Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Instrumentación, control y acceso remoto de un brazo articulado mediante sistema microcontrolador

Proyecto Final de Carrera

Ingeniería Técnica en Informática de Gestión

Autor: Daniel Martín Tórtola

Director: Antonio Martí Campoy

28/08/2014

Tabla de contenidos

1. Introducción

Esta memoria describe el Proyecto Final de Carrera con título “Instrumentación, control y acceso remoto de un brazo articulado mediante sistema microcontrolador” realizado por Daniel Martín Tórtola.

1.1 Motivación y entorno

El proyecto trata sobre la implementación de un sistema de control remoto vía web para un brazo articulado.

Elegí este proyecto porque aun siendo de la rama de Gestión siempre he tenido predilección por la electrónica y por ver “software en movimiento”. Por otro lado en la empresa en la que actualmente trabajo, he podido participar en proyectos donde he aprendido electrónica y software embebido, con lo que la realización de este proyecto, me permite demostrar estos conocimientos adquiridos.

Para llevar a cabo este proyecto, el director del mismo ha puesto a disposición el brazo robot.

Se trata de un “Quick Shot Robot Arm SVI-2000” que es controlado mediante unos joysticks. Este brazo tiene cinco ejes, cuatro de estos se encargan de mover la posición de la pinza y el quinto se encarga de la apertura/cierre de esta.

A su vez, se puede decir que el brazo está dividido en cuatro secciones: base, brazo, antebrazo y pinza

1. El primer eje es el encargado de rotar la base en ambos sentidos.
2. El segundo inclina el brazo respecto a la base.
3. El tercero inclina el antebrazo respecto al brazo.
4. El cuarto rota la pinza.
5. El quinto abre y cierra la misma.

Imagen 1 – Quick Shot Robot ARM SVI 2000

Imagen 2 - Joysticks de control

1.2 Objetivos

El objetivo principal del proyecto es generar la interfaz necesaria para sustituir dichos joysticks por un servidor web que permita el control remoto del brazo articulado, desde cualquier dispositivo con acceso a Internet/Intranet.

Para ello se ha decidido comenzar la implementación de un sistema de control utilizando los nuevos microcontroladores y tecnologías que se pueden encontrar en el mercado actual (Arduino, FTMBED, Raspbery…)

Para ello se tendrá que generar un sistema que de soporte a dicha funcionalidad, es decir:

- una capa de hardware,
- lógica que controle al hardware
- el servidor web.

2. Especificación de Requisitos

A continuación se van a describir los requisitos a cumplir para la implementación del proyecto. Se han subdividido en dos secciones, hardware y software.

2.1 Hardware

- Alimentación del brazo y del sistema:
  El brazo está alimentado mediante 4 baterías D separadas en grupos de dos en espejo. Estos grupos sacan un voltaje nominal de +-3V.
  Para evitar el uso de las pilas debido a su coste, el brazo tenía derivada la alimentación a través de cableado para usar una fuente externa.
  El objetivo sería crear otra fuente externa que alimente el brazo robot y a su vez sirva para alimentar el resto del sistema (sistema microcontrolador, finales de carrera…).

- Sustitución de los joysticks:
  Los joysticks que controlan el brazo son de 2 ejes y tienen también un pulsador. Cada joystick controla dos ejes del brazo y con el pulsador de cada uno se controla la apertura y cierre de la pinza.
  Normalmente los joysticks actúan como contactores, es decir, cuando el eje se mueve en cierta posición, cierra el circuito permitiendo que corra la corriente y el brazo se mueva. Este efecto, se puede simular con relés controlados por el sistema microcontrolador, que a su vez lo protegerían ya que los relés aislarían la etapa de potencia de la de control.

- Finales de carrera de los ejes:
  Todos los ejes, salvo el encargado de la rotación de la pinza tienen tope, es decir, a partir de esa posición lo único que se hace es sobrecargar el motor y forzar la estructura del brazo. Dado que el objetivo del proyecto es controlar remotamente el brazo, se pueden dar casos en los que estemos en esa situación. Con lo que habrá que generar un sistema de finales de carrera que detengan automáticamente el motor para evitar forzar el brazo y hacer saber al usuario dicha situación.

- Elección del sistema microcontrolador:
  Se tiene que elegir un sistema microcontrolador de los que se encuentran en el mercado actual que sea capaz de dar soporte a los requisitos descritos anteriormente, como por ejemplo Arduino, Raspbery, FTMBED…

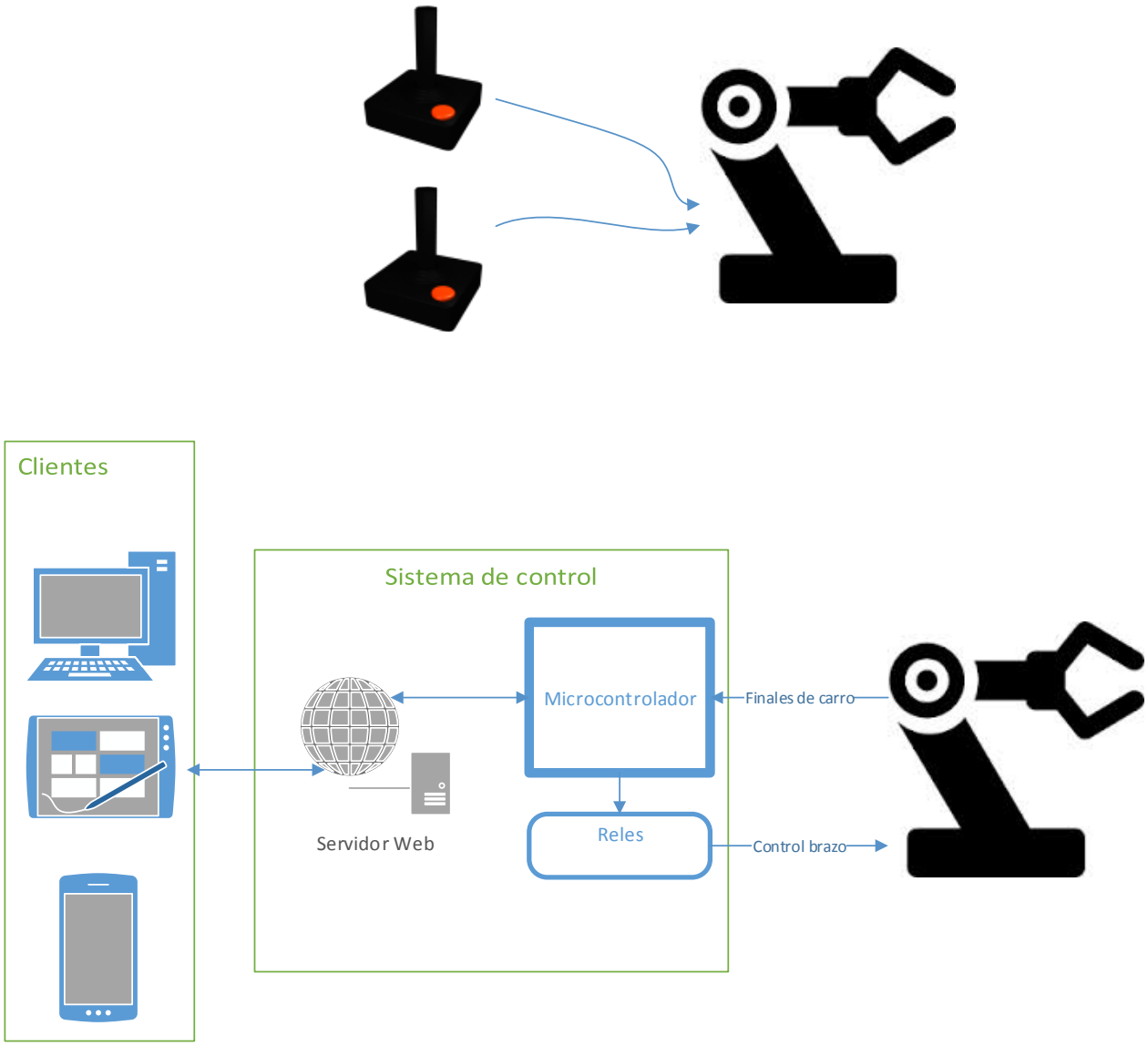

Imagen 3 - Diagrama antes y después del proyecto

Estos diagramas muestran de forma visual todo lo explicado anteriormente en los requisitos.

2.2 Software

- Control remoto: Dado que el brazo se controlará remotamente, es necesario utilizar un protocolo que gestione las peticiones de control. Éste protocolo ha de estar embebido en el sistema microcontrolador, es decir, su implementación ha de estar soportada por las librerías del mismo. Normalmente se puede implementar protocolos como el SSH, Telnet o Web, pero por facilidad de uso, me he decantado por el Web ya que permite al cliente controlar el brazo de forma intuitiva y a su vez, desde cualquier dispositivo. Uno de los inconvenientes del servidor web, es que el cliente siempre inicia la comunicación y esto es algo muy importante a tener en cuenta durante la implementación (actualizaciones de estado, control de accesos…).

- Páginas Web y su almacenamiento: El servidor mostrará las páginas que permitirán al cliente controlar el brazo. Para ello hay que pensar donde guardarlas y como mostrarlas. A su vez hay estudiar qué lenguaje de script es el más adecuado (Jquery, Ajax…).

- Modos de control (automático y manual): Para controlar el brazo se ha barajado el uso de dos modos. Por un lado manual, en el cual se indicará el eje y la dirección. Y por otro lado el modo automático en el cual el usuario con un script indicara los movimientos a realizar por el brazo.

- Gestión de accesos: Dado que al implementar el servidor web convertimos el brazo en un recurso en red, este pasa a estar accesible por todos los usuarios de dicha red, con los problemas que eso conlleva. El primer problema que se aprecia es la posibilidad de indicar órdenes contradictorias al brazo. Con lo cual, hay que evitar que dichas ordenes contrapuestas lleguen a ocurrir.

3. Implementación

3.1 Hardware

En este punto se va a explicar el hardware necesario para llevar a cabo este proyecto y como se ha realizado el montaje del sistema.

3.1.1 Hardware utilizado

Sistema microcontrolador:

Para el sistema microcontrolador he seleccionado un Arduino Mega.

El Arduino Mega consiste en una placa microcontrolador basada en ATmeg1280. Tiene 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reset.

Me he decantado en primer lugar, por el gran número de entradas/salidas que tiene, ya que investigando en el mercado los diferentes sistemas microcontroladores, este es el que más tiene. Ya que por ejemplo la Raspbery Pi tan solo cuenta con 8.

Para este proyecto, son necesarias como mínimo 16 entradas/salidas, ya que el brazo tiene 5 ejes, con lo que necesitaríamos dos salidas por eje y por otro lado en 3 de esos ejes se han instalado finales de carrera con lo que habría que sumarle 6 entradas más.

Imagen 4 - Arduino Mega

En segundo lugar, la familia Arduino, es una de las comunidades de hardware libre con mayor número de colaboradores, lo que hace que el desarrollo de cualquier aplicación para estos sistemas, sea mucho más sencillo y que se tengas a disposición una gran cantidad de herramientas, librerías y ejemplos como se puede observar en el siguiente link http://forum.arduino.cc/.

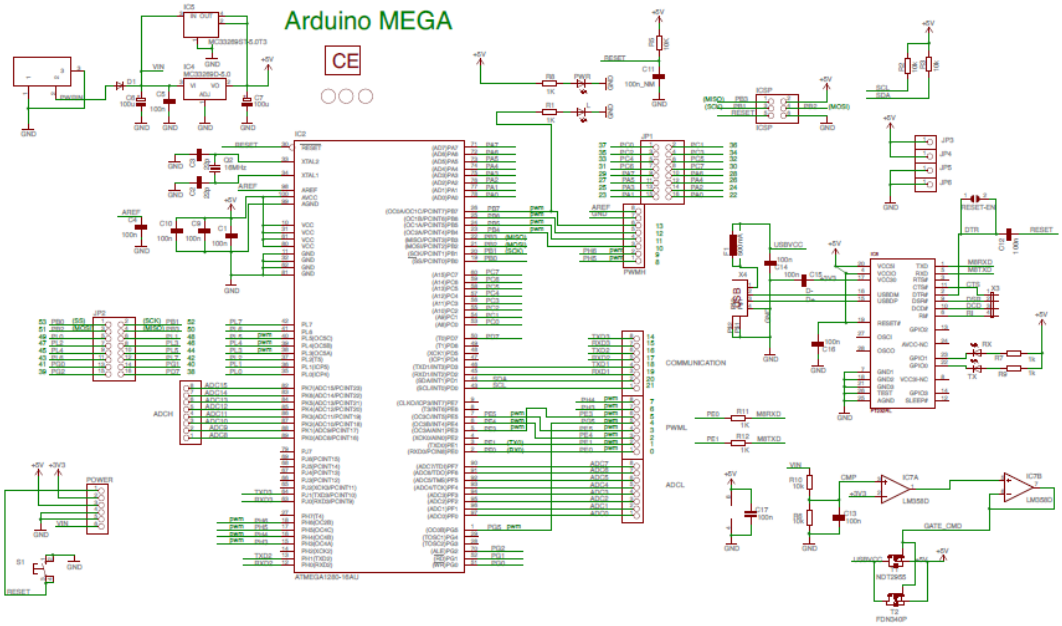

Imagen 5 - Esquema del Arduino Mega

Dado que el Arduino Mega únicamente nos permite comunicarnos con él vía puerto serie (USB) y que para llevar a cabo el proyecto es necesario una comunicación vía Ethernet hay que añadir un "shield" que añada dicha funcionalidad el Arduino.

Por ello se ha añadido el Shield Ethernet para Arduino.

Este shield, aparte de proporcionar la conexión de Ethernet, también añade la posibilidad de aumentar la memoria interna del dispositivo usando tarjetas micro SD, lo que soluciona el almacenamiento de las páginas web.

Imagen 6 - Shield Ethernet para Arduino

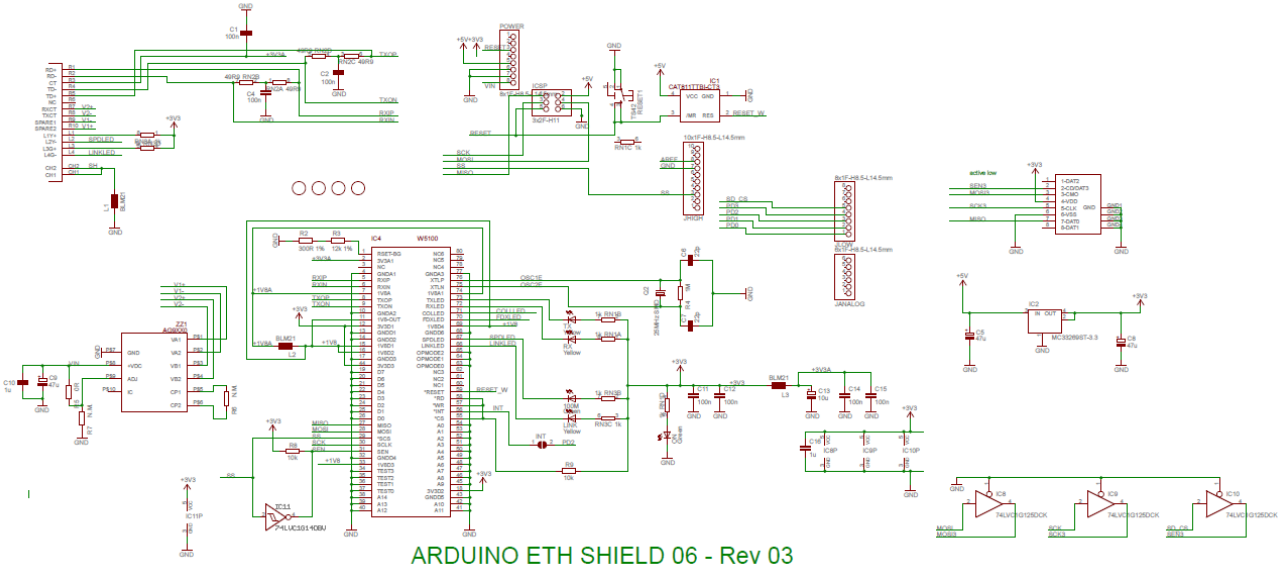

Imagen 7 - Esquema Shield Ethernet para Arduino

Finales de carrera de los ejes:

Para evitar el sobreesfuerzo de los motores al llegar a su tope, se han instalado unos finales de carrera que controlan que esto no ocurra.

Imagen 8 - Final de carrera

Estos contactores tienen tres pines, común, normalmente abierto y normalmente cerrado. Para realizar el circuito de señal, se han conectado los comunes a las entradas de Arduino, los normalmente abiertos a 5V, los normalmente cerrados a tierra y para limitar el paso de corriente por estas vías, se ha añadido una resistencia de 10Kohms en la vía de 5V.

De esta forma, el Arduino siempre estará leyendo 0V a no ser que un final de carrera sea pulsado leyendo 5V.

Sustitución de los joysticks:

Como se ha explicado en los requisitos de hardware, para sustituir los joysticks es necesario utilizar unos relés para que a través del Arduino permitan al cliente mover los ejes del brazo.

Dado que son 5 ejes, es decir 10 movimientos los posibles, necesitamos 10 relés para dar soporte a todos. Investigando las opciones que proporciona el mercado, me decante por estas tarjetas “SainSmart 8-Channel Relay Module”:

Imagen 9 - Tarjetas de 8 relés

Se han tenido que comprar dos tarjetas de 8 relés, ya que no se ha encontrado ninguna tarjeta de relés con una configuración distinta a 2^n y la tarjeta de 16 era demasiado grande.

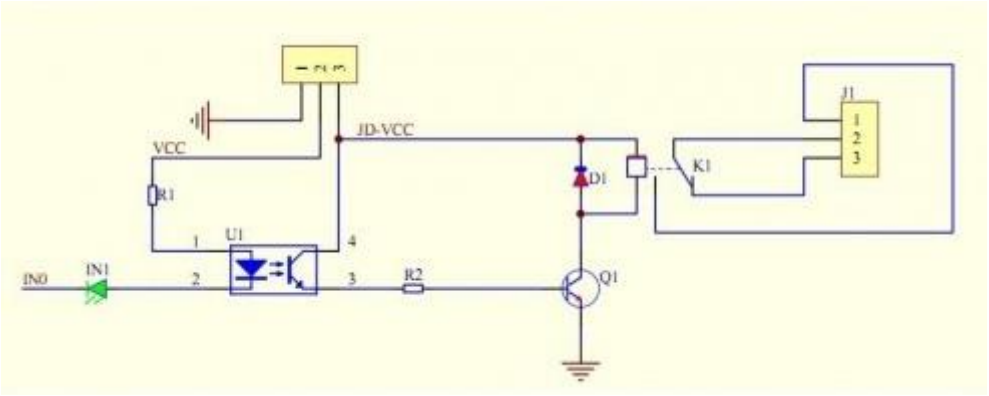

Imagen 10 - Esquema del circuito de activación/alimentación de un relé

Los joysticks están conectados al brazo robot a través de conectores DB9, para completar la sustitución de estos, es necesario hacer que las salidas de los relés vayan conectadas a un terminal DB9 y unir dichos terminales a través de cable serie.

Utilizando un multímetro, se ha descifrado el uso de cada uno de los pines del conector DB9 y su conmutación con el común. El resultado fue este:

Pines	ARM1	ARM2
PIN 8	Común	Común
PIN 8-1	Eje 2 negativo	Eje 4 negativo
PIN 8-2	Eje 2 positivo	Eje 4 positivo
PIN 8-3	Eje 1 negativo	Eje 3 negativo
PIN 8-4	Eje 1 positivo	Eje 3 positivo
PIN 8-5	Sin efecto	Sin efecto
PIN 8-6	Eje 5 negativo	Eje 5 positivo
PIN 8-7	Sin efecto	Sin efecto
PIN 8-9	Sin efecto	Sin efecto

Tabla 1 - Pinout de los conectores y su efecto al conmutar con el común.

Se adquirieron unos cables serie para conectar la salida del sistema con el brazo robot, por equivocación se adquirieron cables series con formato Null Modem que cruzan el pin out de los terminales.

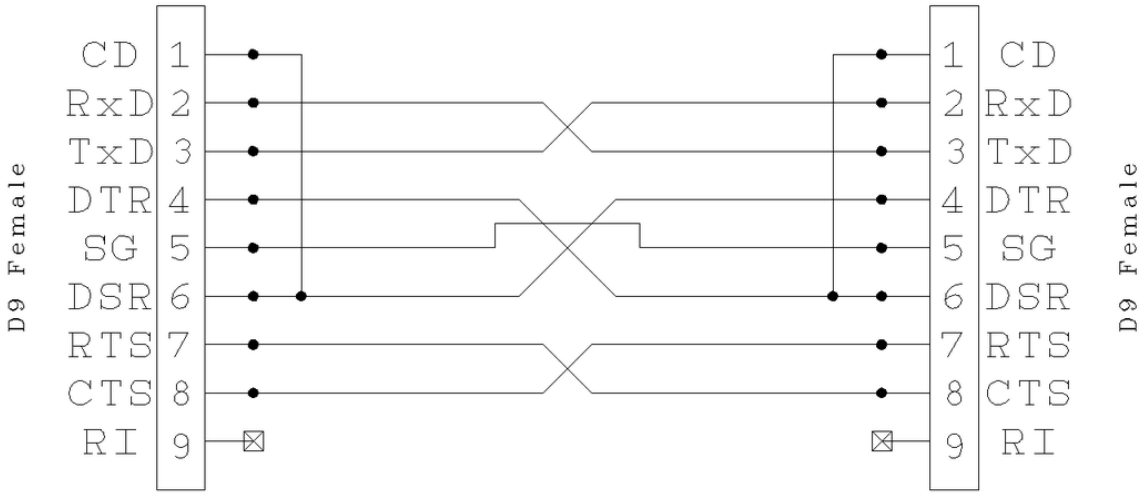

Imagen 11- Cableado DB9 Null Modem

Debido a ello fue necesario hacer manualmente los cables, ya que en las tiendas habituales solo se puede encontrar con formato Null Modem.

Alimentación del brazo y del sistema

Dado que el sistema final necesita muchas tensiones distintas

- Arduino 12V
- Brazo Robot +-3V
- Reles 5V
- Finales de carrera 5V

Me he decantado por usar una fuente de alimentación ATX para PC que saca todas esas tensiones a excepción del +-3V para el brazo robot.

ATX - Conector principal de alimentación 24 Pines(20 pines + 4 pines(11,12 y 23,24))

Tensión	Pin	Color	Color	Pin	Tensión
+3.3 V	1			13	+3.3 V
+3.3 V	2			14	-12 V
Tierra	3			15	Tierra
+5 V	4			16	PS_ON
Tierra	5			17	Tierra
+5 V	6			18	Tierra
Tierra	7			19	Tierra
Power OK	8			20	-5 V(*opcional*)
+5 VSB	9			21	+5 V
+12 V	10			22	+5 V
+12 V	11			23	+5 V
+3.3 V	12			24	Tierra

Tabla 2 - Cableado y tensiones de una fuente ATX

Imagen 12 - Fuente alimentación ATX

Para solucionar dicho problema, he creado un PCB en el cual convierto los +-12V que saca la fuente en +-3V utilizando los circuitos integrados LM317 y LM337.

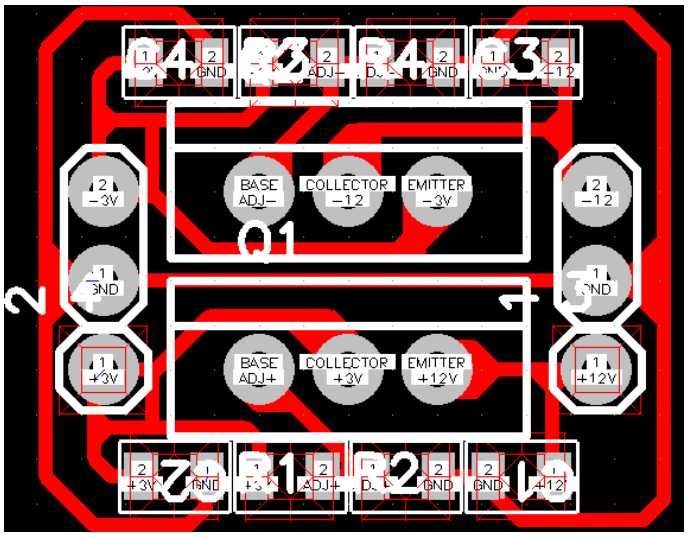

Imagen 23 - PCB conversor de tensión

3.1.2 Montaje

Todos los elementos nombrados en el apartado anterior a excepción de los finales de carrera se han instalado dentro de una caja especial para la implantación de sistemas electrónicos.

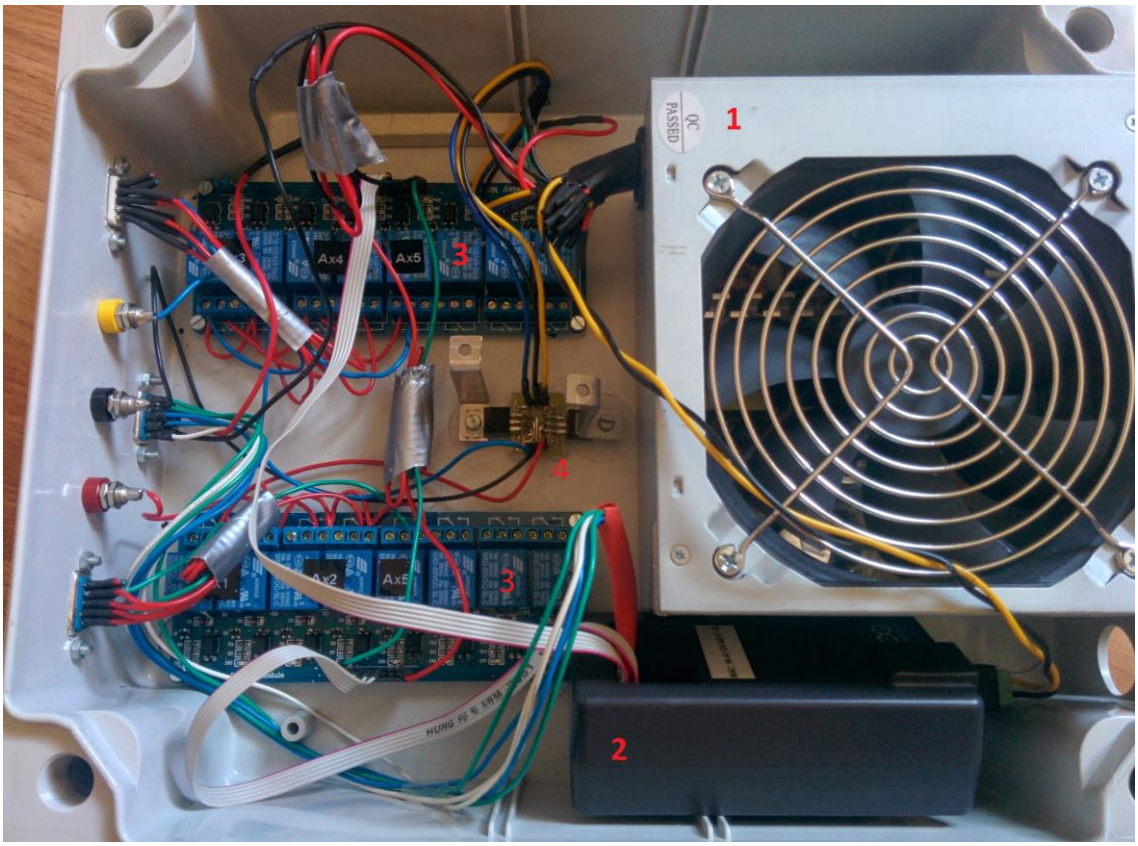

Imagen 34 – Caja vista desde arriba

Como se puede apreciar en esta imagen, dentro de la caja podemos encontrar:

1. La fuente de alimentación ATX,
2. Arduino Mega dentro su propia caja,
3. Dos tarjetas de relés
4. PCB que convierte la tensión del robot.

También se puede ver todo el cableado que interconecta dichos equipos.

- De las placas de relés al Arduino
- De los conectores DB9 a las placas de relés (control de brazo) y al Arduino (finales de carrera). Se aprecia mejor en la imagen siguiente.
- La alimentación de todos los equipos.

Imagen 45 – Frontal de la caja

Esta imagen se puede considerar como la parte frontal. Únicamente contiene una entrada para introducir el cable de Ethernet y el cable de USB (solo para depuración) y al otro lado se ve el conector de la fuente y su interruptor que a la vez hace de interruptor general.

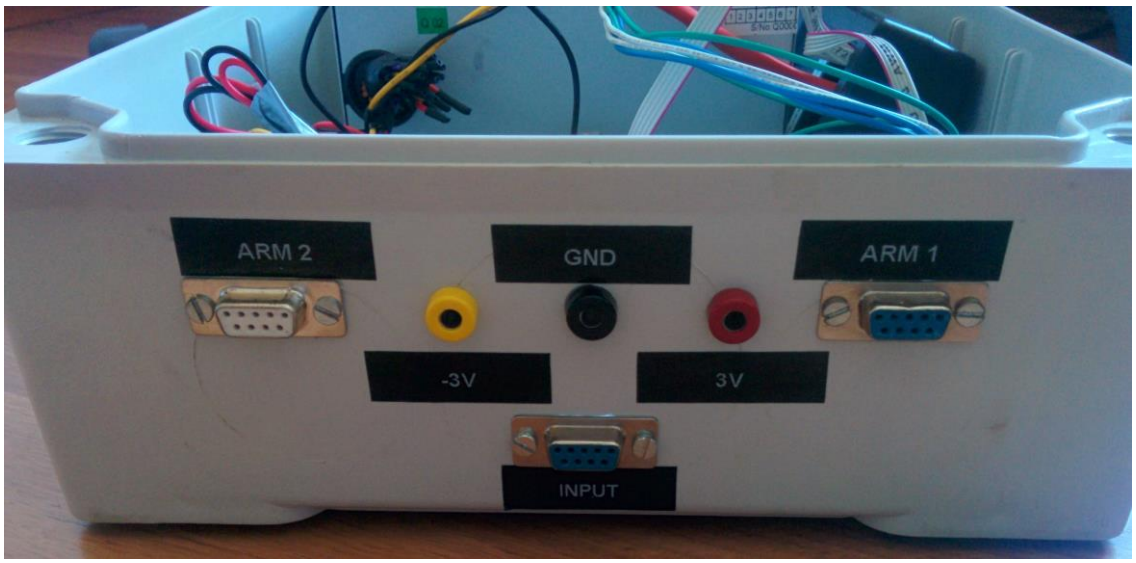

Imagen 56 – Parte trasera de la caja

Esta es la parte trasera en la cual se encuentran todas las conexiones hacia el brazo:

- Emulación del joystick (ARM1 y ARM2)
- Etapa de potencia (+-3V y GND)
- Input, que es la entrada de los finales de carrera

Imagen 67 – Caja cerrada

En cuanto a los finales de carrera a continuación se muestran unas imágenes donde se puede ver como se han situado para detectar el final del recorrido del eje.

Imagen 78 – Lado derecho brazo robot

En la imagen 17 se puede ver sobre el eje 2 (etiquetado como axis 2) un pequeño tope y los dos finales de carrera.

Imagen 89 – Lado izquierdo brazo robot

En la imagen 18 se ven los finales de carrera del eje 3, también en la parte inferior derecha de la misma los homólogos encargados del eje 1 y el apéndice metálico encargado de pulsar los finales de carrera del eje 1.

La imagen 19 muestra la parte trasera del brazo donde se puede ver las conexiones hacia la caja (ARM1 y ARM2) y las bananas de alimentación.

Imagen 20 – Parte trasera del brazo robot

Aunque son cuatro los ejes que tienen tope mecánico (1, 2, 3 y 5) el eje 5, es decir la pinza, no tiene finales de carrera ya que no se pueden situar sin interferir en el funcionamiento de la misma.

3.2 Software

En esta apartado se va a explicar la funcionalidad que realiza el software dentro del sistema sin entrar en detalle en la codificación. Para ello se va a mostrar un diagrama de bloques que muestra la lógica interna y se van a explicar las funciones que lo llevan a cabo.

3.2.1 Diagrama de bloques

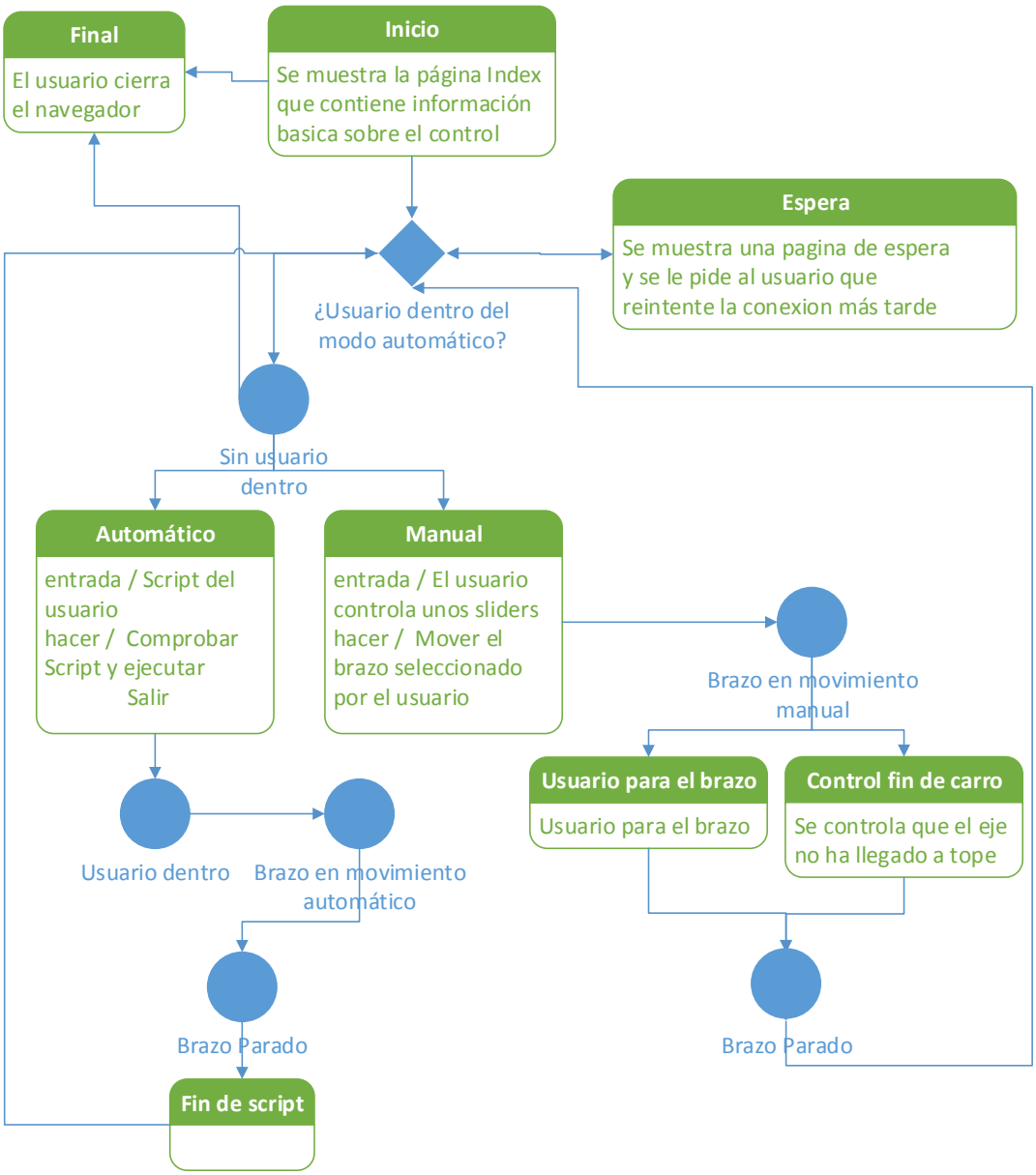

Imagen 21 – Diagrama lógica de la aplicación

En este diagrama se muestra que existen dos formas de controlar el brazo, una automática y otra manual.

La manual solo se encarga de realizar dos funciones, los diferentes movimientos del brazo y parar el brazo ya sea por petición del usuario o cuando los finales de carrera son activados.

La automática, sin embargo, es más restrictiva, es decir, no se debe permitir a más de un usuario ejecutar un script, para ello se ha desarrollado un control de accesos que se encarga de detectar si hay un usuario creando un script o ejecutándolo. De ser así, a todas las demás conexiones entrantes se les inhabilita tanto el acceso mostrando una página de espera, como el control si ya se encuentran dentro.

3.2.2 Software embebido

Dado que el proyecto está realizado usando un Arduino Mega, se ha utilizado C para la programación y las librerías propias de C más las que están desarrolladas para controlar el micro de Arduino.

En este caso se han utilizado las siguientes librerías:

- SPI.h: Permite el uso del puerto USB para generar una comunicación serie. En este proyecto se utiliza para la depuración del servidor web.
- Ethernet.h: Genera una capa Web que permite al Arduino ser tanto servidor como cliente.
- SD.h: Permite la lectura y escritura de archivos localizados en la tarjeta SD.

Todas las aplicaciones desarrolladas para Arduino tienen en común dos funciones que son “obligatorias”. Estos son el setup y el loop.

El setup es la función que se encarga de inicializar tanto el Arduino (configuración GPIO, baud rate puerto serie, MAC e IP del servidor…) como las variables internas para la aplicación.

- GPIO
  - 22 a 31 salidas digitales que controlan el brazo
  - 34 a 39 entradas digitales que leen los finales de carrera.
- MAC e IP: La MAC es dependiente de cada dispositivo red y la IP es variable y se carga desde un fichero localizado en la tarjeta SD.
- Puerto serie: Se utiliza para depurar, únicamente muestra si se han cargado correctamente los archivos desde la tarjeta SD y las peticiones web hechas por el cliente. Su baud rate es: 9600.
- SD: En la inicialización de la tarjeta SD se comprueba que todos los archivos necesarios (webs, ip.txt…) están dentro y son legibles.

Y la función loop como su nombre indica es el bucle principal, es decir es la rutina que se va a repetir a cada tick del microcontrolador. En este caso, se encarga de gestionar las peticiones web y la gestión de los finales de carrera.

Las peticiones web que puede recoger el loop son las siguientes:

- Index, Auto o Wait: al recibir estas cabeceras HTTP se carga la página solicitada y se muestra al cliente.
- Ajax_inputs: Como su nombre indica, son valores enviados desde la web a través de Ajax (se explican más extendidamente en el punto siguiente) y dependiendo de los valores enviados se mueve el brazo.
- Auto_script: Contiene una línea con la sentencia de script a ejecutar por el brazo robot.

Los finales de carrera se comprueban en cada loop y cada vez que se indica mover el brazo. Esto es denominado rutina polling, es decir, es una petición constante del estado de esas variables, para actuar en consecuencia.

En un principio se pensó en implementar los finales de carrera con interrupciones hardware, es decir, interrupciones que paran el loop principal y ejecutan una subrutina, pero por limitación del propio Arduino Mega, no es posible.

El Arduino Mega tiene disponibles 6 pines para la interrupción hardware que serían suficientes para esta aplicación, pero al añadirle el shield de Ethernet, solo deja útiles 4, que no son suficientes.

Además dado a que la velocidad tanto de reacción como de los movimientos del brazo robot es bastante lenta usar rutinas polling en este caso es una buena solución para evitar el sufrimiento de los motores y de la mecánica.

Por otro lado también se almacena el estado de cada eje. Para ello se ha creado una estructura que almacena toda la información correspondiente de cada eje y que envía por XML la información leída a la web para indicar constantemente el estado del brazo robot. Esta estructura de enteros contiene por cada eje:

- Dos pines de control, uno por cada salida digital usada para mover el brazo.
- Dos pines de finales de carrera, uno por cada entrada digital que muestran al usuario en la página web si se ha alcanzado un final de carrera.
- Estado actual, en movimiento o parado indicado con estas variables constantes:
    - const int NEG = -1;
    - const int POS = 1;
    - const int STOP = 0;

Por ejemplo, información correspondiente al eje 1:

```
Axis[0].Out_Neg = 22;
Axis[0].Out_Pos = 24;
Axis[0].In_Neg = 35;
Axis[0].In_Pos = 34;
Axis[0].Estado = STOP;
```

La estructura Axis, es un array de 5 posiciones que guarda la información de cada eje, las variables out corresponden a los pines de microcontrolador encargados de activar los relés, y las in son pines de microcontrolador encargados de leer los finales de carrera.
Esto es una ventaja si hubiera que modificar el hardware, ya que únicamente editando los valores de esta estructura tendríamos el software embebido de nuevo compatible con el hardware.

El XML que se le envía a la web únicamente contiene los valores actuales de las salidas/entradas de cada eje sin almacenamiento intermedio, es decir, usando las variables in y out de cada eje se lee el valor actual y genera el XML siendo el resultado enviado así:

```
<inputs>
        <switch>ON</switch>
        <switch>OFF</switch>
        <switch>OFF</switch>
        <switch>OFF</switch>
        <switch>OFF</switch>
        <switch>OFF</switch>
        <axis>0</axis>
        <axis>1</axis>
        <axis>-1</axis>
        <axis>0</axis>
        <axis>0</axis>
</inputs>
```

En el anterior texto en formato XML, se pueden ver seis switches y cinco axis.

Cada entrada switch corresponde a un final de carrera y están agrupados por parejas (positivo y negativo) que representan los tres ejes con finales de carrera instalados, es decir, las dos primeras entradas switch corresponden el eje y así consecutivamente.
En este caso se puede apreciar que el final de carrera negativo del eje uno está activado.

Cada entrada axis represente el estado actual de cada eje. En este caso todos están parados a excepción del eje 2 que se mueve positivamente y del eje 3 que se mueve negativamente.

A su vez también se han creado unas funciones auxiliares que se encargan de mover el brazo a partir de los datos de la estructura Axis y de temporizar las acciones (tiempo ejecución script, tiempo sin usuario…)

En el siguiente diagrama se puede ver todo lo explicado en este apartado de una manera más gráfica.

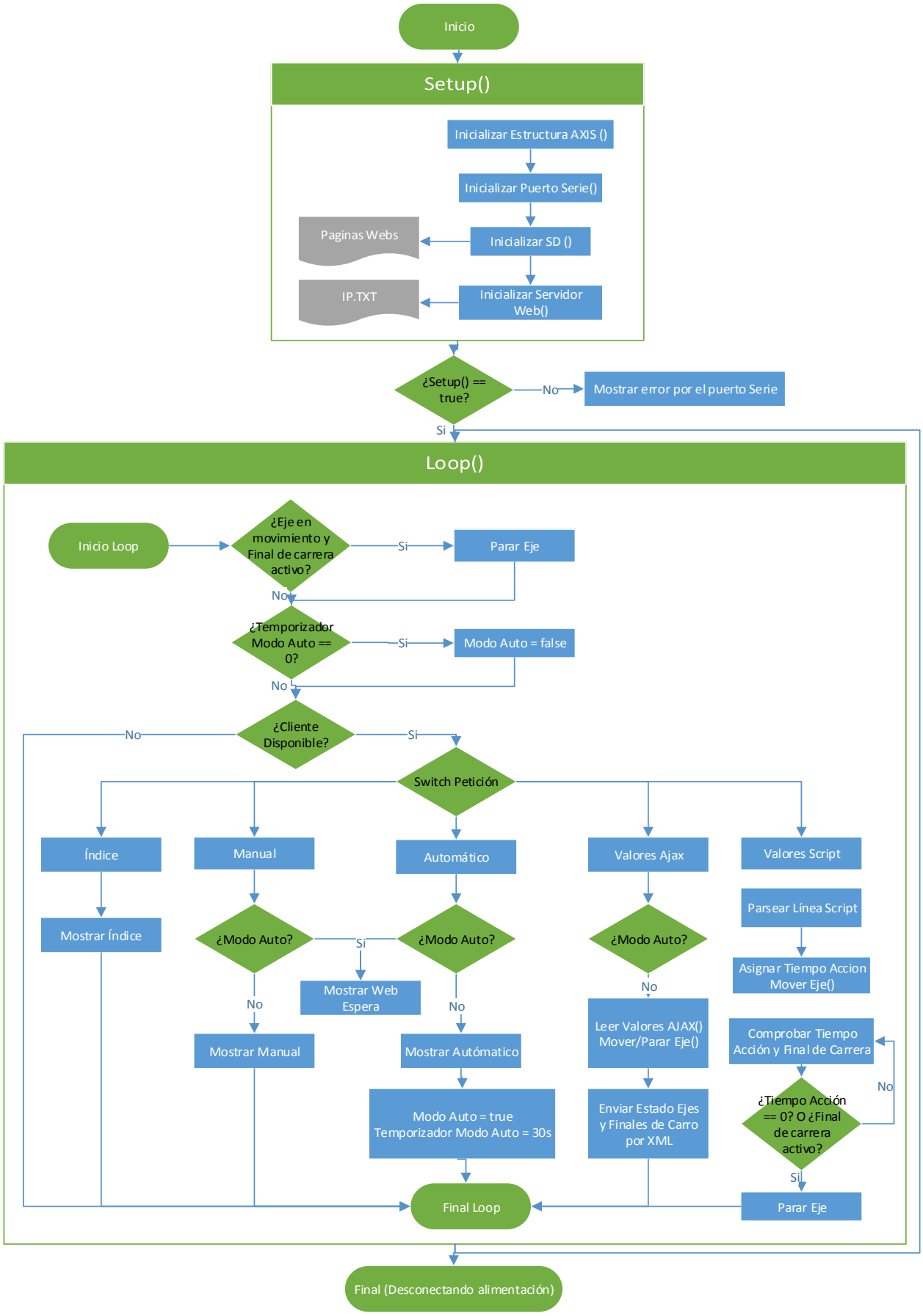

Imagen 22 – Diagrama funciones software

3.2.3 Desarrollo Web

La capa web está basada en cuatro páginas web que son: índice, manual, automática y espera.

Para el diseño se han utilizado estilos CSS que es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML y Jquery que son unas librerías para Javascript que por un lado permiten que la generación de scripts sea más intuitiva y por otro permite aplicar diseños predefinidos bastante atractivos.

Además se ha añadido funcionalidad con AJAX para mantener actualizado todas las webs con el estado actual en el que se encuentra el brazo. El AJAX permite hacer peticiones al servidor sin necesidad de la intervención del usuario y actualizar el estado de la web mostrada sin necesidad de recargarla completamente.

Índice:

Imagen 23 – Página web índice

En esta página únicamente muestra la información para conocer la anatomía y los modos de control del brazo.

También permite seleccionar los modos de control pulsando los links de la descripción o los botones situados en la parte superior de la página.

Manual:

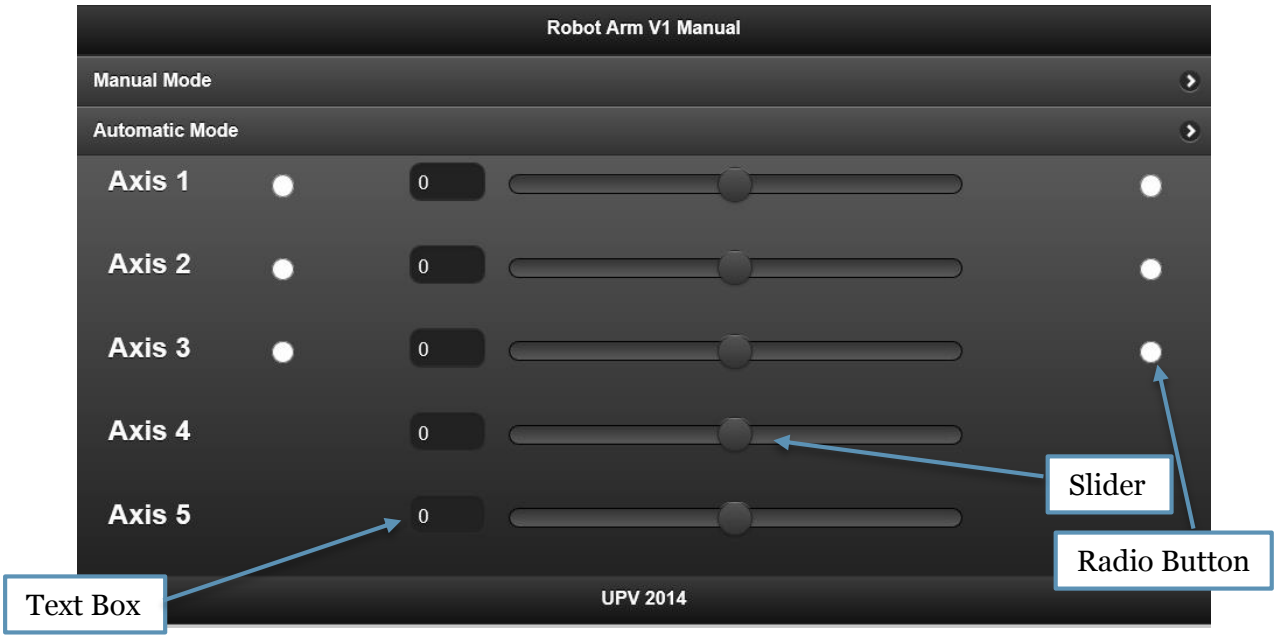

Imagen 24 – Página web para el control manual

Esta página es la que se encarga del control manual del brazo robot, es decir, es una emulación directa del joystick. Básicamente la página permite al usuario que se indique el eje y la dirección en la que mover el brazo. Por otro lado el cuadro de texto indica si el moviemiento esta siendo ejecutado, es decir, si la accion se ha efectuado correctamente y los radio buttons situados a cada lado del slider indican si se ha alcanzado el final de carrera.

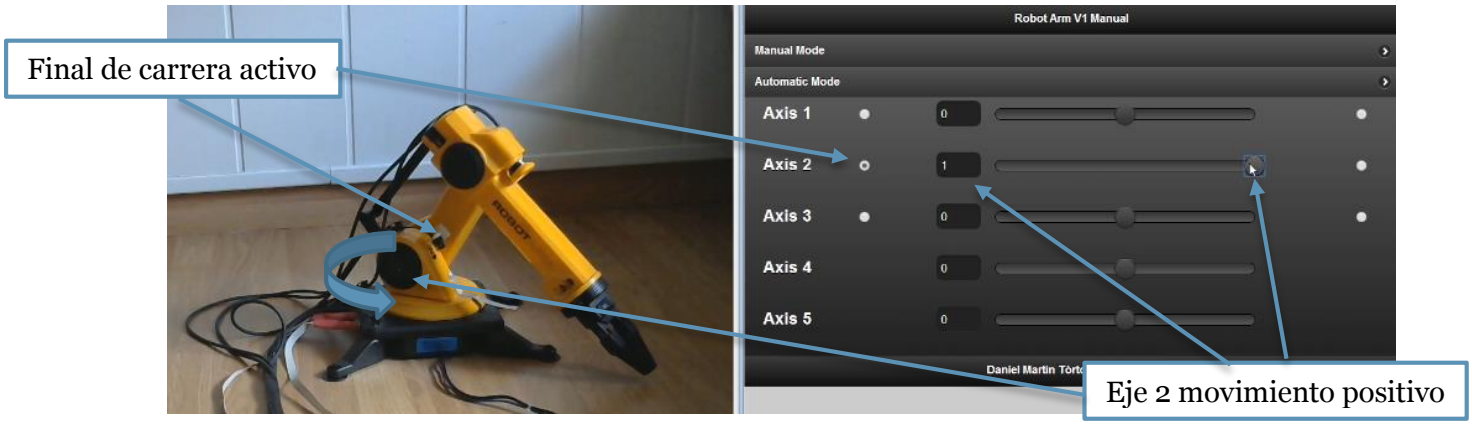

Imagen 25 – Página web para el control manual en uso

En esta imagen de la web manual en uso, se puede ver al brazo en el tope negativo del eje 2 y comenzando a mover positivamente el eje 2 igualmente.

Tambien hay que añadir que esta página puede ser utilizada por varios clientes a la vez. Cuando un cliente ordene un nuevo movimiento, a los demas clientes conectados aparecerá dicha orden en los cuadros de texto al lado de cada eje.

Automático:

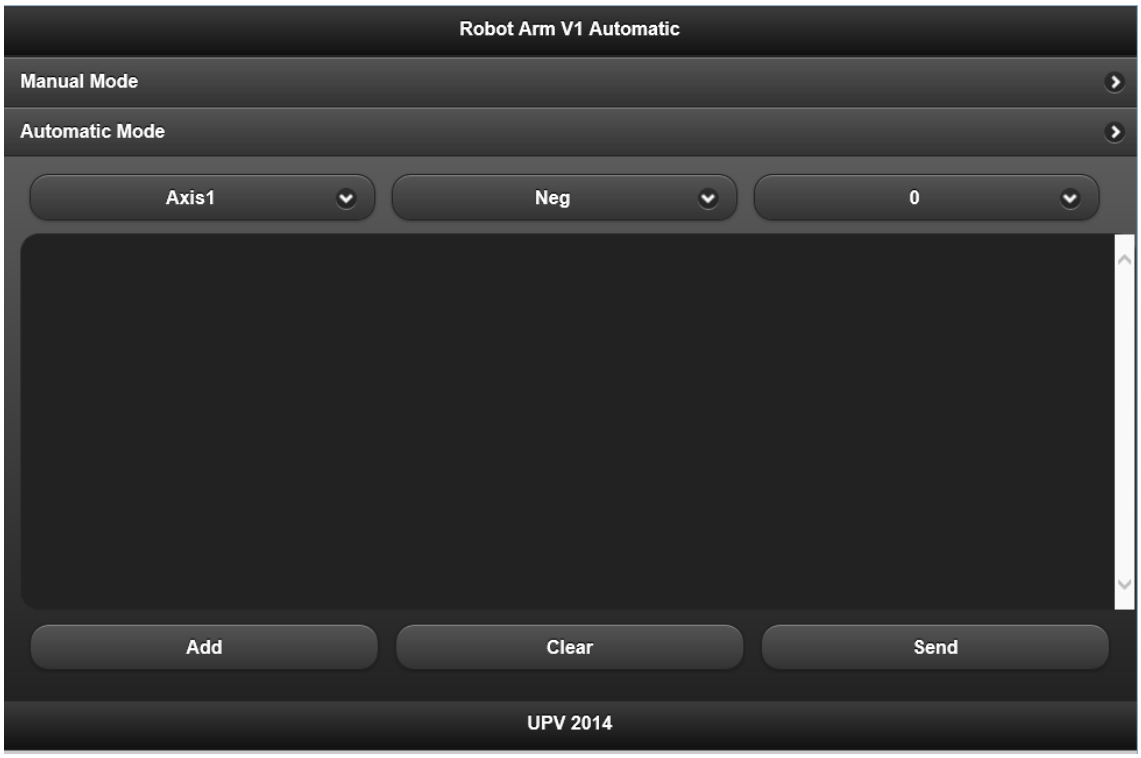

Imagen 26 – Página web para el control automático

Está pagina se utiliza para crear los scripts que posteriormente el brazo ejecutará. Durante el desarrollo se planteó el uso de lenguajes de script (javascript) o lenguajes interpretados (phyton) para crear los scripts, pero esto obligaba que el usuario conociera algo de programación.

Por lo que al final se realizo de la siguiente manera:

Mediante unos selectores se elige el eje, dirección y tiempo a mover. Una vez hecho esto, se pulsa el boton “añadir” para añadir esa instrucción al script.

Cuando se ha finalizado el script deseado se pulsa el boton “enviar” y el brazo robot ejecuta linea a línea el script generado.

Con el botón “limpiar” se vacia la consola del script.

Ejemplo de script:

axis1,-1,1

axis1,1,2

axis4,-1,1

axis5,-1,4

Como se puede leer el primer paso del script movera el eje 1 durante 1 segundo en dirección negativa.

Espera:

Imagen 27– Página web de espera

Como se ha indicado en el diagrama de bloques, el modo automático es restrictivo, es decir en el momento que un usuario accede al modo automático se inhabilita el acceso a nuevos usuarios e igualmente se dejan de atender todas las peticiones de los clientes en modo manual.

En el caso de los clientes conectados en modo manual verán únicamente que las acciones que indican al brazo no están siendo ejecutadas, pero en el caso de las nuevas conexiones se mostrará la página web de espera.

3.3 Herramientas utilizadas

3.3.1 SVN

Subversion (SVN) es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD.

Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones (delta), optimizando así al máximo el uso de espacio en disco. SVN permite al usuario crear, copiar y borrar carpetas con la misma flexibilidad con la que lo haría si estuviese en su disco duro local. Dada su flexibilidad, es necesaria la aplicación de buenas prácticas para llevar a cabo una correcta gestión de las versiones del software generado.

SVN puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores.

3.3.2 Visual Micro

Visual Micro es un plugin para Microsoft Visual Studio 2008-2013 y para Atmel Studio 6.1-6.2 que permite que cualquier Proyecto Arduino sea desarrollado, compilado y cargado en cualquier placa Arduino.

Básicamente este plugin emula todas las funcionalidades que tiene el Arduino IDE y le añade las siguientes:

- Depuración en vivo.
- Intellisense, escritura predictiva.
- Y unas cuantas más que no se ha llegado a usar.

Decidí usar esta opción al estar familiarizado con el entorno de Visual Studio, pero tenía una serie de bugs que muchas veces bloqueaban el desarrollo completamente, por ejemplo, el bloqueo de los puertos USB cuando se lanzaba la comunicación serie.

Por este y muchos otros motivos volví a usar Arduino IDE que aun siendo más básico es completamente funcional.

3.3.3 OrCAD

OrCAD es una herramienta de diseño de PCBs y la he utilizado para crear el conversor de tensión.

3.3.4 Arduino IDE

Es un editor de C para Arduino muy básico, no permite depuración, y únicamente busca errores de compilación, pero aun así es la herramienta más sencilla y efectiva para crear una aplicación para Arduino y programarla en el microcontrolador.

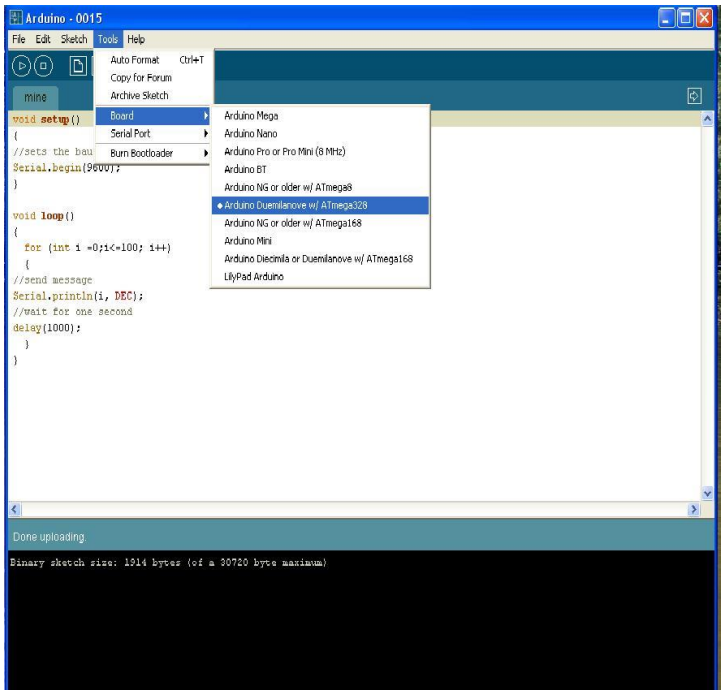

Imagen 28– Arduino IDE

3.3.5 Dreamweaver

Es una aplicación en forma de estudio (basada en la forma de estudio de Adobe Flash) que está destinada a la construcción, diseño y edición de sitios, vídeos y aplicaciones Web basados en estándares.

En este proyecto se ha usado tanto para crear las páginas web, como para comprobar el funcionamiento de los scripts.

3.3.6 Varios

Obviamente también se han utilizado otras herramientas mucho más comunes como las contenidas en el paquete Office tanto para generar la memoria como para crear los diagramas y tablas.

3.3.7 Lenguajes de programación

ANSI C: Es el lenguaje de programación compilado que se ha utilizado para programar el firmware que lleva el micro embebido.

AJAX (Asynchronous JavaScript And XML): Es una tecnología asíncrona que se utiliza para intercambiar información entre el servidor y el cliente. En este proyecto se ha utilizado para refrescar el estado en el que se encuentra el brazo robot y comunicarlo al cliente web.

Javascript: Es un lenguaje interpretado que se utiliza en el cliente, en este caso en las páginas web. Sirve para poder acceder a los objetos DOM (Modelo de Objetos del Documento) y poder acceder a sus propiedades para leerlas o modificarlas. En el proyecto se ha usado como soporte de AJAX en el refresco de las webs y para facilitar la creación de los scripts del modo automático.

Jquery: Es una biblioteca que genera una capa sobre Javascript que simplifica la interacción con los objetos DOM, a su vez también crea estilos y animaciones muy vistosas para generar páginas web más intuitivas para el usuario. En el proyecto se ha usado para crear el diseño de las webs y dar soporte a las funciones AJAX.

HTML (HyperText Markup Language): Es el lenguaje utilizado para desarrollar páginas web y ese ha sido su principal uso en este proyecto.

4. Presupuesto

Artículo	Cantidad	Precio Unitario	Precio Total
Arduino Mega rev3	1	43,06 €	43,06 €
Shield Ethernet	1	28,55 €	28,55 €
Caja Arduino Mega	1	14,86 €	14,86 €
Placa 8 relés	2	11,00 €	22,00 €
Tarjeta micro SD 2Gb	1	2,99 €	2,99 €
Fuente de alimentación 500W	1	16,42 €	16,42 €
Caja	1	41,00 €	41,00 €
Conector Hembra DB9	4	1,37 €	5,48 €
Conectores Macho DB9	2	1,29 €	2,58 €
Cable alimentación	1	5,20 €	5,20 €
Bobina Cable 20m	1	10,00 €	10,00 €
Conector Banana hembra	3	1,30 €	3,90 €
Final de carrera	6	0,922 €	5,53 €
Total			201,57 €

Tabla 3 – Presupuesto total del proyecto

5. Pruebas

Este apartado pretende explicar de manera más visual el resultado final obtenido tras la implementación del sistema.

5.1 Puesta en Marcha

El sistema está compuesto de tres partes como se ha explicado previamente, brazo, sistema de control y servidor. Cada uno de ellos tiene una pequeña puesta en marcha y una serie de comprobaciones para asegurarse del buen funcionamiento del mismo.

Brazo Robot:

El brazo robot tiene dos conexiones de entrada, una de salida y la alimentación. Todas estas conexiones han de conectarse con la caja donde se encuentra el sistema de control.

Las conexiones de entradas son ARM 1 y ARM 2 y han de conectarse usando los cables serie a sus homónimas en la caja.

La conexión de salida o cableado de los finales de carrera ha de conectarse en el terminal serie etiquetado en la caja como INPUT.

Y por último los cables de alimentación han de ir conectados a la caja usando el sistema de colores de las bananas y los conectores.

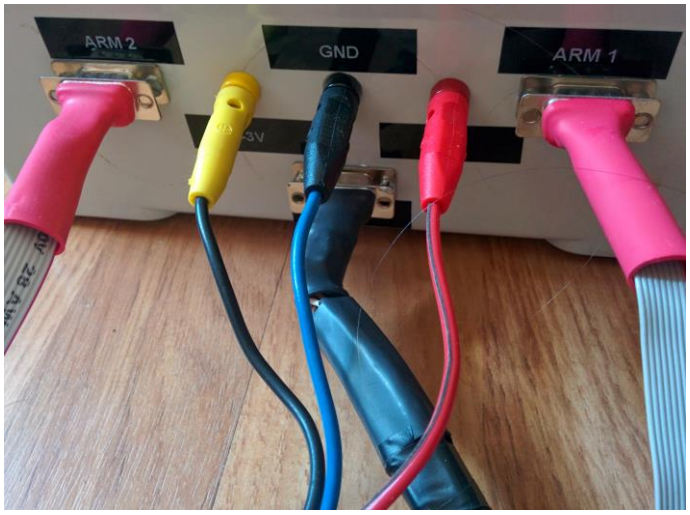

Imagen 29 – Conexionado entre la caja y el brazo

Sistema de Control:

Dentro de la caja hay que conectar con Arduino todos los demás componentes que se encuentran dentro (fuente de alimentación, tarjetas de relés, y cableado de los finales de carrera).

Ambas tarjetas de relés tienen un cable plano con uno de los cables de color rojo. Ese cable se considera el que indica la polaridad u orden. Hay que conectar la tarjeta de relés etiquetada con el Ax1 y Ax2 al puerto 22 y la otra (etiquetado como Ax3 y Ax4) en paralelo, es decir en el puerto 23. Se sobreentiende, que tanto en el puerto 22 como 23 tiene que ir el cable rojo de los cables planos.

Los finales de carrera terminan en un conector con un capuchón rojo hecho con plástico termo retráctil. Dicho conector tiene que conectarse ocupando los puertos del 34 al 41, siendo los dos puertos 40 y 41 a los pines que no está cubiertos por el capuchón, es decir, estos pines son los que han de estar conectados al final.

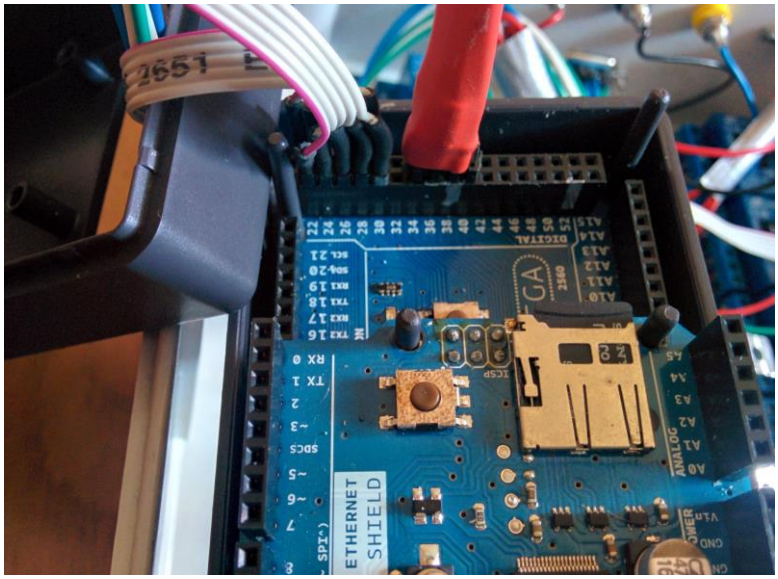

Imagen 30 – Conexionado puertos Arduino y tarjeta SD

La fuente de alimentación alimenta con 12V al Arduino a través del terminal Jack.

Por otro lado, también hay que conectar el cable de Ethernet y el de USB (si se quiere depurar).

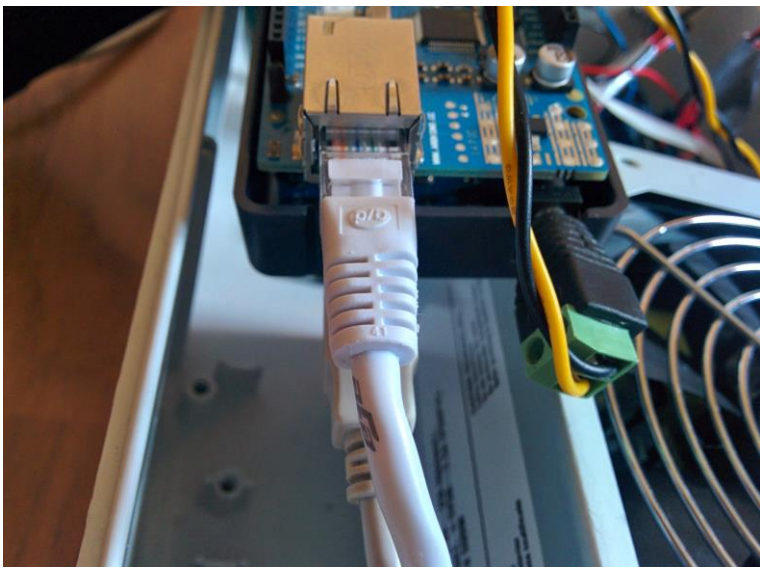

Imagen 31 – Conexionado puertos Arduino y tarjeta SD

Servidor Web:

Lo único que se puede configurar del servidor es la IP a través de la cual se va a manejar el brazo. Para hacer eso hay que extraer la tarjeta micro SD conectada en el Arduino e indicar en el archivo "ip.txt" la dirección deseada. En el caso de haber algún error en el formato de la IP o en la ausencia del archivo, el web se mostrará en la IP por defecto 192.168.1.200.

5.2 Videos

En estos videos alojados en YouTube se muestra todo es sistema en funcionamiento y en sus distintos modos de manejo, manual, manual cooperativo y automático.

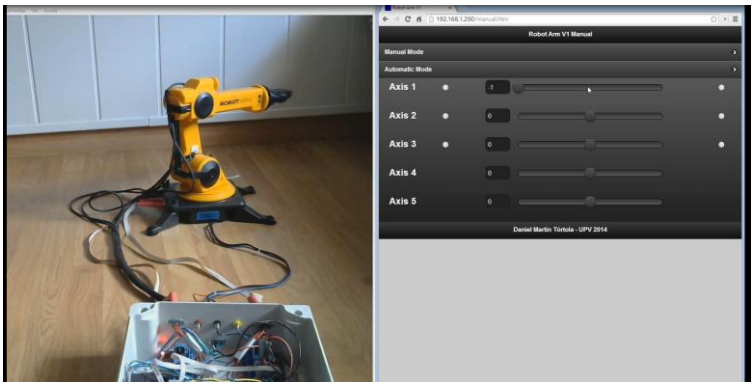

Imagen 32 – Video control manual (link pulsando imagen)

Imagen 33 – Video control manual cooperativo (link pulsando imagen)

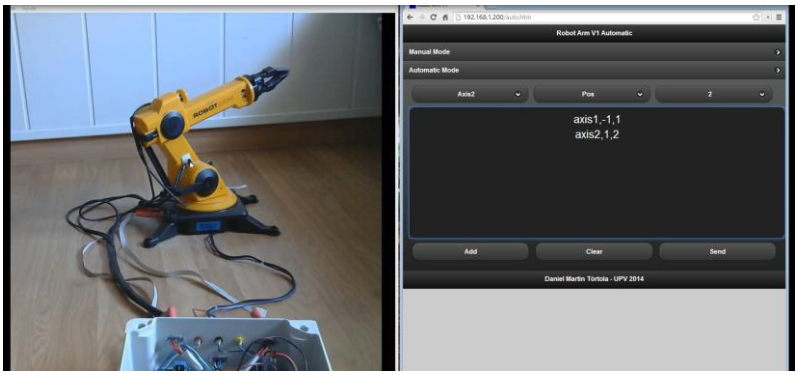

Imagen 34 – Video control automático (link pulsando imagen)

6. Conclusiones

Con la elaboración de este proyecto, he podido trabajar en campos completamente distintos a los que me había centrado durante el desarrollo de la carrera, ya que soy un estudiante de la rama de Gestión y este proyecto abarca una gran parte de electrónica y software de bajo nivel.

Realizando este proyecto me he encontrado con problemas que no habían surgido nunca durante el desarrollo de aplicaciones para PC o Web ya que estos sistemas no tienen las limitaciones que se tienen al programar en un microcontrolador, por ejemplo, velocidad a la hora de realizar operaciones, saturación de RAM…

Me ha sido un reto en aspectos como la parte mecánica, ya que en el mundo del software no se usan demasiadas tuercas, pegamentos, soldadores, cables y otros tipos de tortura. Pero no hay que ser negativo en ese aspecto ya que aunque sea mucho el tiempo perdido en pruebas, es muy bonito, ver el software "moverse".

En cuanto a la capa web, he tenido muchos quebraderos de cabeza a la hora de desarrollar el servidor. En un principio empecé haciendo el servidor con una librería open source llamada WebDuino. Esta librería a modo muy básico permitía generar un servidor rápidamente, pero cuando se le pedía algo más de funcionalidad de la que técnicamente soportaba (Ajax, XML, Jquery…) el servidor daba fallos aleatorios. Por lo que tuve que replantear el servidor y montarlo usando las librerías propias de Arduino, algo menos intuitivas pero funcionales.

Siguiendo en la capa web, me he encontré con el dilema de hacer el servidor restrictivo totalmente, es decir, permitir que un único usuario manejara el brazo, pero decidí investigar como se suele resolver estos dilemas y hallé Ajax que como se explica en esta memoria resuelve dicho problema satisfactoriamente.

Este proyecto me ha portado unos conocimientos en electrónica, más específicamente, placas microcontrolador que seguramente me serán muy útiles en mi futuro profesional tanto como en el personal, ya que me gustaría investigar por mi cuenta soluciones domóticas para el hogar.

Para finalizar este proyecto debo decir que me siento orgulloso y realizado ya que elegí este proyecto como un reto personal y creo que su desarrollo es satisfactorio.

Debo dar las gracias al director de este proyecto ya que ha resuelto todas mis dudas y me ha aportado ideas, también a GND S.A. por facilitarme todo el instrumental y paciencia de soportar mis momentos de falta de inspiración y a mi pareja por aguantar el tener la casa llena de "juguetes".

Con ello doy por finalizada mi etapa estudiantil.

7. Bibliografía

Arduino. (s.f.). Obtenido de http://arduino.cc/en/pmwiki.php.

JQuery. (s.f.). Obtenido de http://jquery.com/.

Mbed. (s.f.). Obtenido de https://mbed.org.

Raspberry Pi. (s.f.). Obtenido de http://www.raspberrypi.org/.

Subversion. (s.f.). Obtenido de https://subversion.apache.org/.

Visual Micro. (s.f.). Obtenido de http://www.visualmicro.com/.

W3Schools. (s.f.). Obtenido de http://www.w3schools.com/.

Webduino. (s.f.). Obtenido de https://code.google.com/p/webduino/.

Wikipedia Org. (s.f.). Obtenido de http://es.wikipedia.org/.