

UNIVERSIDAD DE HUELVA  
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA  
PROYECTO FIN DE CARRERA

---

## Banco de Calibración de Zowi

---

*Autor:*  
Marcial RODRÍGUEZ

*Coordinador:*  
Juan Manuel ENRIQUE

*Proyecto Fin de Carrera  
para la titulación de Ingeniería Industrial*

*realizado en*

Grupo Robótica Industrial y Automatización  
Extinto Dpto. Innovación y Robótica  
BQ

March 16, 2017



# Contents

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Objetivo . . . . .	1
1.1.1	Requisitos . . . . .	1
1.2	Estructura del proyecto . . . . .	1
<b>2</b>	<b>Marco Teórico</b>	<b>3</b>
2.1	Contexto . . . . .	3
2.1.1	Zowi . . . . .	3
2.2	Motivo del proyecto . . . . .	4
2.2.1	Servos . . . . .	4
2.2.2	Problema a resolver . . . . .	6
2.2.2.1	Calibración . . . . .	7
2.3	Possibles alternativas . . . . .	8
2.3.1	Sensor IMU . . . . .	8
2.3.1.1	Acelerómetro . . . . .	8
2.3.1.2	Giróscopo . . . . .	9
2.4	Herramientas y tecnologías utilizadas . . . . .	9
2.4.1	Máquinas . . . . .	9
2.4.2	Diseño . . . . .	10
2.4.3	Programación . . . . .	10
2.4.4	Post-desarrollo . . . . .	10
<b>3</b>	<b>Desarrollo</b>	<b>13</b>
3.1	Introducción . . . . .	13

3.2	Prototipo de validación de concepto	13
3.2.1	Cámara visión por computador	13
3.2.2	Sensores IMUs	14
3.2.2.1	Teoría	14
3.2.2.2	Prueba inicial	17
3.2.3	Conclusiones	18
3.3	Prototipo de laboratorio	18
3.3.1	Línea de evolución	19
3.3.1.1	Primeras mejoras	19
3.3.1.2	Tomando forma	21
3.3.1.3	Prototipo final: Banco de Calibración	24
3.3.2	Pruebas realizadas	25
3.4	Armarios finales	26
3.4.1	Componentes e implementación física	28
3.4.1.1	Arquitectura y conexiones	28
3.4.1.2	Armario eléctrico	29
3.4.1.3	Mega	29
3.4.1.4	Shield Mega	31
3.4.1.5	Raspberry PI 2	32
3.4.1.6	Power Boost	32
3.4.1.7	LCD Display	33
3.4.1.8	Otros componentes	34
3.4.1.9	Zowi - Zum	34
3.4.1.10	Banco soporte Final de Zowi y zapatos	34
3.4.2	Software	35
3.4.2.1	Mega	35
3.4.2.2	Zowi	38
3.4.2.3	Raspberry	39

<b>4 Desarrollo</b>	<b>43</b>
4.1 Conclusiones y trabajos futuros . . . . .	43
4.1.1 Conclusiones . . . . .	43
<b>A Manual de usuario</b>	<b>45</b>
<b>B Planos Eléctricos: Cabinet 1</b>	<b>51</b>
<b>C Planos Eléctricos: Cabinet 2</b>	<b>59</b>
<b>D Mega PINOUT</b>	<b>67</b>
<b>E Mega Custom Shield</b>	<b>69</b>
<b>F Máquina de estados Mega</b>	<b>79</b>
<b>G Configuración IMUS</b>	<b>85</b>
<b>H Estructura base de datos</b>	<b>87</b>
<b>I BOM de los armarios finales</b>	<b>89</b>



# List of Figures

2.1	Robot Zowi . . . . .	4
2.2	Servo Futaba S3003 . . . . .	5
2.3	Diagrama de bloques servomotor . . . . .	5
2.4	Posición según PWM . . . . .	6
2.5	Eje dentado servo . . . . .	7
2.6	Lectura acelerómetro . . . . .	9
2.7	Lectura giróscopo . . . . .	9
2.8	Hephestos . . . . .	10
2.9	Witbox . . . . .	10
2.10	Cyclone . . . . .	11
3.1	BQ Zum Bluetooth empleada para mover los servos . . . . .	14
3.2	Celda con cámara de Fanuc . . . . .	15
3.3	Calibración tobillo con visión . . . . .	15
3.4	Calibración cadera con visión . . . . .	15
3.5	MinIMU-9 v3 de Pololu . . . . .	16
3.6	Posiciones calibración IMU . . . . .	17
3.7	Tobillo gira sobre eje Z . . . . .	17
3.8	Cadera gira sobre eje Z . . . . .	17
3.9	Sensor anclado con grapa impresa . . . . .	19
3.10	Bancos impresos para Zowi y zapato . . . . .	20
3.11	MinIMU-9 v3 Pin-out . . . . .	20
3.12	Banco 1 zapato . . . . .	20

3.13 Banco con inclinación regulable . . . . .	21
3.14 Banco con conexión MEGA-ZUM . . . . .	22
3.15 Arquitectura versión MEGA-ZUM por I <sup>2</sup> C . . . . .	23
3.16 Prototipo final Zowi SLS . . . . .	23
3.17 Banco prototipo final . . . . .	25
3.18 Prueba pre MP . . . . .	26
3.19 Segundos por cada calibración de Zowi . . . . .	27
3.20 OK:1   NOK:0 - para cada calibración de Zowi . . . . .	27
3.21 Armarios finales preparados para su envío a fábrica . . . . .	28
3.22 Esquema de conexiones . . . . .	29
3.23 Armario final: Lateral . . . . .	30
3.24 Armario final: Interior . . . . .	30
3.25 Freaduino Mega2560 . . . . .	31
3.26 PCB Shield . . . . .	31
3.27 Shield soldada . . . . .	31
3.28 Raspberry PI 2 - 1GB Ram . . . . .	32
3.29 Power Boost . . . . .	33
3.30 Backpack para Display LCD . . . . .	33
3.31 Banco soporte y zapatos finales . . . . .	34

# List of Tables



## Chapter 1

# Introducción

## 1.1 Objetivo

Diseño y construcción de un sistema que forma parte de la cadena de montaje de un juguete comercial: el robot educativo Zowi, desarrollado por BQ. La principal función de éste sistema es la de ajustar, de forma automática, las posiciones de los servomotores del juguete. Adicionalmente, se implementan otras funcionalidades importantes tal como la descarga del software final en el controlador del robot.

### 1.1.1 Requisitos

Algunos de los requisitos marcaron el camino a seguir hasta la solución final, facilitando la toma de decisiones en diferentes puntos del proyecto. Las peticiones más relevantes para el diseño fueron las siguientes:

- Plazo de finalización fijado en 2 meses.
- Replicable fácilmente; a poder ser, por terceros.
- Utilizable por personal con poca o ninguna formación técnica.
- Cadencia aproximada de la línea de producción: 30-60 segundos.
- Deseable: fácil instalación.

## 1.2 Estructura del proyecto

La información se presenta de la siguiente forma:

- En éste Capítulo 1 se presenta una breve introducción del proyecto.
- En el Capítulo 2 se describe el motivo del proyecto, una pequeña descripción de las posibles soluciones y una base teórica sobre los principios y componentes más importantes del sistema, así como una pequeña mención a las tecnologías y herramientas que han sido útiles o necesarias para su desarrollo.

- El Capítulo 3 se centra en la línea de desarrollo, mostrando las diferentes etapas y prototipos por los que se ha pasado hasta llegar a la versión final, con una descripción de las funciones de los componentes electrónicos dentro del sistema y del software creado o utilizado.
- En el Capítulo 4 se sintetizan los resultados.
- En los anexos se recoge gran cantidad de la información del proyecto, conteniendo código, planos, esquemáticos o tablas de gran tamaño, entre otros. Será frecuente el uso de referencias a los anexos durante todo el documento.

## Chapter 2

# Marco Teórico

### 2.1 Contexto

Una de las líneas de desarrollo de BQ es la robótica educativa, un conjunto de productos y servicios orientados a educar en las tres partes de todo proyecto tecnológico: Diseño, hardware y software. Cuenta para ello con diferentes productos (impresoras -Witbox, hephestos-, "mi primer kit de robótica", o los printbots), programas (Bitbloq -basado en scratch-), así como contenido y soporte en la web ("Do it with others" -[www.diwo.bq.com](http://www.diwo.bq.com)-).

El presente proyecto surge de la necesidad de automatizar una parte del proceso de fabricación de un nuevo robot educativo, Zowi. Hasta ahora, los robots desarrollados (printbots, por su carácter de chasis y componentes no-electrónicos totalmente imprimibles en 3d) usaban para moverse servos de rotación continua y ruedas, Zowi es el primer robot bípedo desarrollado por BQ (utiliza dos servos de posición para los piés y otros dos para las caderas) y la librería de osciladores sinusoidales -Oscillator-, desarrollada por Juan "Obijuan" González, entonces jefe del departamento, para robots serpiente modulares.

#### 2.1.1 Zowi

Zowi, también llamado el robot de Clan TVE, es un robot educativo para niños que les inicia en el mundo de la tecnología y de la programación. Se puede ver el juguete en la Figura 2.1.

Este robot, movido por una controladora basada en Arduino y diseñada a medida para él, además de los servos que lo convierten en un robot bípedo, incorpora unos sensores de ultrasonidos como ojos que le permiten detectar obstáculos, una matriz de leds como boca para expresar "emociones" y tres pequeños botones para alternar entre sus modos de funcionamiento, entre otros componentes.

Se ha creado una aplicación móvil, Zowi App, que permite controlar los movimientos de Zowi y reprogramarlo fácilmente con diferentes juegos. Zowi tiene muchas posibilidades más allá de lo predefinido en su aplicación. BQ tiene una plataforma online para programar a sus robots y controladoras, esta plataforma se llama Bitbloq. En Bitbloq se puede programar a Zowi y a otros tipos de placas Arduino de forma muy sencilla por medio de bloques, sin necesidad de saber ningún

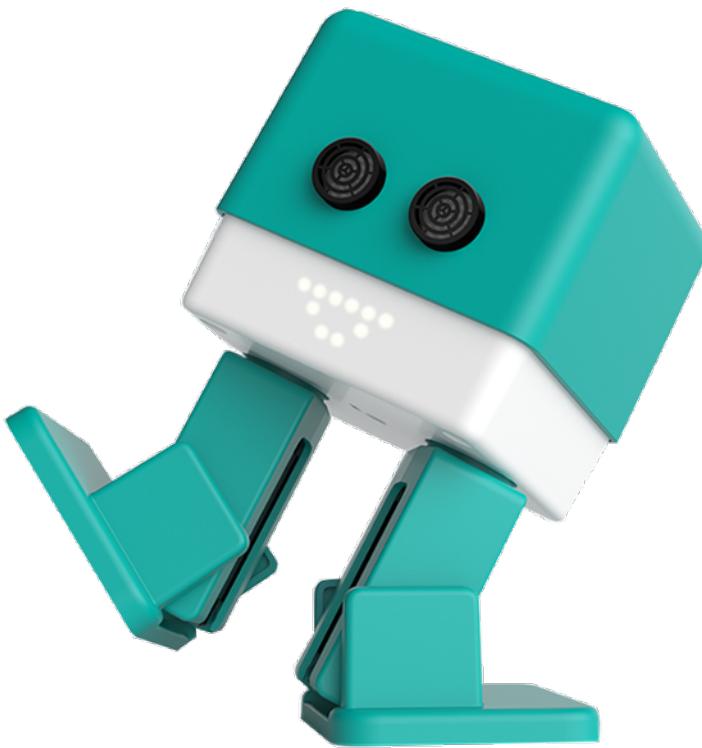


FIGURE 2.1: Robot Zowi

lenguaje de porgramación. La opción de emplear código directamente tambien está disponible.

## 2.2 Motivo del proyecto

La tarea asignada al autor del proyecto fue la de idear y desarrollar el prototipo de **un sistema capaz de ajustar de forma automática cada uno de los cuatro servos que componen las piernas de Zowi**, lo que permite al robot ejecutar sus programas de fábrica, con los que saldrá a la venta, de forma correcta; entre ellos: realizar diferentes bailes, caminar, etc. Dicho sistema debe ser utilizado por operaradores sin conocimientos de Arduino ni de electrónica en general. La nacionalidad de los operarios tambien es desconocida por no saberse el emplazamiento de la fábrica hasta más avanzado el proyecto (razón por la que parte de la documentación anexada se encuentra en inglés: menú de la interfaz, comentarios del código, etc).

### 2.2.1 Servos

Los servos son un tipo especial de motor DC que se caracterizan por su capacidad para posicionarse de forma inmediata en cualquier posición dentro de su intervalo de operación. Para ello, el servomotor espera un tren de pulsos que se corresponde con el movimiento a realizar. Están generalmente formados por el motor, un sistema

reductor formado por ruedas dentadas, un potenciómetro y un circuito de realimentación, todo en una misma caja de pequeñas dimensiones. En la Figura 2.2, se muestra un servomotor del tipo empleado en Zowi.



FIGURE 2.2: Servo Futaba S3003

El circuito electrónico de realimentación del servo es el encargado de recibir la señal PWM y traducirla en movimiento del Motor DC. El eje del motor DC está acoplado a un potenciómetro, el cual permite formar un divisor de voltaje. El voltaje en la salida del divisor varía en función de la posición del eje del motor DC. En la Figura 2.3 se muestra un diagrama del funcionamiento de un servo de posición.

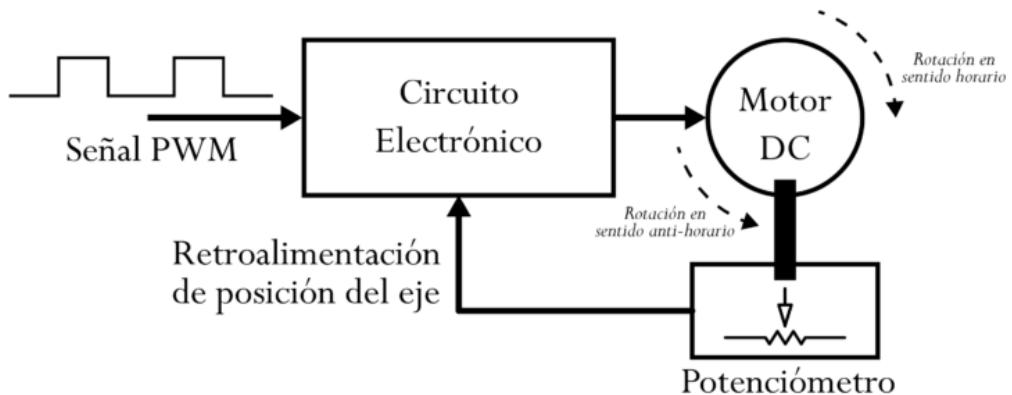


FIGURE 2.3: Diagrama de bloques servomotor

La modulación por anchura de pulso, PWM (Pulse Width Modulation), es una de los sistemas más empleados para el control de servos. Este sistema consiste en generar una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, manteniendo el mismo período, con el objetivo de modificar la posición del servo según se desee. Arduino permite generar PWM en algunos de sus pines de forma sencilla, además, para hacerlo aún más fácil, tiene la librería `servo.h`, que nos

permite dar un valor de posición angular (grados) a un determinado pin, y él sólo genera el PWM.

Las señales de PWM requeridas por el circuito de control electrónico son similares para la mayoría de los modelos de servo. Esta señal tiene la forma de una onda cuadrada. Dependiendo del ancho del pulso, el motor adoptará una posición fija. Un ejemplo visual en la Figura 2.4.

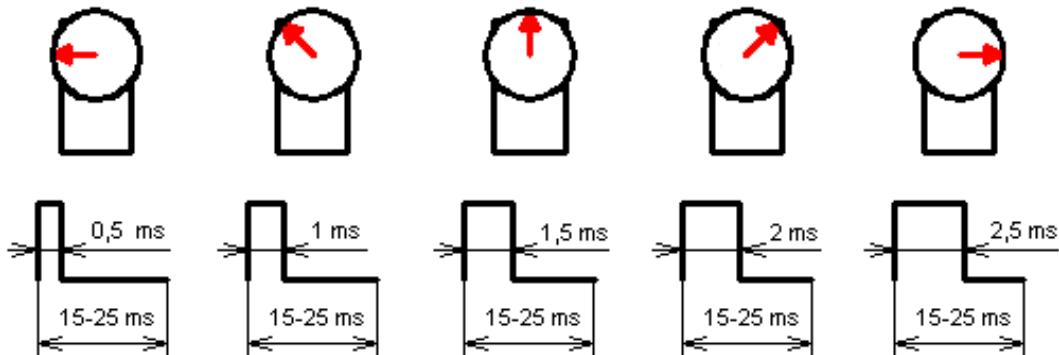


FIGURE 2.4: Posición según PWM

## 2.2.2 Problema a resolver

Los servos empleados son una versión clónica del Futaba S3003 que vimos en la Figura 2.2, con un rango de funcionamiento algo mayor (en torno a  $193^\circ$ ). Para el caso de Zowi, el rango empleado será bastante menor, por cuestiones de estabilidad del juguete y por el diseño de las piezas, mecánicamente sería imposible torcer las articulaciones en todo el rango del servomotor.

La continuación del eje del servo, eje dentado, engrana en la pieza de Zowi. Como podemos ver en la Figura 2.5 a modo ilustrativo, hay un determinado número de dientes en dicho engranaje; para el servo empleado en Zowi, hay 24 dientes en el engranaje, por lo que un diente supone un cambio de  $15^\circ$  respecto al anterior.

El ensamblaje de los servos en sus piezas correspondientes se hace de forma manual, se hace que el servo llegue a hacer tope en uno de los sentidos como referencia y es engranado de forma que tenga disponible el rango de funcionamiento necesario. Sin embargo, dentro de los  $15^\circ$  que se comentaban anteriormente, y por la misma fabricación de las partes del servo, las piezas podrían quedar montadas con un desfase de hasta 7.5 grados, o mayor aún si el operador no elige bien la orientación al engranar. Adicionalmente, por el fabricante del servo, la relación entre tren de pulsos PWM y la posición final del servo puede variar -estar un poco desplazada-, incluso variar muy ligeramente entre servos del mismo fabricante.

Se tienen entonces varios factores que hacen que cada servo sea montado en una posición distinta -o considerada distinta por el programa- y tenga que ser configurado por software, además, el ajuste se adaptará para cada servo, no se puede definir una regla válida para todos, de ahí que se emplee un método iterativo para solucionarlo.



FIGURE 2.5: Eje dentado servo

Lo ideal sería que la posición del servo intermedia coincidiese con la posición neutra de la articulación de Zowi, dejando la mitad del rango de giro del servomotor para cada sentido. Sin embargo, como el rango de movimiento del juguete es menor que el rango completo del servo, se tiene un margen bastante generoso que se puede corregir en el ajuste.

#### 2.2.2.1 Calibración

La calibración de los servos es de importancia vital para un buen funcionamiento de las funciones programadas en el juguete y en cualquier aplicación que utilice servos, en general. En Arduino, utilizando la librería `servo.h`, se consiguen generar los pulsos PWM de forma transparente, empleando directamente grados sexagesimales (números enteros). El primer paso para utilizar un servo es siempre definir qué posición es la de partida del servo, es decir, fijarle un  $0^\circ$ . Ésto se realiza guardando un valor ("trim" u "offset") que podrá ser positivo o negativo y se empleará para corregir las orientaciones.

Cuando el usuario quiere mover el servo a una posición determinada utilizando la librería `servo.h` directamente, tendrá que moverlo a la posición deseada más el valor de "offset". El offset obviamente siempre será el mismo para cada servo, salvo que se desmonte y monte el servo en otra posición o en otra pieza y se tenga que recalibrar. Este ajuste es habitual cuando se trabaja con servos, en robótica o radiocontrol por ejemplo.

Sabemos que Zowi es un producto pensado para ser reprogramado y "trastornado", para que el usuario aprenda sobre electrónica y programación; sin embargo, éste usuario no tiene por qué tener ningún conocimiento previo en el momento de adquirir el robot, es por ésto que Zowi sale a la venta con programas por defecto, programas que muestran sus posibilidades. Zowi tendrá los valores de "offset" de los servos, calculados para su montaje en fábrica, guardados en determinadas direcciones de la EEPROM, zona de la memoria de la placa controladora que no se borra al ser reprogramada por el usuario.

## 2.3 Posibles alternativas

Anteriormente, se ha explicado cuál es el problema a resolver. Para darle solución, la clave es poder leer las orientaciones reales de los servos, es decir, las orientaciones respecto al mismo juguete; en base a estas posiciones se debe actuar sobre los servos hasta colocarlos en la posición neutra y guardar en la memoria el desfase observado para cada servo. Se plantean dos posibles caminos:

- **Visión por computador:** Emplear un sistema de visión para, mediante matching y detección de líneas, observar el desvío y poder actuar sobre los servomotores hasta llevarlos a la posición deseada.
- **Sensor IMU:** Utilizar utilaje impreso y sensores de tipo inercial (acelerómetro-giróscopo) para calcular las inclinaciones producidas por la posición de los servos.

Se elige la segunda opción por tener una estimación de tiempo de desarrollo más ajustada, además de un coste considerablemente menor.

### 2.3.1 Sensor IMU

Se describe brevemente la utilidad del sensor IMU ya que todo el sistema se construye en torno a este componente.

Una IMU, o unidad de medición inercial, es un sensor diseñado para combinar las características de un acelerómetro y un giróscopo, y mostrar información completa sobre aceleración, posición, orientación, velocidad...

Las medidas crudas de los componentes del acelerómetro, giróscopo y magnetómetro (encapsulado junto al acelerómetro) estarán directamente relacionadas con la fuerza G, velocidad angular y campo magnético, respectivamente. Dichas medidas pueden ser procesadas y tratadas con un microcontrolador o microprocesador para obtener posiciones, orientaciones, etc.

#### 2.3.1.1 Acelerómetro

El acelerómetro mide la aceleración en las 3 dimensiones del espacio. La gravedad de la Tierra tiene una aceleración perpendicular al suelo. Así pues, la IMU también detecta la aceleración de la gravedad terrestre. Gracias a ésta se pueden usar las lecturas del acelerómetro para saber cuál es el ángulo de inclinación respecto al eje X o eje Y. Figura 2.6.

Dado que el ángulo se calcula a partir de la gravedad, no es posible calcular el ángulo Z empleando únicamente el acelerómetro. Para hacerlo se necesita otro componente: el magnetómetro, que es simplemente un tipo de brújula digital.

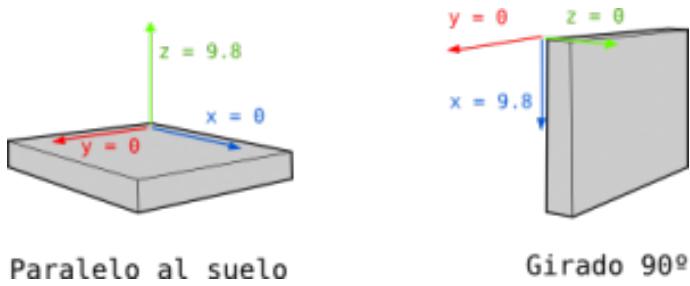


FIGURE 2.6: Lectura acelerómetro

### 2.3.1.2 Giróscopo

El giróscopo es un dispositivo que mide la velocidad angular en cada uno de los ejes. Figura 2.7.

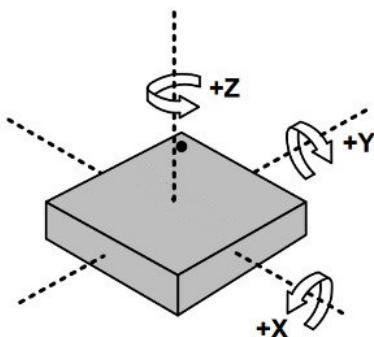


FIGURE 2.7: Lectura giróscopo

## 2.4 Herramientas y tecnologías utilizadas

Además de los componentes electrónicos que se han introducido anteriormente y los que se tratarán en el capítulo de desarrollo: Capítulo 3, cabe mencionar las tecnologías y herramientas que han resultado útiles durante todo el proyecto.

### 2.4.1 Máquinas

La tecnología de impresión 3D por FFF ha aportado agilidad al proyecto al permitir la creación de piezas en el mismo lugar en cuestión de horas. Se han empleado máquinas **Hephestos** (Figura 2.8) y **Witbox** (Figura 2.9) de BQ, ambas con firmware **Marlin**, para la creación del utilaje provisional y componentes menores como los soporte de las placas dentro del armario o la caja que contiene el display LCD y el zumbador.

Otra máquina interesante fue la fresadora **Cyclone** (Figura 2.10), entonces en fase beta. Se montó y empleó para el primer prototipo de la PCB de la shield desarrollada.



FIGURE 2.8: Hephestos

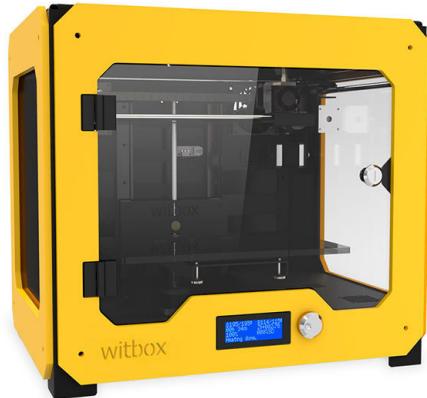


FIGURE 2.9: Witbox

#### 2.4.2 Diseño

Para el diseño de las piezas impresas se ha utilizado **Inventor** y **FreeCad**, el laminado se ha llevado a cabo con **Cura**.

El diseño de la PCB y la generación de los gerbers se logró mediante **KiCad**. Su fresado fue posible gracias a **FlatCAM** y **CNCCodeController**.

Los planos eléctricos se han creado con **DraftSight**.

#### 2.4.3 Programación

El desarrollo de los programas de Arduino y la programación de las controladoras se ha llevado a cabo empleando el mismo IDE de Arduino clásico: **Arduino 1.0.6**.

El sistema operativo instalado en Raspberry ha sido la versión adaptada de Debian: **Raspbian**, la versión concreta, última a la fecha: Raspbian Wheezy.

El programa principal de la Raspberry ha sido desarrollado en **Python 2.7.11**, última versión de Python 2 a la fecha. Ha sido útil el editor **Atom** y su capacidad de acceder a los directorios remotos directamente, mediante SFTP.

#### 2.4.4 Post-desarrollo

Para acceso remoto, troubleshooting y extracción de los datos se han utilizado las siguientes herramientas:

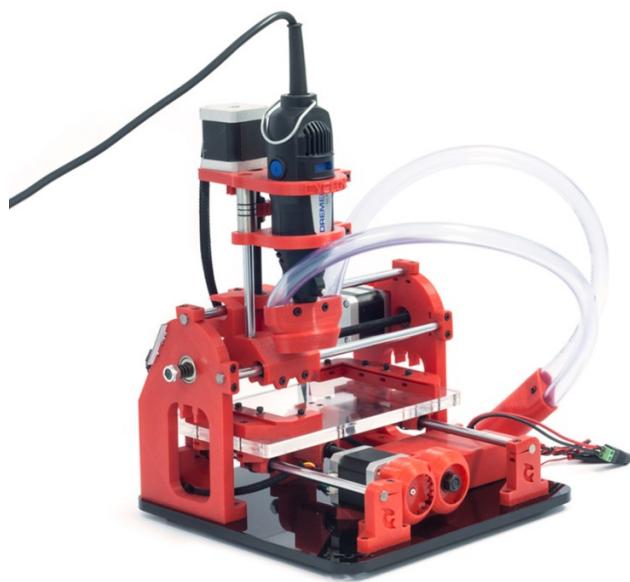


FIGURE 2.10: Cyclone

- **Putty** para las conexiones SSH.
- **Filezilla** y **WinSCP** para acceso remoto a ficheros.
- **HeidiSQL** para acceso a la base de datos.

La documentación del proyecto se ha realizado en **L<sup>A</sup>T<sub>E</sub>X** con **Atom** y **TeXworks**.



## Chapter 3

# Desarrollo

### 3.1 Introducción

En el presente capítulo se exponen las distintas vías propuestas para abordar la resolución del problema, con el desarrollo de unos prototipos para validación de concepto. Se muestra, también, la evolución del prototipo hasta la versión funcional de laboratorio, así como los detalles de la versión final desarrollada y empleada en la línea de producción.

### 3.2 Prototipo de validación de concepto

Para dar solución al problema planteado, inicialmente se realizan dos pruebas con tecnologías diferentes. Por un lado, y por petición de la empresa, se presentó una versión basada en visión por computador, tecnología utilizada por el autor del presente proyecto en proyectos anteriores. Por otra parte, se dió una solución alternativa basada en sensores giróscopo-acelerómetro.

Para ambas pruebas, se cuenta con un prototipo impreso de Zowi con una placa controladora genérica: BQ ZUM BT (ver Figura 3.1), versión propia de BQ de la conocida Arduino UNO. Tanto el juguete como su placa controladora se están desarrollando a la fecha de las pruebas.

#### 3.2.1 Cámara visión por computador

Se realiza una prueba piloto aprovechando el sistema de visión instalado para otro proyecto, compuesto por la cámara fija de Fanuc, 2 paneles LED de luz difusa (60W) y el software de visión Fanuc: iVision. Se elige realizar las pruebas en ésta plataforma, y no en un ordenador convencional con matlab u opencv, por inmediatez, el sistema está ya calibrado y en funcionamiento, además de que cuenta con un rack de salidas y entradas digitales; como se ha dicho, el objetivo era comprobar la viabilidad.

El sistema de visión está conectado al controlador de un brazo robótico LRMate 200iD, con un módulo I/O digitales a 24 voltios. Por lo que se realiza una interfaz

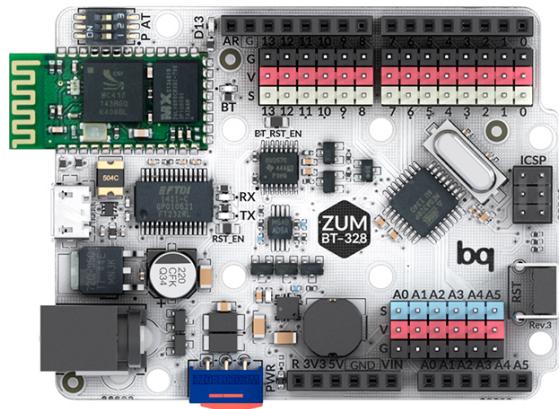


FIGURE 3.1: BQ Zum Bluetooth empleada para mover los servos

con 2 relés para comunicar con la placa Arduino (5V), encargada de calibrar los servos del juguete.

En la figura 3.2 podemos ver el área de trabajo de la celda robótica y su cámara.

El programa creado en Arduino recibe como entradas dos señales que indican en qué sentido se debe girar el servo. Como salida, actúan sobre el servo haciéndolo corregir su posición iterativamente 1 grado cada segundo hasta alcanzar la posición deseada.

La posición deseada es definida utilizando iRvision (Software de visión de Fanuc), mediante técnicas de visión por computador (Localización de elementos entrenados, medición de distancias y ángulos entre ellos). Para el caso del "tobillo", buscando perpendicularidad con la "cadera", y para ésta, consiguiendo cierta distancia entre dos líneas paralelas. Se pueden ver las Figuras 3.3 y 3.4.

### 3.2.2 Sensores IMUs

Para la segunda prueba se emplea una placa que combina un giróscopo, acelerómetro y magnetómetro para formar una unidad de medición inercial (de ahora en adelante IMU) con una precisión de hasta 0.1 grados.

#### 3.2.2.1 Teoría

La MinIMU-9 v3 (Figura 3.5) fue seleccionada de entre varias opciones, como la famosa MPU6050, la GY-87 o la versión v2 de MinIMU (todas de un precio similar), por la precisión y repetibilidad de las lecturas, además de parecer un dispositivo ampliamente utilizado y probado por la comunidad.

Tras estudiar el comportamiento del sensor, y una fase de documentación acerca de las opciones de que se dispone, surgen tres posibilidades:

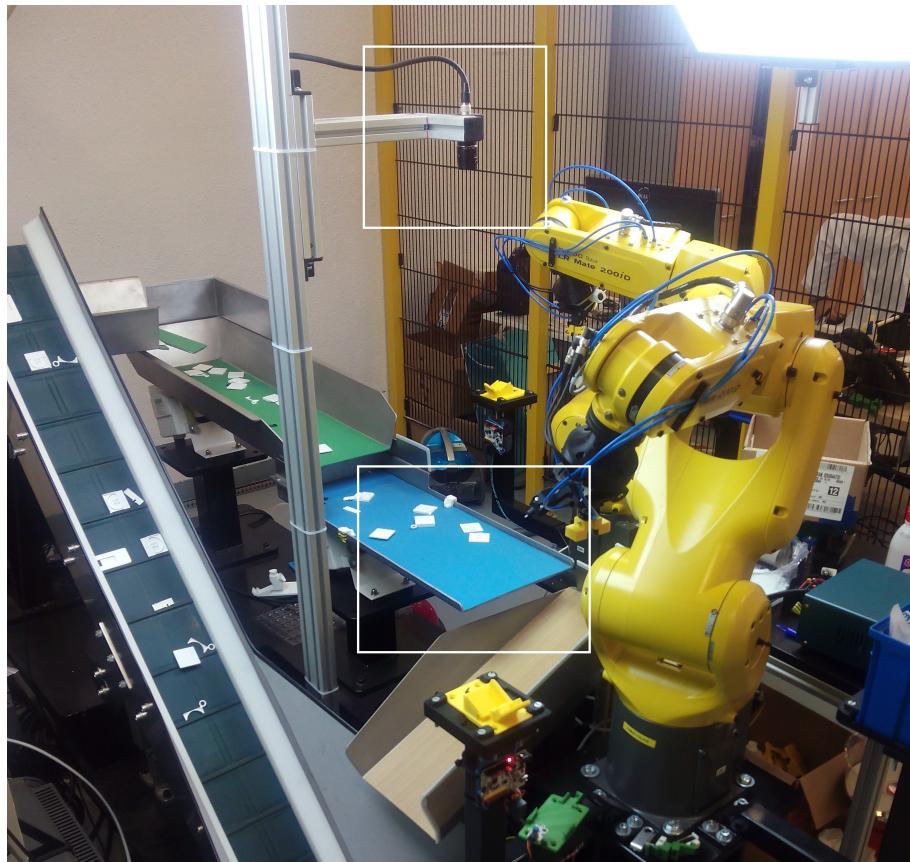


FIGURE 3.2: Celda con cámara de Fanuc



FIGURE 3.3: Calibra-  
ción tobillo con visión



FIGURE 3.4: Calibra-  
ción cadera con visión

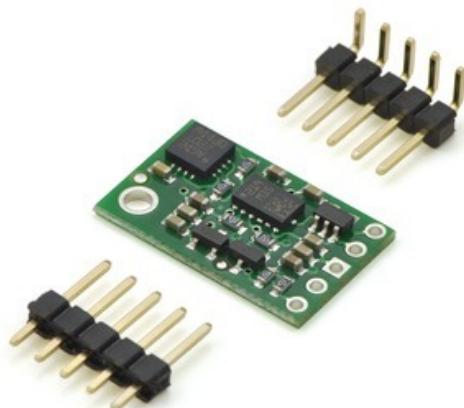


FIGURE 3.5: MinIMU-9 v3 de Pololu

- Librería de Pololu "MinIMU-9 AHRS"
- Librería "Open IMU"
- Desarrollar la matemática tomando los valores RAW del sensor

A pesar de lo atractivo de la tercera opción, no se disponía de suficiente tiempo para seguir adelante con ella, se trató de comprender el funcionamiento de las 2 librerías ya creadas.

Por simplicidad, Open IMU resultaba llamativa, pero rápidamente se percibió que podía quedarse demasiado corta y que las lecturas obtenidas empleando ésta librería (los ángulos euler de pitch, roll y yaw) parecían no tener la repetibilidad necesaria para nuestra aplicación, por otro lado, hacía más de 3 años que no se realizaban cambios en su repositorio GIT, por lo que posiblemente estaba preparada para funcionar con la versión anterior de la placa, MinIMU-9 v2, con una exactitud de hasta 1 grado solamente.

La línea elegida fue la de la librería recomendada por Pololu (fabricante del dispositivo), dejando un poco limitado el modo de funcionar pero ofreciendo un acceso rápido a las mediciones del sensor en forma de posición en ángulos euler.

Para un correcto funcionamiento del magnetómetro, se han de configurar los valores máximos y mínimos de las lecturas del acelerómetro, estos 2 sensores se encuentran en el encapsulado LSM303, la librería de dicho chip proporciona un programa de Arduino que captura dichas mediciones y salva la mayor mientras movemos la placa en todas las orientaciones posibles, mostrándola por el puerto serie. Los valores obtenidos se han de escribir, sustituyendo los "por defecto", en la cabecera de la misma librería.

La limitación de ésta librería que se había comentado antes, es que para obtener con precisión los ángulos euler en todas las orientaciones, se debe llamar una función de la librería **calibrateIMU()** con la placa colocada en una de las 2 posiciones que se ven en la Figura 3.6, lo más paralelo al suelo posible, con ésta función definimos en el espacio el 0 de los ejes X e Y del sistema, por lo que se corrigen desvíos si la posición no es totalmente horizontal, sin embargo, ésto deriva en errores que crecen

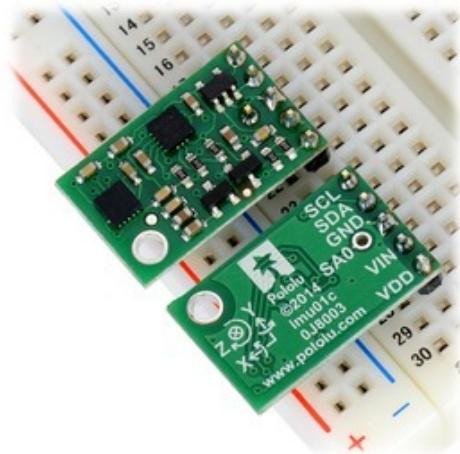


FIGURE 3.6: Posiciones calibración IMU

FIGURE 3.7: Tobillo  
gira sobre eje ZFIGURE 3.8: Cadera  
gira sobre eje Z

en proporción a la inclinación que se de en alguno de los ejes. Se elige una de ellas configurando ciertos parámetros de la librería de Pololu en el programa principal.

La orientación en Z depende de la brújula y su tiempo de estabilización es lento, por ello, se piensa en emplear los giros en los otros 2 ejes, X e Y, por lo que la posición del robot ha de ser la de acostado sobre un lado, descartando las posiciones que, de entrada, parecen más razonables como boca arriba/boca abajo, o con las piernas hacia arriba. Ver Figuras 3.8 y 3.7.

### 3.2.2.2 Prueba inicial

Se programa el algoritmo empleando una máquina de estados en la misma placa controladora de Zowi con un pulsador para pasar de un estado a otro. Versión muy básica para probar el concepto. Se emplea comunicación serie para conocer el estado del sensor y una grapa impresa para acoplarlo al pié del Zowi.

El procedimiento programado sería:

- Iniciar el sistema – Automáticamente se produce la calibración de la IMU
- Colocar el sensor en el pie de Zowi
- Pulsar el botón – Se inicia la calibración del servo

El modo de calibración del servo consiste en dar un primer valor a la posición que debería tener el servo, es decir,  $0^\circ$  para la cadera o  $90^\circ$  para el pie, entonces se lee el sensor para ver el error cometido y se ajusta la posición del servo teniendo en cuenta la diferencia del valor deseado y la nueva lectura, se repite hasta 2 veces.

Se obtienen buenas sensaciones con éste método y se nota de inmediato que el éxito del mismo radica en la correcta calibración inicial del sensor y en una buena fijación al pie del robot, por lo que se desarrolla un zapato para montar en él el sensor y un soporte para el robot, junto a otro soporte para realizar la calibración del sensor (ya integrado en el zapato) en la misma pieza. De esta forma se garantiza la correspondencia y paralelismo de los planos de apoyo de ambas partes (robot – zapato).

### 3.2.3 Conclusiones

El método de calibración empleando visión muestra buenos resultados y es muy mejorable si se emplea software creado específicamente para la aplicación, sin embargo, las condiciones de iluminación y posición han de ser las mismas o muy parecidas para cada banco de trabajo para garantizar un buen funcionamiento del software, por lo que su instalación, puesta en marcha y configuración, son delicados y se deberían hacer in-situ. En el momento de las pruebas de validación aún se desconoce la ubicación de la fábrica y la nave de montaje -posiblemente en China o Polonia-, la solución se encarece considerablemente si se han de realizar desplazamientos para llevar a cabo la instalación y puesta en marcha.

Con las IMUs se obtuvieron buenos resultados y se tenía bastante claro que se quería seguir adelante con ellas por su bajo coste y su fácil implementación. Quedaba afinar el algoritmo de calibración implementando una máquina de estados mucho más robusta y endureciendo el criterio de validación automática de una buena calibración para obtener mejores resultados. En el siguiente subcapítulo se desarrolla su evolución.

## 3.3 Prototipo de laboratorio

Se decide continuar con la solución de las IMUs y se producen reuniones cada pocos días/semanas entre los diferentes grupos implicados en el desarrollo del juguete Zowi (mecánica, hardware, producto, automatización...) dónde las especificaciones el banco de calibración van tomando forma en función de los resultados obtenidos. En la presente sección se mostrará la línea de evolución del prototipo, mencionando las características y mejoras más relevantes añadidas, así como las pruebas realizadas antes de pasar al desarrollo de la versión final.

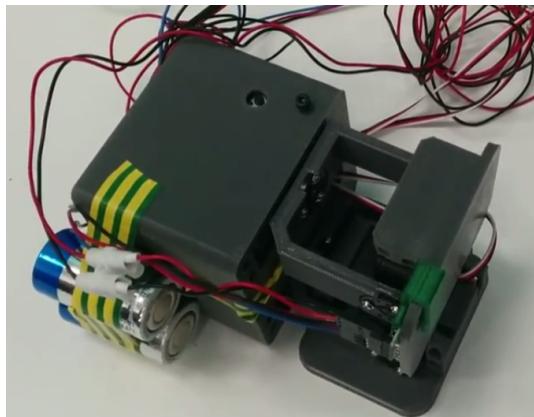


FIGURE 3.9: Sensor anclado con grapa impresa

### 3.3.1 Línea de evolución

Tras elegir la línea de los sensores IMU, se comienza a trabajar de inmediato en la mejora de la solución.

#### 3.3.1.1 Primeras mejoras

La máquina de estados es mejorada, diferenciando calibraciones malas de calibraciones buenas. Se añaden una indicador led y uno acústico para resaltar el resultado con diferentes sonidos. Esta mejora se mantendrá hasta la versión final del producto.

La siguiente necesidad a cubrir sería garantizar que el robot comparte el plano con el sensor, que ha de ser calibrado antes de colocarse en el robot como explicamos en la sección anterior. Hasta ahora el sensor era fijado utilizando una grapa impresa (Ver Figura 3.9). Se monta el sensor en una pieza especialmente diseñada e impresa para colocarla en el pié de Zowi, razón por la que es llamada "Zapato". Se crea también una base donde se coloca el Zowi, con un área para colocar el zapato en el momento de su inicialización, dónde el sensor es calibrado. Se puede ver el banco en la Figura 3.10.

El algoritmo hasta ahora pasaba por calibrar una pierna del juguete solamente, haciendo ambas partes -tobillo y cadera-. Se mejora este punto, conectando a la placa controladora 2 sensores IMUs. La comunicación con la MinIMU-9 se realiza a través de I<sup>2</sup>C, y su interfaz permite seleccionar fácilmente entre 2 direcciones distintas para el giróscopo, y lo mismo para el acelerómetro/brújula. Para ello solamente tenemos que conectar SA0 a GND. (Ver Figura 3.11 y Tabla ??)

Es necesario cambiar las direcciones de una de las placas, de lo contrario ambas tendrían las mismas direcciones de esclavo I<sup>2</sup>C y causaría un mal funcionamiento si intentar leer de una de ellas. Además, se mejora el banco impreso para poder albergar los 2 zapatos y la máquina de estados, implementando la calibración de ambas piernas y aumentando número de iteraciones e incluyendo un filtrado de múltiples medidas para evitar leer ruido. Ver Figura 3.12.

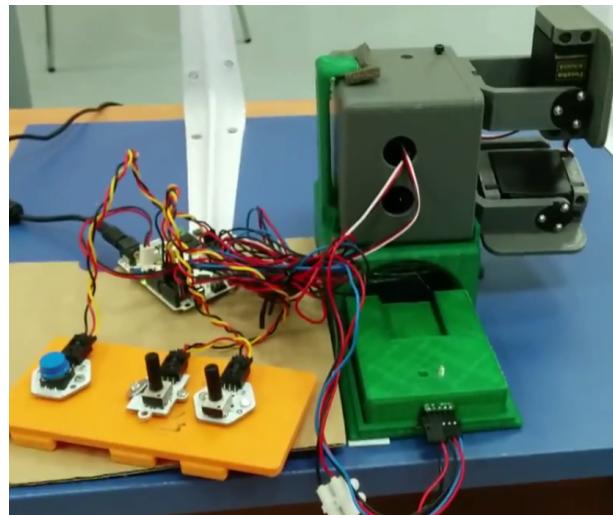


FIGURE 3.10: Bancos impresos para Zowi y zapato

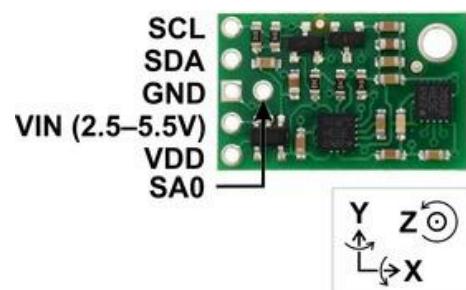


FIGURE 3.11: MinIMU-9 v3 Pin-out

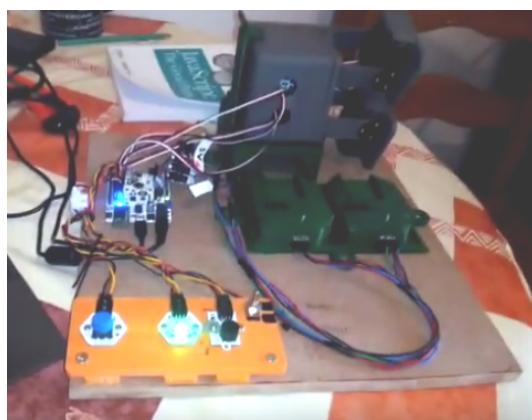


FIGURE 3.12: Banco impreso para Zowi y 2 zapatos

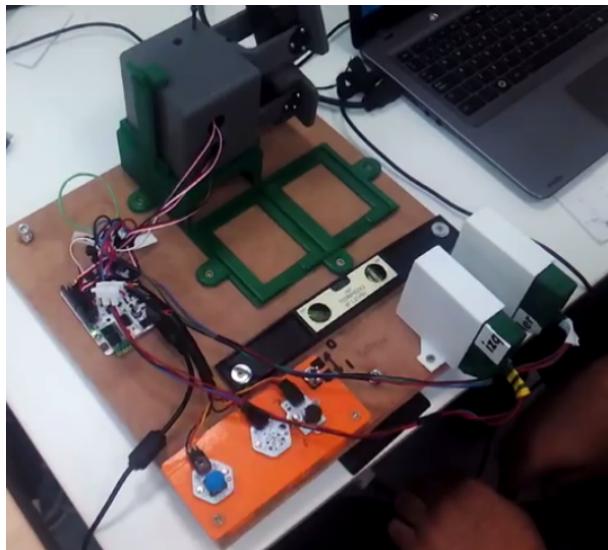


FIGURE 3.13: Banco con inclinación regulable

La calibración/inicialización de los sensores muestra resultados, como se ha comentado anteriormente, no tan buenos cuando el sensor se inclina notablemente. Por ello que el éxito de la calibración de la cadera ( $0^\circ$  de set point) es mayor que el de la calibración del pie ( $90^\circ$ ). Se considera corregir el set point de los pies en la fase de calibración de los sensores.

Se diseñan unos cajetines para el ajuste del sensor a los  $90^\circ$ , de modo que antes de comenzar la calibración de los juguetes, el sensor (zapato), debe ser calibrado en las posiciones horizontal y vertical.

Se impide avanzar en el procedimiento si tras calibrar los sensores al inicio, las medidas obtenidas difieren demasiado de  $0^\circ$  en ambos ejes, se vuelve a un estado en el que se espera pulsar el botón para volver a realizar la calibración, es decir, se comienza a implementar una gestión de errores.

Se añaden al conjunto un nivel y unos tornillos para regular la inclinación y mantener todo lo más horizontal posible antes de comenzar. Ver Figura 3.13.

### 3.3.1.2 Tomando forma

Hasta ahora todas las pruebas se habían realizado en la misma controladora de Zowi. Se recibe información sobre la placa que será empleada para los robots (en desarrollo por el Dpto. de Hardware), se conoce que el número de entradas/salidas será reducido y ajustado a los sensores/actuadores del propio juguete, por lo que realizar todas las conexiones que planteamos en nuestras pruebas en la misma placa junto a todos los periféricos de zowi no es una opción.

Se plantea el uso de otra placa conectada por I<sup>2</sup>C a la placa controladora del Zowi (por ahora una ZUM BT) y la librería Wire.h de Arduino. Zum BT es una versión de la Arduino UNO con bluetooth incorporado, y una etapa de potencia capaz de sacar hasta 3.2 amperios por sus salidas digitales.

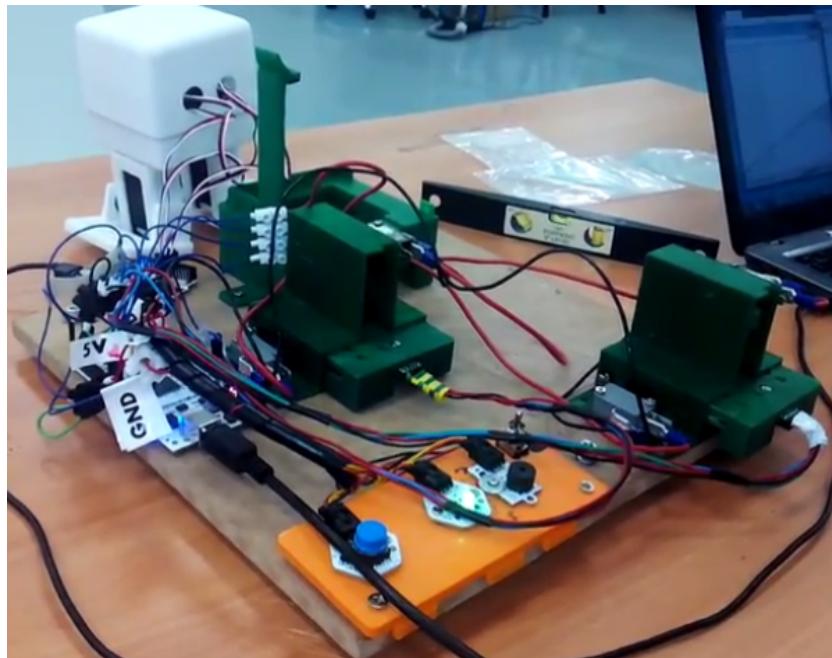


FIGURE 3.14: Banco con conexión MEGA-ZUM

La nueva placa empleada sería una Freeduino Mega, versión basada en la Mega, es elegida por ser compatible con todo lo desarrollado y por su bajo coste para la empresa y por presentar una cantidad de entradas y salidas mayor que los modelos UNO.

Se vuelven a diseñar unos bancos para calibrar los sensores tanto horizontal como verticalmente, es decir, inicializar ejes y ajustar el eje Y en vertical. Todo en el mismo espacio, como se puede ver en la Figura 3.14.

El empleo de una Mega nos proporciona una gran cantidad de pines de entrada salida, se incorporan al sistema unos finales de carrera para detectar que los zapatos están en la posición correcta para su calibración.

Como característica **principal**, se diseña una interfaz de comunicación entre ambas placas mediante I<sup>2</sup>C, se construyen determinadas "instrucciones" en la placa Mega (placa con la máquina de estados) y un programa que funciona como intérprete que es cargado en la ZUM (placa del robot), se encarga de traducir las instrucciones recibidas de la otra placa por I<sup>2</sup>C en movimientos de servos, entre otras cosas. Puede verse la arquitectura del sistema en la Figura 3.15.

La máquina de estados se hace más robusta en general, detectando fallos y facilitando la recuperación para un correcto funcionamiento.

Al final de esta etapa se conoce la controladora que llevará el juguete, así como la versión -prototipo- de Zowi definitiva, sinterizada por láser, en lugar de inyectada, pero ya con sus dimensiones y forma finales. Se puede ver en la Figura 3.16.

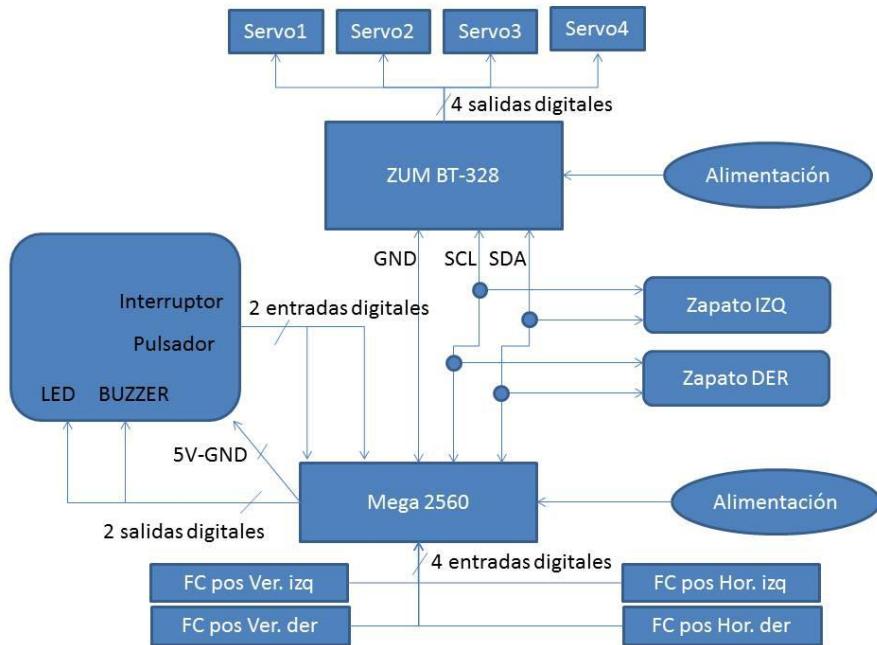
FIGURE 3.15: Arquitectura versión MEGA-ZUM por I<sup>2</sup>C

FIGURE 3.16: Prototipo final Zowi SLS

### 3.3.1.3 Prototipo final: Banco de Calibración

En este punto se produce el mayor cambio en el proyecto. La solución de comunicación por I<sup>2</sup>C entre las diferentes placas asumía que la controladora de Zowi tendría accesibles los pines al micro para I<sup>2</sup>C y SPI (cosa incierta) y, además, hacía necesario el acceso al interior del robot, que en principio, podría llegar ya montado de las etapas de producción anteriores. Se sugiere implementar la comunicación a través del puerto USB que tendrá la placa de Zowi.

Las controladoras Arduino pueden ser esclavos serie por USB, no maestros, realizar una comunicación serie de una a otra no es posible, de ahí que se plantee usar una Raspberry PI (Figura 3.28) como maestro Serie de ambas placas, esclavos serie, sustituyendo la interfaz I<sup>2</sup>C implementada en la versión anterior y haciendo de puente entre ambas.

Se contempló la posibilidad de quitar la Mega y emplear solamente la Raspberry como núcleo del sistema, sin embargo, eso conllevaría rehacer gran parte del trabajo y encontrar la forma de hacer funcionar los sensores con éste dispositivo, lo que requería un tiempo que no se tenía.

Una primera iteración fue la de sustituir la comunicación por I<sup>2</sup>C, por la comunicación serie, se adaptaron los programas de MEGA y de ZUM. La comunicación era unidireccional. Mega > Raspberry > Zowi. La Raspberry PI funcionaba como mero driver de comunicaciones:

- Se modificaron los comandos enviados con la librería Wire.h desde la free-duino Mega para ser comandos enviados por serie (Librería Serial.h) con ciertos caracteres de inicio y fin de comunicación a modo de protocolo (desarrollado específicamente para esta aplicación).
- Se implementó en la Raspberry un programa en Python que hacía de intérprete. Haciendo escucha a la comunicación serie abierta con la Freeduino Mega, y enviando comandos a la placa controladora de Zowi a través de la comunicación Serie abierta con ella.

La incorporación al sistema de una Raspberry PI, un ordenador al fin y al cabo, abría gran cantidad de mejoras de fácil implementación, por lo que se aprovechó la ocasión para desarrollar un prototipo "casi definitivo", no solamente a nivel de software.

Se decide introducir la electrónica en un armario eléctrico y, a su vez, emplear el mismo como estructura del soporte de Zowi y de los zapatos. Ver Figura 3.17. Se integran nivel y patas regulables.

Se introdujo un display LCD conectado a la Mega por I<sup>2</sup>C que hacía más fácil la interacción del usuario con el banco de calibración y se trabajó en mejorar la máquina de estados para hacer el uso del sistema más amigable; y se mejoró el conexiónado añadiendo una PCB custom como shield de la Mega, que ofrecía borneros para conectar los leds, display LCD, IMUs, finales de carrera y botones.

Durante el proceso de calibración se guardarán los offsets calculados en ciertas posiciones de la memoria EEPROM de la placa controladora de Zowi. Se usa para

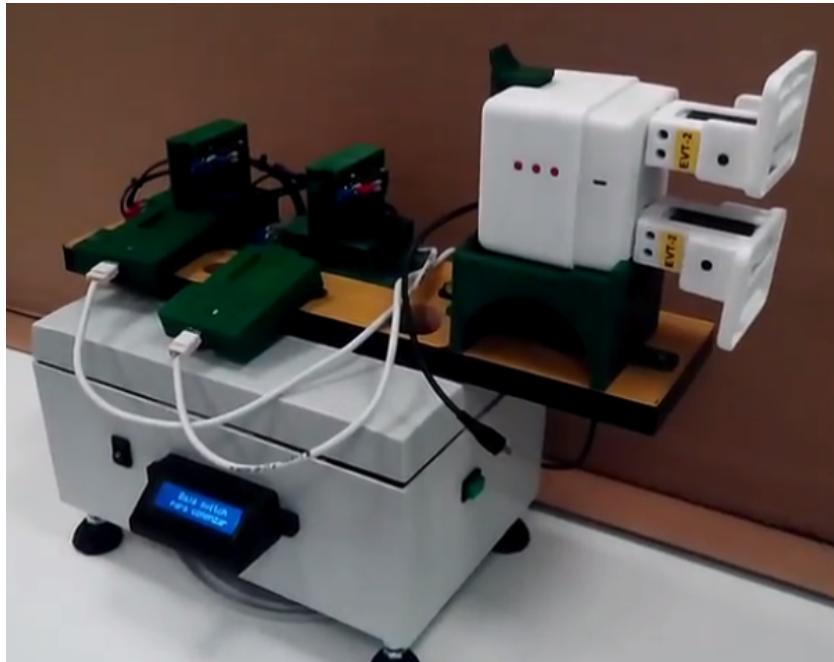


FIGURE 3.17: Banco prototipo final

ello la librería EEPROM.h. Esto es necesario para el funcionamiento de los programas por defecto, cuyas librerías (zowi.h y oscillator.h) están configuradas para usar los valores en dichas posiciones de la memoria.

Multitud de mejoras software definitivas fueron implementadas en esta fase para aumentar su funcionalidad, en el subcapítulo siguiente se verán en mayor detalle, a destacar:

- Raspberry utilizada como programador, empleando la utilidad AVRDUDE cargará el programa de calibración en cada Zowi.
- Cargará, del mismo modo y al finalizar la calibración, el programa de fábrica con que saldrá a la venta el producto. Sobrescribiendo el de calibración.
- Se le habilita acceso remoto por SSH y por XRDП para poder acceder a ella desde fuera.
- Se instala un servidor de MySQL y se configura para guardar datos tanto en local como en nuestros servidores.
- El entorno de programación de Arduino es instalado, para poder modificar la Mega directamente desde Raspberry.

### 3.3.2 Pruebas realizadas

Durante todo el desarrollo se han realizado pruebas, sin embargo, se comentan los resultados la prueba previa a MP (massive production), realizada con el prototipo final. La Figura 3.18 muestra el armario de calibración preparado para la prueba.



FIGURE 3.18: Prueba pre MP

Los operadores fueron compañeros que no habían tenido contacto alguno con la máquina, haciendo de operarios de forma distraída.

Se logran calibrar satisfactoriamente 62 juguetes, con solamente 1 contratiempo: Al no respetar el protocolo en cuanto a orden de operaciones, concretamente, intento de programar la placa sin energizar. Si se intenta programar la placa conectada pero sin encender, el programador AVRDUDE realiza hasta 10 intentos de programarla sin éxito; al energizarla durante los intentos, según en qué momento se haga, puede ser recuperable, o puede llevar al sistema a un estado de fallo. Se tuvo que reiniciar y volver a hacer la calibración de los zapatos. A pesar de ser un error de operador, se intenta corregir para la versión final.

Un total de 4 veces se produce una calibración mala, inducidas para poner a prueba el sistema, el sistema notifica (Display y sonido) la calibración como mala y se necesita que se vuelva a pulsar el botón. De ahí una calibración de apenas unos segundos en las gráficas siguientes, por no tener que conectar y calzar al juguete).

Analizando la información extraída de la base de datos tras las calibraciones, se muestra (Figura 3.19) Segundos vs. n° Zowi, una segunda gráfica (Figura 3.20) muestra 1 para calibraciones válidas, 0 para las declaradas fallidas. Los tiempos del primer y último registro no son representativos.

### 3.4 Armarios finales

El trabajo asignado al autor del proyecto debió haber concluido con el prototipo presentado en el subcapítulo anterior, dejando por diseñar una versión definitiva de los soportes y zapatos del banco al Dpto. de Mecánica de BQ, y el montaje de tantos armarios y bancos como fuese necesario para abordar la producción de todos los juguetes a Rosti (empresa de extrusión de plásticos y montaje subcontratada en

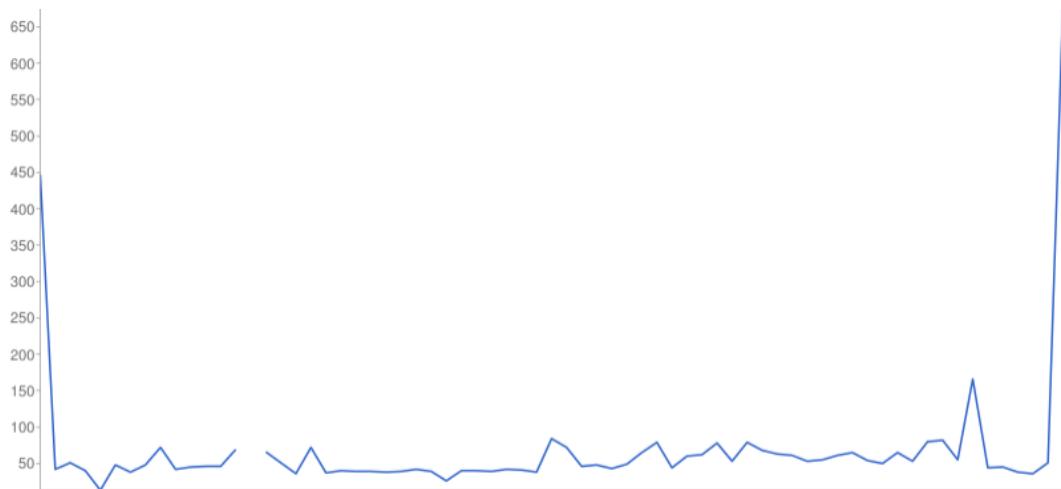


FIGURE 3.19: Segundos por cada calibración de Zowi

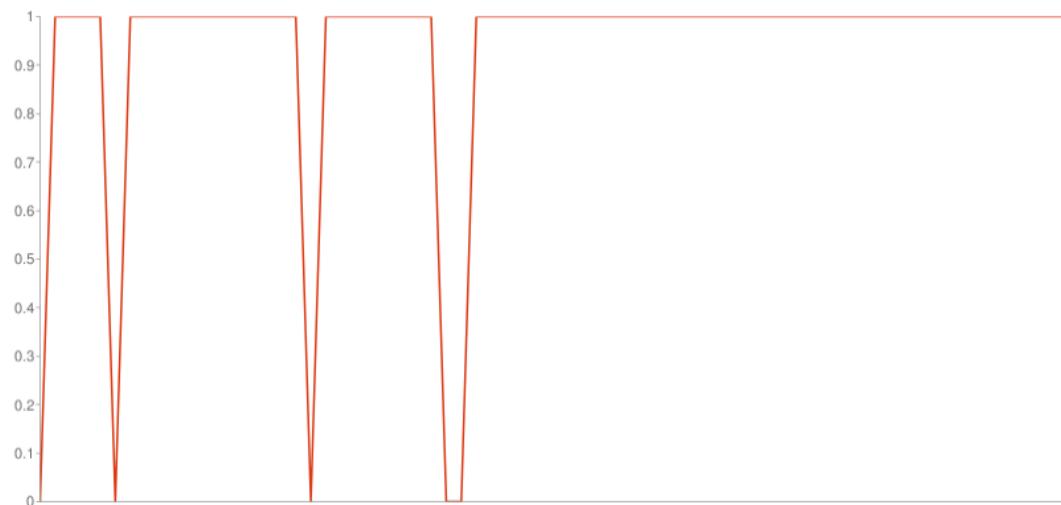


FIGURE 3.20: OK:1 | NOK:0 - para cada calibración de Zowi

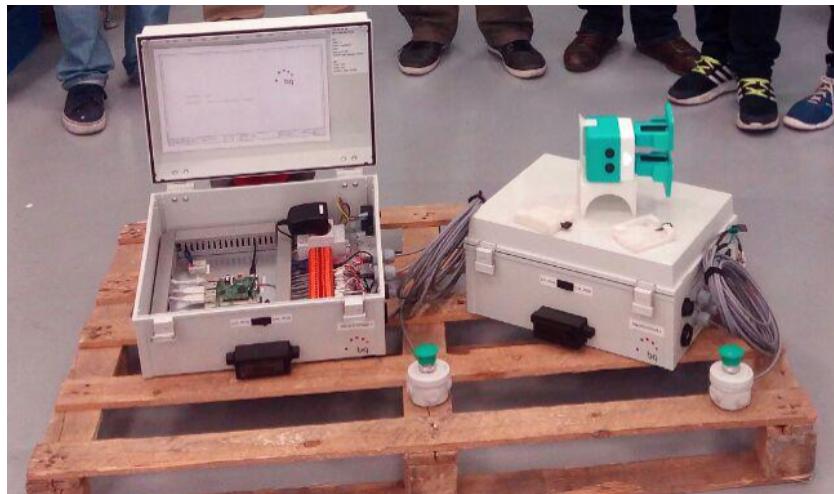


FIGURE 3.21: Armarios finales preparados para su envío a fábrica

Polonia), encargados del ensamblado de los robots Zowi; sin embargo, se decidió realizar el montaje de 2 armarios con ligeras modificaciones y planos eléctricos para enviar a Polonia ya probados y validados. Ver armarios en Figura 3.21.

Modificaciones destacables respecto al prototipo:

- Armario de componentes eléctricos separado de la parte mecánica, se siguieron utilizando los diseños impresos de versiones anteriores para validación del funcionamiento, pero las piezas finales quedaban a cargo del Dpto. de Mecánica. Éste nuevo armario sería más amplio que el anterior.
- Mangueras de cables más largas (pues armario y banco de calibración irán separados).
- Pulsador exterior con una manguera de cable para ser colocado donde se desee en fábrica, en lugar del pulsador en el lateral del armario de la versión anterior.
- Conexión USB y Ethernet por pasamuros al interior del armario.
- Cambios en el diseño del cajetín del display LCD, pues el armario irá colocado a la altura de los ojos del operador.

### 3.4.1 Componentes e implementación física

#### 3.4.1.1 Arquitectura y conexiones

El diagrama que se puede ver en la Figura 3.22, muestra, a grandes rasgos, un esquema de las conexiones entre los principales dispositivos que intervienen en el sistema. El sistema completo consta del armario eléctrico, además del juguete Zowi conectado al armario. Por ello se muestran en el esquema tanto las placas internas del armario (Mega y Raspberry), como la placa controladora de Zowi (Zum).

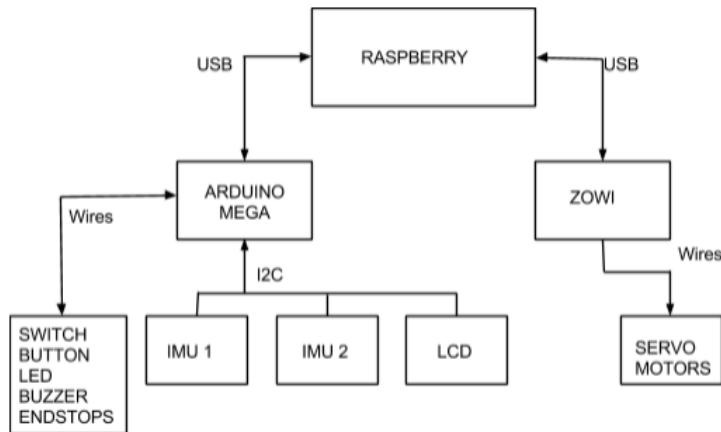


FIGURE 3.22: Esquema de conexiones

### 3.4.1.2 Armario eléctrico

Se elige instalar la parte fija del sistema en una placa de montaje dentro de un armario eléctrico de 40x30x18cm. Se emplean canaletas y bornas en un carril DIM para facilitar las conexiones al exterior del armario. Se utilizan pasamuros para las mangueras a pulsador y sensores IMU, así como para conexión USB (Raspberry - Zowi) y RJ45.

En el carril DIM se instala una toma de Schuko, al que se conectarán la Raspberry PI para alimentar toda la electrónica. El armario es energizado por un conector a red eléctrica JR-101 con interruptor y fusible.

Se pueden ver las Figuras 3.24 y 3.23, interior y lateral del armario, respectivamente. Para detalle de las conexiones al clemero, se pueden consultar los planos eléctricos en el Anexo ??.

### 3.4.1.3 Mega

Esta placa hace de cerebro del sistema a pesar de ser alimentada a través de la Raspberry, y de comunicar a través de ella con Zowi.

Recibe las lecturas de los sensores IMU, tiene conectados los finales de carrera, interruptor, pulsador, panel LCD, buzzer y leds. Todas las conexiones cableadas de entrada/salida, así como la alimentación pasan por un shield diseñado para ésta aplicación.

Define una máquina de estados que mueve al operador por todos los pasos de la calibración y controla a la raspberry mediante un protocolo de instrucciones implementado para este fin que usa como canal el puerto serie por USB, también empleado para hacer llegar instrucciones a la placa de Zowi.

Podemos decir que Mega es la encargada de interactuar con el exterior y, dentro de nuestro sistema, realiza las funciones de un PLC. Para ello se ha elegido cuidadosamente cómo conectar los diferentes componentes a sus entradas/salidas, en el Anexo ?? se puede consultar el pin-out de ésta placa:

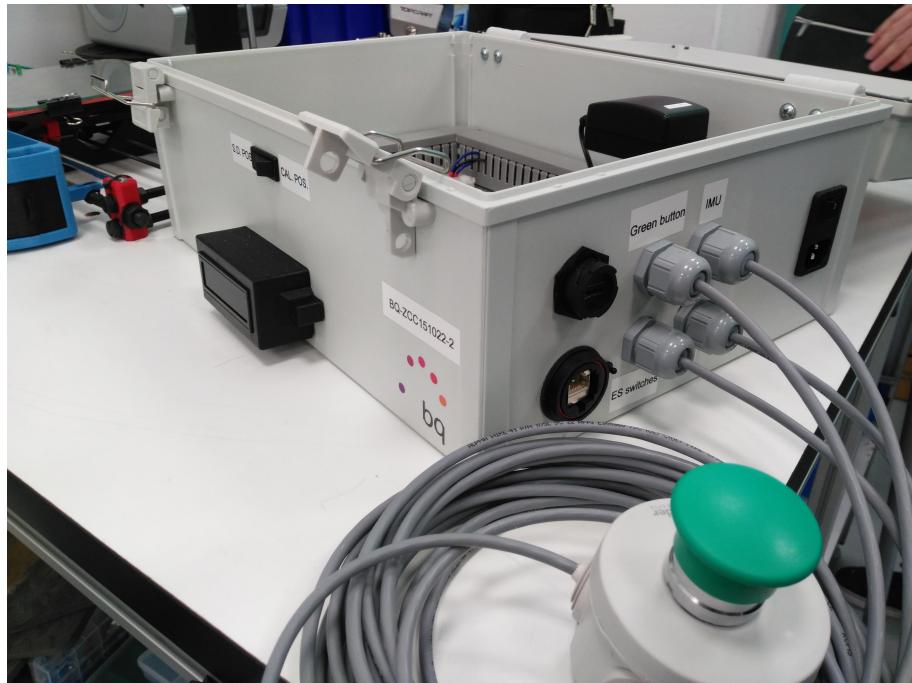


FIGURE 3.23: Armario final: Lateral



FIGURE 3.24: Armario final: Interior

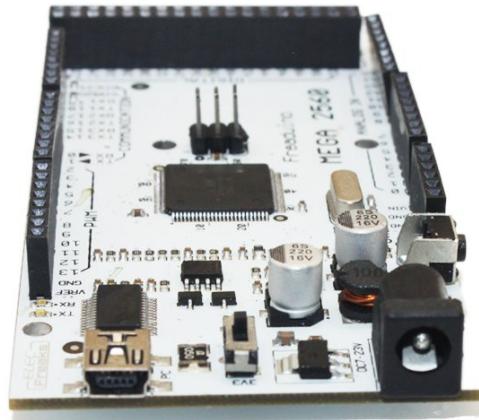


FIGURE 3.25: Freeduino Mega2560

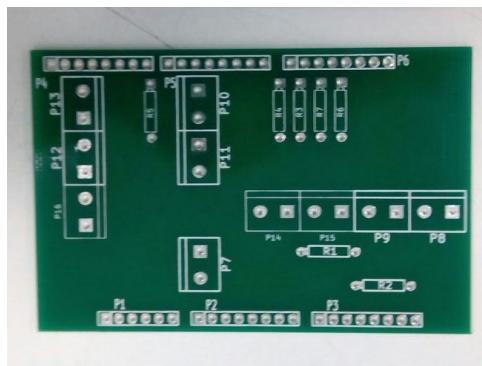


FIGURE 3.26: PCB Shield



FIGURE 3.27: Shield soldada

- Leds y buzzer necesitan PWM para poder configurar los sonidos/colores.
- Los pulsadores y finales de carrera se han conectado a entradas que tienen asociadas resistencias de pull-up, conectadas fácilmente por software.
- Se ha alimentado por Vin en lugar de por USB para poder alimentar a todos sus dispositivos.

Información sobre software en la Subsección 3.4.2. Para conocer con detalle las conexiones se pueden consultar los documentos del Anexo ??.

#### 3.4.1.4 Shield Mega

Se emplea el diseño de la shield de la versión prototipo del armario. En este caso se mandan a fabricar las PCBs, ya que nuestra fresadora no dió unos resultados tan profesionales.

Esta shield es colocada sobre los pines de la Mega y nos proporciona unos borneros para facilitar la conexión cableada a los dispositivos de entrada y salida, así como a la alimentación.

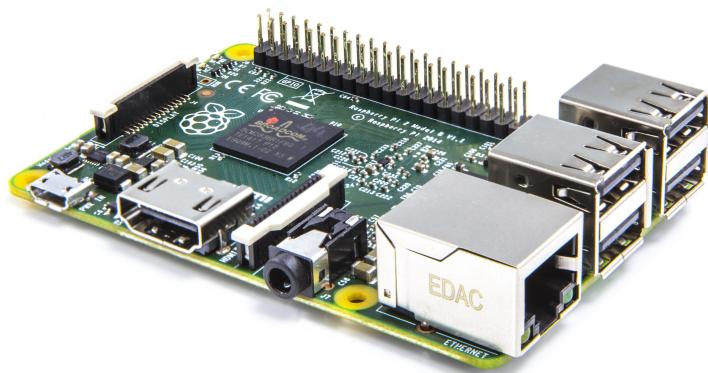


FIGURE 3.28: Raspberry PI 2 - 1GB Ram

Los esquemáticos y gerbers con el layout del diseño de la PCB se pueden consultar en el Anexo ???. Para información más clara sobre conexiones, se pueden consultar los planos eléctricos, en el Anexo ???.

### 3.4.1.5 Raspberry PI 2

Este ordenador (Figura 3.28) aporta muchísimas mejoras de fácil implementación al sistema, además de su función principal, hacer de enlace entre Mega y la controladora de Zowi. Éstas mejoras serán definidas en la Subsección 3.4.2, apartado dedicado al software.

Comunica con Mega y con las controladoras de los Zowi (Zum), para ello utiliza protocolo serie por cable USB con cada una de las placas. Además, por Ethernet, se puede acceder a ella por SSH (puerto 22) y por XRD (puerto 3389) desde la misma red local o desde internet configurando propiamente el NAT; y una IP y puerto pueden ser configurados para guardar datos de calibración contra una base de datos remota.

Es alimentada por su toma microUSB, a través de un transformador conectado al Schuko del armario. Se encarga de alimentar y programar por USB a los Zowi, para lo que ha sido necesario desbloquear el límite de corriente suministrada por puerto USB por el sistema operativo (Se necesita suficiente potencia para mover los servos). También alimenta a la Mega por su pin de 5V+, para lo que ha sido necesario elevar dicha tensión. Una vez más, se pueden ver los planos eléctricos para mayor detalle, Anexo ???.

### 3.4.1.6 Power Boost

La Mega debe ser alimentada a más de 7V. Para reducir número de transformadores y el espacio ocupado, se decide alimentarla a través de la raspberry.

Este dispositivo (Figura 3.29) acepta una tensión de entrada de 1.5V a 25V y, mediante el potenciómetro, permite ajustar su salida a valores entre 4V y 25V.

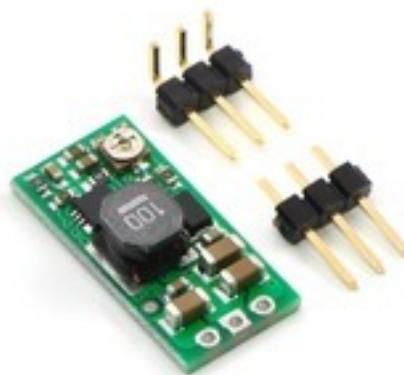


FIGURE 3.29: Power Boost

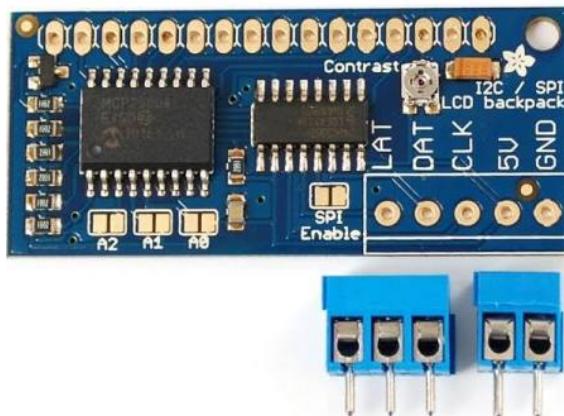


FIGURE 3.30: Backpack para Display LCD

El power boost es ajustado para elevar la tensión suministrada por pines de la Raspberry de 5V a 9V para alimentar la Mega. Necesario para no tener que dar la corriente por el puerto USB. Aún desbloqueando el límite software de intensidad de la Raspberry, con lo que se consigue hasta 1.2A a 5V, el sistema no funciona correctamente al alimentar por USB ambas placas, Zowi carga su batería y mueve sus servos utilizando dicha corriente. De modo que se alimenta la Mega (y todo lo conectado a ella) desde el pin de 5v de Raspberry, sin usar el puerto USB y sin tener que introducir otro transformador en el armario.

#### 3.4.1.7 LCD Display

Una pantalla es instalada para indicar al operador instrucciones e información sobre el uso del sistema. Se elige un display de 16 filas x 2 columnas y un backpack de Adafruit (Figura 3.30) que nos permite comunicar con el display utilizando I<sup>2</sup>C en lugar de una buena cantidad de salidas digitales, y que funciona con librerías más que probadas hechas por la comunidad.

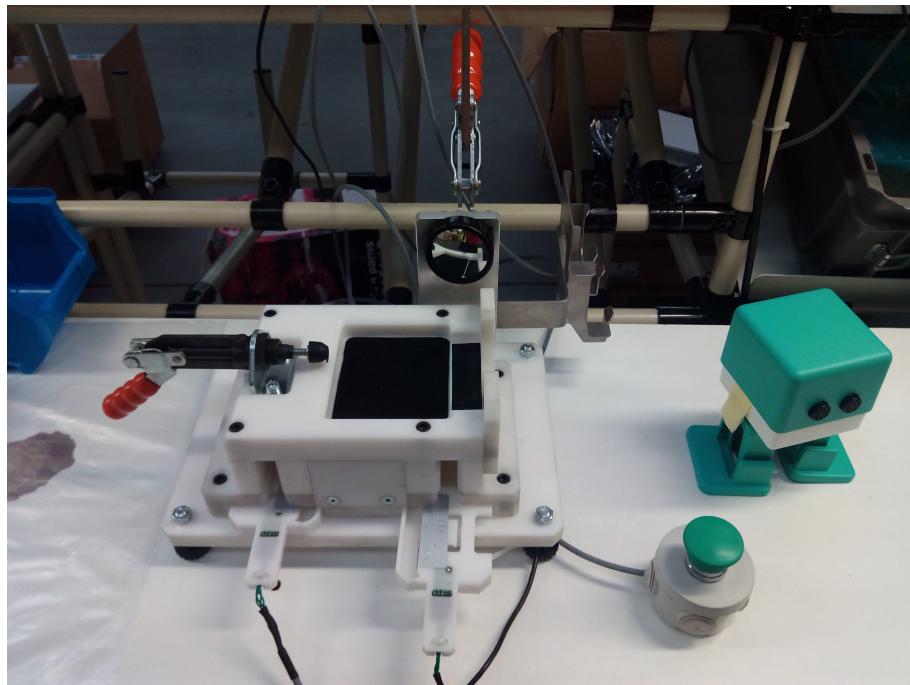


FIGURE 3.31: Banco soporte y zapatos finales

#### 3.4.1.8 Otros componentes

Además de los componentes principales comentados anteriormente, y de los sensores IMU tratados en el Marco Teórico (Capítulo ??), se emplean otros componentes menores como son los leds, el buzzer, los botones, finales de carrera, así como sus resistencias, ademas de las clemas, puntas, y más. En los planos eléctricos (Anexo ??) y hoja de costes (Anexo ??), se puede consultar más información sobre su conexión o modelo.

#### 3.4.1.9 Zowi - Zum

Como se ha comentado anteriormente, Zowi también forma parte del sistema. Su controladora tendrá conectados, entre otras cosas, los 4 servomotores responsables de mover las articulaciones. Comunica solamente con la Raspberry PI utilizando protocolo serie por cable USB. La Raspberry le descargará un software para poder interpretar las instrucciones recibidas, y ser capaz de mover los servos o guardar información en su EEPROM.

#### 3.4.1.10 Banco soporte Final de Zowi y zapatos

Para validaciones y durante todo el desarrollo del proyecto -así como para su uso en Madrid, tras acabar la fase de producción en fábrica- se han empleado, y emplean, los diseños realizados por el autor, cuyos planos podemos consultar en el Anexo ?. No se especifican cotas de tolerancias por estar pensados para fabricar con impresoras 3D de filamento fundido, donde las dimensiones finales varían por parámetros como la temperatura o la velocidad de extrusión de la máquina.

Sin embargo, el utilaje empleado en cadena de montaje (Figura 3.31) 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.

### 3.4.2 Software

En esta subsección se explicarán los aspectos relacionados con la programación del sistema y los algoritmos empleados. Se ha dividido 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.

#### 3.4.2.1 Mega

Este dispositivo se puede programar utilizando el lenguaje de Arduino, que es una adaptación de C++ que proviene 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 I<sup>2</sup>C.
- Ficheros de la librería MinIMU, que extiende a su vez de las librerías del giroscopio 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.

##### 3.4.2.1.1 Protocolo de comunicación

Para lograr enviar instrucciones a la controladora del robot utiliza como intermedio 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°, utilizando la librería servo.
- **MHOC:** Ordena a Zowi mover todos los servos a la posición 90° 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 también resultaron útiles durante el desarrollo, sin embargo no se utilizan en la versión final.

### 3.4.2.1.2 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 estándar 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.

### 3.4.2.1.3 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 90°, se recuerda que el rango del servo es 0-180°, y que el juguete necesita un rango de solamente unos 80°.

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 pie. 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 90°), y se calcula la nueva posición necesaria a establecer en el servo para obtener los 90° deseados.

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

### 3.4.2.1.4 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 90° como ángulo deseado. Realmente, el valor éste ángulo para las caderas se corresponde con el 0°, mientras que para los pies se corresponde con el 90°, por lo que también es ajustado en esta etapa. Tras valorar que los ángulos 0° han sido bien definidos y su lectura

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

#### 3.4.2.1.5 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 mínimos 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.

#### 3.4.2.2 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 I<sup>2</sup>C, 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 I<sup>2</sup>C, 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.

##### 3.4.2.2.1 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**:

#### 3.4.2.3 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, Wheezy.

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.

### 3.4.2.3.1 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 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.

#### 3.4.2.3.2 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 ??.

#### 3.4.2.3.3 Configuración adicional

El servicio de escritorio remoto XRD 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 1.2A por USB para alimentar a Zowi (por defecto, limitado 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 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.

## Chapter 4

# Desarrollo

## 4.1 Conclusiones y trabajos futuros

### 4.1.1 Conclusiones

Intento de programar una placa no energizada. Puertos no liberados, bind-unbind.



## Appendix A

# Manual de usuario

Manual de usuario generado.

# User manual

## 1. Workbench preparation

First of all, it is necessary to adjust the workbench calibration using the four adjustable legs installed at the bottom side and the level incorporated. It is important to set the work area as parallel as possible to the ground plane, and fix the bank to ensure that the position does not change during the process.

## 2. Switching on the system

When switched on the main switch (at the back side), it is necessary to wait some seconds (around 30-50s) until the system has completely started. During this time, the blue led will be blinking slowly and the LCD will be showing “*Zowi workbench. Starting...*”. After this, the workbench will emit a sound and the LED will stop blinking.

At this point, it depends on the switch position (at the front side) to start the process.

If the switch is set at “1” the LCD will show “*Change switch to start process*”, whilst whether the position is set at “0”, the LCD will show “*Place shoes in Horizontal box*”, and it will be ready to start the calibration process.

### Notes:

**It is really important not to have any Zowi plugged when the system is switching on, to avoid communication problems (usb port assignment done by raspberry).**

**It helps to the horizontal offset calibration having the shoes placed in the horizontal box when the system is switching on, as the IMU sensors will be getting right measures all the time.**

## 3. IMUs calibration

The operator must follow the display instructions to calibrate the offset of the sensors correctly. Firstly, the horizontal position is calibrated. When this calibration is right, the LCD will show “*Horizontal done. Insert Vertical*” to continue with the vertical calibration.

It is really important not to disturb the system, trying to avoid vibrations, movements etc that could interfere in the sensors calibration. If after trying to calibrate the sensor five or six times, the LCD shows “*Horizontal calib failed. Retry*”, switch off the system as shown at step 6 (Shutting down the system), and switch on it again, being sure that the shoes are correctly placed in the horizontal box.

When the sensors calibration is finished, the LCD will show “*Plug, Switch on, Put on and Push*” and the blue LED will start blinking slowly.

**Note: If the workbench is moved at any point of the process, it must be repeated this step to ensure the correct calibration relative to the ground.**

#### 4. Zowi calibration

After calibrating correctly the offset of the sensors, it will proceed to calibrate as many Zowis as desired.

To calibrate each Zowi, place it at the workbench with the face looking to the operator (or with the connector looking to the operator depending on the program loaded in the system initially, by default the option with the connector looking to the operator is loaded) and put the shoes at each foot, paying attention to use the correct shoe (top, bottom). After this, switch on Zowi, plug it and push the button. The LCD will show "*initiating communication*".

The Raspberry Pi will try to upload the calibration program to the Zowi board. There are three possible situations:

- The calibration program is uploaded correctly. The system will continue with the calibration of each joint.
- The Zowi board does not respond after 40 seconds. The LCD will show "*Timeout. Check Zowi and push*". Check Zowi is switched on and push the button to retry.
- The Zowi board responds but it is impossible to upload the calibration program. The LCD will show "*Connection fault. Check Zowi*". Check Zowi is plugged and push button to retry.

When the calibration program is uploaded correctly, the LCD will show "*Communication OK. Calibrating...*" and the LED will blink quickly. Zowi will move its hips and feet to calibrate them. After various iterations trying to calibrate the four joints, there are three possible situations:

- The calibration has failed. It has been impossible to reach a calibrated position. The LCD will show "*CALIBRATION FAILED*" and the red LED will turn on. The LCD will show "*Push for a new calibration*" after a second.
- The calibration is right, so the Raspberry Pi will try to upload the Final Test program. The LCD will show "*Calibration OK. Loading Test Prg*". If it is not possible to upload this program the LCD will show "*CALIBRATION FAILED*" and the red LED will turn on. The LCD will show "*Push for a new calibration*" after a second.
- The calibration is right, so the Raspberry Pi will try to upload the Final Test program. The LCD will show "*Calibration OK. Loading Test Prg*". If the program is uploaded correctly the green LED will turn on and the LCD will show "*Push for a new calibration*".

Once the process is done, switch off Zowi and unplug it. It is possible to connect other Zowi to repeat the Zowi calibration pushing the button.

**5. Reset the system**

At any point of the process, excepting for the step 2(Switching on the system) and 6 (Shutting down), it is possible to switch to “1” the front switch, and the system will change to step 3 (IMUs calibration).

**6. Shutting down the system**

At step 3 (IMUs calibration), it is possible to switch off the whole system. To do it, it is necessary to push the button for at least two seconds. The LCD will show “*Shutting down...*” and afterwards, the LCD will show nothing. When this happens, switch off the power switch (at the back side).





## Appendix B

### Planos Eléctricos: Cabinet 1

A      B      C      D      E      F

9  
8  
7  
6  
5  
4  
3  
2  
1  
0

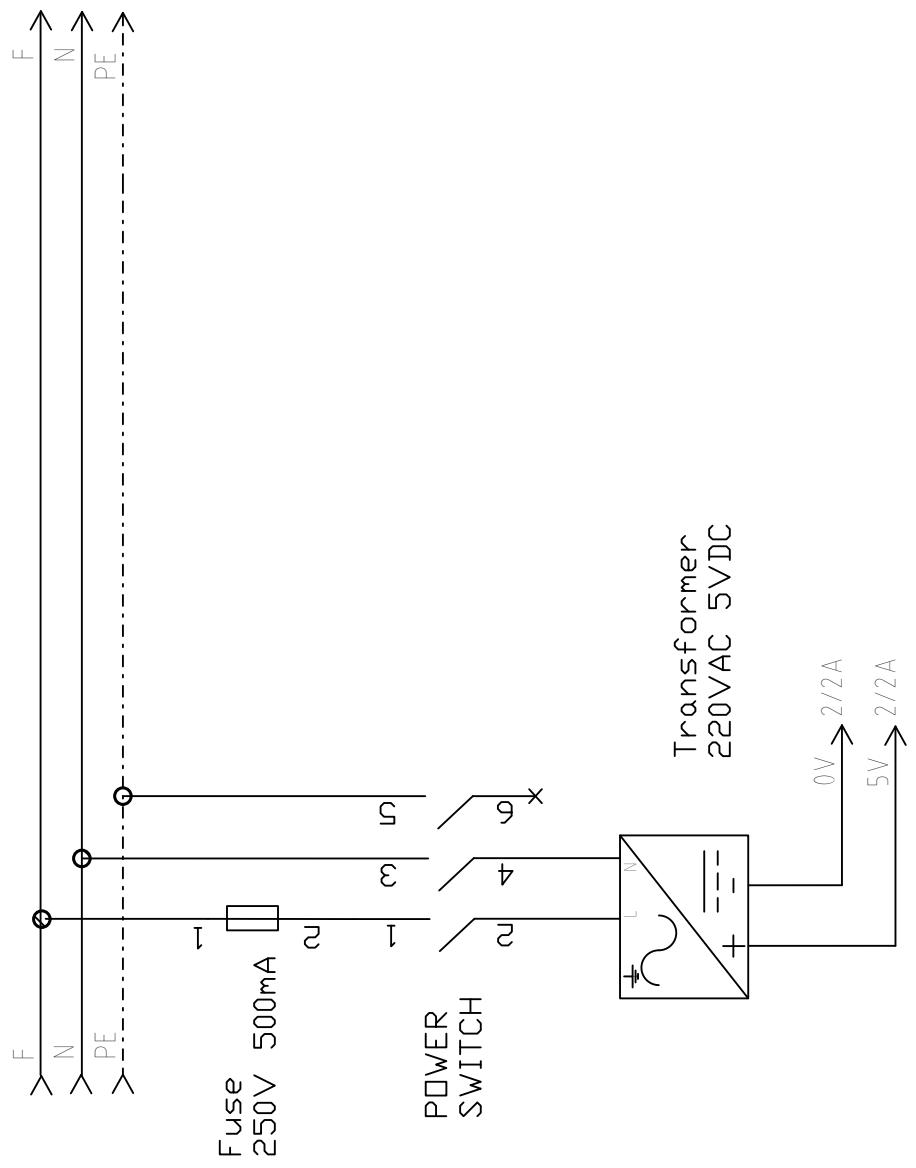


COMPANY: BQ

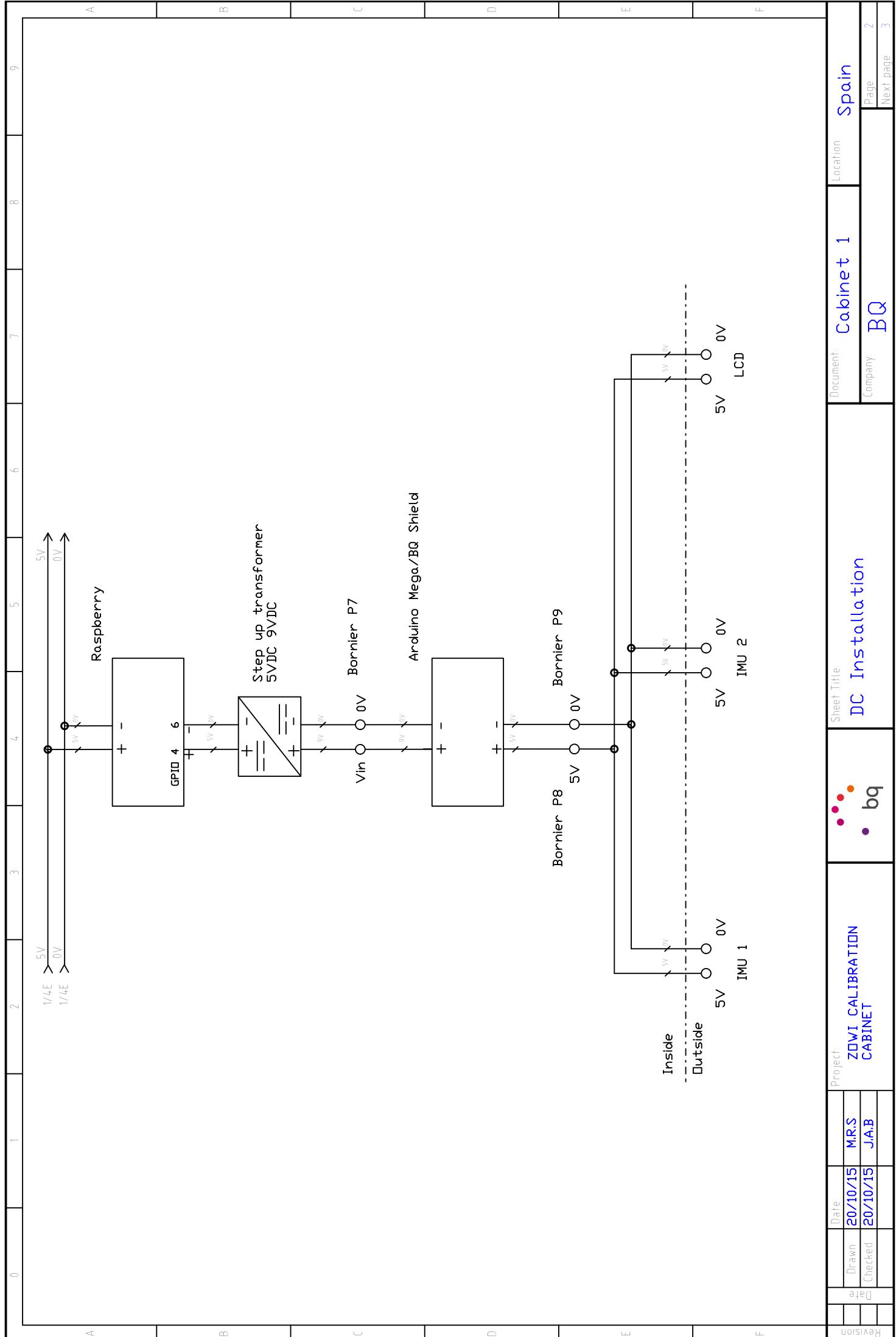
PROJECT: ZOWI CALIBRATION CABINET

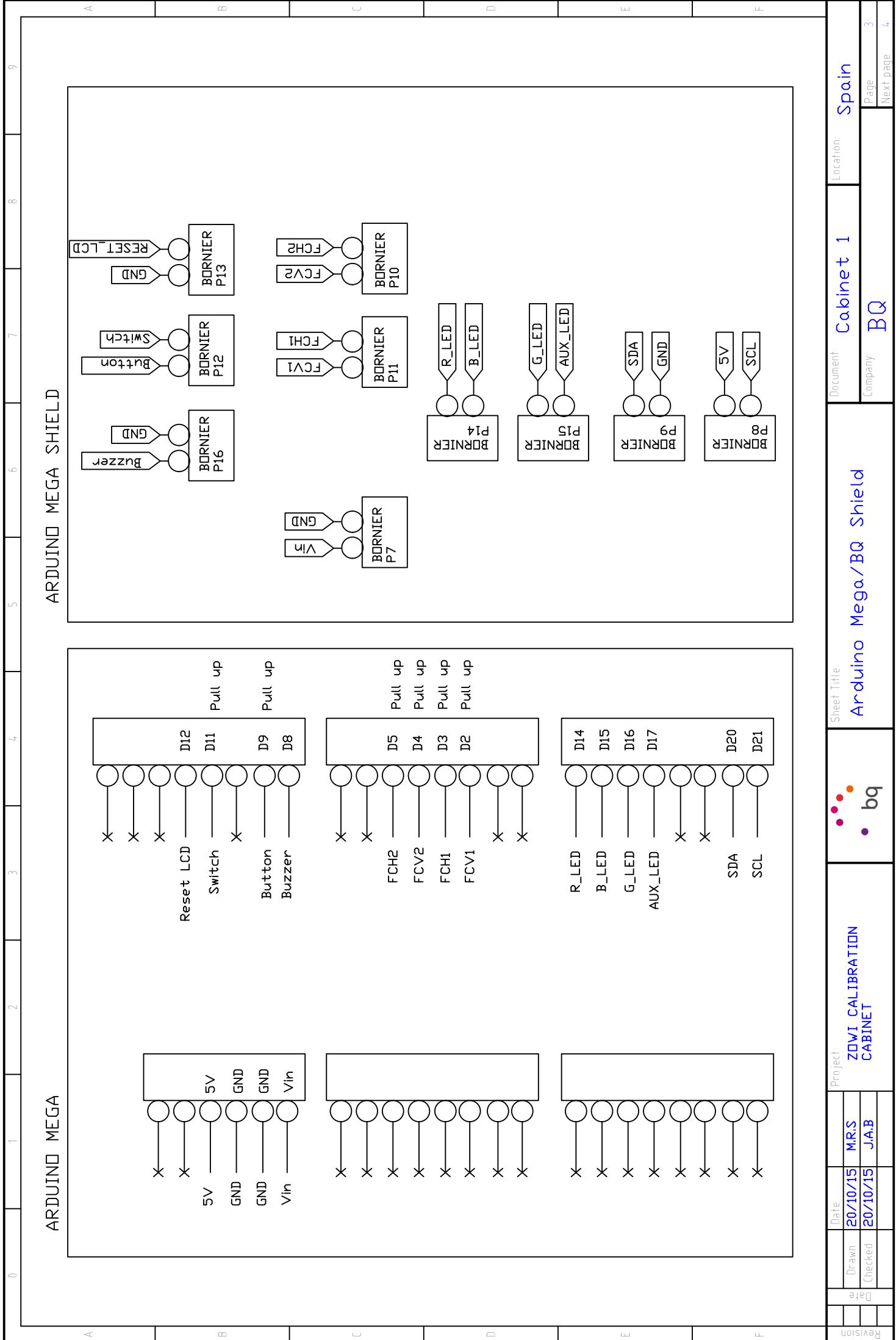
Project:	ZOWI CALIBRATION CABINET	Sheet Title:	FRONT PAGE
Date Drawn:	20/10/15	J.A.B	bq
Date Checked:	20/10/15	R.D.S	

Document:	Cabinet 1	Location:	Spain
Company:	BQ	Page:	0
		Next Page:	1



Revision	Date	Project	Sheet Title	Location	Document
A	Drawn	ZWI CALIBRATION	• ● ●	Spain	
B	Checked	ZWI CABINET	• bq		
C					Company: BQ
D					Page 1
E					Page Next Page
F					2





A 8

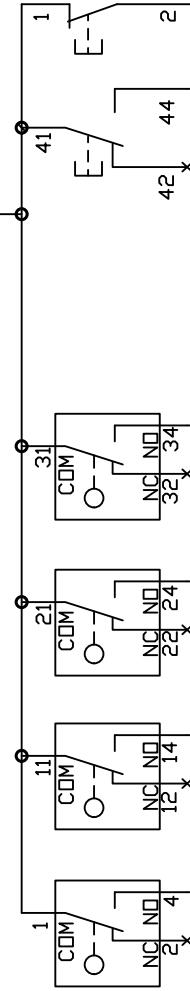
B 7

C 6

D 5

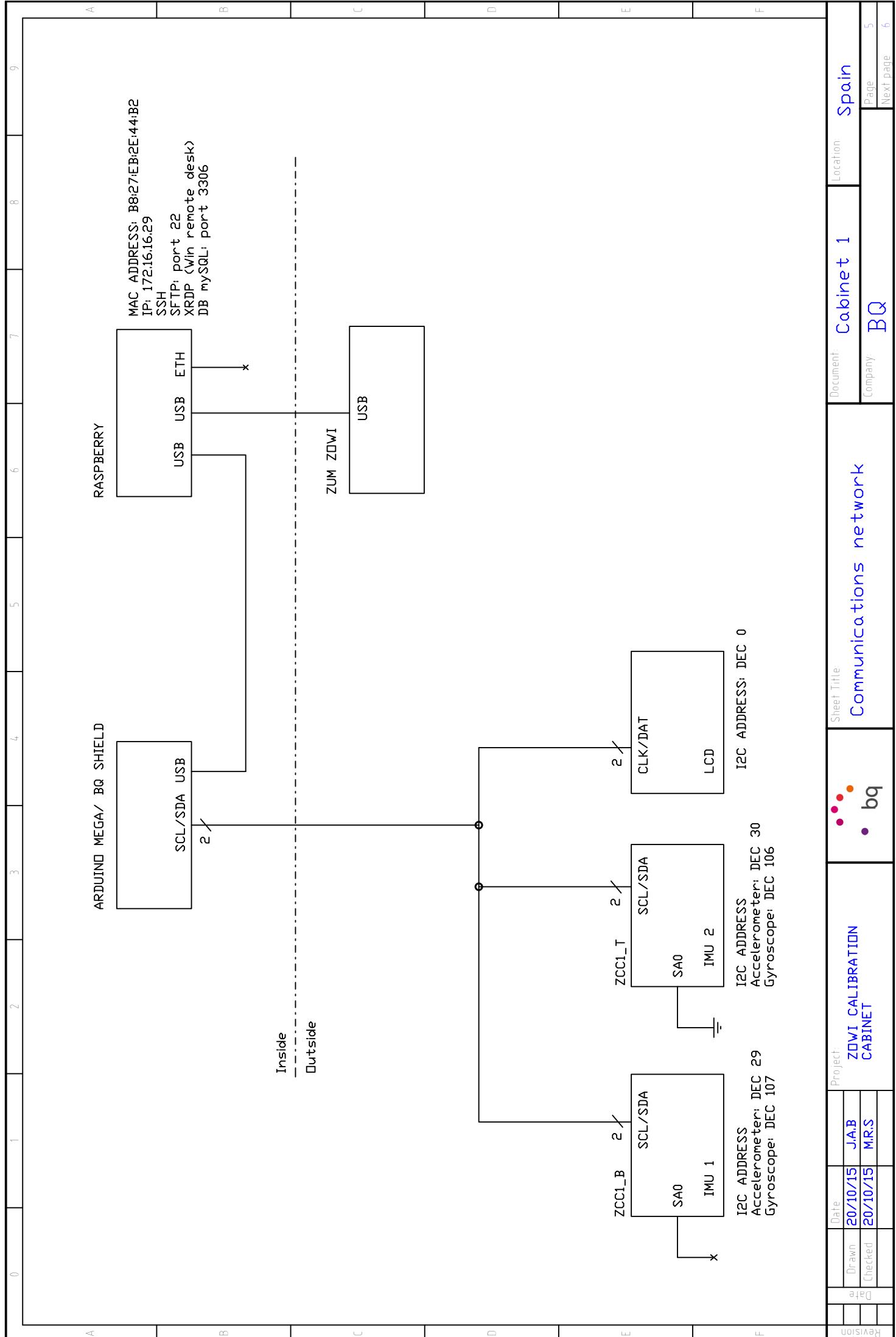
E 4

F 3

**BØRNIER**  
13**RESET\_LCD****GND****SWITCH BUTTON****BØRNIER**  
P12**FCV2****BØRNIER**  
P10**FCV1****BØRNIER**  
P11Document: **Cabinetet 1**  
Company: **BQ**Location: **Spain**  
Page: **4**

Next Page

Revision: **A**  
Drawn: **20/10/15**  
Checked: **20/10/15**Project: **ZWI CALIBRATION**  
CABINETDate: **J.A.B**  
M.R.S



A	B	C	D	E	F																																																																								
9																																																																													
8																																																																													
7																																																																													
6																																																																													
5																																																																													
4																																																																													
3																																																																													
2																																																																													
1																																																																													
0																																																																													
A	B	C	D	E	F																																																																								
																																																																													
																																																																													
<p style="text-align: center;">Bornier connections</p> <table border="1"> <thead> <tr> <th>Component</th> <th>Color</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>Vin Boost</td><td>White</td><td>Vin Boost</td></tr> <tr><td>Vout Boost</td><td>White</td><td>Vout Boost</td></tr> <tr><td>VCC Mega 1</td><td>Red</td><td>VCC Mega 1</td></tr> <tr><td>VCC Mega 2</td><td>Blue</td><td>VCC Mega 2</td></tr> <tr><td>0V</td><td>Black</td><td>0V</td></tr> <tr><td>0V</td><td>Black</td><td>0V</td></tr> <tr><td>SDA</td><td>White</td><td>SDA</td></tr> <tr><td>SCL</td><td>Orange</td><td>SCL</td></tr> <tr><td>Switch</td><td>Grey</td><td>Switch</td></tr> <tr><td>Button</td><td>Red</td><td>Button</td></tr> <tr><td>FCH2</td><td>Red</td><td>FCH2</td></tr> <tr><td>FCV2</td><td>Green</td><td>FCV2</td></tr> <tr><td>FCH1</td><td>White</td><td>FCH1</td></tr> <tr><td>FCV1</td><td>Orange</td><td>FCV1</td></tr> <tr><td>Green LED</td><td>Grey</td><td>Green LED</td></tr> <tr><td>Blue LED</td><td>Grey</td><td>Blue LED</td></tr> <tr><td>Red LED</td><td>Grey</td><td>Red LED</td></tr> <tr><td>Buzzer</td><td>Grey</td><td>Buzzer</td></tr> <tr><td>Black USB</td><td>Black</td><td>Black USB</td></tr> <tr><td>Red USB</td><td>Red</td><td>Red USB</td></tr> <tr><td>Green USB</td><td>Green</td><td>Green USB</td></tr> <tr><td>White USB</td><td>White</td><td>White USB</td></tr> <tr><td>Shielded USB</td><td>Grey</td><td>Shielded USB</td></tr> </tbody> </table>						Component	Color	Description	Vin Boost	White	Vin Boost	Vout Boost	White	Vout Boost	VCC Mega 1	Red	VCC Mega 1	VCC Mega 2	Blue	VCC Mega 2	0V	Black	0V	0V	Black	0V	SDA	White	SDA	SCL	Orange	SCL	Switch	Grey	Switch	Button	Red	Button	FCH2	Red	FCH2	FCV2	Green	FCV2	FCH1	White	FCH1	FCV1	Orange	FCV1	Green LED	Grey	Green LED	Blue LED	Grey	Blue LED	Red LED	Grey	Red LED	Buzzer	Grey	Buzzer	Black USB	Black	Black USB	Red USB	Red	Red USB	Green USB	Green	Green USB	White USB	White	White USB	Shielded USB	Grey	Shielded USB
Component	Color	Description																																																																											
Vin Boost	White	Vin Boost																																																																											
Vout Boost	White	Vout Boost																																																																											
VCC Mega 1	Red	VCC Mega 1																																																																											
VCC Mega 2	Blue	VCC Mega 2																																																																											
0V	Black	0V																																																																											
0V	Black	0V																																																																											
SDA	White	SDA																																																																											
SCL	Orange	SCL																																																																											
Switch	Grey	Switch																																																																											
Button	Red	Button																																																																											
FCH2	Red	FCH2																																																																											
FCV2	Green	FCV2																																																																											
FCH1	White	FCH1																																																																											
FCV1	Orange	FCV1																																																																											
Green LED	Grey	Green LED																																																																											
Blue LED	Grey	Blue LED																																																																											
Red LED	Grey	Red LED																																																																											
Buzzer	Grey	Buzzer																																																																											
Black USB	Black	Black USB																																																																											
Red USB	Red	Red USB																																																																											
Green USB	Green	Green USB																																																																											
White USB	White	White USB																																																																											
Shielded USB	Grey	Shielded USB																																																																											
																																																																													
<p>Sheet Title: Bornier connections</p>																																																																													
<table border="1"> <tr><td>Project</td><td>ZWI CALIBRATION CABINET</td></tr> <tr><td>Date Drawn</td><td>20/10/15</td></tr> <tr><td>Date Checked</td><td>20/10/15</td></tr> <tr><td>J.A.B</td><td>M.R.S</td></tr> </table>						Project	ZWI CALIBRATION CABINET	Date Drawn	20/10/15	Date Checked	20/10/15	J.A.B	M.R.S																																																																
Project	ZWI CALIBRATION CABINET																																																																												
Date Drawn	20/10/15																																																																												
Date Checked	20/10/15																																																																												
J.A.B	M.R.S																																																																												
<table border="1"> <tr><td>Revision</td><td>6</td></tr> <tr><td>Page</td><td>1</td></tr> <tr><td>Next page</td><td></td></tr> </table>						Revision	6	Page	1	Next page																																																																			
Revision	6																																																																												
Page	1																																																																												
Next page																																																																													
<p style="text-align: right;">Location: Spain</p>																																																																													
<table border="1"> <tr><td>Document:</td><td>Cabinet 1</td></tr> <tr><td>Company:</td><td>BQ</td></tr> </table>						Document:	Cabinet 1	Company:	BQ																																																																				
Document:	Cabinet 1																																																																												
Company:	BQ																																																																												

## Appendix C

### Planos Eléctricos: Cabinet 2

A      B      C      D      E      F

9  
8  
7  
6  
5  
4  
3  
2  
1  
0

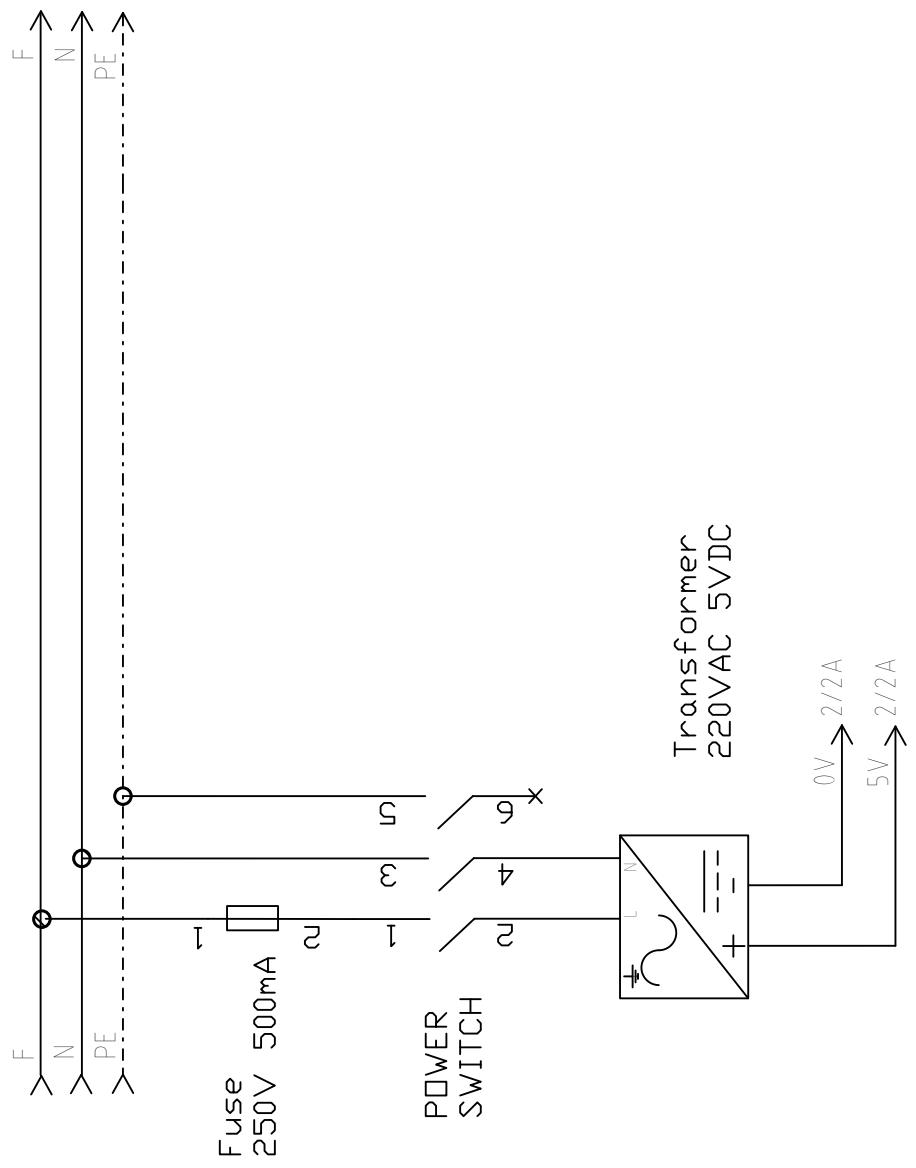


COMPANY: BQ

PROJECT: ZOWI CALIBRATION CABINET

Project:	ZOWI CALIBRATION CABINET	Sheet Title:	FRONT PAGE
Date Drawn:	20/10/15	Date Checked:	J.A.B R.D.S
Revised:		Page:	0

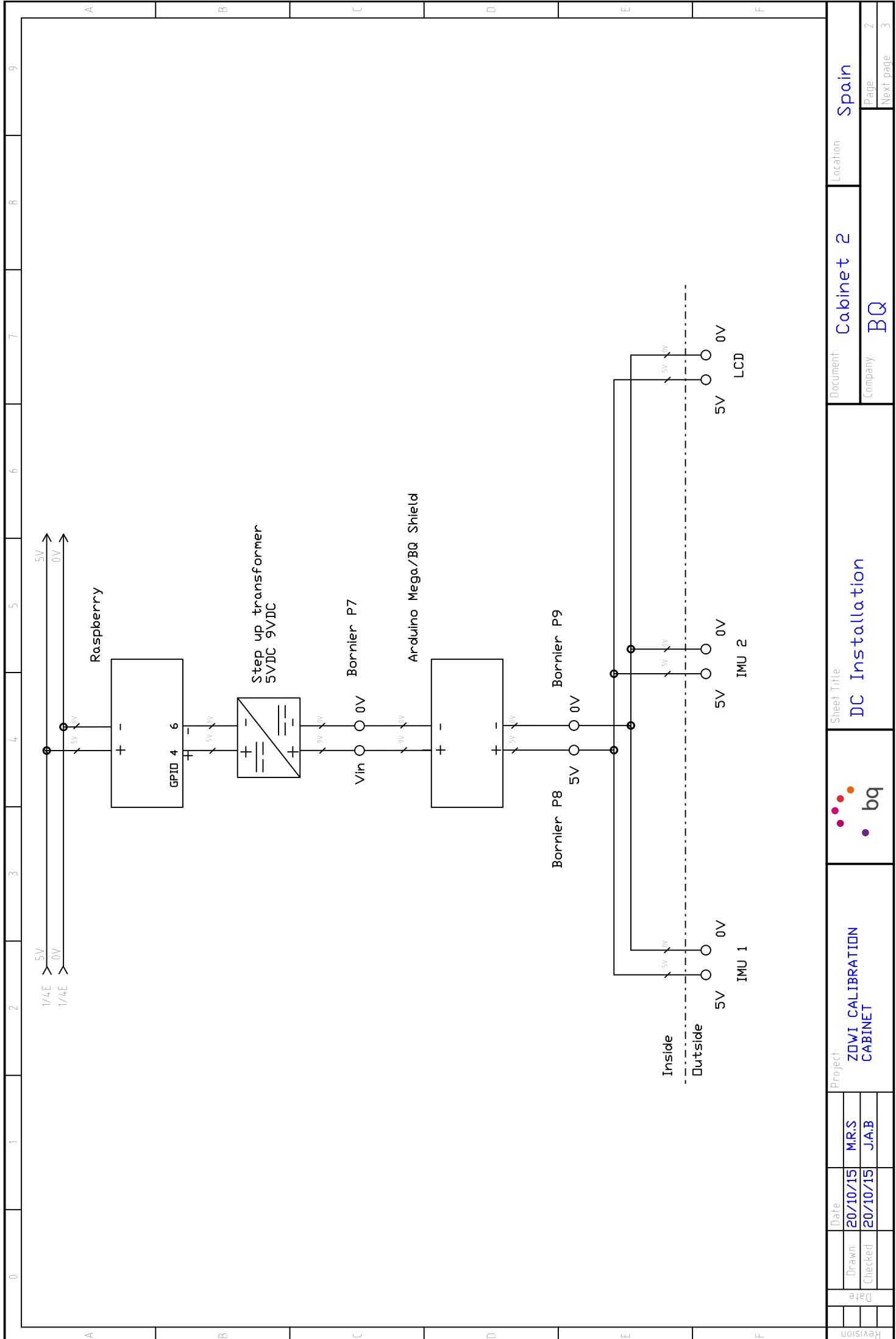
Document:	Cabinet 2	Location:	Spain
Company:	BQ	Page:	1



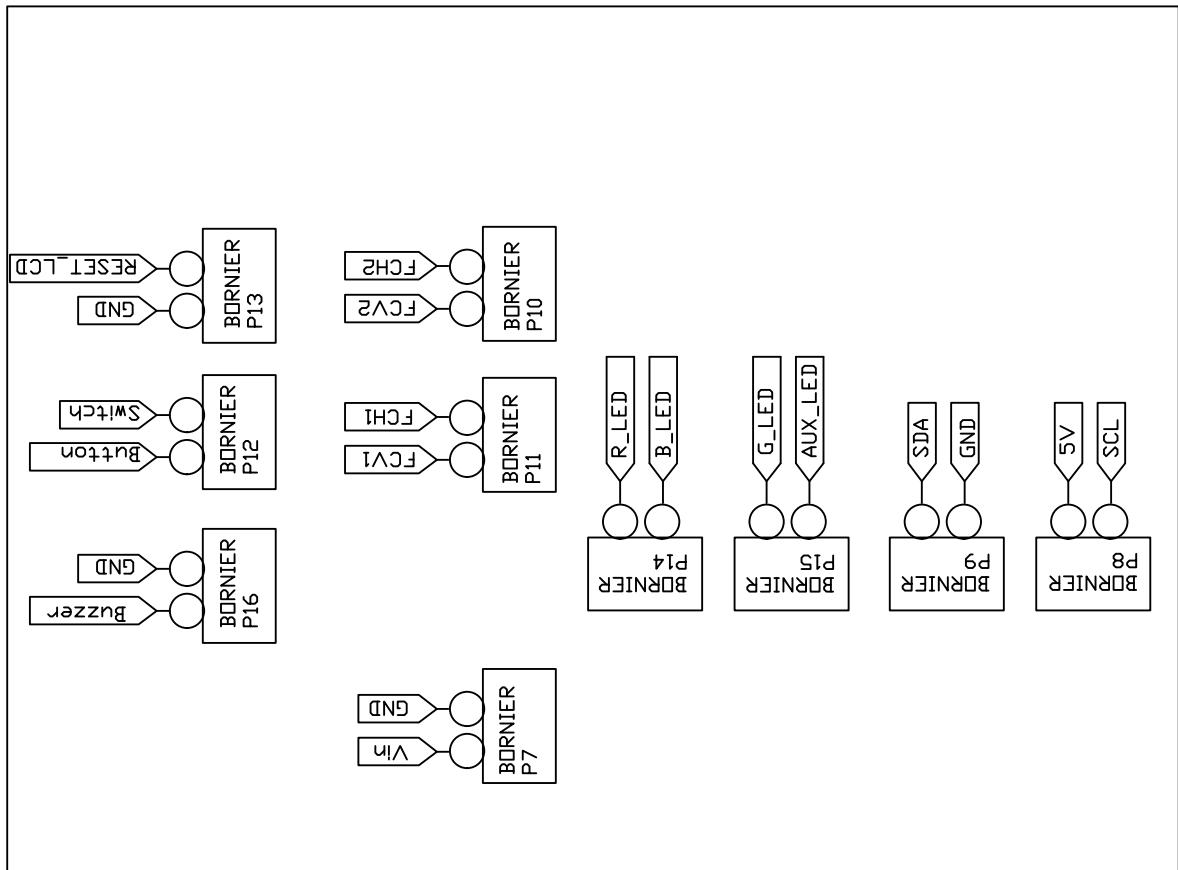
9 8 8 7 7 6 6 5 5 4 4 3 3 2 2 1 1 0 0

ଶ୍ରୀ କମଳାନାଥ ପାତ୍ର

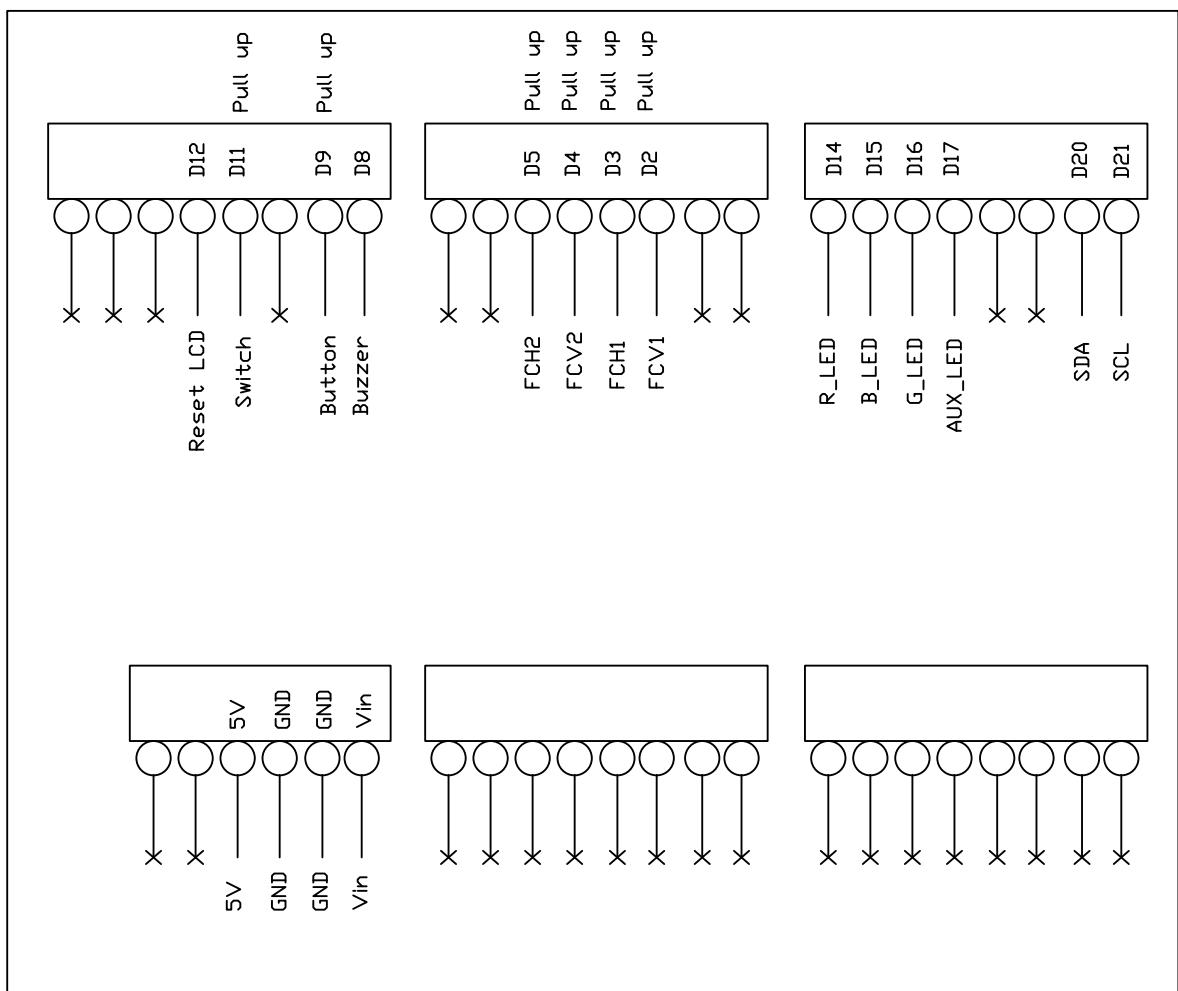
Project:	Sheet Title:	
	AC Installation	Spain
Date	Document:	Cabinet 2
Drawn	Company	BQ
Checked	Page	1
Revised	Page	2



## ARDUINO MEGA SHIELD



## ARDUINO MEGA



Revision

Date	Drawn	20/10/15	M.R.S
Date	Checked	20/10/15	J.A.B

Project  
ZOWI CALIBRATION

Document: Cabinet 2  
Company: BQ

Location: Spain  
Page: 3  
Next page: 4

Document: Cabinet 2  
Company: BQ

A 8

B 7

C 6

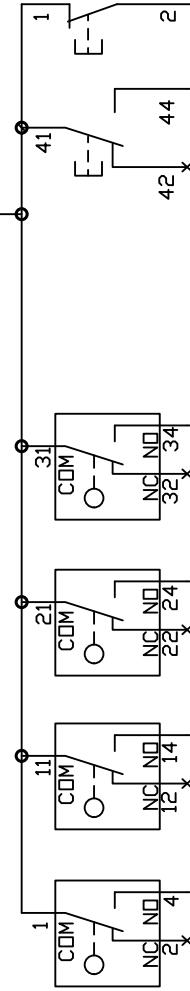
D 5

E 4

F 3

**BØRNIER**  
13**RESET\_LCD**

GND

**SWITCH BUTTON****BØRNIER**  
P12**FCV2****BØRNIER**  
P10**FCV1****BØRNIER**  
P11Document: **Cabinet 2**

Location:

Spain

Project:

ZWI CALIBRATION

Company:

BQ

Sheet Title:

DI Signals

Page:

4

Next Page

A

B

C

D

E

F

A

B

C

D

E

F

6

4

2

3

1

0

A

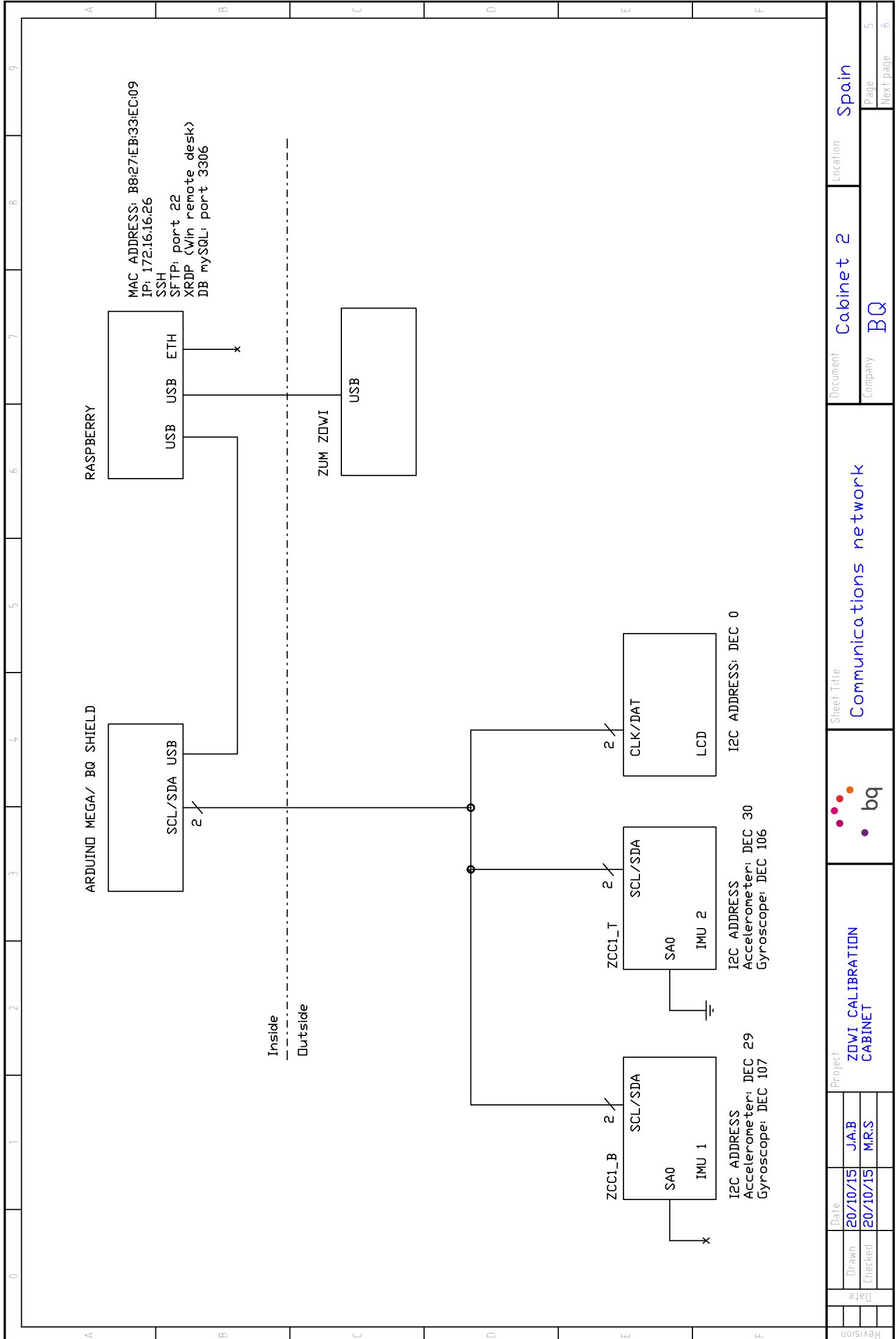
B

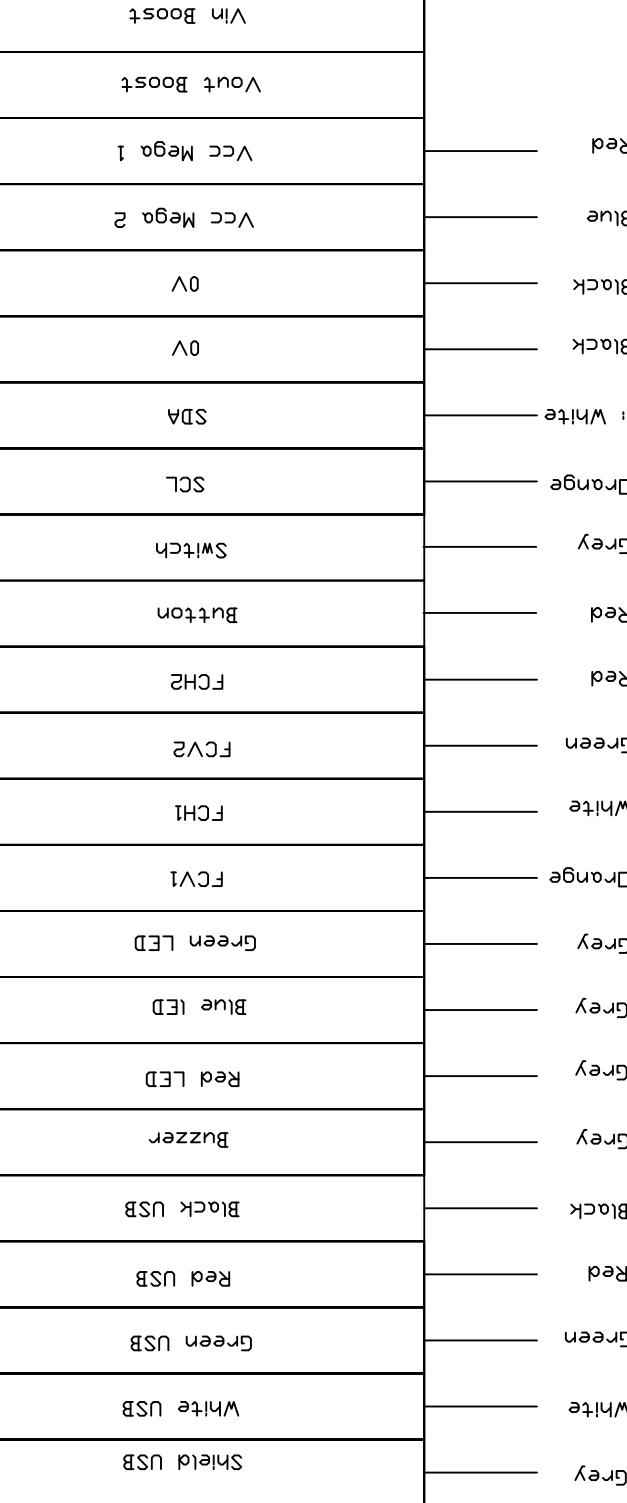
C

D

E

F

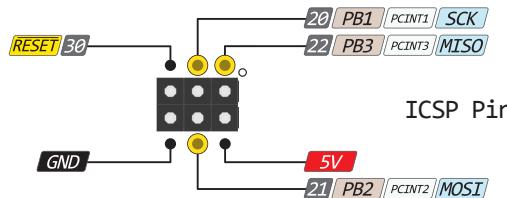


A	B	C	D	E	F																		
9																							
8																							
7																							
6																							
5																							
4																							
3																							
2																							
1																							
0																							
A	B	C	D	E	F																		
																							
																							
<p><b>Sheet Title:</b> Bornier connections</p> <p><b>bq</b></p> <table border="1"> <tr><td>Project</td><td>ZWI CALIBRATION CABINET</td></tr> <tr><td>Date Drawn</td><td>20/10/15</td></tr> <tr><td>Checked</td><td>20/10/15</td></tr> <tr><td>M.R.S.</td><td></td></tr> <tr><td>Document</td><td>Cabinet 2</td></tr> <tr><td>Company</td><td>BQ</td></tr> <tr><td>Location</td><td>Spain</td></tr> <tr><td>Page</td><td>6</td></tr> <tr><td>Next page</td><td></td></tr> </table>						Project	ZWI CALIBRATION CABINET	Date Drawn	20/10/15	Checked	20/10/15	M.R.S.		Document	Cabinet 2	Company	BQ	Location	Spain	Page	6	Next page	
Project	ZWI CALIBRATION CABINET																						
Date Drawn	20/10/15																						
Checked	20/10/15																						
M.R.S.																							
Document	Cabinet 2																						
Company	BQ																						
Location	Spain																						
Page	6																						
Next page																							
																							

## Appendix D

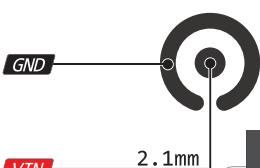
# Mega PINOUT

# MEGA PINOUT



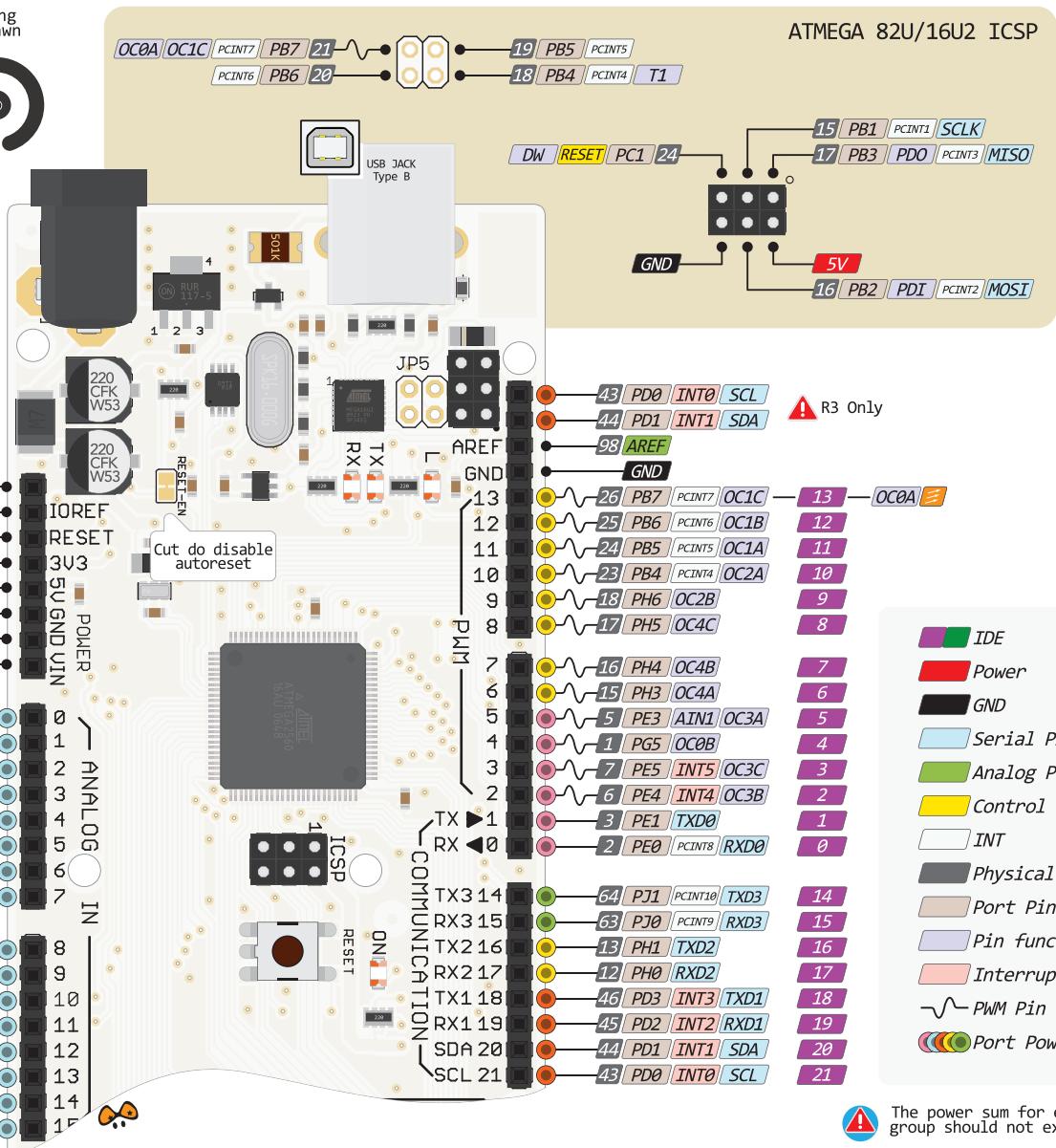
## ICSP Pinout

 7-12V Depending  
on current drawn

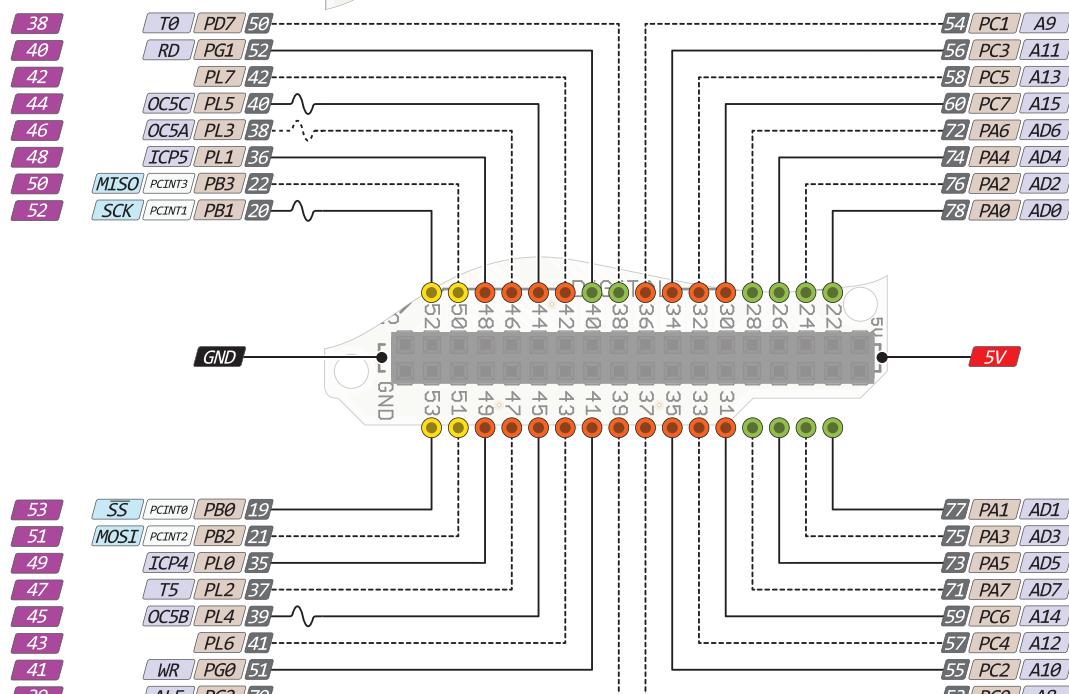


VTN

The input voltage to the board when it is running from external power.  
Not USB bus power.



The power sum for each pin's group should not exceed 100mA



 Absolute MAX per pin 20mA  
recommended 10mA

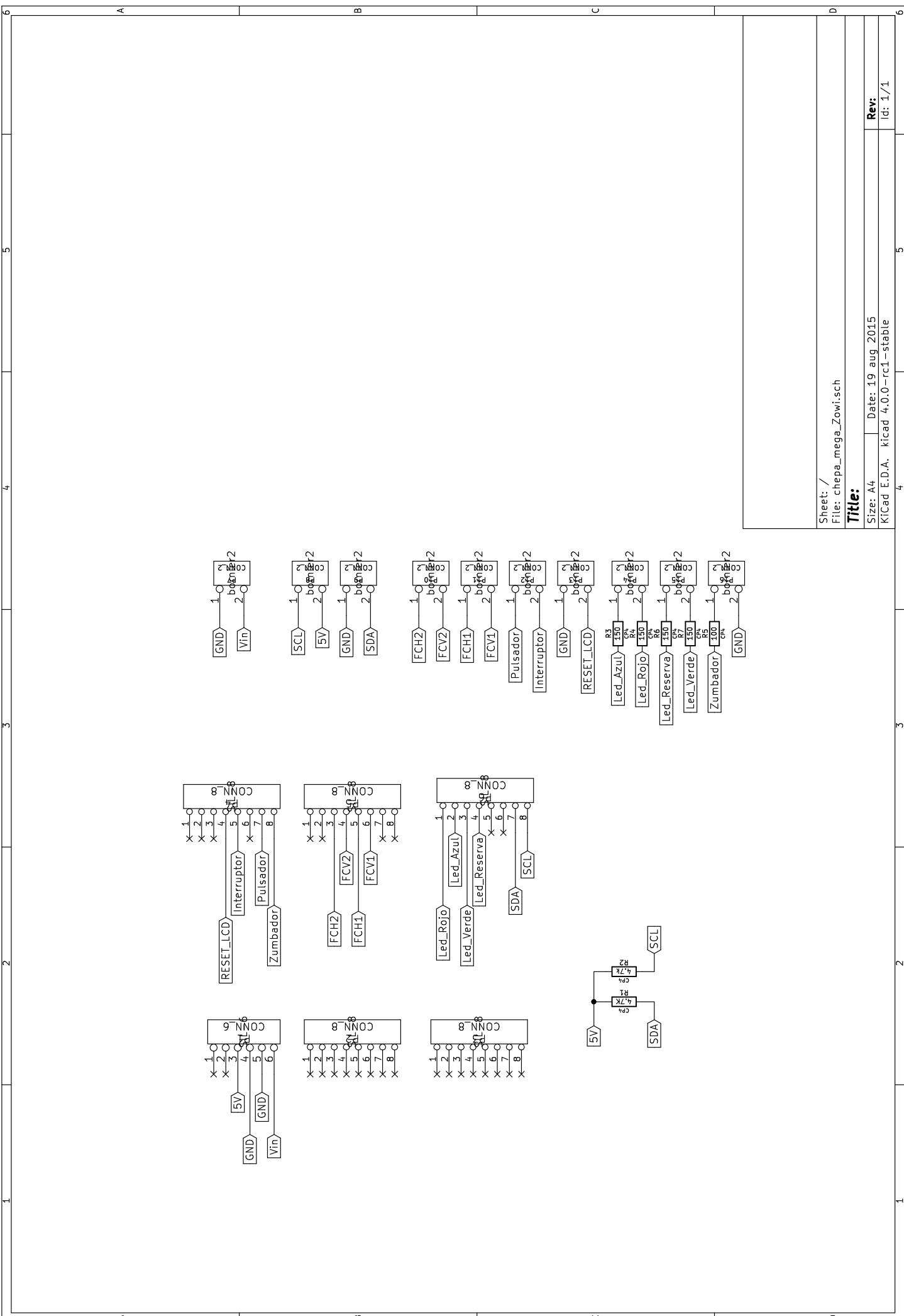
 Absolute MAX 200mA  
for entire package

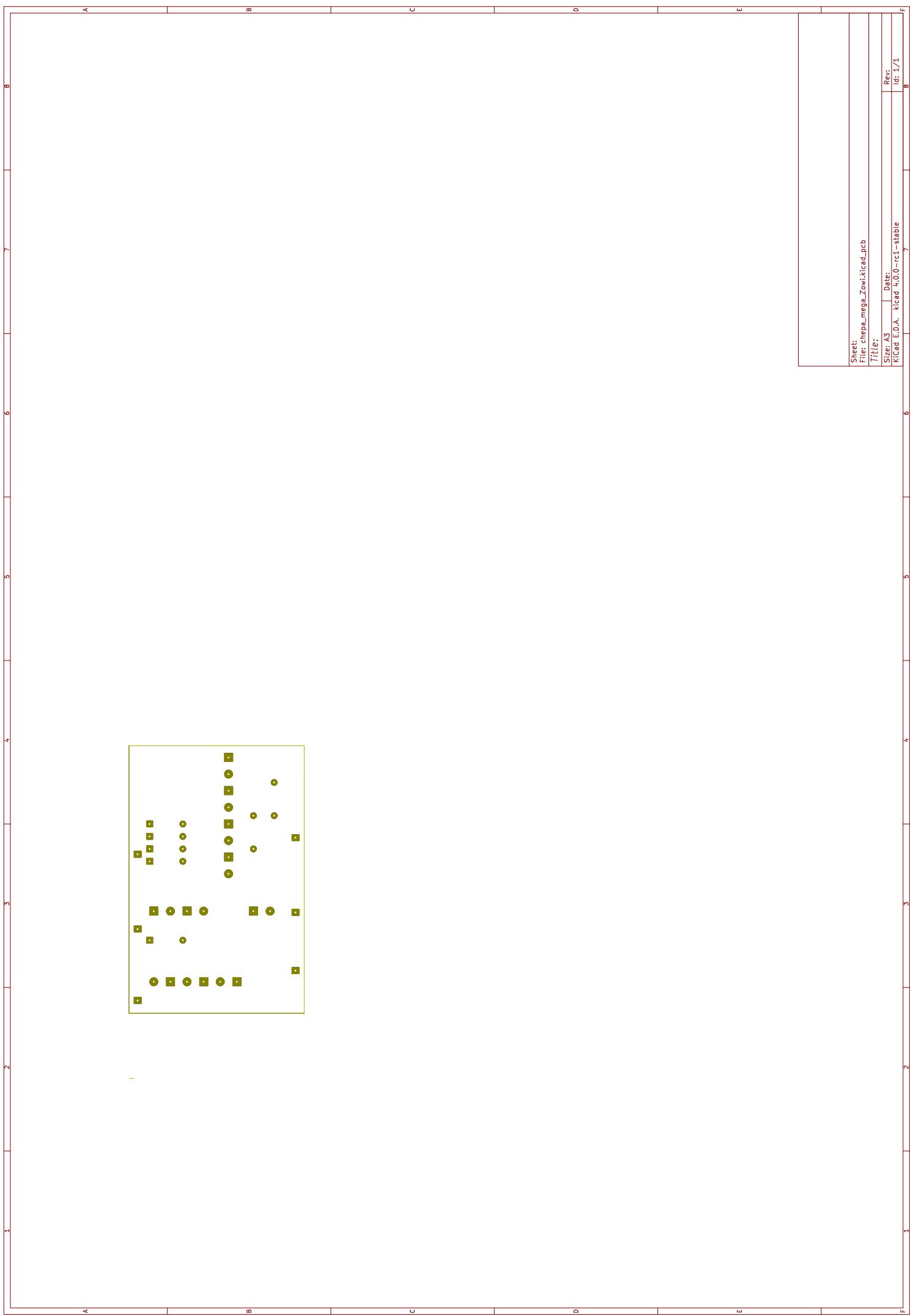
bq  
www.bq.com

## Appendix E

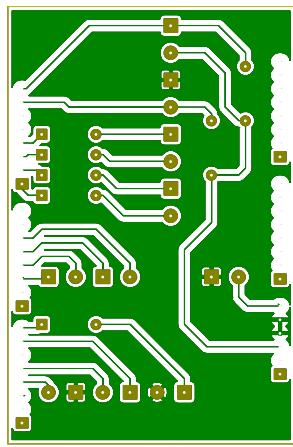
# Mega Custom Shield

Esquemático y gerbers del shield diseñado para Mega.

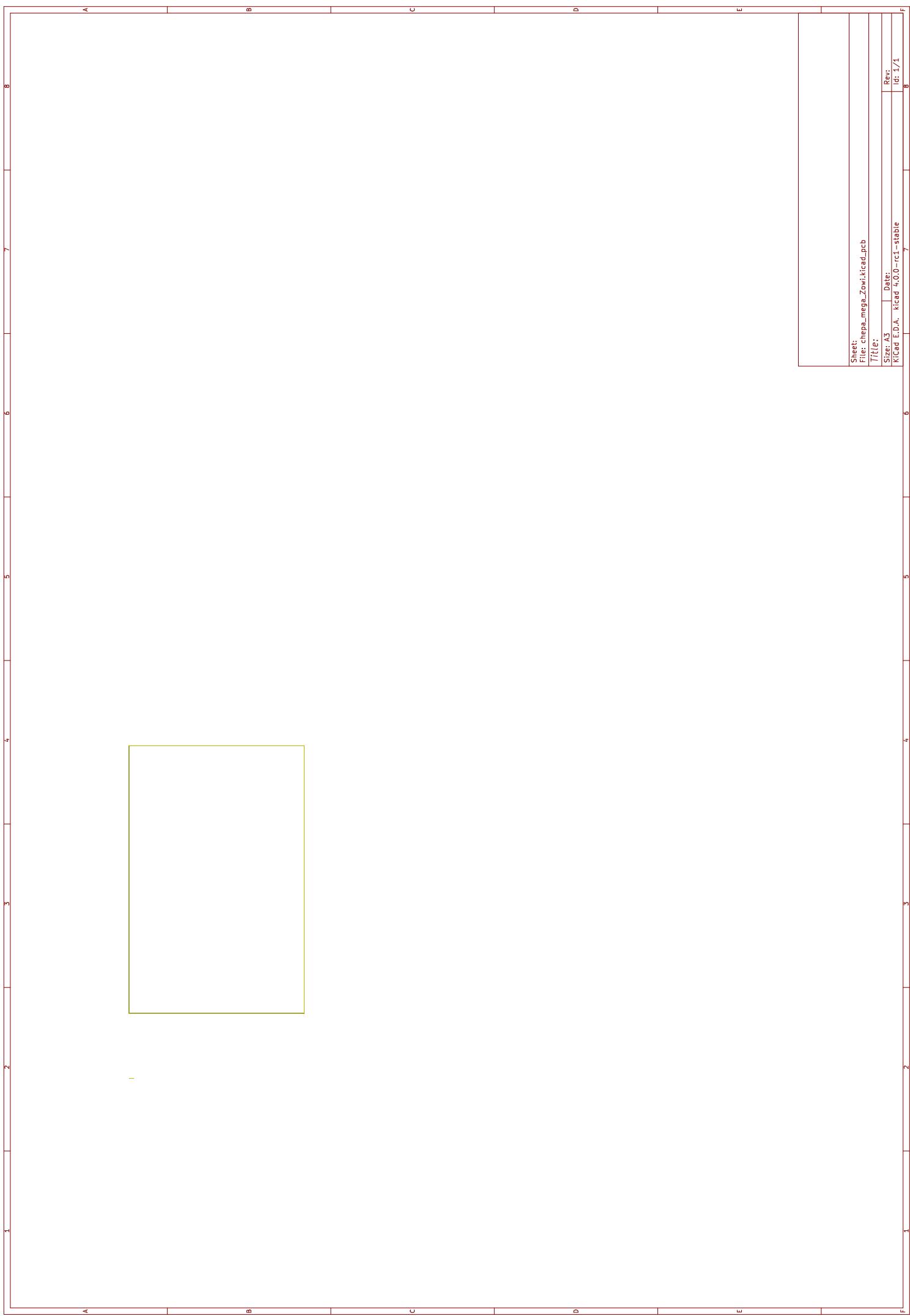




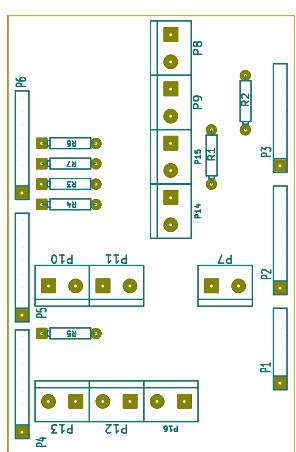
Sheet:  
File: chpa\_mega\_Zow.kicad\_pcb  
Title:  
Size: A3 Date: --/--/--  
KiCad E.D.A. KiCad 4.0.0 -rc1-stable Rev: 1/1

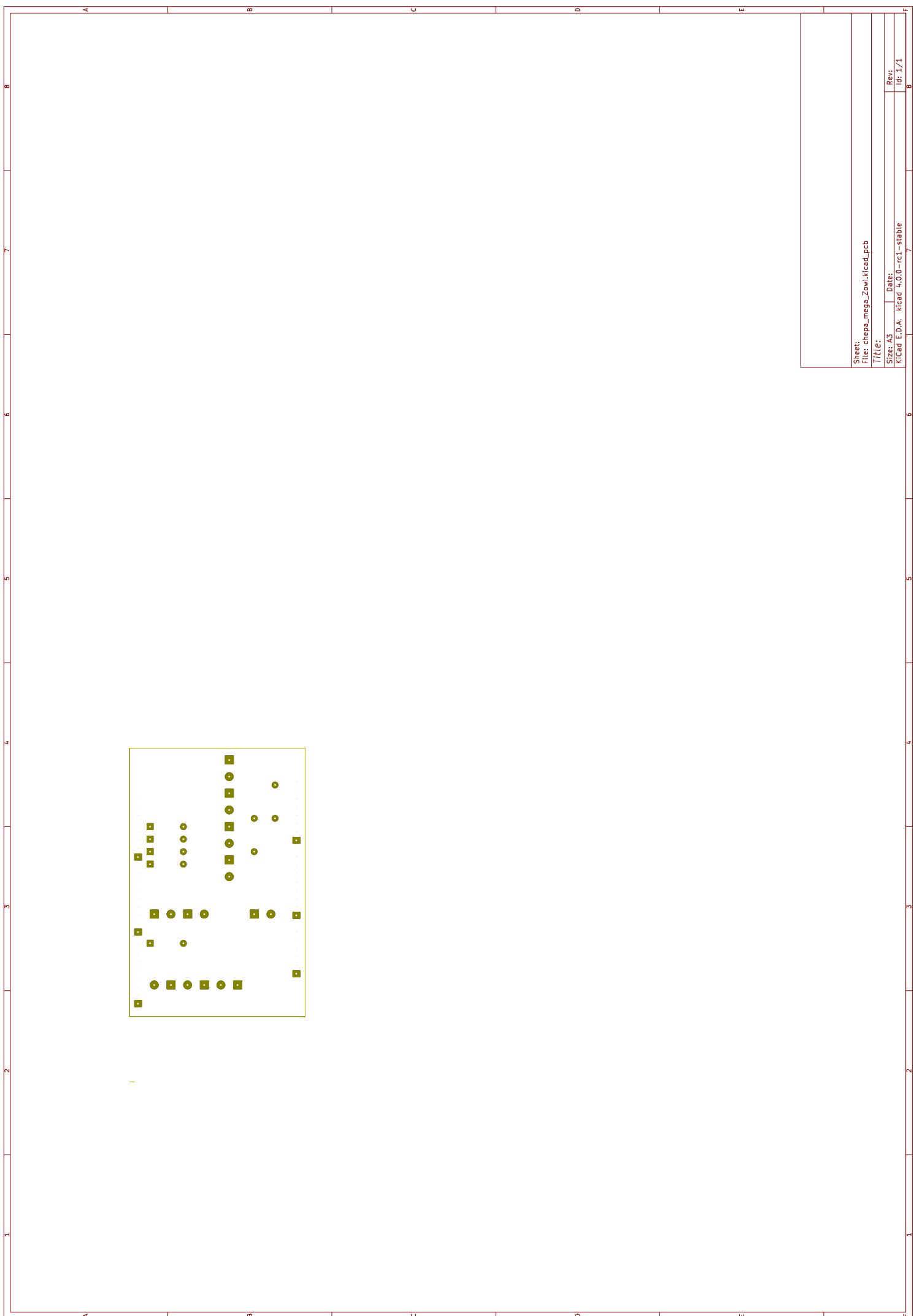


Sheet:  
File: chpa\_mega\_20.kicad\_pcb  
Title:  
Size: A3 Date:  
KiCad EDA, kicad 4.0.0 -rc1-stable Rev:  
Id: 1 / 1

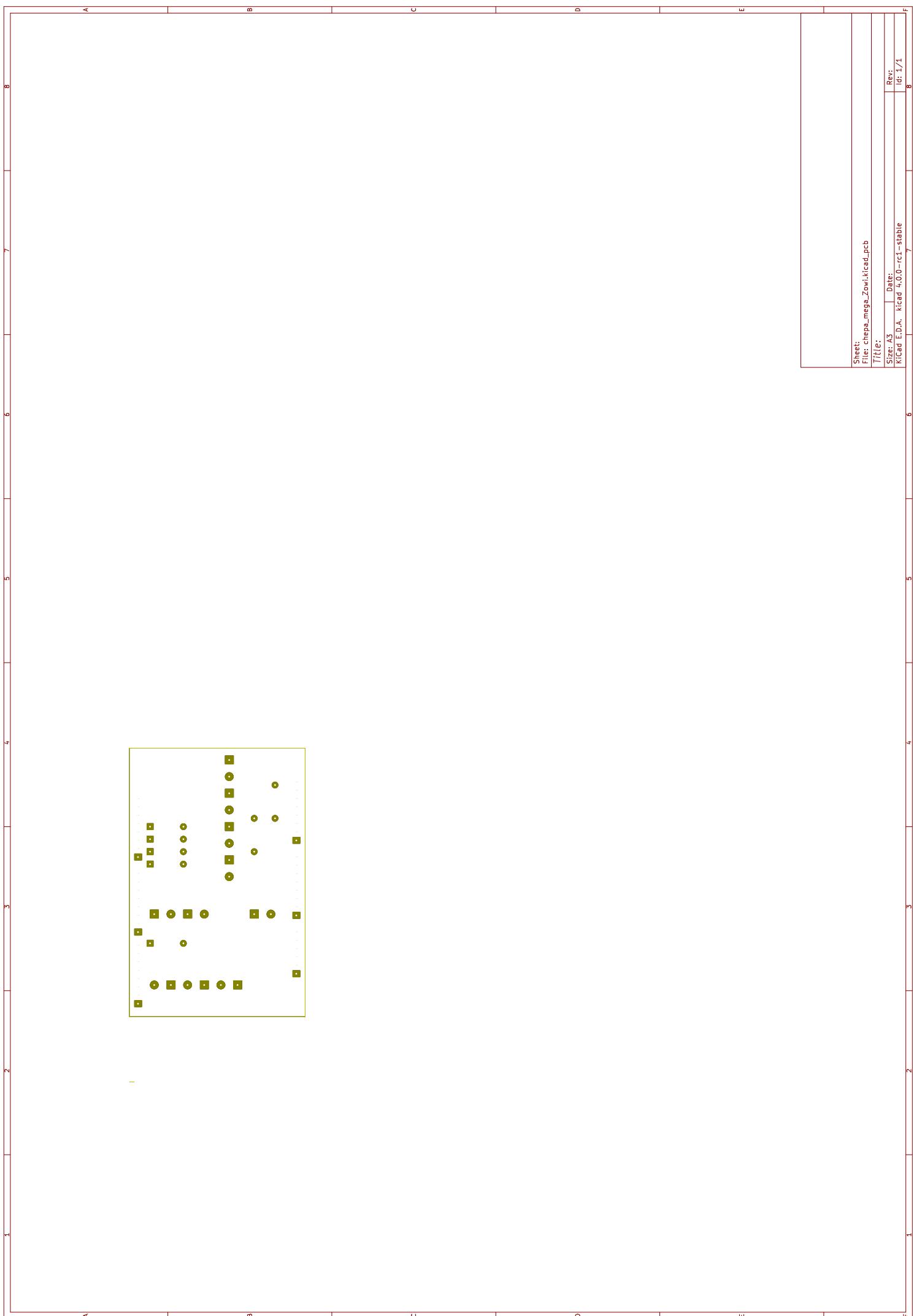


Sheet:  
File: chpa\_mega\_Zowi.kicad\_pcb  
Title:  
Size: A3 Date:  
KiCad E.D.A. kicad 4.0.0 -rc1-stable Rev:  
Id: 1 / 1

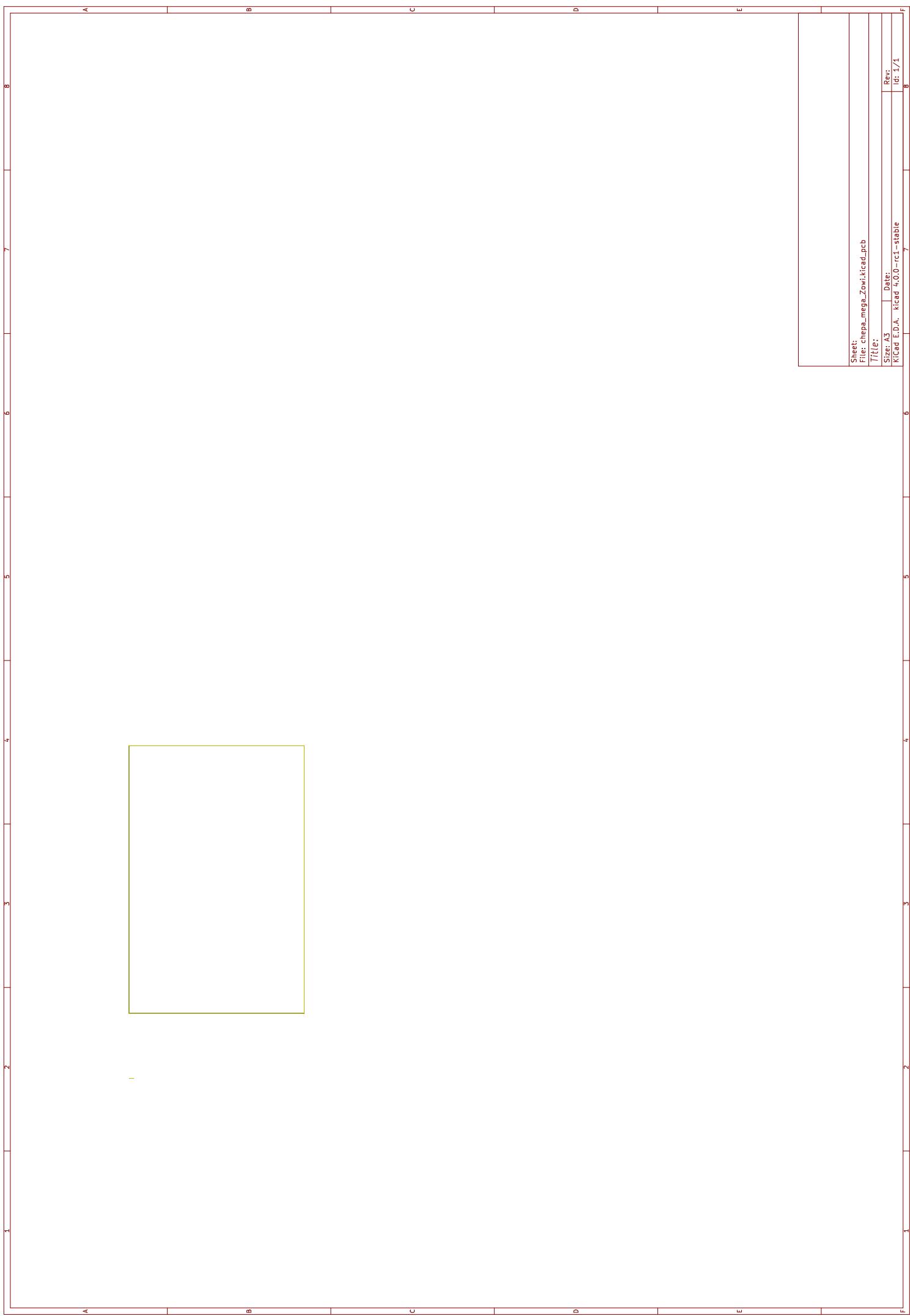




Sheet:  
File: chpa\_mega\_Zow.kicad\_pcb  
Title:  
Size: A3 Date: --/--/--  
KiCad E.D.A. KiCad 4.0.0 -rc1-stable Rev: 1/1



Sheet:  
File: chpa\_mega\_Zow.kicad\_pcb  
Title:  
Size: A3 Date: --/--/--  
KiCad E.D.A. KiCad 4.0.0 -rc1-stable Rev: 1/1

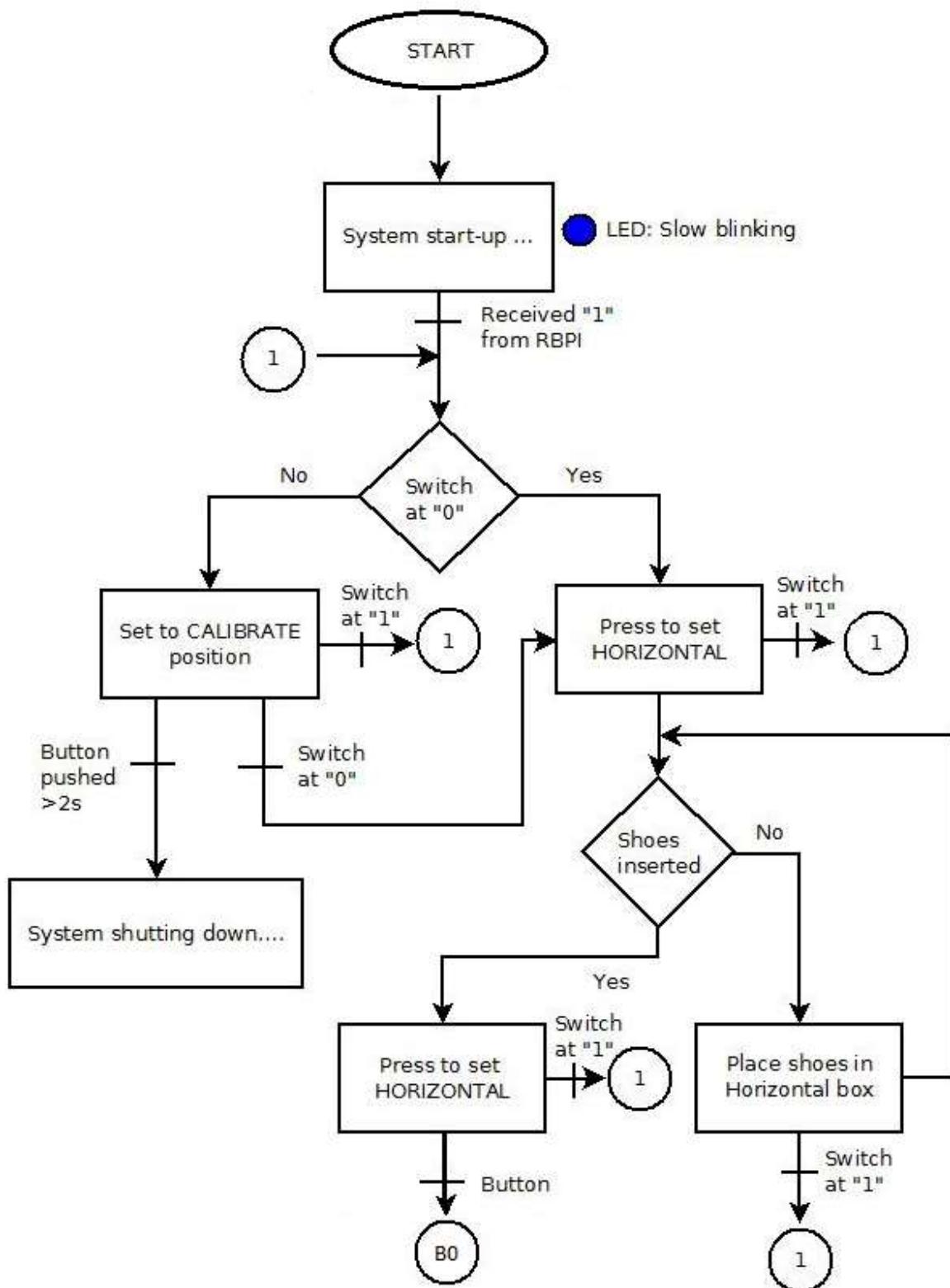


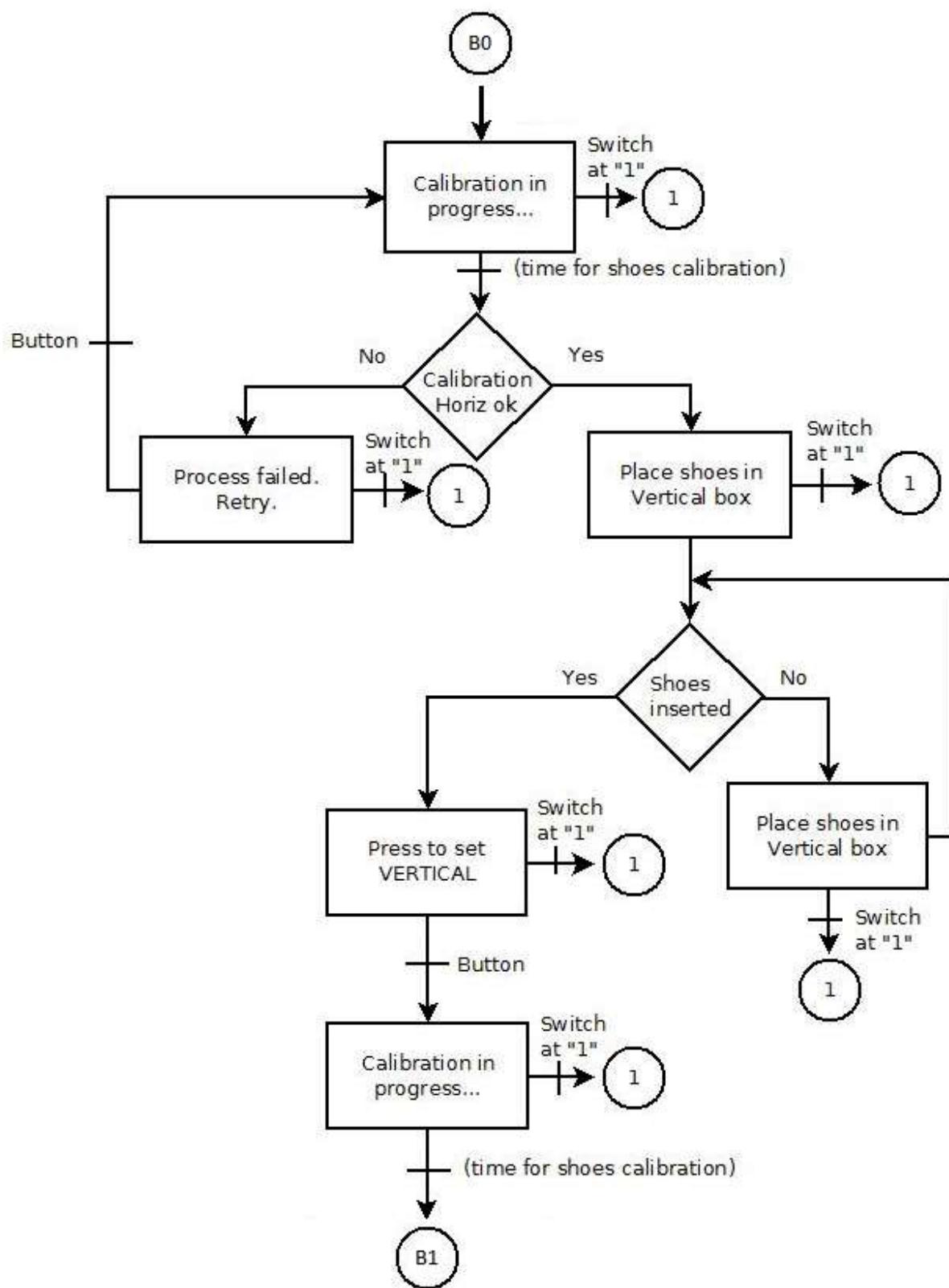


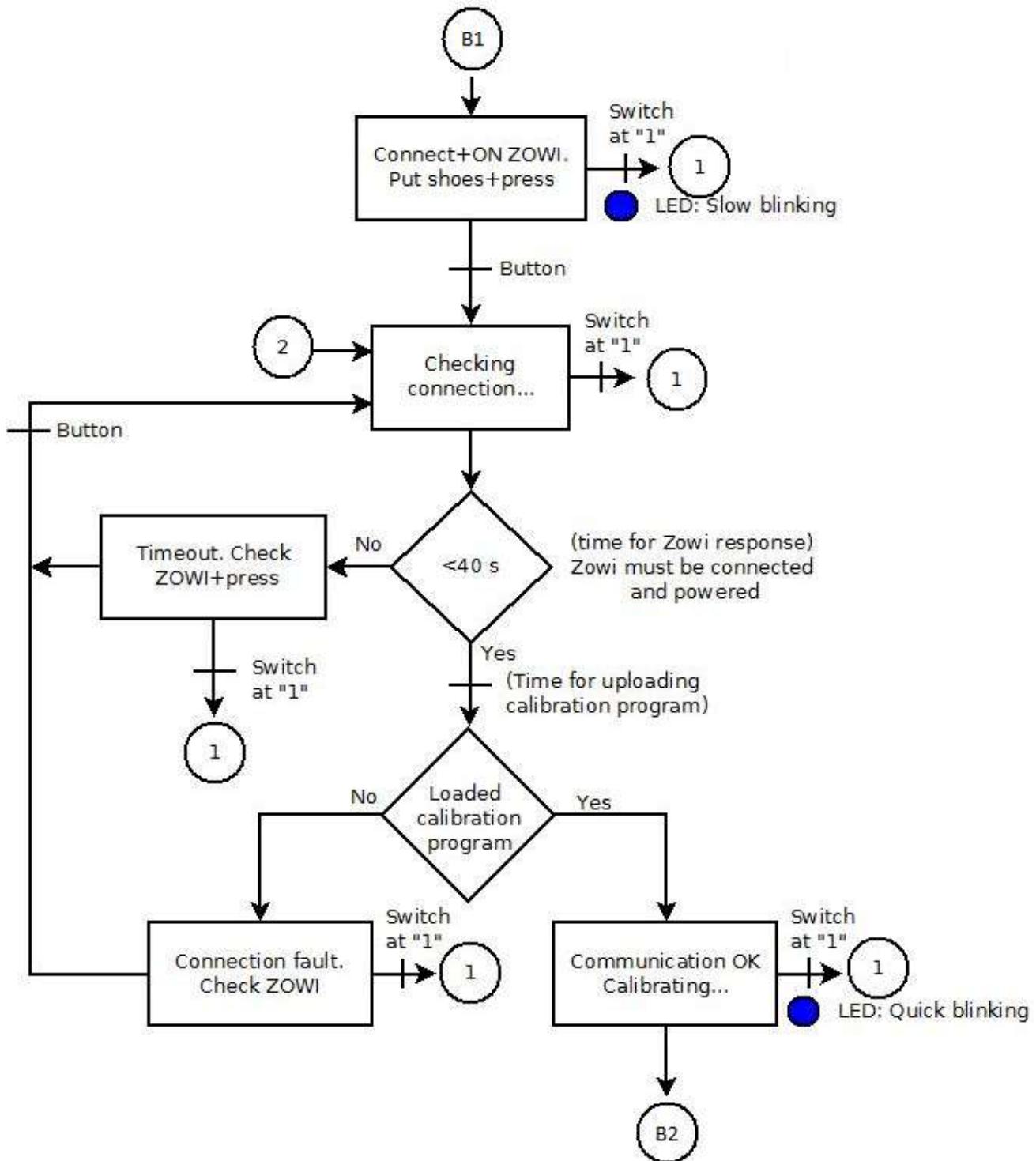
## Appendix F

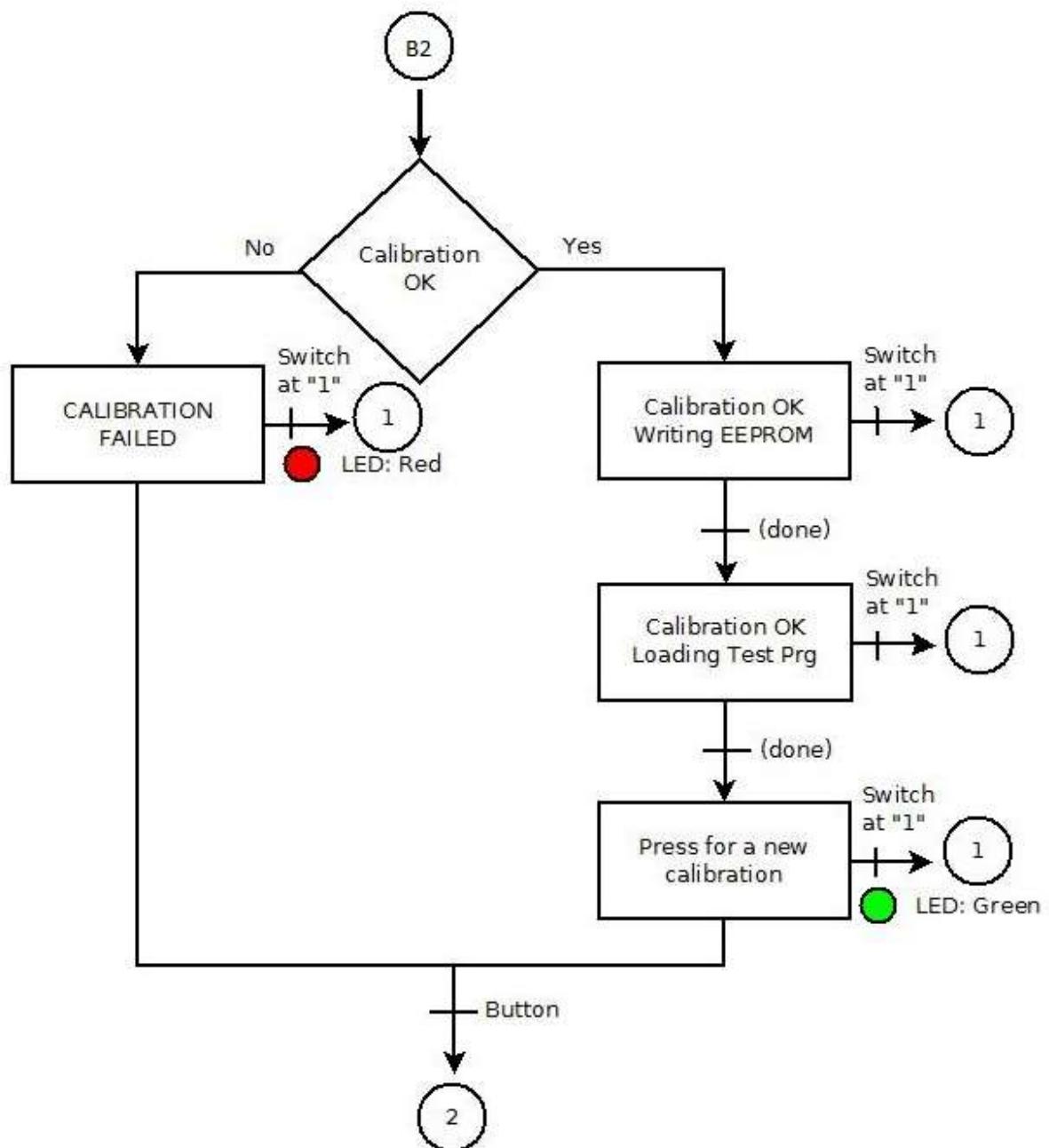
# Máquina de estados Mega

Diagrama de la máquina de estados implementada en Mega.











## Appendix G

# Configuración IMUS

Valores máximos y mínimos de la calibración de las IMUs instaladas en los dos armarios. Además de información de sus pines SA0, selector de direcciones de esclavo I<sup>2</sup>C.

## IMUS Zowi Calibration Cabinet

### **ZCC1\_T.**

min: { -3212, -3273, -2653} max: { +3505, +3260, +4601}

sa0 - gnd      bench: 1      foot: Top

### **ZCC2\_T.**

min: { -2566, -2723, -2993} max: { +5121, +3067, +3483}

sa0 - gnd      bench: 2      foot: Top

### **ZCC1\_B.**

min: { -2666, -3446, -4782} max: { +4625, +3695, +4383}

sa0 - no gnd      bench: 1      foot: Bottom

### **ZCC2\_B.**

min: { -3325, -3341, -4089} max: { +2887, +3480, +2803}

sa0 - no gnd      bench: 2      foot: Bottom

## Appendix H

# Estructura base de datos

En este documento se muestra la estructura de la base de datos, además de las direcciones de la EEPROM de Zowi donde se guardan las posiciones "offset" de los servos.

Zum EEPROM (Zowi):

[0]	[1]	[2]	[3]	[4]	[5]	[6]	
new home for left hip	new home for right hip	new home for left foot	new home for right foot	empty	1st letter for custom robot name (default value #)	2nd letter for custom robot name (default empty)	...

Database table columns (local and remote have same structure):

id	timestamp	serial_number	e_left_hip	e_right_hip	e_left_foot	e_right_foot	state	h_left_hip	h_right_hip	h_left_foot	h_right_foot
id	time stamp	Zum SN	calibration joint error	calibration joint error	calibration joint error	calibration joint error	OK/ NOK	new home position	new home position	new home position	new home position

## Appendix I

# BOM de los armarios finales

Desglose de precios de los componentes del sistema final, para ambos armarios.

<a href="http://www.farnell.es">www.farnell.es</a>			[€] Unitary cost	[€] Cost
Quantity	Ref	Description		
2	1123635	Fibox 398x298x182	88.16	176.32
2	1123670	Fibox EKIV43 placa montaje	10.16	20.32
45	149295	Entelec Uk 010500220 Terminal block, 2 pos, 10 awg	0.772	34.74
5	147494	Entelec uk 011836816, end plate	0.484	2.42
4	1750512	Entelec UK 1SNK50515R0000 ground block 4mm	3.25	13.00
2	852156	Canalización pvc 25x40, 2m	9.96	19.92
2	3094704	Legrand 0428 power socket, 10A/16A,250V,2P+E, din	16.37	32.74
2	2461029	Raspberry-pi 2, 1GB ram	32.87	65.74
2	2427499	Stontronics T5582DV, power adaptor, euoro,uk, 90V,264V, 10W,5V,2A	7.11	14.22
2	1516058	Multicomp JR-101-1-FRSG-02 INLET, IEC, with fuse holder	6.94	13.88
2	9667725	Bulgin PX0833socket (ethernet mount panel)	26.22	52.44
2	9652850	USB montaje panel receptaculo	17.21	34.42
2	2468268	Multicomp MC000950 cable, usb 2.0 a-micro b-male, 2m, black	2.85	5.70
1	1712543	Manguera 5 hilos, 30.5m, 22awg, 5.11mm diametro cable	48.49	48.49
10	1462823	Molex 50-57-9405 5 vias, 2.54mm	0.117	1.17
60	1777073	Conectores crimpars,26-22awg	0.112	6.72
10	2429543	Conector de Cable a Placa, Agujero Pasante, Header, 5, 2.54 mm	1.1	11.00
12	1178944	Lapp Kabel 53015130 Glandula de cable, 9mm gris	1.04	12.48
12	1178903	Contratuercas gris	0.448	5.38
2	3729497	Actuador del int, pulsador, 40mm, verde	10.27	20.54
2	3052989	Bloque de contacto 1NC, tornillo	5.93	11.86
2	1201896	Cable USB A to B 0,83m	3.1	6.20
2	1355986	Caja de empalmes IP55, gris, 60mmx40mm	1.35	2.70
4	4252019	Racor tubo a tubo 4 a 6 mm	4.23	16.92
2	4251969	Racor en T tubo,tubo,tubo 8mm	4.94	9.88
2	1671744	Cable red 2.5m, IEC a enchufe red (enchufe polonia C,E)	12.67	25.34
24	2008019	Bloque terminal PCB 3 vias, 26-12awg, 5,08mm	0.628	15.07
10	1593422	Conector placa a placa vertical, 2,54mm	0.109	1.09
50	9341315	Resistencia150 Ohm	0.0459	2.30
50	9341099	Resistencia 100Ohm	0.0459	2.30
50	9341951	Resistencia 4.7k	0.0449	2.25
10	1735362	Microinterruptor, SPDT, 10A, Long roller	2.06	20.60
			Total:	708.13

<a href="http://es.rs-online.com/">http://es.rs-online.com/</a>			[€] Unitary cost	[€] Cost
Quantity	Ref	Description		
1	528-132	Tuerca hexagonal nylon M2.5 (Bolsa 50 unidades)	3.92	3.92
1	291-329	Tornillo nylon avellanado M2.5x6 (Bolsa 100 unidades)	7	7.00
			Total:	10.92

<a href="http://www.bricogeek.com">www.bricogeek.com</a>			[€] Unitary cost	[€] Cost
Quantity	Ref	Description		
2	LCD-0003	LCD 16x2	14.5	29.00
2	PRO-0114	Adafruit LCD backpack I2C/SPI	8.95	17.90
2	PRO-0026	Conversor DC ajustable 4-25V 2A	10.4	20.80
			Total:	67.70

<a href="http://www.electronicaembajadores.com/Productos/Detalle/-1/SSACIM2/modulo-minimu-9">http://www.electronicaembajadores.com/Productos/Detalle/-1/SSACIM2/modulo-minimu-9</a>			[€]	[€]
Quantity	Ref	Description	Unitary cost	Cost
4	SSACIM2	Modulo MinIMU-9 v3	33.84	135.36
		Total:		135.36

<b>ELECROW</b>			[€]	[€]
Quantity	Ref	Description	Unitary cost	Cost
10 -		PCBs shield Arduino Mega para armario calibración Zowi	1.29	12.90 \$
		Shipping		25.56 \$
		Aduanas		25.39 €
		Total:		59.23 €
			<b>TOTAL;</b>	<b>981.35</b>