Sin embargo, el utillaje empleado en cadena de montaje (Figura 2.31) fue rediseñado por el Dpto. de Mecánica y fabricado por sinterizado selectivo por láser, para lograr un acabado más preciso y mayor resistencia. Le añadieron palancas para fijar cada juguete y mejoraron los zapatos para facilitar la colocación.

### 2.4.2 Software

En esta subsección se explicarán los aspectos relacionados con la programación del sistema y los algoritmos empleados. Se ha divido en tres partes, una para cada uno de los tres dispositivos principales, Mega, Raspberry y Zowi. Puede ser útil, para entender el funcionamiento del sistema, consultar los anexos XX,XX,XX, donde se muestra el Manual de Usuario y el Código desarrollado.

#### 2.4.2.1 Mega

Este dispositivo se puede programar utilizando el lenguaje de Arduino, que es una adaptación de C++ que proveniene de avr-libc y provee de una librería de C de alta calidad para usar con GCC en los microcontroladores AVR de Atmel y muchas funciones específicas para los MCU AVR de Atmel. Su simplicidad y la implicación por parte de la comunidad hacen realmente fácil y rápido su uso y aprendizaje.

El programa principal ha sido creado en el entorno de Arduino, en un fichero (.ino). Se emplean las funciones de Arduino precocinadas y se extiende de librerías de Arduino, programadas en C++, para todo lo posible, a destacar:

- LiquidCrystal para escribir al display LCD.
- Wire para la comunicación I2C.
- Ficheros de la librería MinIMU, que extiende a su vez de las librerías del giróscopo y el acelerómetro, L3G.h y LSM303.h, respectivamente.

El programa de Mega sigue una estructura de máquina de estados que se encarga de llevar al operador por los diferentes pasos del proceso de calibración, es aquí dónde se toman las lecturas de las posiciones de los servos y se decide cuánto se han de mover, dónde se valida el resultado de la calibración y dónde se produce la interacción entre la máquina y el usuario. Se puede decir que es el principal dispositivo del sistema.

Para conocer la lógica de la máquina de estados, se pueden consultar los diagramas del anexo XX, para conocer el proceso de uso, como operador, se puede consultar el manual de usuario del anexo XX.

##### 2.4.2.1.1 Protocolo de comunicación

Para lograr enviar instrucciones a la controladora del robot utiliza como intermediario a la Raspberry, que establece una comunicación serie por USB con ambas controladoras. Se desarrolla un protocolo de instrucciones que tiene las siguientes partes:

- Signo de dólar ($) como carácter inicial de la trama.
- 4 caracteres que indican el comando función en cuestión.
- Dos puntos (:) para indicar inicio de valores de parámetros.
- Parámetros de la función, si los tuviera, separados por el carácter (*).

Instrucciones desde Mega:

- **IZUM:** Comando que indica a Raspberry que cargue el programa calibración en Zowi.
- **ROFC:** Comando leer offset de las articulaciones de Zowi.
- **WOFC:** Ordena a Zowi escribir los valores de offset en su EEPROM. Se envía con la instrucción un número (1 o 2) para indicar pierna derecha/izquierda, además del separador (*) y las posiciones en grados, 3 dígitos cada una, separadas por el asterisco (*).
- **M90C:** Ordena a Zowi mover todos los servos a la posición $90^o$, utilizando la librería servo.
- **MHOC:** Ordena a Zowi mover todos los servos a la posición $90^o$ con la corrección guardada en la EEPROM, si la hay.
- **MHOC:** Ordena a Zowi mover todos los servos a la posición indicada, necesita como parámetro un número de tres cifras que indique la posición en grados.
- **MSxC:** Comando a Zowi para mover el servo indicado a la posición indicada. Necesita un número del 1 al 4 que indica el servo a mover, el separador (*), además de 3 dígitos que indiquen el grado.
- **WERC:** Comando que envía los errores medidos tras la calibración a Raspberry. Para cada servo se envía el error con 3 dígitos enteros más 2 decimales, separados por punto. Los datos de cada servo están separados por (*).
- **FZUM:** Ordena a Raspberry que cargue el programa final en Zowi.
- **ROFF:** Comando a Raspberry para apagar el sistema (Shut down).
- **WSQL:** Ordena Raspberry que cierre comunicación con Zowi y registre una entrada de fin de calibración en la base de datos.

Si bien las instrucciones anteriores fueron útiles durante el desarrollo, se acabó utilizando, principalmente, el movimiento de un solo servo por ser más controlado, se observaron algunos funcionamientos no deseados al tener en movimiento los 4 servos simultáneamente, además de la batería en carga.

Como entrada, Mega recibirá 'M' o 'B' desde Raspberry en los estados que requieren feedback para determinar el siguiente paso en la máquina de estados, indicando estos 'Mal' y 'Bien', respectivamente. Las funciones de lectura como ROFC tambien resultaron útiles durante el desarrollo, sin embargo no se utilizan en la versión final.

#### 2.4.2.1.2 Configuración

Mega permite hasta 3 canales adicionales de comunicación serie, Serial1 utiliza los pines 19 como RX y 18 como TX para una comunicación serie TTL a 9600 baudios. Gracias a un cable TTL-USB, es posible debugear y desarrollar fácilmente, aún teniendo utilizado el puerto serie habitual para comunicar con Raspberry.

Programación para los componentes acústicos y luminosos utilizando las librerías stándar de Arduino, que permiten emitir un sonido o establecer el color de una luz con una simple línea de código, también se recurre al uso de las resistencias de pull-up integradas, activadas en la declaración del tipo de pin, para reducir la cantidad de electrónica necesaria para los finales de carrera y pulsadores.

#### 2.4.2.1.3 Algoritmo básico de calibración

Este algoritmo se ejecuta para la correcta calibración de cada uno de los juguetes.

Inicialmente se mandan los servos del robot Zowi a las posiciones correspondientes a 90º, se recuerda que el rango del servo es 0-180º, y que el juguete necesita un rango de solamente unos 80º.

La calibración consiste en tomar la lectura de los sensores IMU, de cada sensor interesan 2 orientaciones, una de ellas directamente relacionada con la posición de la cadera del robot, y la otra con el pié. Leyendo ambos sensores se obtienen las 4 posiciones de los servos.

Se calcula el desfase entre la posición leída y la posición actual según el programa de Zowi (inicialmente 90º), y se calcula la nueva posición necesaria a establecer en el servo para obtener los 90º deseados.

El proceso de lectura-corrección se lleva a cabo de forma iterativa hasta obtener una lectura con error inferior a 1º respecto a 90º.

#### 2.4.2.1.4 Calibración de los sensores

Esta calibración se ha de realizar cada vez que el sistema es iniciado o reiniciado.

Mediante las funciones de la librería de MinIMU, se establecen los ángulos cero para cada eje, para ello el operario es guiado a insertar los zapatos (que contienen los sensores) en los cajetines del banco de calibración, en éste paso se toman medidas hasta valorar que la calibración ha sido correcta. En esta etapa se corrigen pequeñas inclinaciones que pueda haber en el espacio de trabajo. Es la parte más crítica del proceso, porque la calibración se ve muy afectada por el movimiento, vibraciones o incluso campos magnéticos.

Durante el algoritmo de calibración se ha hablado del ángulo de 90º como ángulo deseado. Realmente, el valor éste ángulo para las caderas se corresponde con el 0º, mientras que para los piés se corresponde con el 90º, por lo que tambien es ajustado en esta etapa. Tras valorar que los ángulos 0º han sido bien definidos y su lectura

es estable, se ha de girar cada zapato 90º utilizando los cajetines verticales. En esta fase se corregirá el pequeño error que pueda tener el sensor en esos 90º (no suele ser mayor de 1º).

#### 2.4.2.1.5 Calibración del acelerómetro

Configuración necesaria cada vez que se reemplace el sensor. Ésta es la única configuración necesaria para replicar el sistema.

Esta calibración no forma parte del programa del sistema, sino de su configuración e instalación. En los 2 armarios que se han creado, se utilizan en total 4 sensores IMUs, los circuitos integrados que contiene no son exactamente iguales y, por recomendación del fabricante y creador de la librería de MinIMU, se han de ajustar los valores máximos y minímos crudos del magnetómetro (mismo encapsulado que el acelerómetro). Para ello se utiliza un programa del fabricante que muestra, por el puerto serie, el valor máximo y mínimo crudos registrados mientras el sensor es movido en todos los ángulos. Estas constantes son introducidas en el programa de cada armario para ser utilizadas por la librería del acelerómetro, LSM303.

### 2.4.2.2 Zowi

El programa definido para la controladora de Zowi es un intérprete de los comandos definidos. La Raspberry descargará dicho intérprete en el microcontrolador de Zowi para poder comunicarle las posiciones a las que ha de mover los servos, y los valores de calibración que tendrá que guardar en su memoria EEPROM, memoria que no es sobrescrita en el proceso de programación habitual de la placa controladora.

Adicionalmente, la controladora de Zowi cuenta con un módulo de EEPROM por I2C, donde se recogen datos de fabricación de la placa. De esta forma se tienen 2 módulos de EEPROM, dejando el módulo del microcontrolador disponible para ser modificado por los usuarios de la placa.

Es en el módulo habitual donde se registran los valores de calibración del servo, pudiendo ser configurados por los usuarios en caso de que desmonten el juguete o reemplacen algún componente. Por otro lado, desde el módulo EEPROM por I2C, se obtendrá el número de serie, que almacenará la Raspberry en la base de datos con fines de trazabilidad.

Las librerías que se utilizan serán I2C_eeprom.h y las estándar Servo.h para el movimiento de los servos y EEPROM.h para guardar los valores de calibración.

#### 2.4.2.2.1 Protocolo visto desde Zowi

Las siguientes instrucciones son recibidas por Zowi en la última versión del intérprete:

- **MSxC:** Al recibir este comando, Zowi mueve el servo indicado a la posición indicada. El primer parámetro es un número del 1 al 4 que indica el servo a mover, seguido del separador (*) y 3 dígitos que indican el grado.
- **WOFC:** Zowi calcula el valor de offset a partir de las posiciones recibidas y los escribe en su EEPROM. Las instrucciones contienen un número (1 o 2) para indicar pierna derecha/izquierda, además del separador (*) y las posiciones en grados, 3 dígitos cada una, separadas por el asterisco (*).
- **M90C:** Zowi mueve todos los servos a la posición 90º por defecto.
- **MHOC:** Zowi lee el valor del offset de las articulaciones en su EEPROM y mueve todos los servos a la posición 90º con la corrección.

Se implementa una instrucción de salida, utilizando el mismo protocolo que se ha visto en Mega (instrucción delimitada por "$" y "#"), Zowi lee el número de serie de su controladora (6 dígitos) y se lo envía a la Raspberry con la instrucción **OKNS:**.

#### 2.4.2.3 Raspberry

La Raspberry resulta ser un componente tan importante como la Mega dentro del sistema. Inicialmente fue incluída para hacer de nexo en la comunicación entre ambos controladores (Mega y ZUM de Zowi), pero inmediatamente se implementaron algunas funcionalidades adicionales.

Este ordenador tiene instalado un sistema operativo Raspbian, versión adaptada de Debian a Raspberry, concretamente se usa la última versión estable en el momento de desarrollo, Wheezie.

El lenguaje elegido para hacer el software es python, por lo rápido que resulta desarrollar con éste lenguaje y por ser el más familiar para el autor. Se instalan por tanto los paquetes necesarios (el entorno python-dev, el gestor de paquetes PIP y el gestor de entornos virtuales, virtualenv). Las librerías empleadas son serial para las comunicaciones serie, os para poder ejecutar comandos del sistema operativo y pymysql para poder escribir en bases de datos MySQL.

Para que el controlador de Zowi interprete las instrucciones que recibirá, se ha de descargar el intérprete creado. Esto requiere que Raspberry programe el micro ATMega328p de Zowi, para ello se utiliza la herramienta AVRdude como instrucción del sistema operativo, directamente desde el programa corriendo en python. El programa en Arduino del intérprete está convertido en un fichero hexadecimal ya compilado con el entorno de Arduino, AVRdude escribirá este programa en el microcontrolador de Zowi.

Al arrancar el sistema operativo y la script, se inicia comunicación con Mega usando el puerto ttyUSB0 y envía un “1” para inicializar el programa. La Raspberry permanecerá a la escucha, los dispositivos conectados (Mega -y Zowis tras la descarga del intérprete-) envían instrucciones utilizando el protocolo implementado ya comentado en los apartados anteriores, cualquier instrucción a Raspberry irá contenida entre “$ … #”. Las instrucciones de Mega que han llegar a Zowi, se envían ya sin los delimitadores ($ y #), es decir, solamente el contenido de la instrucción.

Además de delimitar fácilmente qué datos son instrucciones, nos permite poder hacer eco del resto de datos recibidos por serie, para debug o información de los programas de las Arduino en Raspberry.

#### 2.4.2.3.1 Máquina de estados en Raspberry

Se muestra la respuesta de la script de python a las diferentes instrucciones recibidas:

**IZUM:** Cuando se recibe este código, Raspberry programará la ZUM de Zowi con el software de calibración (el intérprete), ubicado en el siguiente directorio:

/home/pi/zowi/python/zowi_offset_i2c.cpp.hex

El programa es subido usando la instrucción de AVRdude:

avrdude -patmega328p -carduino -P/dev/ttyUSB1 -b 115200 -D
-Uflash:w:/home/pi/zowi/python/zowi_offset_i2c.cpp.hex:i

Como se ha dicho anteriormente, éste programa es necesario para interpretar los códigos e instrucciones enviados desde Mega. Se envía un código de confirmación con el texto “M” o “B” indicando a Mega indicando el resultado del proceso.

**FZUM:** Cuando se recibe este código, se programa ZUM de Zowi con el programa final del juguete. El fichero está está en el siguiente directorio:

/home/pi/zowi/python/ZOWI_BASE_v0.cpp.hex

Se envía un código de confirmación con el texto “M” o “B” indicando a Mega indicando el resultado del proceso.

**ROFF:** Indica que el sistema se debe apagar, cuando se recibe este comando desde Mega, se apaga el sistema operativo.

**WERC:** Este código indica los errores de calibración de cada articulación, es decir, cuántos grados de diferencia se han conseguido entre los valores teóricos y los medidos tras finalizar una calibración, sea exitosa o fallida. Se almacenan dichos valores para ser guardados, posteriormente, en la base de datos.

**MSxC:** El resto de instrucciones de movimiento de servos ya comentadas en los apartados de software de Mega y Zowi, se envían directamente de Mega a Zowi. Para el caso de movimiento de un servo específico, **MSxC**, se almacenan los últimos valores enviados, para poder ser volcados a la base de datos cuando se reciba la instrucción adecuada.

**WOFC:** Este código es enviado indicando las nuevas posiciones de los servos de Zowi, las posiciones que hacen que sus articulaciones estén alineadas y calibradas. Raspberry envía estos valores a Zowi para que éste calcule los desfases y los escriba en su memoria EEPROM.

**OKNS:** Es la única instrucción que proviene de Zowi, e indica el número de serie de la placa controladora del ejemplar, Raspberry almacena dicho número para su escritura en base de datos.

#### 2.4.2.3.2 Base de datos

Se instala en el sistema una base de datos MariaDB (MySQL) local, contra la que se registran los eventos que suceden durante el uso del sistema. Además, la script del programa principal de python está también preparada para escribir a una base de datos remota. Se escribe en la base de datos tras cada calibración, sea buena o mala. Para lo que se utiliza un campo de "Estado" que valdrá 1 o 0, respectivamente. Se registran los errores de calibración leídos en Arduino Mega para cada uno de los 4 servos. Se registran además las 4 posiciones "trim" de los servos, las mismas que se hacen llegar a la placa de Zowi para que guarde en su EEPROM para salir a venta. Se registra la hora, con fines estadísticos de tiempos y tambien el número de serie del Zowi conectado con fin de trazabilidad en caso de errores/reclamaciones.

Cuando se inicia o apaga el sistema, se escriben todos los campos a 1 salvo la hora para el caso de inicio. Y todos los valores a 0 salvo la hora para el fin.

Durante el uso de los primeros días en producción surgieron algunos problemas, para los que se definieron unos códigos de error y se implementaron ciertas correcciones de forma remota, de la misma forma que para inicio y final se usan valores a 1 y a 0, para estos errores utilizamos:

- **Valores a 2:** Excepción del cable, producido por pérdida de comunicación con Mega.
- **Valores a 3:** Otros errores no conocidos, por excepción en el programa principal.
- **Valores a 4:** Error conexión fallida al intentar descargar intérprete a Zowi.
- **Valores a 5:** Error conexión fallida al intentar descargar programa final a Zowi.

Para ver la estructura de la tabla, se puede consultar el Anexo **??**.

#### 2.4.2.3.3 Configuración adicional

El servicio de escritorio remoto XRDP es instalado, lo que nos permite utilizar la interfaz gráfica de forma remota.

El entorno de Arduino es también instalado, permitiendo modificar el programa de Mega desde Raspberry.

En el fichero de configuración de Raspberry **/boot/config.txt** se configura *max_usb_current=1* lo que nos permite suministrar hasta to 1.2A por USB para alimentar a Zowi (por defecto, límitado a 600mA).

Se configura el Login automático mediante el fichero **/etc/inittab**. De este modo el sistema funciona como servidor en lugar de esperar que un usuario introduzca sus credenciales. Ref/Anexo.

Se crea una una script, **running.sh** que funciona como monitor, hará que la script principal sea arrancada siempre que no esté corriendo. Esta medida rearma el sistema ante algunos errores, como por ejemplo, intentar programar una placa que deja de estar energizada o es pagada o desconectada durante la programación. Anexo xx.

Se configura el arranque automático de la script anterior al encender el sistema, dicha script levantará el programa principal nada más ser iniciado el sistema. Configuración en el ficher **ZowiInit** que se puede consultar en el Anexo xx.

Tras observar el funcionamiento del sistema en producción durante algunos días, se detectan algunos errores al conectar y desconectar repetidas veces los robots. El sistema podría no funcionar si se asigna un puerto diferente a Mega, o si se inicia el sistema con un Zowi ya conectado. Para mejorar ésto, se crean unas reglas (fichero **50-usbportsbq.rules**) para definir el puerto según el ID del fabricante de la placa, asignando a cualquier puerto USB conectado a Raspberry el alias de "mega" o "zowi" en lugar de ttyUSBX, nombre por defecto) . Dichas reglas son instaladas dentro de **/etc/udev/rules.d/**. El programa de python es modificado para asumir estos cambios. El fichero de las reglas se puede ver en el Anexo XX. Además, se implementa una nueva función en el código que desconecta y conecta eléctricamente la bahía del puerto USB de Zowi por software (utilizando **bind** y **unbind**) antes de cada programación. Los errores y reinicios se reducen inmediatamente.