## 24 y 25 de septiembre

### Horas trabajadas: 12+

* 24 de septiembre: Reunión con Santiago para especificar los requisitos.
* Creado cliente Android.
  + Captura los valores del acelerómetro y los muestra por pantalla.
  + Fijado en modo horizontal.
* Montado robot
* Se ha instalado la distribución Raspbian sobre la Raspberry y se han actualizado los paquetes así como la instalación de diversas utilidades.
* Instalado xrdp para crear servidor de escritorio remoto.
* Conectado e instalado módulo WiFi. Configurada la red para que tenga la IP estática 192.168.0.123
* Se ha establecido una comunicación USB entre la Raspberry y la Arduino. Se ha creado un pequeño script a modo de prueba para comprobar la comunicación. La Raspberry envía por un script en Python un número N a Arduino. Cuando Arduino lo recibe, hace parpadear un led N veces.
  + Script Python

import serial  
ser = serial.Serial('/dev/ttyACM0', 9600)  
ser.write('5')

* + Script Arduino

const int ledPin = 52;

void setup(){

pinMode(ledPin, OUTPUT);

Serial.begin(9600);

}

void loop(){

if (Serial.available()) {

light(Serial.read() - '0');

}

delay(500);

}

void light(int n){

for (int i = 0; i < n; i++) {

digitalWrite(ledPin, HIGH);

delay(100);

digitalWrite(ledPin, LOW);

delay(100);

}

}

## 26 de septiembre

### Horas trabajadas: 8

* Realizada investigación exhaustiva para sopesar las distintas tecnologías que emplear y cómo realizar las distintas fases del proyecto. De momento se profundiza en Java (Android), Python (Raspberry Pi) y C++ (Arduino).
* Añadidos al AndroidManifest.xml los permisos Internet y network state y wake lock
* Añadida propiedad keepScreenOn a la actividad principal para impedir bloqueo de pantalla
* Primera versión de comunicación Android-Raspberry
  + Creado servidor en Python que abre un socket TCP en el puerto 5000 y espera recibir una cadena de 1024 bytes.
  + Creado cliente en Android que envía un mensaje. El mensaje llega al servidor pero provoca un error. Es decir, muestra el mensaje por pantalla y acto seguido lanza una excepción y finaliza la ejecución del servidor.

## 27 de septiembre

### Horas trabajadas: 10+ (día insano)

* General
  + Creado repositorio GIT en bitbucket
* Android
  + Cambiado el valor a medir de los sensores. Antes era ‘Y’ y ‘Z’ del acelerómetro, ahora son ‘Y’ y ‘Z’ de la orientación. Hay mejoras pero me sigue sin convencer, en un futuro hay que mejorar esto.
  + Realizado mapeo de valores. El valor Y (rotación) se mapea del rango [-90,90] a [255,0] (para que quepa en un byte, la precisión es suficiente), y el valor Z (aceleración) de [0,90] a [255,0].
  + Realizado cliente socket, que envía un datagrama UDP con un mensaje de 3 bytes (comando=1, param1=aceleración, y param2=rotación).
* Raspberry
  + Realizado server.py, que abe un socket UDP en el puerto 5000 para recibir datagramas con una longitud de 3 bytes y mostrar por pantalla tanto la dirección IP del emisor del datagrama como los 3 bytes del mensaje (convertidos a unsigned int para facilitar su lectura). Recibe los datagramas correctamente y los envía a la placa Arduino.
* Arduino
  + A partir del código del taller de robótica, he reescrito la clase GoShield para recibir los parámetros de la placa Arduino y transformarlo en movimiento del motor. **¡Funciona!**

## 28 de septiembre

### Horas trabajadas: 4

* Android
  + Mejorado diseño del código.
  + Añadidas definiciones de datagramas.
  + Creado conmutador parar-moverse, que envía el comando STOP o MOVE.
* Arduino
  + Interpreta los comandos con un switch. Cuando recibe el comando 0 para. Con el comando 1 se mueve: Move(speed, balance);
  + Mejorado el código
* Raspberry
  + Intento (fallido) de crear una red Ad Hoc

## 1 de octubre

* Grabado video demostración: <http://www.youtube.com/watch?v=Fz93w9QeBjI>

## 2 de octubre

### Horas trabajadas: 4

* Arduino
  + Pequeñas mejoras en el código, comentado código
* Raspberry
  + Separado servidor de producción (sin salidas por pantalla) de servidor de desarrollo
  + Creado servicio robotManager que arranca el servidor al iniciar la Raspberry. Ejecuta python2.7 /home/pi/server/server.py
  + Otro intento de ad-hoc. Esta vez he conseguido la red, pero no que le llegue la mandanga al robot. Me puedo conectar por ssh, pero tarda mucho en iniciar sesión ¿?
    - He seguido este tutorial: <http://elinux.org/RPI-Wireless-Hotspot>

## 9 y 10 de octubre

### Horas trabajadas: 8

#### Objetivo de la sesión: Organización, conceptualización, testeo y detección de errores.

* Montaje
  + Ahora el robot se puede mover de forma independiente, gracias a la incorporación de las pilas y el cableado para alimentar todos los elementos.
    - Video demostración: <http://www.youtube.com/watch?v=RlNEMqJr_3E>
  + Comenzado el diseño en 3D para el montaje de la parte frontal, que sostendrá la cámara y el parachoques frontal.
    - Fichero de partida <http://www.thingiverse.com/thing:128617>
* Arduino
  + Detectados unos errores en el constructor de la clase GoShield. Define como elementos de entrada unos elementos de salida. Luego se sobrescribe como entrada, por lo que todo queda en orden, aunque hay que corregirlo.
  + Se debe reestructurar el fichero Types.h eliminando las variables que cuentan el número de elementos de entrada y de salida, para hacerlo directamente con respecto a la longitud del vector que los define.
* Android
  + Detectado fallo al suspender/restaurar la aplicación. En algunos casos sigue emitiendo señales de coordenadas cuando no debería hacerlo.
* Raspberry
  + Más intentos del punto de acceso WiFi. Parece que ya empieza a ir
    - NO funciona cuando ejecuto:
      * update-rc.d hostapd defaults  
        update-rc.d udhcpd defaults   
        update-rc.d robotManager defaults
        + Yo creo que es por el orden de las cosas. Necesito poner requires en etc/init.d/robotManager. Que necesite tener hostapd activado, udhcpd activado (de forma secuencial, no antes de hostapd) y una ip configurada, y una vez tenga una ip en mi propia red ejecuto el script de Python.
    - Funciona a veces cuando ejecuto:
      * update-rc.d robotManager defaults
        + …y dentro de etc/init.d/robotManager hago:
        + service hostapd start  
          service udhcpd start  
          sleep 3  
          python2.7 /home/pi/server/server.py &
    - **FUNCIONA CUANDO ESTÁ CONECTADO EL HDMI ¿!?!!?!** poniendo esto mismo al final de /etc/rc.local:
      * Es necesario hacer primero:  
        sudo update-rc.d hostapd, udhcpd y robotManager remove
      * service hostapd start  
        service udhcpd start  
        python2.7 /home/pi/server/server.py &
* General
  + Creado documento de objetivos y conceptos. En él se explican los objetivos a cumplir de forma bimensual (octubre, diciembre y febrero) y una larga lista de ideas que realizar una vez alcanzados los objetivos, es decir, después de febrero.

## 11 de octubre

### Horas trabajadas: 7

* Raspberry
  + He acabado desquiciado con el punto de acceso wifi. He probado a usar otro servidor de DHCP llamado isc-dhcp-server en lugar de udhcpd. No he conseguido hacerlo funcionar. Llevo 5 horas con esto y me rindo.
* Arduino
  + Ahora recibe el código Claxon ON/OFF y enciende o apaga el pito.
  + Arreglado fallo en el constructor en la inicialización de pines de entrada y salida. Ahora cada pin se inicializa una vez (y se pone a nivel bajo para ahorrar energía), ya que antes algunos pines de salida (LED) también se definían como de entrada y luego se definían como salida. No tiene fallo visual ya que se arregla el fallo al sobrescribir a lo que le correspone, pero no era elegante.
* Android
  + Separado socket de mensajes. Es decir, por un lado está la clase encargada de enviar el socket, y por otro la de tener el repertorio de funciones que el robot hace (moverse, pitar, parar, …)
  + Hecho lo del pito. He probado enviando 2 veces el mensaje o 1 vez y la diferencia es inapreciable de primeras.
  + Preguntar qué formas de transmisión hago. De momento envío todo a través del mismo socket UDP.  
    Quizás es mejor abrir tener un socket exclusivo para el bombardeo de coordenadas / movimiento del robot y otro para el resto. Por otra parte, quizás me conviene también un socket TCP…