conexión mediante el módulo externo).
Se probó en primer lugar la conexión entre módulos y luego la recepción de datos serie a través de la conexión Bluetooth.

Relevancias acerca del desarrollo

Porqué tiembla?
El movimiento del cubo no es constante y totalmente sincronizado debido a lo siguiente: el acelerómetro sensa una aceleración exacta estando quieto. Si se sacude, se mueve, o se deja caer, la aceleración que sensa no está completamente basada en la gravedad, y esto se traduce en la reducción de la precisión de las mediciones.
De todas formas, para que la medición estando quieto sea exacta, se debía despreciar la fuerza gravitatoria. Decidimos no hacer eso, porque entonces al moverlo no podríamos despreciarla, y el foco del proyecto era el movimiento, no el estado de quietud del objeto manipulado.

Porqué no sensa los movimientos yaw? (y qué es yaw???)
Se denomina yaw al movimiento similar a expresar “no” con el gesto de nuestra cabeza. Los acelerómetros no miden este movimiento, simplemente por el hecho de que los valores que éstos sensan (en este caso particular) no cambian. En caso de querer sensarlos, será necesario que trabajen junto con un giróscopo, o una brújula digital. Es interesante destacar que esto no es trivial.

Pero yo quiero unas mediciones más acertadas...
Bueno, entonces sería necesario utilizar este sistema, en conjunto con un giroscopio `[11]`; ambos formarían lo que se conoce como IMU (Inertial Measurement Unit, o Unidad de Medida Inercial). Un giroscopio es muy bueno para medir rotación, pero no comprende orientaciones. Un acelerómetro es bueno para determinar orientación, pero no tiene la habilidad de mantener registro de rotaciones.
Se debería coordinar ambos dispositivos, para que trabajen en conjunto.

Experiencia de Desarrollo

Dividimos el proceso de desarrollo en dos partes: el modelado y manipulación de elementos por un lado, y la comunicación Bluetooth por el otro.

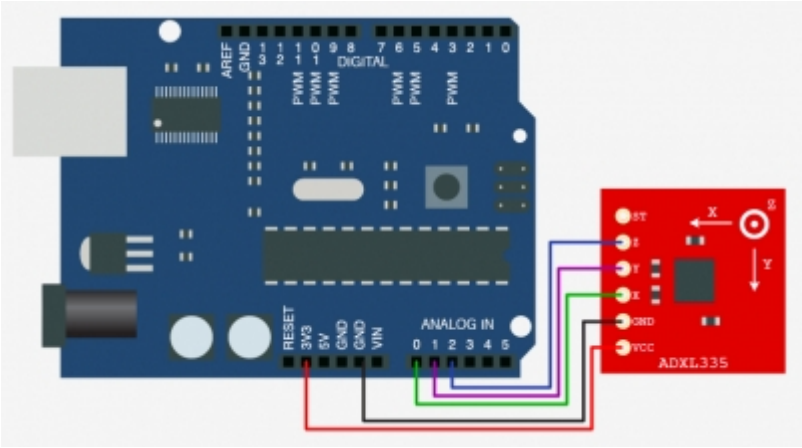
Modelado y manipulación de elementos (Gastón)

Luego de informarme acerca de lo que queríamos hacer, y entenderlo, busqué en Internet acerca de posibles proyectos que utilizacen los mismos recursos que nosotros. Encontré el siguiente tutorial, que me pareció muy interesante: http://blog.bricogeek.com/noticias/tutoriales/tutorial-arduino-acelerometro-3-ejes-adxl335-3g/ .
Entonces, simplemente bajé los programas, instalé Processing, y los corrí.
Para mi sorpresa*, no anduvo.
Luego de investigar el tema, llegué a la conclusión siguiente: Processing utiliza las libraries de RXTX para comunicarse con el Puerto Serial, y la versión que estaba utilizando tenía un conflicto en las mismas que impedía la detección y manejo de eventos de Serial. Así, que, comencé a ver otras tecnologías que pudieran sernos útiles (ver apartado).
Luego de decidir usar Java, y ya teniendo el código que usaría en el Arduino, utilicé un ejemplo (http://www.ubicuos.com/2010/03/04/rotacion-de-un-cubo-usando-java/) para tener un cubo 3D que pudiera manipular. Inspeccionando el código e informándome acerca de cómo sucedía, ví que el ejemplo se movía gracias a Events que se generaban a partir de movimientos del mouse. Así que investigué acerca de cómo podría hacerlo a través del puerto Serial, y finalmente volví a caer en RXTX. Utilizando una versión que sabía funcionaba (amablemente el profesor nos la cedió de un proyecto que había hecho antes) [12].
Restaba lograr que la comunicación se estableciera... así que, con ese fin, seguí adaptando código.

Luego de varios intentos, y gracias a la ayuda de un compañero que sabía utilizar el Puerto Serial (a diferencia mía, en ese entonces), logré establecer la comunicación, y recibir los datos de forma adecuada.
Pero me encontré con un problema: los movimientos eran bruzcos y casi completamente diferentes a los hechos en la realidad.
Probé calibrando los valores antes de enviarlos, pero no noté cambios significativos.

Fue así que caí en este tutorial muy interesante: http://bildr.org/2011/04/sensing-orientation-with-the-adxl335-arduino/ . Me percaté de que no alcanzaba con simplemente enviar los datos sensados, había que mapear cada dato a una posición dentro de los 360 grados posibles de cada eje. Entonces, luego de cambiar el programa en el Arduino, modifiqué el código del Applet para que las rotaciones no fueran de la forma “si hubo un cambio de posición, girá en torno al cambio aplicado”; las actuales tendrían la forma “si hubo un cambio de posición, manipulá el cubo de acuerdo a cada movimiento, teniendo en cuenta la intensidad”.

La conexión del acelerómetro y el Arduino fue así:



Finalmente, luego de tanto esfuerzo, pude hacer funcionar mi parte. Restaba la integración con el dispositivo Bluetooth.

**El término sorpresa es relativo; de hecho, ni siquiera es relativo, directamente no es aplicable :) Tuvimos muchísimas complicaciones desde el principio de las clases (ver apartado respectivo), por lo que cuando esto sucedió ya estábamos habituados a que las cosas no anden. De todas formas, el proceso fue altamente didáctico, puesto que, con cada problema, aprendimos un nuevo trasfondo, y una nueva solución posible.*

Comunicación Bluetooth (Eugenio)

Al principio busqué información tanto del módulo bluetooth (BT) y del adaptador genérico que tenía en mi poder para darle capacidades inalámbricas a la computadora que ejecutaría el applet. Luego de buscar bastante, no encontre proyectos que se propusieran el mismo sentido de transmisión de información que nosotros, la mayoría hacía foco en controlar un arduino a través de BT pero no de sensar datos enviados por este unicamente.
Esto me confundió un poco, ya que se hablaba de modos master y slave para un módulo BT, y al parecer el que teníamos estaba configurado en slave y yo entendí que debía estar configurado en master para hacer lo que deseábamos. Para ello me puse a buscar información sobre esto, pero no resultaba un proceso fácil (si era siquiera era posible). Abandoné la primer búsqueda desilusionado y retome a la clase siguiente, luego de navegar bastante terminé entendiendo de que los modos estaban relacionados con quien iniciaba la conexión y quien la aceptaba y no con el sentido del flujo de información, ambos aceptando flujos en ambos sentidos. Recién allí comencé a intentar conectar el adaptador BT USB para darle capacidades de conexión BT a la computadora, este adaptador no resultó ser para nada facil de configurar como yo lo deseaba (al menos en el entorno de Ubuntu - Linux). Es decir, al conectarlo el SO lo detectaba y permitía emparejar con dispositivos estilo celulares y otras cosas; pero lo que necesitabamos nosotros era que este adaptador fuera tomado por el SO como un puerto serie, de esta forma podríamos tomar los datos enviados por el arduino sobre conexión Bluetooth y que la computadora los interpretara como datos serie (que era en realidad lo que eran esos datos, gran paradoja :p). Luego de intentar con una aplicación que supuestamente servía a tal fin (Blueman) e incluso probar a través de línea de comandos con el configurador por defecto de Ubuntu no pude encontrar forma de hacerlo. Hasta aquí sabía que el adaptador andaba, pero no si podría usarlo como un puerto serie sobre Bluetooth.

Luego de probar en Ubuntu, me pase a probar en Windows 7 (W7), y al parecer todo empezó bien, el adaptador era reconocido y W7 ofrecía una interfaz sencilla para elegir que se tomara como un puerto serie. Pero no eran todas buenas, cuando quise emparejar el adaptador con el módulo BT conectado al arduino, al parecer emparejaban pero no veía nada en el monitor serie que viene por defecto en el IDE de arduino (ni en otros programas que sirven para monitorear puerto serie como Putty o la misma consola de W7) mientras que el arduino si enviaba datos serie con el programa cargado actual.

Al final desistí con el adaptador y me puse a probar con el modulo BT integrado de una MacBook, el cual fué relativamente sencillo de configurar como puerto serie y para mi sorpresa funcionó de inmediato, recibiendo los datos serie enviados por el arduino a traves de BT. La contra de ello es que estaba usando OSX como SO y me hubiera gustado que se hiciera todo el proyecto en Linux. Por cuestiones de tiempo y de disponibilidad de herramientas no pude probar en otra maquina con W7 o Ubuntu con el adaptador integrado; pero creo (al menos en W7) puede ser simple de hacerlo funcionar.

Qué otras tecnologías consideramos?

- **Python** [13]: nos pareció particularmente atractivo por su simpleza en la comunicación con el Puerto Serial. Finalmente lo descartamos, por no encontrar herramientas de modelado 3D lo suficientemente simples de usar.
- **Panda3D** [14]: es un game engine, programado en Python. Lo descartamos porque apuntaba a modelados más complejos, y por consecuente más complicados.
- **OpenGL**: otra posibilidad para modelado 3D, que fue descartada por su complejidad (transformaciones de matrices HEAVYS).
- **GLUT** [19]: es un toolkit para OpenGL, con bindings en distintos lenguajes (particularmente probamos el de C++).
- **HOpenGL** [15]: binding de OpenGL para Haskell. En caso de haber migrado, se hubiera utilizado el módulo serialport [16] de Haskell para comunicación con el Puerto Serial.
- **Coin** [17]: conjunto de herramientas para modelado 3D.
- **Pivy** [18]: binding de Coin para Python.
- **Houdini** [20]: similar a Blender, para el modelado 3D. Lo descartamos por no considerarlo tan programático como queríamos que fuera la herramienta que necesitábamos (se orientaba más a la manipulación interactiva).

Problemas encontrados

- La utilización del Puerto Serial era de muy bajo nivel, comparada con los lenguajes que habitualmente usábamos (Haskell, Java, Smalltalk, Python, etc.).
- Desconocíamos acerca de protocolos de comunicación en Puerto Serial. Finalmente definimos uno propio.
- Encontramos un bug en la integración Processing + RXTX, motivo por el cual debimos cambiar de tecnología. No tuvimos el tiempo suficiente para reportar el bug de forma apropiada.
- El equipamiento en clase hacía dificultosa la práctica debido a las restricciones de conectividad. Esto resultó particularmente infructuoso en el momento de probar nuevas tecnologías.
- Antes de comenzar el desarrollo del Trabajo Final, pruebas con un Arduino hicieron que el mismo quedase “en estado vegetativo”. Luego de un tiempo, logramos volverlo a la normalidad, y volvió a estar funcional.
- Las tecnologías inalámbricas son hermosas hasta que uno se pone a trabajar para intentar que funcionen con un propósito un poquito diferente para el que fueron diseñadas en principio.

Código del proyecto

Se puede acceder al código del proyecto, así como las librerías utilizadas, en el repositorio de GoogleCode del proyecto. La dirección del mismo es la siguiente: http://code.google.com/p/alfduino/

REFERENCIAS

[1]Arduino
http://www.arduino.cc/es/
http://arduino.cc/en/Main/arduinoBoardDuemilanove

[2]Lenguaje de Programación Arduino
http://es.wikipedia.org/wiki/Arduino#Lenguaje_de_programaci.C3.B3n_Arduino
http://arduino.cc/es/Reference/HomePage

[3]Bluetooth
http://es.wikipedia.org/wiki/Bluetooth
http://es.wikipedia.org/wiki/Perfil_Bluetooth

[4]JY-MCU (Chip CSR BC417143)
http://www.csr.com/products/29/bluecore4-ext

[5]Acelerómetro
http://es.wikipedia.org/wiki/Acelerómetro

[6]ADXL335
http://www.analog.com/static/imported-files/data_sheets/ADXL335.pdf

[7]Puerto Serial
http://es.wikipedia.org/wiki/Puerto_serie

[8]Java
http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

[9]Eclipse
http://es.wikipedia.org/wiki/Eclipse_%28software%29

[10]Applet
http://es.wikipedia.org/wiki/Applet

[11]Giroscopio
http://es.wikipedia.org/wiki/Gir%C3%B3scopo

[12]EL PROYECTO QUE EL PROFESOR HABIA HECHO ANTES XD
https://github.com/CrearAyT/Guitarra-Vas-a-Llorar/blob/master/TuxGuitar-Duino/src/org/herac/tuxguitar/io/guitarduino/GDPPlugin.java

[13]Python
http://www.python.org/

[14]Panda3D
https://www.panda3d.org/

[15]HOpenGL
http://www.haskell.org/haskellwiki/Opengl#HOpenGL_Resources

[16]Haskell serialport
http://hackage.haskell.org/package/serialport

[17]Coin
http://www.coin3d.org/

[18]Pivy
http://pivy.coin3d.org/

[19]GLUT
http://www.opengl.org/resources/libraries/glut/

[20]Houdini
http://www.sidefx.com/