**Universidad Nacional de La Matanza**

**Departamento de Ingeniería e Investigaciones Tecnológicas**

**Sistemas Operativos Avanzados**

**2017**

# Sistemas Embebidos, Android e IoT

**Días de Cursada:** Martes  **Turno:** Noche

**Docentes:**  [Sebastián Barillaro](http://so-unlam.com.ar/moodle/user/view.php?id=15&course=1)

[Esteban Carnuccio](http://so-unlam.com.ar/moodle/user/view.php?id=10&course=1)

[Graciela de Luca](http://so-unlam.com.ar/moodle/user/view.php?id=11&course=1)

[Gerardo Garcia](http://so-unlam.com.ar/moodle/user/view.php?id=12&course=1)

[Waldo Valiente](http://so-unlam.com.ar/moodle/user/view.php?id=7&course=1)

[Mariano Volker](http://so-unlam.com.ar/moodle/user/view.php?id=13&course=1)

**Integrantes:** Leonardo Andrade

Gabriel Bosco

Marcos Vittorio

**Proyecto:** Monitor de bebés

#### Introducción

En el siguiente informe se describen las experiencias compartidas durante la realización de nuestro proyecto, desde la compra de los materiales hasta el desarrollo de las aplicaciones correspondientes para cada sistema (embedido y Android).

Muchas fueron las dificultades que se nos presentaron y, en más de una ocasión, nos encontramos replanteándonos la utilización de uno u otro módulo/sensor o los métodos de comunicación que utilizaríamos finalmente. La lógica aplicada en ambos dispositivos también nos trajo algunas complicaciones, ya sea por la comunicación entre ambos sistemas (Http Request), como la atención de las interrupciones en el sistema embebido o el manejo de servicios en la aplicación Android. A continuación, lo realizado.

#### Elección del entorno a utilizar

Inicialmente la idea era trabajar con las placas Galileo del laboratorio pero, luego de ver las limitaciones en documentación y la disponibilidad de la misma fuera del laboratorio lo que nos da mayor tiempo para trabajar con este equipamiento, optamos por enfocarnos en las placas Arduino, que procedimos a comprar. El modelo elegido fue Arduino Uno R3.

#### Experiencias con el Hardware

Por la idea original de trabajar con las Galileo, en primera instancia sólo se adquirió el acelerómetro. Luego del cambio de plataforma, y junto con la placa, se adquirieron en combo un sensor PIR (se utilizó para incursionar en el mundo Arduino), un módulo LCD cd 16x2 con I2C y algunos LED (Se observó con estos últimos que, según su color, algunos de ellos soportaban conexión directa a 5v; el rojo y amarillo perecieron durante las pruebas; se estima que por su frecuencia tienen menor tolerancia que el verde, azul y blanco).

Por último, se adquirió el módulo WIFI (ESP8266). Este módulo funciona a 3.3 volts. Si bien la placa cuenta con un pin con este voltaje, su fuente da un máximo de 50mA y el consumo del módulo, en su arranque, es bastante superior. Por este motivo y de acuerdo a lo sugerido en los foros consultados, se intentó conectar una fuente externa de 12v y 1A (según la documentación la placa soporta hasta 20V y 1.5A), dando como resultado una placa quemada (se estima que se quemaron ambos microcontroladores).

Se adquirió una nueva placa y, con ella, una fuente de protoboard que dispone de pines que permiten setear el voltaje en 5v y 3.3v (dos salidas independientes; una para cada banda de alimentación de la protoboard).

Durante las pruebas, se instaló la fuente en la protoboard y se realizaron todas las conexiones del módulo WIFI con extremo cuidado (5 pines, dos a 3.3v y uno a GND) y se encendió la fuente. Luego de unos segundos se observó un humo blanco saliendo del esp8266, se apagó la fuente y se procedió a revisar daños y causa. Se notó que, si bien el módulo estaba conectado a la línea ‘+’ de la protoboard, la pata ‘-‘ de la fuente estaba conectada en esta línea, por lo cual el módulo realmente se conectó con la polaridad invertida (RIP esp8266), o eso creímos por lo menos. Luego de revisar todos los sensores y la placa, se corrigió la conexión de la fuente y se encendió nuevamente. En este caso, el sensor inició, había cierta comunicación, pero no era posible recibir correctamente lo que enviaba el módulo en ninguna de las velocidades del puerto serie. Tras varias pruebas y una larga investigación en internet, se descubrió que el problema venía dado por una aparente diferencia de potenciales entre los sensores y la placa. Para solucionar esto se conectaron ambos GND de la protoboard con el GND de la placa. Luego, se logró establecer la primera conexión exitosa y empezar a trabajar con el módulo.

#### Lista de componentes utilizados

* Placa Arduino Uno R3
* Sensor Acelerómetro y Giróscopo MPU6050
* Sensor de Sonido KY-038
* Módulo WIFI ESP8266
* Fuente para protoboard MB102
* LED
* Protoboard + cables

#### Experiencias con el Software

Una vez solucionados los inconvenientes de hardware, se comenzó a trabajar con las aplicaciones. En un primer acercamiento, se intentó crear un socket TCP para comunicar ambos sistemas pero no se alcanzaron los resultados buscados. Antes de continuar con configuraciones aisladas se optó por seguir una serie de tutoriales de Android que nos introdujeron en la utilización de servicios, asynctask y manejo de notificaciones, entre otros. Los videos fueron de gran utilidad, ya que aclararon el panorama y nos permitieron ver cuáles eran nuestras necesidades puntuales. Los enlaces son:

Android Studio 6 Servicio y Mediaplayer:

https://www.youtube.com/watch?v=euBcjzWUGt8

Android Studio 7 Servicio en Android y Mediaplayer:

https://www.youtube.com/watch?v=ArLLpUCUIec

Android Studio 8 Servicios y Notificaciones: <https://www.youtube.com/watch?v=qt8iKFtVgzc&t=125s>

Android Studio 10 Tarea Asíncrona en Servicio:

<https://www.youtube.com/watch?v=WNVYbYUVFxA&t=63s>

Para la aplicación en Arduino, se trabajó con un código y librerías aportados en los foros del tema y se ajustaron a nuestras necesidades. Como se mencionó anteriormente, el mayor inconveniente que se nos presentó fue el manejo de las interrupciones generadas por el sensor de sonido (cada vez que se supera el umbral se atiende la interrupción y se contabiliza la magnitud de sonido medida).

#### Detalle del Software Utilizado

* Arduino IDE 1.8
* Android Studio (última versión)
* Putty (pruebas de conexión durante el desarrollo de la comunicación)

#### Cuestiones particulares de los sensores

MPU6050 (Acelerómetro y Giróscopo)

Este sensor en general requiere de una calibración. Para nuestro proyecto, en donde leemos los valores iniciales cada vez que se inicia el sistema, no fue necesario.

El MPU6050 no viene con los pines integrados por lo que fue necesario soldarlos.

KY-038 (Sonido)

Cuenta con un regulador de sensibilidad. Fue necesario ajustarlo hasta lograr un umbral acorde al nivel de ruido de los lugares donde se trabajó.

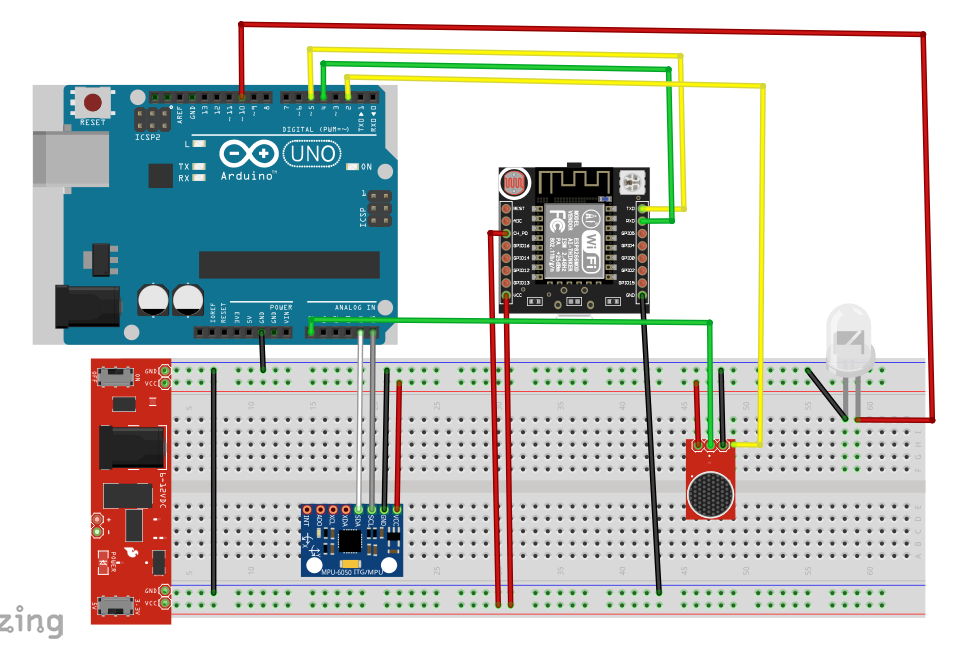
ESP8266 (Módulo WIFI)

El módulo trabaja a 3.3 volts y, durante el arranque, consume una potencia muy superior a la entregada por la placa, por lo cual se utilizó una fuente auxiliar.

#### Conexión

Para la conexión entre los sistemas se genera un servicio con una tarea asíncrona que realiza una HTTP Request. Al iniciarse el servicio, se realiza un while infinito con la petición conformada por la IP del módulo, el número de puerto y un parámetro que le indica a la placa el inicio del monitoreo.

Diagrama de Conexiones



**Nota:** el sensor de sonido utilizado realmente no se encontró en el sistema utilizado para el gráfico. El sensor original tiene un cuarto pin (digital) que envía un pulso alto cada vez que el umbral fijado es superado.

#### Detalle de conexiones

MPU6050

* Rojo: Alimentación (5v)
* Negro: Tierra
* Gris: SCL
* Blanco: SDA

KY-038

* Rojo: Alimentación (5v)
* Negro: Tierra
* Verde: Analógico (A0)
* Amarillo: Digital (D0)

ESP8266

* Rojo I: Alimentación (3.3v)
* Rojo II: CH\_PD (3.3v; se requiere para poder utilizar el módulo)
* Negro: Tierra
* Amarillo: Tx
* Verde: Rx