

Universidad de Sevilla - Ingeniería Informática

ScorClaw

Trabajo de Robótica

José M. Camacho – camachososa@josemazo.com

Gabriel E. Muñoz – munozrios22@gmail.com

26 de Abril del 2013

Contenido

1. Introducción	2
2. Tratamiento del problema	3
3. Implementación	5
4. Manual de uso	9
5. Conclusiones.....	11
6. Bibliografía	12

1. Introducción

¡Bienvenidos! ¡Welcome! ¡Benvenuti! ¡Willkommen! ¡La feria ha llegado al F0.33!

ScorClaw es una aplicación web para móviles que convierte a tu robot SCORBOT en un *claw crane*, el típico juego del “gancho” de nuestra infancia.



La realización de un proyecto de este tipo surgió con la idea de poder controlar el SCORBOT mediante el móvil. En principio tuvimos multitud de ideas, pero el hecho de que el SCORBOT sea un robot bastante limitado nos restringía el espacio de posibles problemas abordables.

Este fue el motivo principal por el que nos decantamos por este juego, en el que hemos podido desarrollar todos los movimientos necesarios para nuestro robot.

En este documento explicaremos todo lo realizado en nuestro proyecto, desde el tratamiento del problema para explicar cómo hemos desarrollado nuestra solución hasta la implementación en código del mismo. En este punto haremos hincapié en las tecnologías usadas, Python y JQuery Mobile, ya que son bastante novedosas en nuestro ámbito.

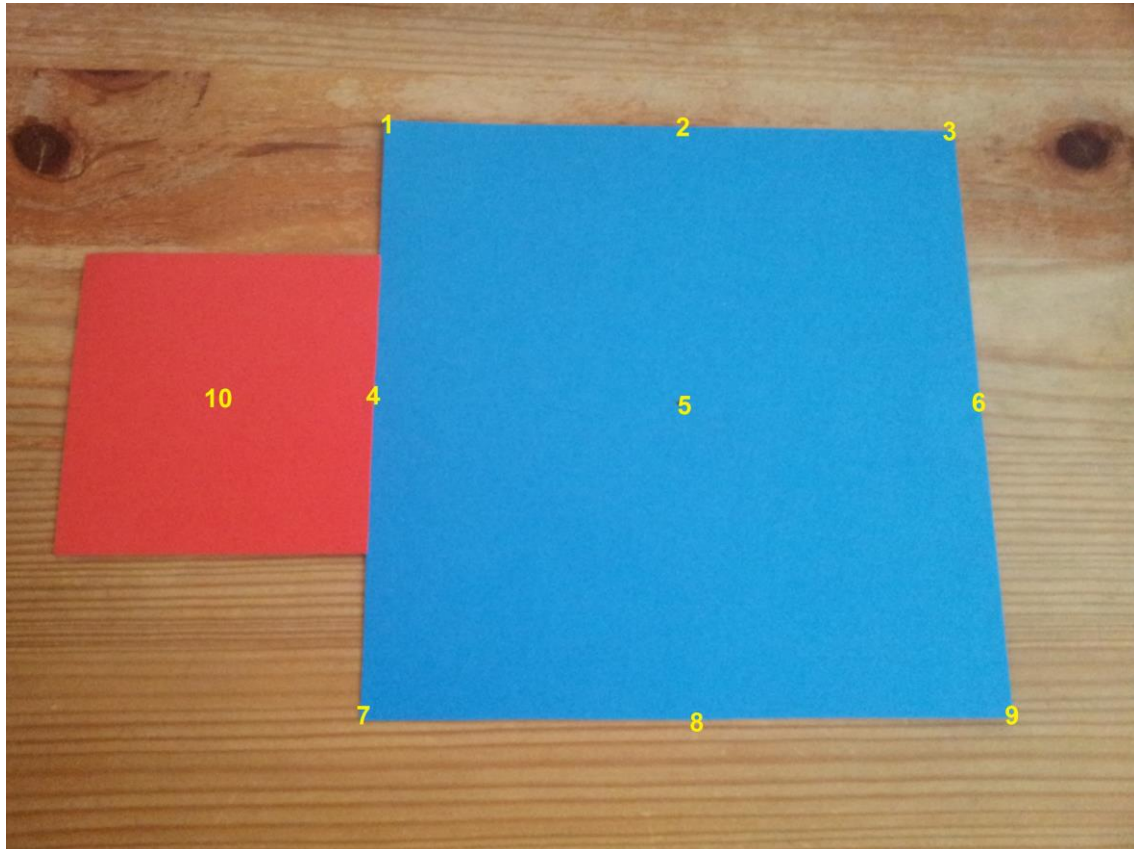
Para finalizar tendremos dos puntos. En primer lugar, los pasos a seguir para configurar y poder jugar con nuestro SCORBOT y como último punto del

documento expondremos las conclusiones a las que hemos llegado con realización de este proyecto.

¡No siga leyendo, aproveche el tiempo para jugar con su ScorClaw y diviértase!

2. Tratamiento del problema

Ya que el SCORBOT tiene un alcance limitado y sufre de malfuncionamiento cuando se lleva a los límites, tuvimos que establecer unos límites en un plano horizontal para que no se saliese de esos puntos. Para ello usamos una plantilla y la colocamos como sale en el video de la sección *Manual de uso*. La plantilla es la siguiente:



La parte azul es la zona de juego y la roja donde se deja el premio. La zona de juego es un cuadrado de 20cm x 20 cm. Lo que hicimos fue buscar un lugar donde el SCORBOT pudiera acceder a todos los puntos de la plantilla, no solo en la mesa, sino también a ciertas alturas. Así encontramos los siguientes puntos:

Punto	(x,y)
1	(-711, -3103)
2	(-699, -2036)
3	(-570, -977)
4	(97, -3064)
5	(97, -1959)
6	(48, -913)
7	(975, -3080)
8	(848, -2006)
9	(646, -933)
10	(97, -3683)

Aquí se pueden observar varias cosas. Según la tabla, el eje Y es el eje vertical en la imagen anterior, positivo hacia abajo y negativo hacia arriba. El eje X es el eje horizontal en la imagen, negativo a la izquierda y positivo a la derecha. El eje Z positivo saldría de la imagen, y el negativo entraría.

Además, apreciamos cierta desviación del SCORBOT. Puede deberse a que la plantilla no esté totalmente alineada con el origen del coordenadas del robot, o que no haya una correspondencia directa con los movimientos físicos de este. En cualquier caso no es un problema para nosotros, así que escogimos los siguientes valores máximos y mínimos para los ejes X e Y.

Nombre	Valor
X mínima	-700
X máxima	900
Y mínima	-3100
Y máxima	-910

Ya seleccionados los puntos al nivel de la mesa teníamos que darle cierta altura, un aumento en el eje Z, para ver si el SCORBOT podía moverse sobre la plantilla a esa altura. Probando escogimos $Z = 1864$, y esa es la altura en la que el jugador controlará al robot.

El problema venía ahora. El valor del eje Z cambiaba drásticamente en la misma altura física cuando se encontraba cerca de la altura de la mesa. Esto ocurría cuando la pinza recorría el eje Y, no cuando se desplazaba sobre el X. Se podría decir que no controla bien los cambios del eje Z a bajas alturas cuando la pinza se aleja o se acerca al robot.

Esta era la relación de los puntos en la altura física donde queríamos agarrar el “premio”, unos 3.5 cm:

Punto	(y, z)
6	(-913, 845)
5	(-1959, 522)
4	(-3064, 265)
10	(-3683, 224)

Entonces escogiendo los puntos extremos calculamos el vector que los unía:

$$(-3683, 224) - (-913, 845) = [-2770, -621]$$

Tras eso calculamos su vector normal:

$$[-2770, -621] \rightarrow [-621, 2770]$$

Y usando el punto (-913, 845) calculamos la ecuación de la recta:

$$\begin{aligned} -621(y + 913) + 2770(z - 845) &= 0 \\ -621y - 566973 + 2770z - 2340650 &= 0 \\ -621y + 2770z - 2907623 &= 0 \end{aligned}$$

Solo falta obtener el valor de z:

$$z = ((2907623 + 621y) / 2770) - 50$$

El -50 del final lo tuvimos que meter para bajar medio centímetro a la solución, ya que si usamos la ecuación con los puntos Y centrales no obtenemos la Z que queríamos, sino un poco más alta. Así que con medio centímetro más hacía abajo en toda la recta no teníamos ningún problema en los extremos y todo iba bien en las posiciones centrales.

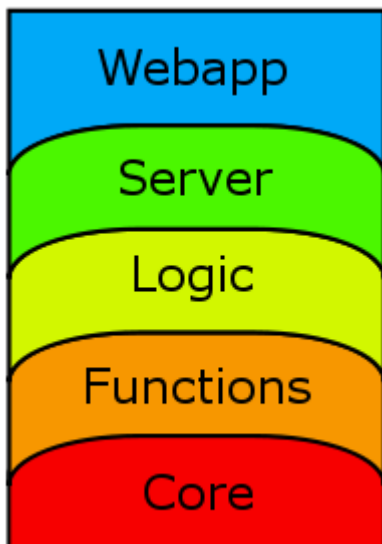
Ya solo tenemos que mover el SCORBOT. A intervalos de 1 cm nos pareció buena idea y cuando se juega con el parece que fue una buena elección. Así que ya analizado por completo el problema pasamos a la implementación.

3. Implementación

Antes de empezar con esta sección tenemos que decir que todo el código y la documentación se encuentran en GitHub en la siguiente dirección:

<http://github.com/codeministers/scorclaw>

Para llevar a cabo la implementación de nuestro proyecto hemos separado el mismo en dos partes, una de servidor y otra de aplicación móvil. La parte del servidor se encargará de recibir las peticiones HTTP que serán las órdenes enviadas mediante la aplicación móvil. De esta forma, una premisa fundamental es que la aplicación móvil pueda comunicarse con el servidor.



Esta ilustración muestra de manera gráfica los niveles de abstracción de nuestro sistema. Como comentábamos en el primer párrafo de este punto, la comunicación por las peticiones se producirá entre los niveles “Webapp” y “Server” y será en los módulos de “Server” (“Logic”, “Functions” y “Core”) donde se interpretarán esas peticiones y se producirá la comunicación con el SCORBOT.

Para la parte de servidor hemos usado Python. Python, según Wikipedia, es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Hasta aquí las definiciones “formales”. Nuestra elección de usar Python se debe principalmente a dos motivos. El primero era el deseo de conocerlo ya que es un lenguaje que lleva unos años en auge y que está evolucionando muy rápidamente y de manera muy robusta. El segundo, y relacionado con el primero, es la poca diversidad de lenguajes que nos “obligan” a aprender en nuestros estudios en la Universidad. De ahí nuestra elección para dar nuestros primeros pasos con él en este proyecto.

Para la comunicación con el puerto serie hemos usado una librería que funciona muy bien en Python llamada *py-serial* en su versión 2.6. No nos ha dado ningún problema y su uso es fácil e intuitivo. Para más información sobre la misma, en la bibliografía estará disponible un enlace a la página de su documentación.

En esta parte de servidor hemos implementado una serie de archivos/módulos con el objetivo de ir abstrayendo los niveles de complejidad en la comunicación con el SCORBOT. En el siguiente gráfico se observa nuestro objetivo de una manera más clara:

Ahora explicaremos estos módulos de forma más detenida, dando una breve explicación de cada uno de los métodos que contienen.

core.py: Esta clase contiene los métodos básicos para comunicarse con SCORBOT o con cualquier dispositivo serie. Es la clase de más bajo nivel de nuestra implementación.

__init__(self, serialPort): Método para iniciar atributos.
open_con(self): Abre la conexión con el puerto serie.
close_con(self): Cierra la conexión con el puerto serie.
write(self, command): Manda al SCORBOT, a través del puerto serie, el comando pasado por parámetro.

functions.py: Esta clase es la que se usará por la lógica de cualquier programa para abstraer los métodos de la clase *core.py*.

__init__(self, serialPort): Método para iniciar atributos.
open_con(self): Abre la conexión con el puerto serie.
close_con(self): Cierra la conexión con el puerto serie.
home(self): Manda el comando *home* al SCORBOT.
open(self): Manda el comando *open* al SCORBOT.
close(self): Manda el comando *close* al SCORBOT.
def_pos(self, *pos): Manda el comando *defp* al SCORBOT. El parámetro *pos* es el que contiene la información de los parámetros necesarios para la función.
teach_pos(self, pos, x, y, z, p, r): Manda el comando *teach* al SCORBOT. Los parámetros son los valores necesarios para definir todas las posiciones de nuestro robot.
speed(self, s): Manda al SCORBOT el comando *speed* con el valor para la velocidad pasada por parámetro.
move(self, *pos): Manda al SCORBOT el comando *move* con el valor de la posición a donde debe moverse.
setp(self, pos1, pos2): Manda al SCORBOT el comando *setp* con los parámetros *pos1* y *pos2* de tal forma que *pos1* tomará el valor de *pos2*.
set_x(self, pos, value): Manda al SCORBOT el comando *setpvc* con los parámetros *pos* y *value* tal que nos quedaría *setpvc pos x value*.
set_y(self, pos, value): Manda al SCORBOT el comando *setpvc* con los parámetros *pos* y *value* tal que nos quedaría *setpvc pos y value*.
set_z(self, pos, value): Manda al SCORBOT el comando *setpvc* con los parámetros *pos* y *value* tal que nos quedaría *setpvc pos z value*.

logic.py: Esta clase es la que se encarga de manejar la lógica de nuestro problema.

__init__(self, serialPort): Método para iniciar atributos.
close_con(self): Cierra la conexión con el puerto serie.

fast(self): Función que aumenta la velocidad del SCORBOT, concretamente cambia su velocidad a un valor de 50.

medium(self): Función que cambia la velocidad del SCORBOT a un valor medio, concretamente a 20.

slow(self): Función que disminuye la velocidad del SCORBOT, concretamente cambia su velocidad a un valor de 7.

initialize(self): Inicializa todos los valores necesarios del SCORBOT para comenzar correctamente la ejecución del programa.

calculate_z(self, y): Calcula el valor de z a partir de la fórmula obtenida en el apartado “tratamiento del problema” y el parámetro y.

update_x_y(self, x=None, y=None) Actualiza los valores para las coordenadas x e y.

more_x(self): Incrementa el valor de la coordenada x.

less_x(self): Decrementa el valor de la coordenada x.

more_y(self): Incrementa el valor de la coordenada y.

less_y(self): Decrementa el valor de la coordenada y.

home(self): Coloca al SCORBOT en el punto de inicio de nuestro programa. Cuidado, no realiza la acción “home” del SCORBOT.

catch(self): Coge una pieza y la lleva al lugar donde el usuario la puede recoger. Seguidamente, la suelta y vuelve al punto de inicio de nuestro programa.

Los módulos que veremos a continuación nos ayudarán a unir todo de una manera elegante y sencilla. Los veremos por encima, sin meternos en detalle, ya que la mayoría son módulos de “apoyo” sin una complejidad excesiva.

requests.py: Este es el módulo de más complejidad. Esta clase es la que se encargará de manejar las peticiones HTTP que realizaremos desde nuestro móvil a nuestro servidor.

main.py: Este módulo también tiene mucha importancia en nuestro programa. Se encargará de comprobar los parámetros pasados para obtener el puerto para el servidor y el nombre del puerto serie.

variables.py: En este módulo tendremos definidas todas las variables y constantes de nuestro programa.

home.py: La utilidad de este módulo es que su ejecución sitúa al SCORBOT en la posición *home*.

Ahora explicaremos la implementación de la parte de la aplicación móvil. Para decantarnos por qué tecnología usar, la decisión ha sido similar a la tomada para el desarrollo de la parte del servidor.

Para el desarrollo de una aplicación móvil hay muchas alternativas conocidas y de confianza (android, iOS...), sin embargo, hemos querido realizarla con una no tan conocida y que sea multiplataforma. En concreto, JQuery Mobile.

JQuery Mobile es un framework para desarrollo web optimizado para dispositivos táctiles, tales como móviles, tablets, etc. Su nombre viene porque está siendo desarrollado por el equipo del proyecto JQuery y su principal ventaja, como ya indicamos en el párrafo anterior, es su marco compatible con una amplia variedad de dispositivos.

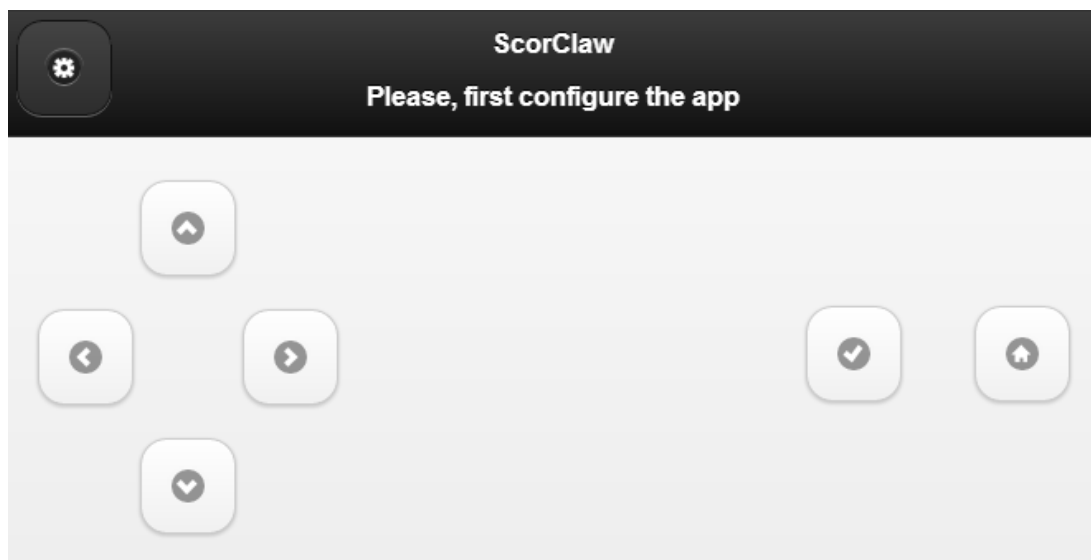
El uso de esta tecnología ha hecho que la parte del desarrollo de la aplicación móvil haya sido bastante más fácil que si nos hubiéramos decantado por alguna de las comentadas

anteriormente. Sólo hemos necesitado desarrollar un archivo *html*, que es el encargado de “dibujarnos” la interfaz del usuario en nuestra pantalla del móvil.

Nuestra interfaz de usuario tendrá dos vistas principales, la vista de juego y la vista para la configuración de nuestro sistema.

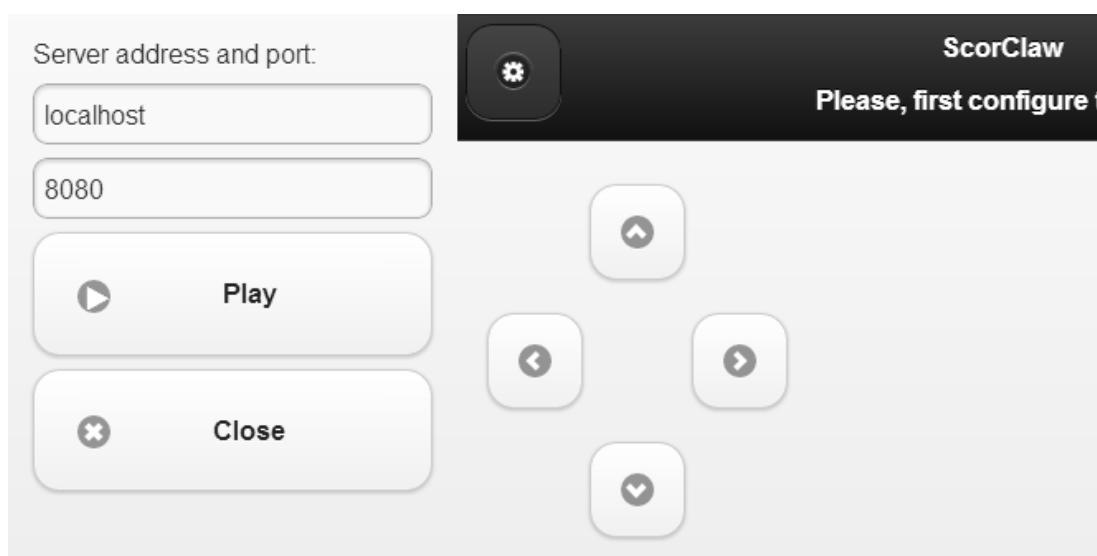
La vista de juego será la que tenga los controles necesarios para poder manejar nuestro SCORBOT. Su estilo será el de las consolas antiguas, con una cruceta de dirección en la parte izquierda y dos botones de acción en la parte derecha, uno para coger un elemento (*check*) y otro para ponerlo en la posición inicial (*casita*).

La imagen siguiente muestra cómo sería esta vista:



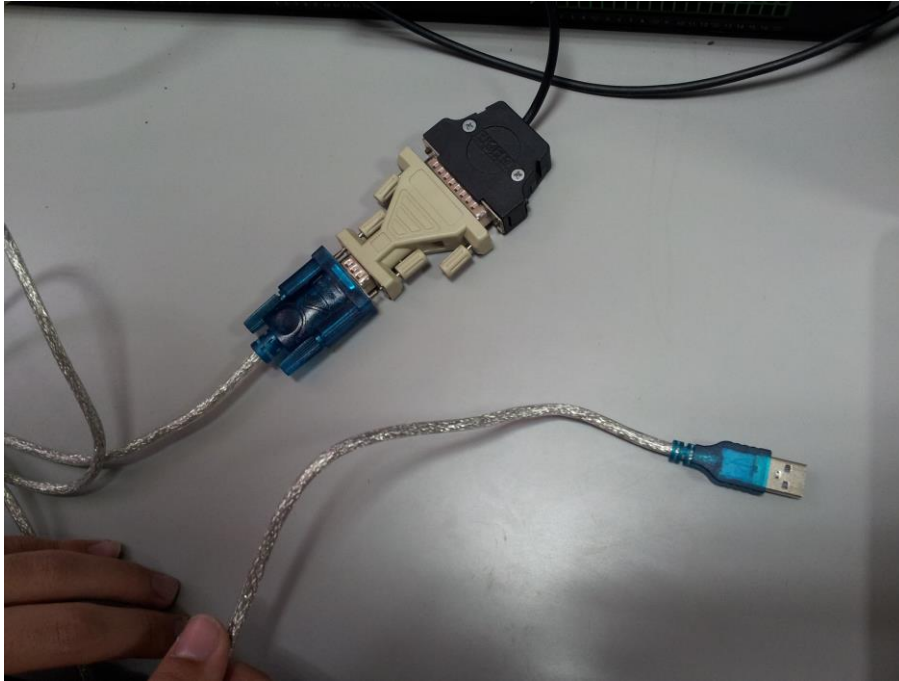
La vista de configuración del sistema es esencial para el correcto funcionamiento del juego. Cuando queremos comenzar a jugar, el primer paso a realizar será configurar correctamente la dirección y puerto de nuestro servidor ya que si no lo hacemos, las peticiones no sería correctamente procesadas (campos *localhost* y *8080*). Los otros dos botones son para parar o reproducir la música (botón *play*) y cerrar el menú de configuración (botón *close*).

Echémosle un vistazo a su aspecto:



4. Manual de uso

1. Comprobar que se tiene el driver del cable USB a puerto serie instalado en la máquina, además del intérprete de Python. Las últimas versiones de la versión 2 son las adecuadas.
2. En primer lugar encendemos el SCORBOT y conectamos el controlador a nuestro cable USB a puerto serie.



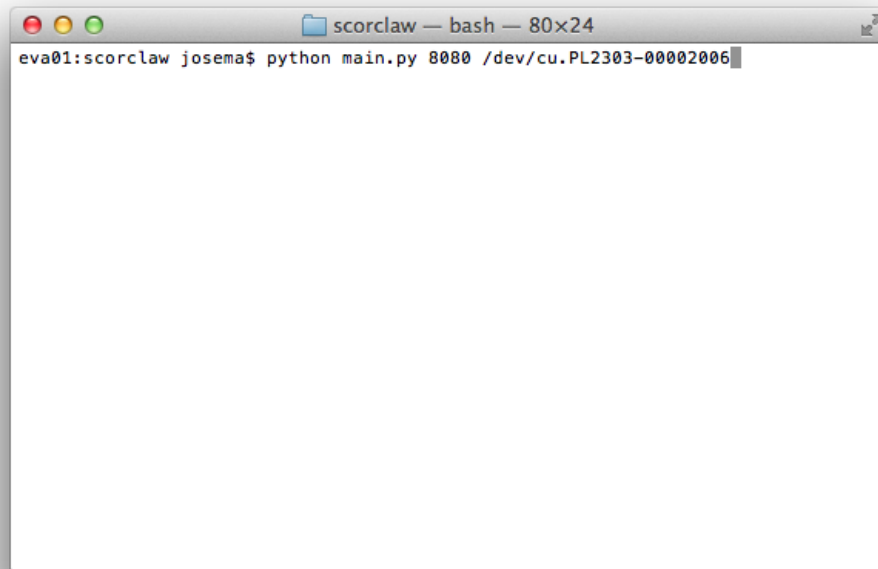
3. Conectar el puerto USB a nuestro ordenador.



4. Después arrancaremos el servidor en nuestro ordenador. Para ello nos situamos con la terminal en el directorio donde se encuentren los archivos `.py` y ejecutamos:

`python main.py "puerto para el servidor HTTP" "nombre del puerto serie"`

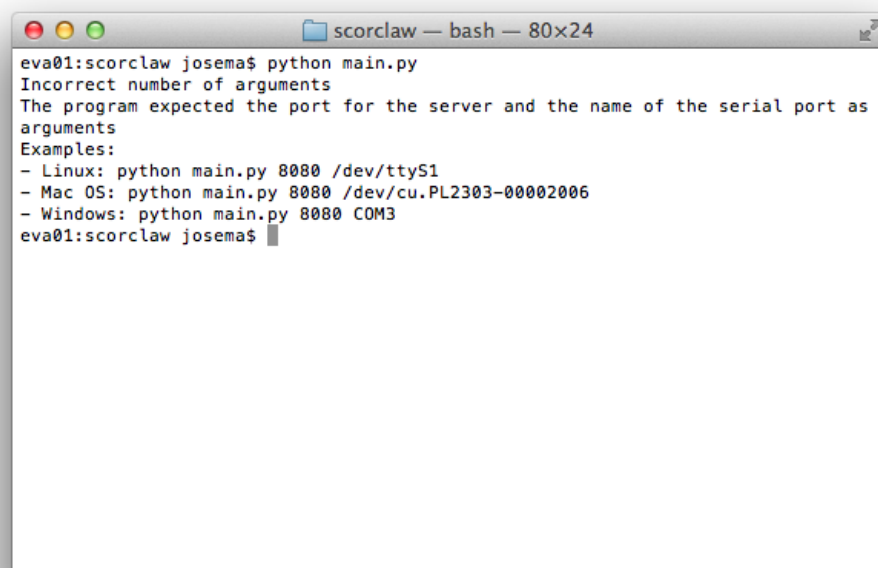
Aquí un ejemplo:



```
scorclaw — bash — 80x24
eva01:scorclaw josema$ python main.py 8080 /dev/cu.PL2303-00002006
```

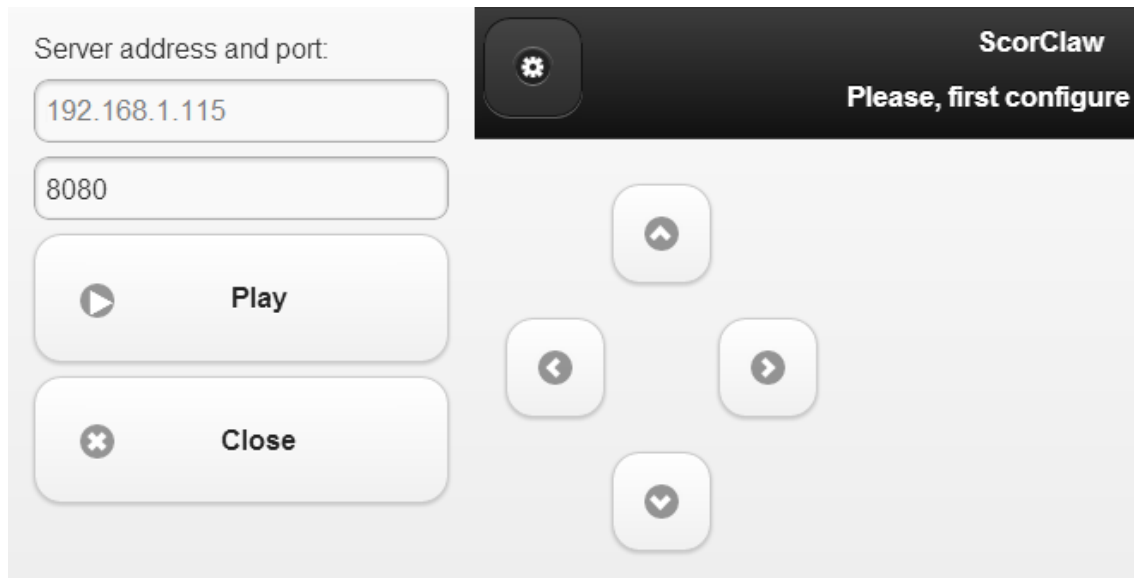
Si todo va bien, la consola imprimirá toda la comunicación que se produzca con el controlador del SCORBOT además de la información sobre las peticiones que la webapp haga con este programa.

Si no pasamos incorrectamente los argumentos al programa, este nos devuelve lo siguiente:



```
scorclaw — bash — 80x24
eva01:scorclaw josema$ python main.py
Incorrect number of arguments
The program expected the port for the server and the name of the serial port as arguments
Examples:
- Linux: python main.py 8080 /dev/ttyS1
- Mac OS: python main.py 8080 /dev/cu.PL2303-00002006
- Windows: python main.py 8080 COM3
eva01:scorclaw josema$
```

5. Accedemos a la webapp y la configuramos.



6. Ya podemos jugar con ScorClaw. Aquí se puede ver jugar a los desarrolladores: <http://youtu.be/1raUCY06ld4>

7. Para acabar, pulsamos varias veces Ctrl+c sobre la terminal. Esto además de acabar con el programa, disparará un manejador para cerrar la conexión serie con el controlador.

Nota: La webapp se puede instalar en cualquier servidor de archivos, no necesita ejecutar ningún lenguaje. Así que se puede meter en el móvil como si de unos simples archivos se tratasen para después abrirla con el navegador. En nuestro caso, para darle un toque más realista, ponemos la webapp sobre la misma máquina que la aplicación servidora y colocándonos con la terminal en el mismo directorio que la webapp ejecutamos:

```
python -m SimpleHTTPServer 8090
```

Así, usando la IP privada de la máquina podemos acceder a la webapp usando el puerto 8090. Si usamos *eduroam*, no podremos acceder con nuestro dispositivo móvil al ordenador servidor, así que tendremos que usar un router wifi o un móvil en modo tethering para crear una red local.

5. Conclusiones

Analizando la realización del proyecto a posteriori, éste ha sido uno de los más completos que hemos realizado en la carrera principalmente por dos motivos, lo útil que nos ha resultado y lo divertido que ha sido desarrollarlo.

Ha sido útil porque, como ya hemos explicado en algún punto anterior, desde un principio hemos visto la oportunidad de poder aprender otras tecnologías y así hemos hecho con el uso de Python y JQuery Mobile. Otras asignaturas que hemos estudiado cursos anteriores han sido

muy restrictivas con las tecnologías a usar y la libertad en ésta ha sido un punto muy positivo a nuestro parecer.

La parte divertida viene por el control de los robots. Esta ha sido la única asignatura de la titulación en la que el desarrollo de una aplicación ha ido unida a obtener resultados inmediatos sobre un dispositivo, cosa que nos ha encantado y que creemos que ayuda a motivar a la realización del mismo.

Como parte no tan positiva, los robots están algo obsoletos. Por supuesto, esto es algo que se escapa al control tanto de profesores como alumnos, pero debido a esto, los últimos días de preparación del proyecto sólo funcionaba correctamente uno de los tres robots de los que disponemos en el aula y la organización con los demás compañeros para poder repartir el tiempo de uso ha sido complicada.

6. Bibliografía

Información sobre Python:

<http://www.python.org/>

<http://es.wikipedia.org/wiki/Python>

<http://pyserial.sourceforge.net/pyserial.html>

Información sobre JQuery Mobile:

<http://jquerymobile.com/>

http://en.wikipedia.org/wiki/JQuery_Mobile

Información sobre *claw crane*:

http://en.wikipedia.org/wiki/Claw_crane

Manual de SCORBOT ER VPlus:

http://www.intelitekdownloads.com/Manuals/Robots/ER_V_plus_manual_100016.pdf

Canción de la webapp, The Medics – Great Is My Fear:

<http://www.jamendo.com/en/track/299446/great-is-my-fear>