## 24 y 25 de septiembre

### Horas trabajadas: 12

* 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: 7

* Añadidos al Android manifest 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

* 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

### 3 horitas

* 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