Universidad Zaragoza
1542

Sensorización inalámbrica de movimientos corporales

# ANEXOS

Claudia Gonzalo Gimeno

2019

# ÍNDICE ANEXOS

# Anexo 1. Boom de componentes

## Componentes

En las tablas 1.1 y 1.2 presentamos los parámetros de los componentes empleados en este proyecto: nombre, comentario, cantidad, manufactura y su ID, vendedor y su referencia, link, precio y precios totales.

Para cada uno de ellos se encuentra definido si es o no necesario en las diferentes opciones de montaje que se plantean: BNO055 soldado, BNO055 en módulo (Adafruit), MPU, con fuente de 5 V, con sensores y con actuadores. El ESP-WROOM-32 y la fuente de 3.3 V están siempre incluidos.

| Designator | Comment | Quanti | Actuadores | BNO_mod | BNO_sol | Manuf | Manuf_ID | MPU | Sensores |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| act_out, Fle1, Fle2, Pulso | TE Connectivity | 4 | SI, NO, NO, NO | NO | NO | TE Connectivity / AMP | 3-282836-3 | NO | NO, SI, SI, SI |
| Bat 3.7V, Bat 5v-17v | Molex 0022272021 | 2 | No,SI | SI, NO | SI, NO | Molex | 22-27-2021 | SI, NO | SI, NO |
| C1, C3 | 0.1uF | 2 | SI | SI | SI | AVX | 12065C104JAT2A | SI | SI |
| C2, C5, C12 | 10uF | 3 | SI | SI, SI, NO | SI, SI, NO | Murata Electronics | GCM31CR71C106KA64L | SI, SI, NO | SI, SI, NO |
| C4 | 1uF | 1 | SI | SI | SI | AVX | 12061C105KAT2A | SI | SI |
| C6, C7 | 0.1uF | 2 | NO | NO | SI | AVX | 08055C104K4Z2A | NO | NO |
| C8, C9 | 22pF | 2 | NO | NO | SI | AVX | 08052A220JAT2A | NO | NO |
| C10 | 6.8uF | 1 | NO | NO | SI | KEMET | C0805C682J5GACTU | NO | NO |
| C11 | 120pF | 1 | NO | NO | SI | AVX | 08055C124KAT2A | NO | NO |
| C13 | 22uF | 1 | SI | NO | NO | AVX | 12066C226KAT2A | NO | NO |
| CLK | 32,768kHz | 1 | NO | NO | SI | ABRACON | ABS07-32.768KHZ-7-T | NO | NO |
| D1 | LED | 1 | SI | SI | SI | Dialight | 598-8291-107F | SI | SI |
| D2, D3, D4 | Bourns CD0805 | 3 | SI | NO | NO | Bourns | CD1206-S01575 | NO | NO |
| FTDI | Header 1 | 1 | SI | SI | SI | Molex | 90120-0763 | SI | SI |
| U1 | Adafruit Industries 2472 | 1 | NO | SI | NO | Adafruit | 2472 | NO | NO |
| J1, J2 | Molex 90120-0762 | 2 | SI | NO, SI | NO, SI | Molex | 90120-0762 | NO, SI | NO, SI |
| L1 | 2.2uH | 1 | SI | NO | NO | Murata Electronics | LQM31PN2R2M00L | NO | NO |
| led_out | 282836-4 | 1 | SI | NO | NO | TE Connectivity | 282836-4 | NO | NO |
| M1 | Espressif Systems ESP32-WRO | 1 | SI | SI | SI | Espressif Systems | ESP-WROOM-32 (16MB) | SI | SI |
| Q1, Q2, Q3 | DMG2302UK | 3 | SI | NO | NO | Diodes Incorporated | DMG2302UK-7 | NO | NO |
| R1, R13, R14, R15, R16, R1 | 10k | 6 | SI, NO, NO, NO, NO, NO | SI, NO, NO, NO, NO, NO | SI, NO, NO, NO, NO, NO | Vishay | TNPW120610K0BEEA | SI, NO, NO, NO, NO, NO | SI |
| R4, R5, R6 | 10k | 3 | NO | NO | SI | TE Connectivity / Neohm | CPF0805B10KE | NO | NO |
| R7, R8, R2, R9, R10 | 0 | 5 | NO | NO, NO, SI, SI, NO | SI, SI, NO, NO, NO | Vishay / Dale | CRCW08050000Z0EAHP | NO, NO, NO, NO, SI | NO |
| R11,R3 | 100k | 1 | SI | NO,SI | NO,SI | Vishay | TNPW1206100KBEEN | NO,SI | NO,SI |
| R12 | 470 | 1 | SI | NO | NO | Panasonic | ERJ-P08J471V | NO | NO |
| S1 | EVQQ2S02W | 1 | SI | SI | SI | Panasonic | EVQ-Q2S02W | SI | SI |
| U2 | Bosch Tools BNO055 | 1 | NO | NO | SI | Bosch Sensortec | BNO055 | NO | NO |
| U3 | TPS62163DSGT | 1 | SI | NO | NO | Texas Instruments | TPS62163DSGT | NO | NO |
| U4 | Torex XC6210B332MR | 1 | SI | SI | SI | Torex | XC6210B332MR | SI | SI |
| V_out_act | 2828362 | 1 | SI | NO | NO | TE Connectivity | 282836-2 | NO | NO |

*Tabla 1.1: Boom de componentes parte uno*

| Designator | V5 | Vendedor | Vendedor_link | Vendedor_ref | Precio_individua | Canti | Precio_total en euros |
|---|---|---|---|---|---|---|---|
| | | | | | | | 0 |
| act_out, Fle1, Fle2, Pulso | NO | Mouser | https://www.mouser.es/ProductDetail/TE-Connectivit | 571-3-282836-3 | 1,67 | 4 | 6,68 |
| Bat 3.7V, Bat 5v-17v | NO,SI | Mouser | https://www.mouser.es/ProductDetail/Molex/22-27-2 | 538-22-27-2021 | 0,236 | 2 | 0,472 |
| C1, C3 | SI | Mouser | https://www.mouser.es/ProductDetail/AVX/12065C10 | c581-12065C104J, 581-12065C104J | 0,314 | 2 | 0,628 |
| C2, C5, C12 | SI | Mouser | https://www.mouser.es/ProductDetail/Murata-Electro | 81-GCM31CR71C106KA4L | 1,03 | 3 | 3,09 |
| C4 | SI | Mouser | https://www.mouser.es/ProductDetail/AVX/12061C10 | 581-12061C105KAT2A | 0,419 | 1 | 0,419 |
| C6, C7 | NO | Mouser | https://www.mouser.es/ProductDetail/AVX/08055C10 | 581-08055C104K4Z2A | 0,236 | 2 | 0,472 |
| C8, C9 | NO | Mouser | https://www.mouser.es/ProductDetail/AVX/08052A22 | 581-08052A220JAT2A | 0,236 | 2 | 0,472 |
| C10 | NO | Mouser | https://www.mouser.es/ProductDetail/KEMET/C0805( | 80-C0805C682J5G | 0,576 | 1 | 0,576 |
| C11 | NO | Mouser | https://www.mouser.es/ProductDetail/AVX/08055C12 | 581-08055C124KAT2A | 0,245 | 1 | 0,245 |
| C13 | SI | Mouser | https://www.mouser.es/ProductDetail/AVX/12066C22 | 581-12066C226KAT2A | 0,62 | 1 | 0,62 |
| CLK | NO | Mouser | https://www.mouser.es/ProductDetail/ABRACON/ABS | 815-ABS07-32.768K7T | 0,908 | 1 | 0,908 |
| D1 | SI | Mouser | https://www.mouser.es/ProductDetail/Dialight/598-8 | 645-598-8291-107F | 0,48 | 1 | 0,48 |
| D2, D3, D4 | NO | Mouser | https://www.mouser.es/ProductDetail/Bourns/CD120 | 652-CD1206-S01575 | 0,131 | 3 | 0,393 |
| FTDI | SI | Mouser | https://www.mouser.es/ProductDetail/Molex/90120-( | 538-90120-0763 | 0,91 | 1 | 0,91 |
| U1 | NO | Mouser | https://www.mouser.es/ProductDetail/Adafruit/2472? | 485-2472 | 30,52 | 1 | 30,52 |
| J1, J2 | SI | Mouser | https://www.mouser.es/ProductDetail/Molex/90120-( | 538-90120-0762 | 0,498 | 2 | 0,996 |
| L1 | SI | Mouser | https://www.mouser.es/ProductDetail/Murata-Electro | 81-LQM31PN2R2M00L | 0,349 | 1 | 0,349 |
| led_out | NO | Mouser | https://www.mouser.es/ProductDetail/TE-Connectivit | 571-2828364 | 0,777 | 1 | 0,777 |
| M1 | SI | Mouser | https://www.mouser.es/ProductDetail/?qs=MLItCLRb' | 356-ESPWROOM3216MB | 3,94 | 1 | 3,94 |
| Q1, Q2, Q3 | NO | Mouser | https://www.mouser.es/ProductDetail/Diodes-Incorpo | 621-DMG2302UK-7 | 0,288 | 3 | 0,864 |
| R1, R13, R14, R15, R16, R17 | SI, NO, NO, NO, NO, NO | Mouser | https://www.mouser.es/ProductDetail/Vishay/TNPW1 | 71-TNPW120610K0BEEA | 0,533 | 6 | 3,198 |
| R4, R5, R6 | NO | Mouser | https://www.mouser.es/ProductDetail/TE-Connectivit | 279-CPF0805B10KE | 0,271 | 3 | 0,813 |
| R7, R8, R2, R9, R10 | NO | Mouser | https://www.mouser.es/ProductDetail/Vishay-Dale/CI | 71-CRCW08050000Z0EAH | 0,21 | 5 | 1,05 |
| R11,R3 | SI | Mouser | https://www.mouser.es/ProductDetail/Vishay/TNPW1 | 71-TNPW1206100KBEEN | 0,725 | 1 | 0,725 |
| R12 | NO | Mouser | https://www.mouser.es/ProductDetail/Panasonic/ERJ | 667-ERJ-P08J471V | 0,148 | 1 | 0,148 |
| S1 | SI | Mouser | https://www.mouser.es/ProductDetail/Panasonic/EVC | 667-EVQ-Q2S02W | 0,262 | 1 | 0,262 |
| U2 | NO | Mouser | https://www.mouser.es/ProductDetail/Bosch-Sensort | 262-BNO055 | 9,25 | 1 | 9,25 |
| U3 | SI | Mouser | https://www.mouser.es/ProductDetail/Texas-Instrume | 595-TPS62163DSGT | 1,75 | 1 | 1,75 |
| U4 | SI | Farnel | https://es.farnell.com/torex/xc6210b332mr/reg-tensi- | 1057803 | 0,518 | 1 | 0,518 |
| V_out_act | NO | Mouser | https://www.mouser.es/ProductDetail/TE-Connectivit | 571-2828362 | 0,777 | 1 | 0,777 |
| | | | | | | | |
| | | | | | Precio_total de todos los cor | | 72,302 |

*Tabla1.2: Boom de componentes parte dos*

En el Anexo 2 presentamos los diagramas de bloques y explicaciones de los mencionados montajes.

## Montaje módulo Adafruit

En la tabla 1.3 presentamos los componentes necesarios para implementar el montaje del proyecto con el módulo de Adafruit. Esto incluye: los sensores, los actuadores, el ESP-WROOM-32, la fuente de 3.3 V y la fuente de 5 V.

Por lo tanto, no se incluyen el BNO055 soldado ni el MPU.

| Designator | Comment | Manuf | Manuf_ID | Precio_individual en euros | Cantidad | Precio_total en euros | Vendedo | Vendedor_link | Vendedor_ref |
|---|---|---|---|---|---|---|---|---|---|
| act_out, Fle1, Fle2, Pulso | TE Connectivity | TE Connectivity / AMP | 3-282836-3 | 1,67 | 4 | 6,68 | Mouser | https://www.mouser.es/ProductDetail/TE-Co | 571-3-282836-3 |
| Bat 3.7V, Bat 5v-17v | Molex 0022272021 | Molex | 22-27-2021 | 0,236 | 2 | 0,472 | Mouser | https://www.mouser.es/ProductDetail/Molex | 538-22-27-2021 |
| C1, C3 | 0.1uF | AVX | 12065C104JAT2A | 0,314 | 2 | 0,628 | Mouser | https://www.mouser.es/ProductDetail/AVX/1 | 581-12065C104J |
| C2, C5, C12 | 10uF | Murata Electronics | GCM31CR71C106KA64L | 1,03 | 3 | 3,09 | Mouser | https://www.mouser.es/ProductDetail/Murat | 81-GCM31CR71C106KA4L |
| C4 | 1uF | AVX | 12061C105KAT2A | 0,419 | 1 | 0,419 | Mouser | https://www.mouser.es/ProductDetail/AVX/1 | 581-12061C105KAT2A |
| C13 | 22uF | AVX | 12066C226KAT2A | 0,62 | 1 | 0,62 | Mouser | https://www.mouser.es/ProductDetail/AVX/1 | 581-12066C226KAT2A |
| D1 | LED | Dialight | 598-8291-107F | 0,48 | 1 | 0,48 | Mouser | https://www.mouser.es/ProductDetail/Dialigh | 645-598-8291-107F |
| D2, D3, D4 | Bourns CD0805 | Bourns | CD1206-S01575 | 0,131 | 3 | 0,393 | Mouser | https://www.mouser.es/ProductDetail/Bourn | 652-CD1206-S01575 |
| FTDI | Header 1 | Molex | 90120-0763 | 0,91 | 1 | 0,91 | Mouser | https://www.mouser.es/ProductDetail/Molex | 538-90120-0763 |
| U1 | Adafruit Industries 2472 | Adafruit | 2472 | 30,52 | 1 | 30,52 | Mouser | https://www.mouser.es/ProductDetail/Adafru | 485-2472 |
| J1, J2 | Molex 90120-0762 | Molex | 90120-0762 | 0,498 | 2 | 0,996 | Mouser | https://www.mouser.es/ProductDetail/Molex | 538-90120-0762 |
| L1 | 2.2uH | Murata Electronics | LQM31PN2R2M00L | 0,349 | 1 | 0,349 | Mouser | https://www.mouser.es/ProductDetail/Murat | 81-LQM31PN2R2M00L |
| led_out | 282836-4 | TE Connectivity | 282836-4 | 0,777 | 1 | 0,777 | Mouser | https://www.mouser.es/ProductDetail/TE-Co | 571-2828364 |
| M1 | Espressif Systems ESP32-V | Espressif Systems | ESP-WROOM-32 (16MB) | 3,94 | 1 | 3,94 | Mouser | https://www.mouser.es/ProductDetail/?qs=M | 356-ESPWROOM3216MB |
| Q1, Q2, Q3 | DMG2302UK | Diodes Incorporated | DMG2302UK-7 | 0,288 | 3 | 0,864 | Mouser | https://www.mouser.es/ProductDetail/Diode | 621-DMG2302UK-7 |
| R1, R13, R14, R15, R16, R17 | 10k | Vishay | TNPW120610K0BEEA | 0,533 | 6 | 3,198 | Mouser | https://www.mouser.es/ProductDetail/Vishay | 71-TNPW120610K0BEEA |
| R2,R10 | 0 | Vishay / Dale | CRCW08050000Z0EAHP | 0,21 | 2 | 0,42 | Mouser | https://www.mouser.es/ProductDetail/Vishay | 71-CRCW08050000Z0EAH |
| R11,R3 | 100k | Vishay | TNPW1206100KBEEN | 0,725 | 1 | 0,725 | Mouser | https://www.mouser.es/ProductDetail/Vishay | 71-TNPW1206100KBEEN |
| R12 | 470 | Panasonic | ERJ-P08J471V | 0,148 | 1 | 0,148 | Mouser | https://www.mouser.es/ProductDetail/Panas | 667-ERJ-P08J471V |
| S1 | EVQQ2S02W | Panasonic | EVQ-Q2S02W | 0,262 | 1 | 0,262 | Mouser | https://www.mouser.es/ProductDetail/Panas | 667-EVQ-Q2S02W |
| U3 | TPS62163DSGT | Texas Instruments | TPS62163DSGT | 1,75 | 1 | 1,75 | Mouser | https://www.mouser.es/ProductDetail/Texas- | 595-TPS62163DSGT |
| U4 | Torex XC6210B332MR | Torex | XC6210B332MR | 0,518 | 1 | 0,518 | Farnel | https://es.farnell.com/torex/xc6210b332mr/r | 1057803 |
| V_out_act | 2828362 | TE Connectivity | 282836-2 | 0,777 | 1 | 0,777 | Mouser | https://www.mouser.es/ProductDetail/TE-Co | 571-2828362 |
| | | | | | | | | | |
| | | | Precio total en euros (Módulo Adafruit): | | | 58,936 | | | |

*Tabla 1.3: Componentes montaje módulo Adafruit*

## Montaje módulo MPU

En la tabla 1.4 presentamos los componentes necesarios para implementar el montaje del proyecto con el módulo MPU 9250/6500. Esto incluye: los sensores, los actuadores, el ESP-WROOM-32, la fuente de 3.3 V y la fuente de 5 V.

Por lo tanto, no se incluyen el BNO055 ni el módulo de Adafruit.

| Designator | Comment | Manuf | Manuf_ID | Precio_individua | Cantidad | Precio_to | Vende | Vendedor_link | Vendedor_ref |
|---|---|---|---|---|---|---|---|---|---|
| act_out, Fle1, Fle2, Pulso | TE Connectivity | TE Connectivity / AMP | 3-282836-3 | 1,67 | 4 | 6,68 | Mouser | https://www.mouser.es/ProductDetail/TE-Conn | 571-3-282836-3 |
| Bat 3.7V, Bat 5v-17v | Molex 0022272021 | Molex | 22-27-2021 | 0,236 | 2 | 0,472 | Mouser | https://www.mouser.es/ProductDetail/Molex/2 | 538-22-27-2021 |
| C1, C3 | 0.1uF | AVX | 12065C104JAT2A | 0,314 | 2 | 0,628 | Mouser | https://www.mouser.es/ProductDetail/AVX/120 | 581-12065C104J |
| C2, C5, C12 | 10uF | Murata Electronics | GCM31CR71C106KA64L | 1,03 | 3 | 3,09 | Mouser | https://www.mouser.es/ProductDetail/Murata-I | 81-GCM31CR71C106KA4L |
| C4 | 1uF | AVX | 12061C105KAT2A | 0,419 | 1 | 0,419 | Mouser | https://www.mouser.es/ProductDetail/AVX/120 | 581-12061C105KAT2A |
| C13 | 22uF | AVX | 12066C226KAT2A | 0,62 | 1 | 0,62 | Mouser | https://www.mouser.es/ProductDetail/AVX/120 | 581-12066C226KAT2A |
| D1 | LED | Dialight | 598-8291-107F | 0,48 | 1 | 0,48 | Mouser | https://www.mouser.es/ProductDetail/Dialight/ | 645-598-8291-107F |
| D2, D3, D4 | Bourns CD0805 | Bourns | CD1206-S01575 | 0,131 | 3 | 0,393 | Mouser | https://www.mouser.es/ProductDetail/Bourns/( | 652-CD1206-S01575 |
| FTDI | Header 1 | Molex | 90120-0763 | 0,91 | 1 | 0,91 | Mouser | https://www.mouser.es/ProductDetail/Molex/9 | 538-90120-0763 |
| J1, J2 | Molex 90120-0762 | Molex | 90120-0762 | 0,498 | 2 | 0,996 | Mouser | https://www.mouser.es/ProductDetail/Molex/9 | 538-90120-0762 |
| L1 | 2.2uH | Murata Electronics | LQM31PN2R2M00L | 0,349 | 1 | 0,349 | Mouser | https://www.mouser.es/ProductDetail/Murata-I | 81-LQM31PN2R2M00L |
| led_out | 282836-4 | TE Connectivity | 282836-4 | 0,777 | 1 | 0,777 | Mouser | https://www.mouser.es/ProductDetail/TE-Conn | 571-2828364 |
| M1 | Espressif Systems ESP32-V | Espressif Systems | ESP-WROOM-32 (16MB) | 3,94 | 1 | 3,94 | Mouser | https://www.mouser.es/ProductDetail/?qs=MLI | 356-ESPWROOM3216MB |
| Q1, Q2, Q3 | DMG2302UK | Diodes Incorporated | DMG2302UK-7 | 0,288 | 3 | 0,864 | Mouser | https://www.mouser.es/ProductDetail/Diodes-I | 621-DMG2302UK-7 |
| R1, R13, R14, R15, R16, R17 | 10k | Vishay | TNPW120610K0BEEA | 0,533 | 6 | 3,198 | Mouser | https://www.mouser.es/ProductDetail/Vishay/T | 71-TNPW120610K0BEEA |
| R9 | 0 | Vishay / Dale | CRCW08050000Z0EAHP | 0,21 | 1 | 0,21 | Mouser | https://www.mouser.es/ProductDetail/Vishay-D | 71-CRCW08050000Z0EAH |
| R11,R3 | 100k | Vishay | TNPW1206100KBEEN | 0,725 | 1 | 0,725 | Mouser | https://www.mouser.es/ProductDetail/Vishay/T | 71-TNPW1206100KBEEN |
| R12 | 470 | Panasonic | ERJ-P08J471V | 0,148 | 1 | 0,148 | Mouser | https://www.mouser.es/ProductDetail/Panason | 667-ERJ-P08J471V |
| S1 | EVQQ2S02W | Panasonic | EVQ-Q2S02W | 0,262 | 1 | 0,262 | Mouser | https://www.mouser.es/ProductDetail/Panason | 667-EVQ-Q2S02W |
| U3 | TPS62163DSGT | Texas Instruments | TPS62163DSGT | 1,75 | 1 | 1,75 | Mouser | https://www.mouser.es/ProductDetail/Texas-In | 595-TPS62163DSGT |
| U4 | Torex XC6210B332MR | Torex | XC6210B332MR | 0,518 | 1 | 0,518 | Farnel | https://es.farnell.com/torex/xc6210b332mr/reg | 1057803 |
| V_out_act | 2828362 | TE Connectivity | 282836-2 | 0,777 | 1 | 0,777 | Mouser | https://www.mouser.es/ProductDetail/TE-Conn | 571-2828362 |
| MPU | MPU9250/6500 | | | | 1 | 11,73 | Amazon | https://www.amazon.es/giroscopio-Aceleraci%C3%B3n-Magn%C3%A9tica-A | |
| | | | | | | | | | |
| | | | | Precio total en euros (MPU): | | 39,936 | | | |
| | | | | | | | | | |
| | | | | Precio total en euros (sin MPU): | | 28,206 | | | |

*Tabla 1.4: Montaje módulo MPU 9250/6500*

## Componentes plan ‘B’

Como se explica en la Memoria, de cara al festival Trayectos, se realiza el diseño e implementación de una nueva placa. En la tabla 1.5 podemos observar los componentes necesarios para el montaje de una de estas placas.

| Designator | Comment | Manuf | Manuf_ID | Precio_in | Cantidad | Precio_ | Vendedor | Vendedor_ref | Vendedor_link |
|---|---|---|---|---|---|---|---|---|---|
| CONN1,CONN2,CONN3,CONN4 | Conector alimentacion,Conector,Conector,Conector | TE Connectivity | 640454-2 | 0,09 | 4 | 0,36 | Mouser | 571-640454-2 | https://www.mouser.es/ProductDetail/TE-Connectivi |
| C1,C2 | 22uF | Panasonic | EEU-FC1V220 | 0,253 | 2 | 0,506 | Mouser | 667-EEU-FC1V220 | https://www.mouser.es/ProductDetail/Panasonic/EE |
| REG1 | XP Power TR10S05 | XP POWER | TR10S05 | 3,38 | 1 | 3,38 | Farnel | 2708277 | https://es.farnell.com/xp-power/tr10s05/convertidor |
| J3 | Jumper Alimentacion | Wurth Electronics | 61300211021 | 0,3 | 1 | 0,3 | Mouser | 710-61300211021 | https://www.mouser.es/ProductDetail/Wurth-Electro |
| J1,J2 | Conector tiras, Conector Entrada Analogica | TE Connectivity / AMP | 5-146285-3 | 0,43 | 2 | 0,86 | Mouser | 571-5-146285-3 | https://www.mouser.es/ProductDetail/TE-Connectivi |
| Q1,Q2,Q3 | DMG2302UK | Wurth Electronics | 61300211021 | 0,3 | 3 | 0,9 | Mouser | 710-61300211021 | https://www.mouser.es/ProductDetail/Wurth-Electro |
| CW3 | MPU9250/6500 | | | | | 11,73 | Amazon | | https://www.amazon.es/giroscopio-Aceleraci%C3%B |
| CW1-CW2 | NodeMCU | | | | | 9,59 | Amazon | | https://www.amazon.es/SeeKool-Tablero-Desarrollo- |
| | | | | | | | | | |
| | | | | SUMA TOTAL EN EUROS: | | 27,626 | | | |

*Tabla 1.5: Componentes plan B*

# Anexo 2. Casos de implementación

El ensamblaje del prototipo se puede dividir en diferentes casos según las necesidades requeridas. Los casos (A, B, C…) se pueden superponer y así ir eligiendo y añadiendo las diferentes características.

Esto simplifica la labor de ensamblaje, pues al tener todos los parámetros ya determinados para cada caso, solo hay que elegir el, o los casos que se van a implementar, y comprar aquellos componentes cuyo valor sea 'SI'. La tabla que incluye todos los casos, componentes y valores se encuentra en el Anexo 1. Todos los casos incluyen el procesador ESP-WROOM-32 y la alimentación interna de 3.3 V. La descripción y del diagrama de bloques del circuito completo se encuentran en la Memoria.

A partir de las especificaciones técnicas, se definen los siguientes casos con sus correspondientes diagramas de bloques.

## Caso A. BNO055 soldado

Este caso se basa en usar el prototipo únicamente para enviar datos vía wifi a partir de la información recogida del sensor BNO055, que se encuentra soldado a la placa con todos sus componentes y conexiones.

Se alimenta con una batería de 3.7 V y el XC6210B332MR (U4) reduce la tensión a 3.3 V, para poder alimentar así el ESP-WROOM-32 y el sensor BNO055. No incluye los sensores de flexión y de pulso, ni los actuadores. Tampoco lo relacionado con la obtención de 5 V en la placa.

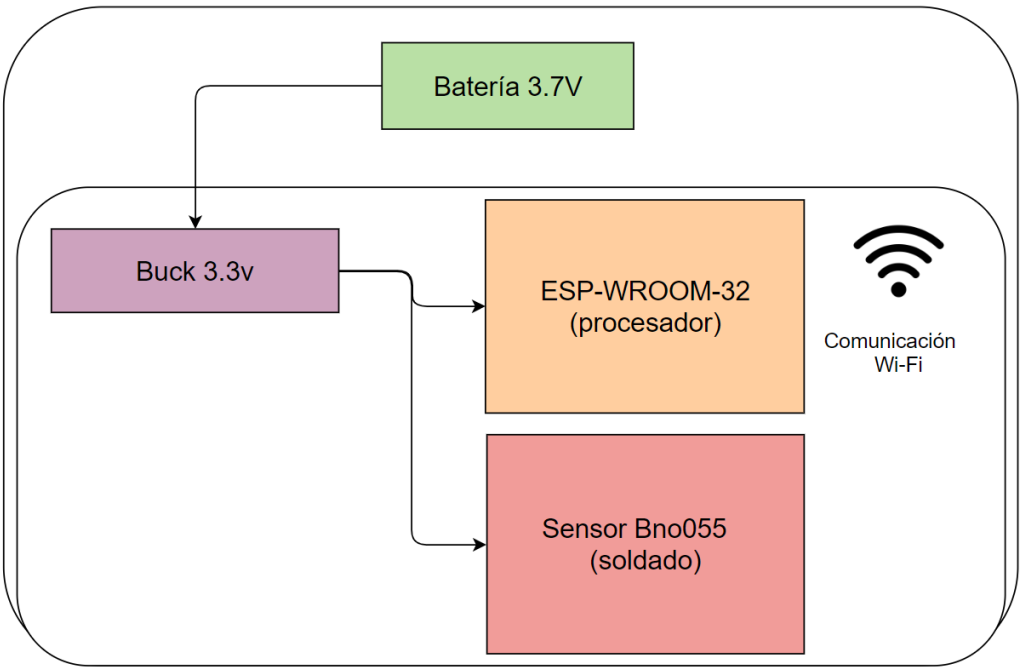



*Figura 2.1: Caso A. BNO055 soldado*

## Caso B. BNO055 en el módulo Adafruit

Este caso se basa en usar el prototipo únicamente para enviar datos vía wifi a partir de la información recogida por el módulo Adafruit Industries 2472, que incluye el sensor BNO055 y todos sus componentes.

Al igual que en el caso anterior, se alimenta con una batería de 3.7 V y el XC6210B332MR (U4) reduce la tensión a 3.3 V, para poder alimentar el ESP-WROOM-32 y el sensor BNO055. No incluye los sensores de flexión y de pulso, ni los actuadores. Tampoco lo relacionado con la obtención de 5 V en la placa.

Esta opción se creó por la dificultad de soldar el BNO055. Es más caro que el caso A, ya que el módulo cuesta aproximadamente el doble de caro que si compráramos el BNO055 y sus componentes sueltos, pero es una opción más segura.

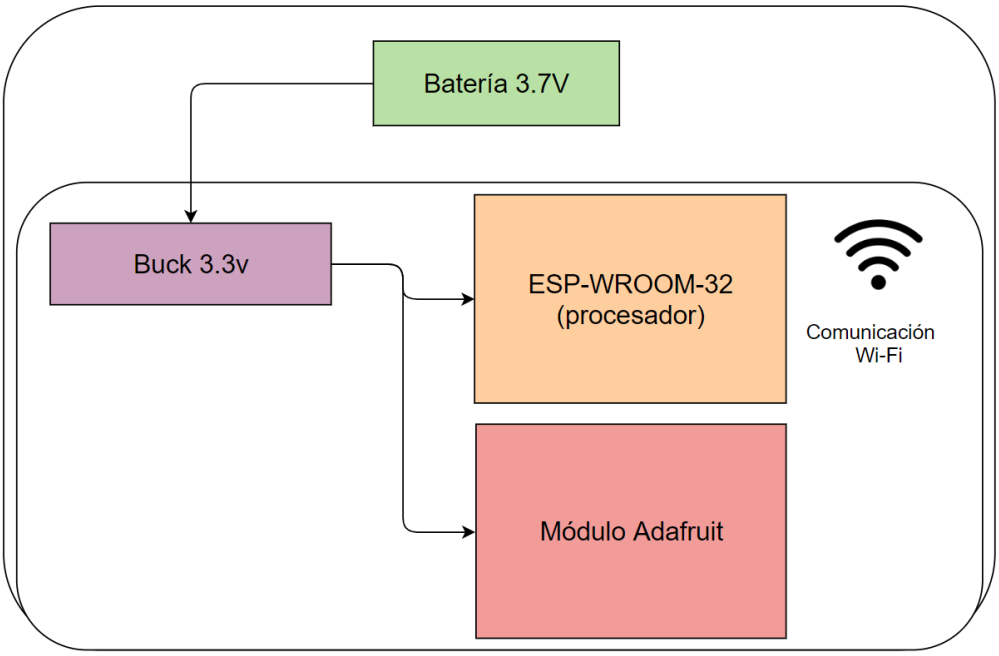



*Figura 2.2: Caso B. Módulo Adafruit*

## Caso C. MPU9250/6500

Este caso se basa en usar el prototipo únicamente para enviar datos vía wifi a partir de la información recogida por el módulo MPU9250/6500. Este módulo es mucho más barato que el del caso B, aunque presenta menor precisión.

Al igual que en los casos anteriores, se alimenta con una batería de 3.7 V y el XC6210B332MR (U4) reduce la tensión a 3.3 V, para poder alimentar el ESP-WROOM-32 junto con al sensor BNO055. No incluye los sensores de flexión y de pulso, ni los actuadores. Tampoco lo relacionado con la obtención de 5 V en la placa.

Además, este módulo ocupa menos espacio que el de los casos anteriores (A y B).

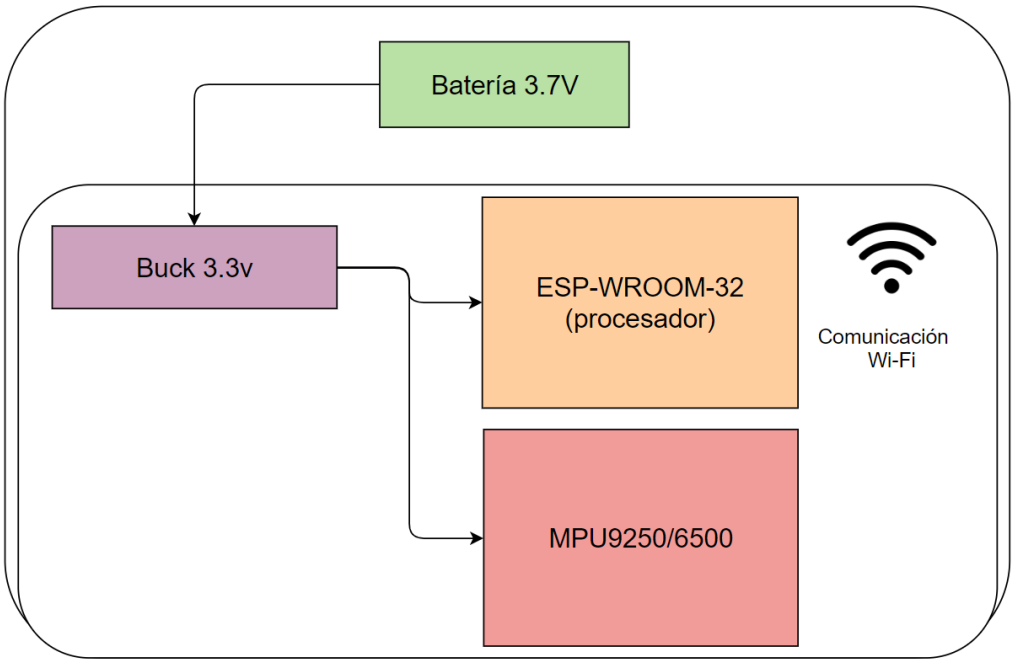



Figura 2.3: Caso C. MPU 92500/6500

## Caso D. Sensores

Se incluye lo necesario para las conexiones al exterior y el funcionamiento de los cinco sensores de flexión y el sensor de pulso. Estas señales se pueden enviar por vía wifi gracias al procesador.

Se alimenta con la batería de 3.7 V, incluyendo además todo lo necesario para poder reducir la tensión a 3.3 V y así poder alimentar el ESP-WROOM-32 y los diferentes sensores. No encontramos en este caso nada relacionado con el sensor de movimiento, los 5 V y los actuadores.

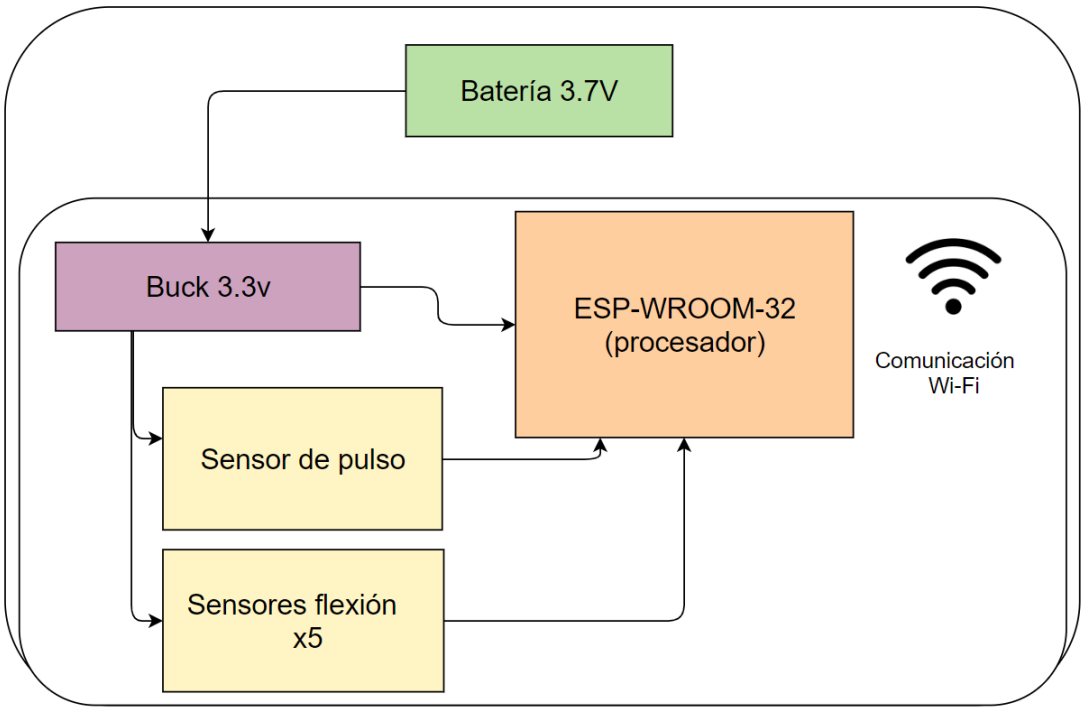



*Figura 2.4: Caso D. Sensores*

## Caso E. 5 V

Se incluye todo lo necesario para que el prototipo tenga de forma interna 5 V. El TPS62163DSGT (U3) se encarga con sus componentes relacionados de la reducción de tensión respecto a la que ofrece la batería. Mediante un jumper (J1) los 5 V que se obtienen se pasan por el XC6210B332MR (U4) y así se obtienen los 3.3 V para poder alimentar el procesador, por si se quiere enviar o recibir información.

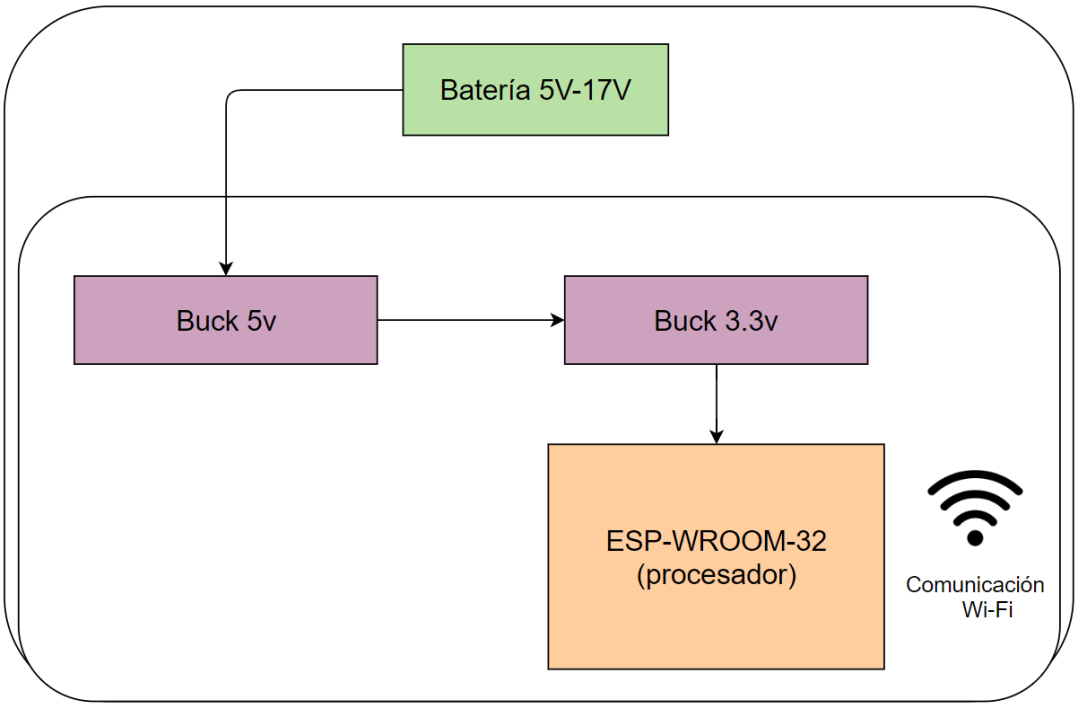



*Figura 2.5: Caso E. 5 V*

## Caso F. Actuadores

Incluye lo necesario para el correcto funcionamiento de los actuadores. Estos están alimentados con V_HIGH o 5 V, por lo tanto se incluye el TPS62163DSGT (U3) y sus componentes para obtener 5 V de forma interna. Encontramos también el jumper con el que obtener los 3.3 V y así poder alimentar el módulo ESP-WROOM-32 y usar su conectividad wifi.

Los actuadores que posee son: una tira de LED y tres salidas de propósito general dispuestas con MOFSET y diodo en antiparalelo. No incluye nada relacionado con el sensor de movimientos ni los sensores de flexión.

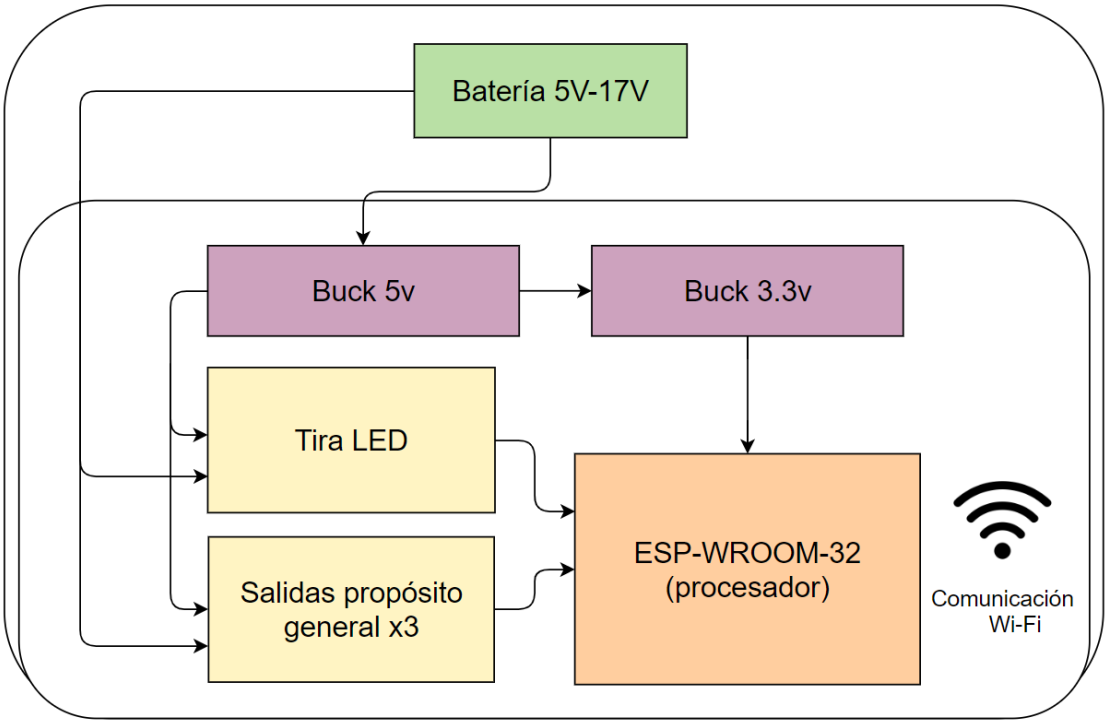



*Figura 2.7: Caso F. Actuadores*

## Anexo 3. Prints PCB

A continuación, se muestran los prints de la placa diseñada para el proyecto, realizados con la herramienta CircuitMaker.

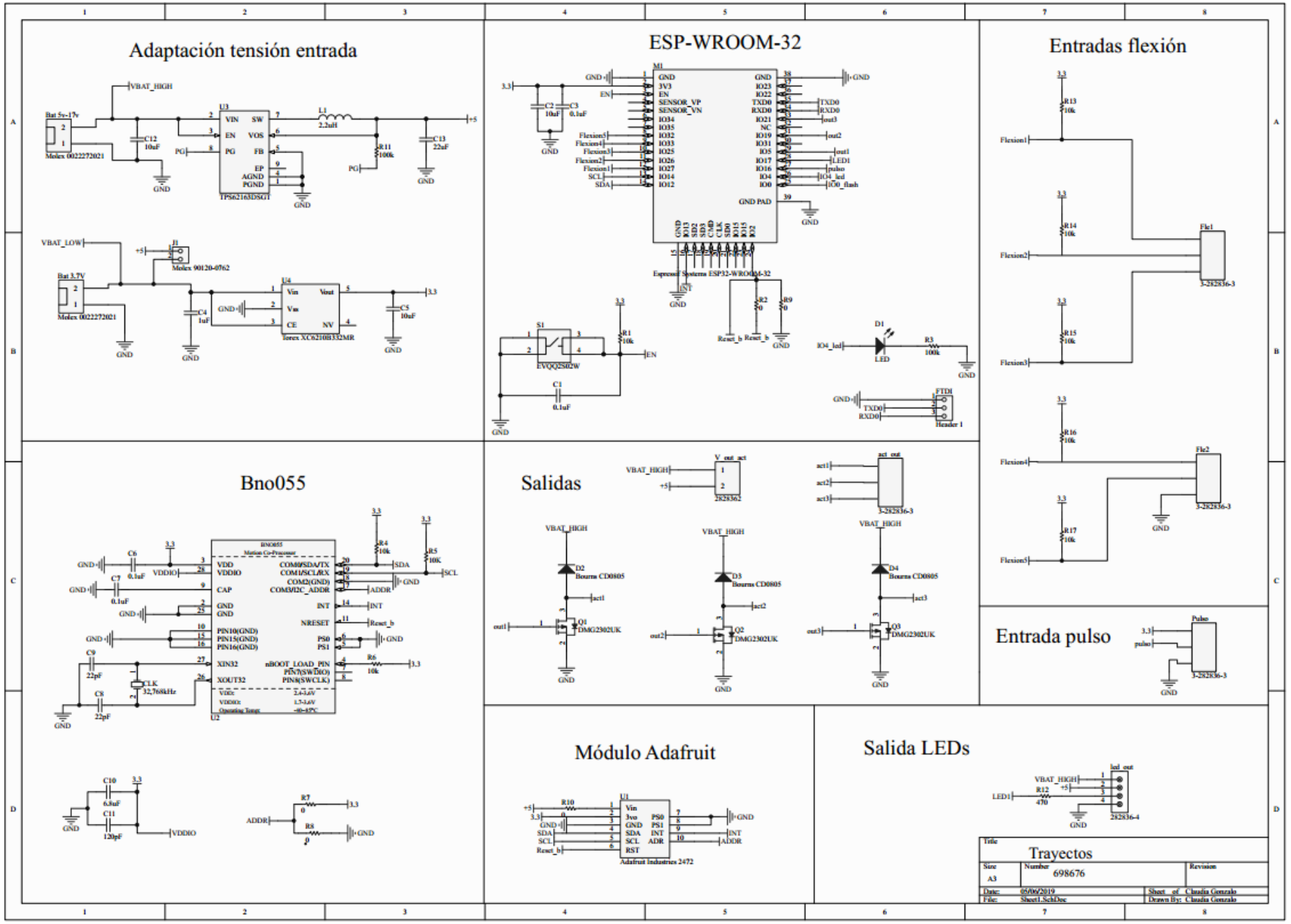
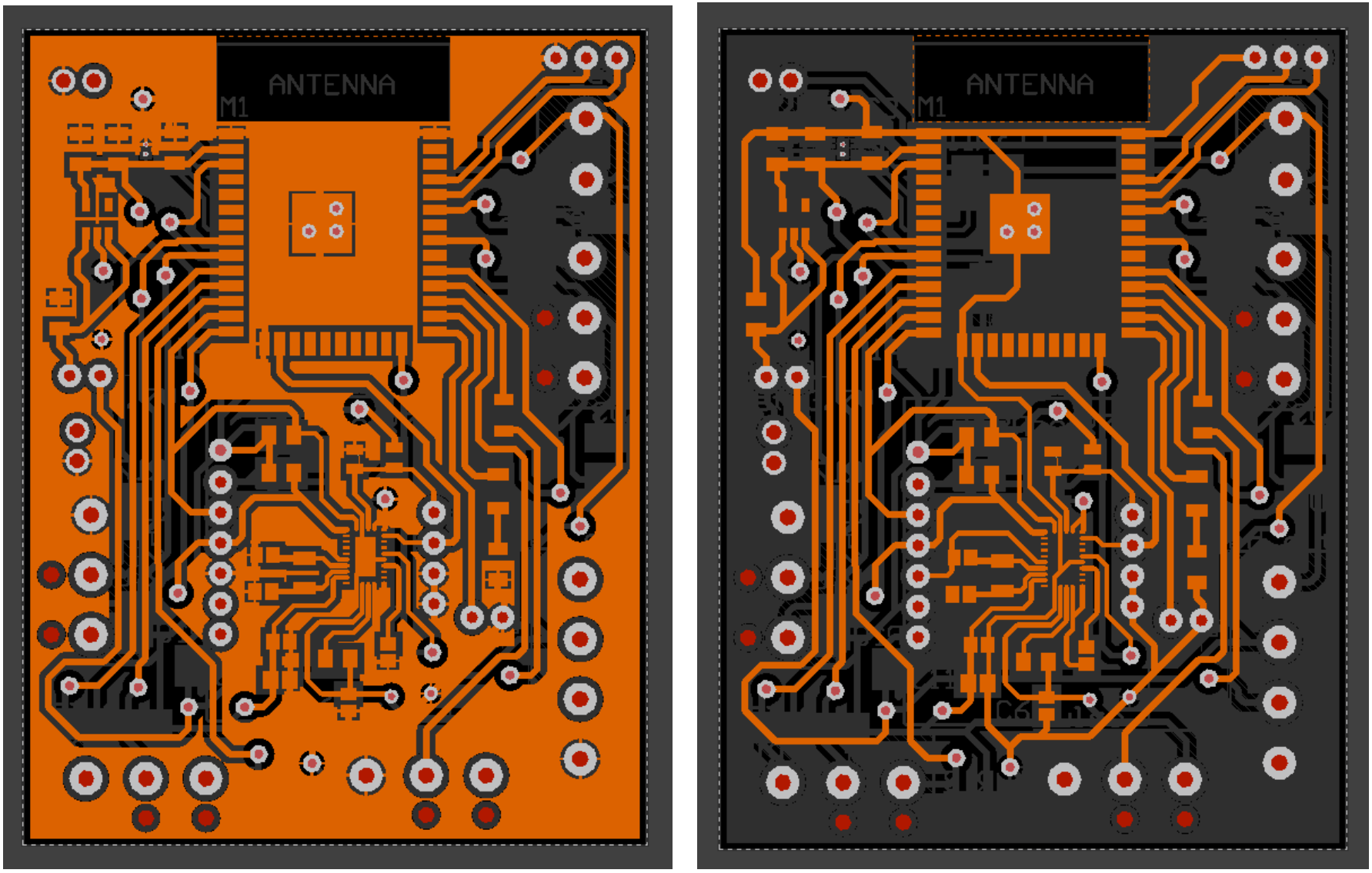
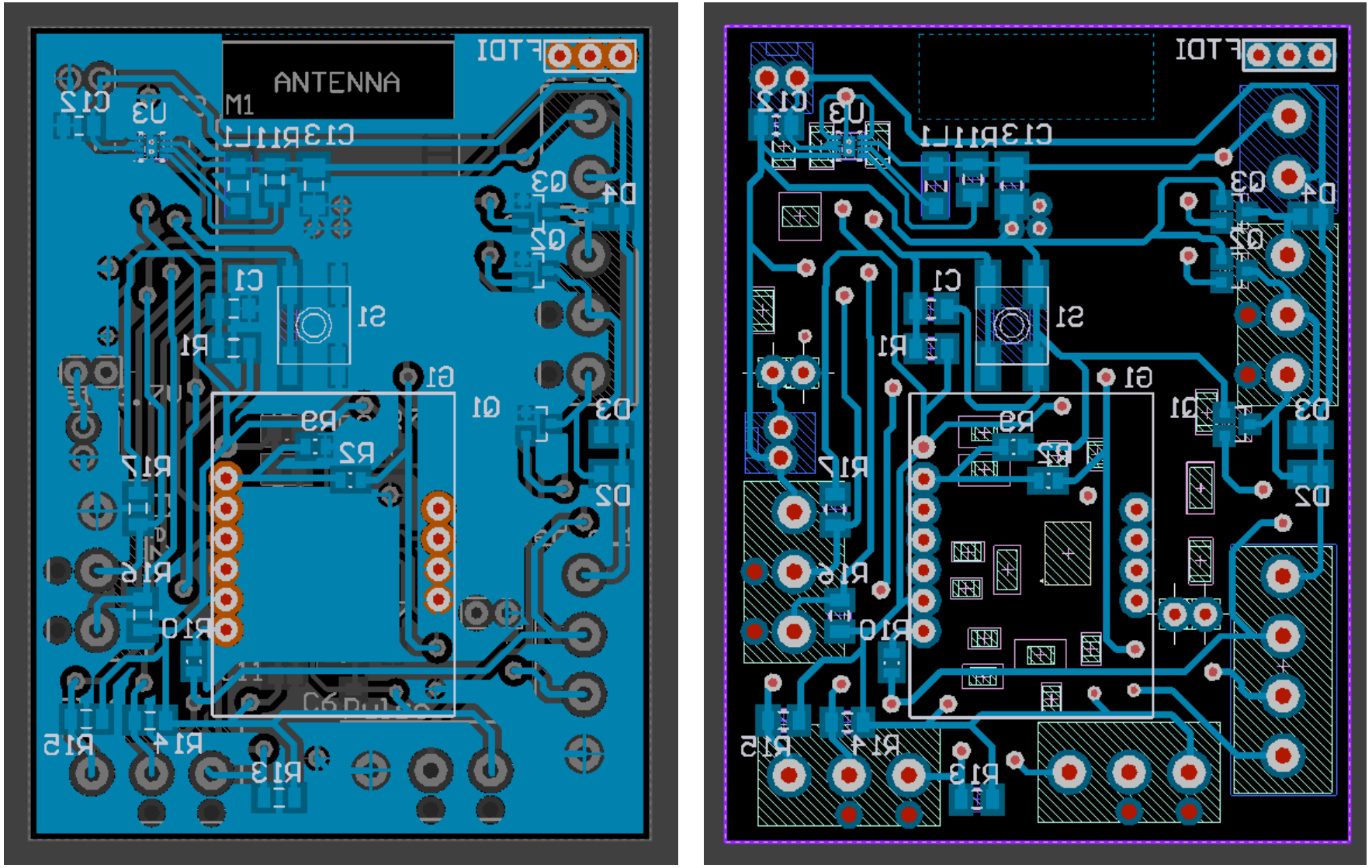



*Figura 3.1: Esquemático*



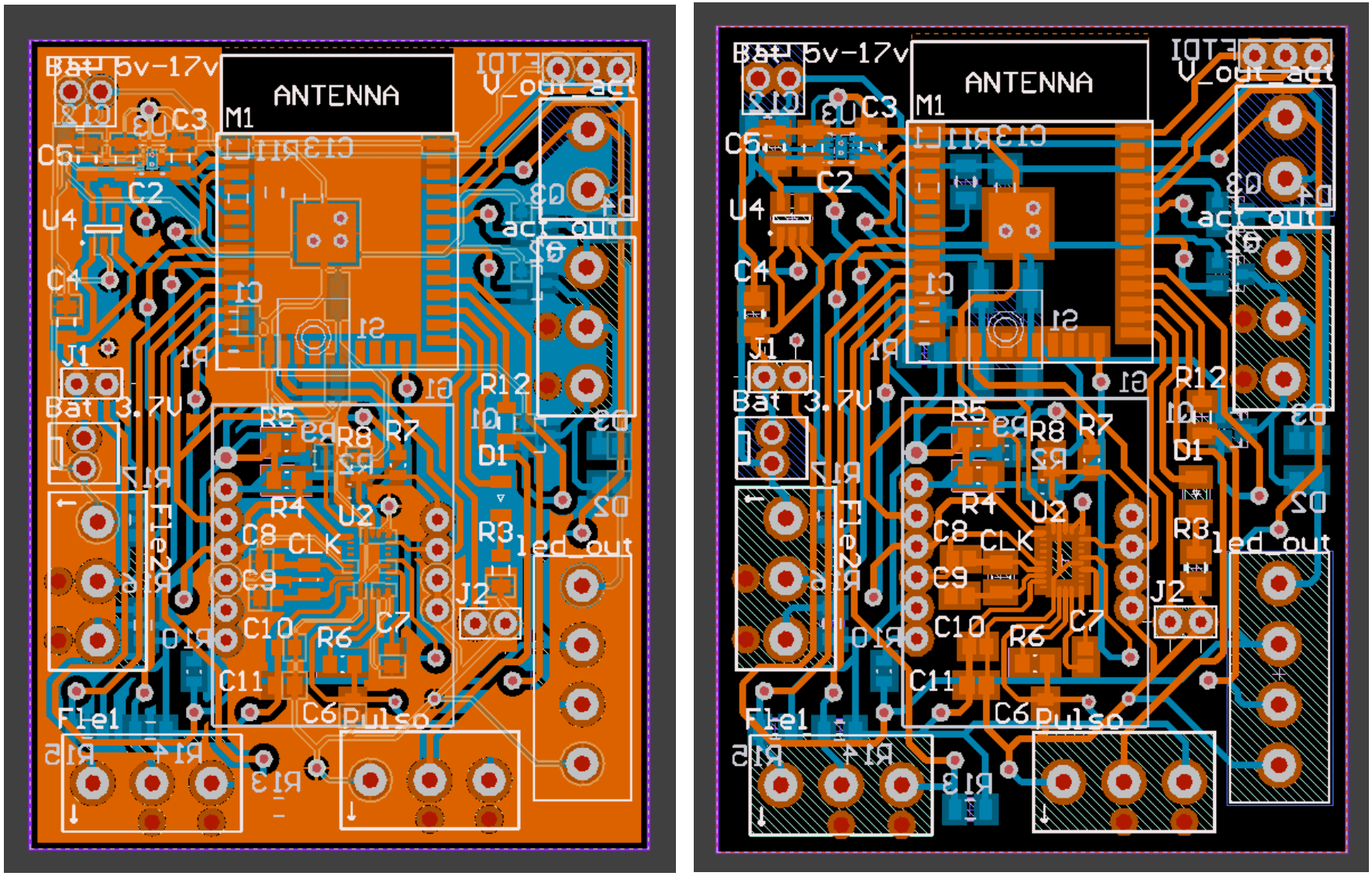

(a) (b)

*Figura 3.2: (a) Plano de masa cara TOP / (b) Pistas, pads y vías cara TOP*



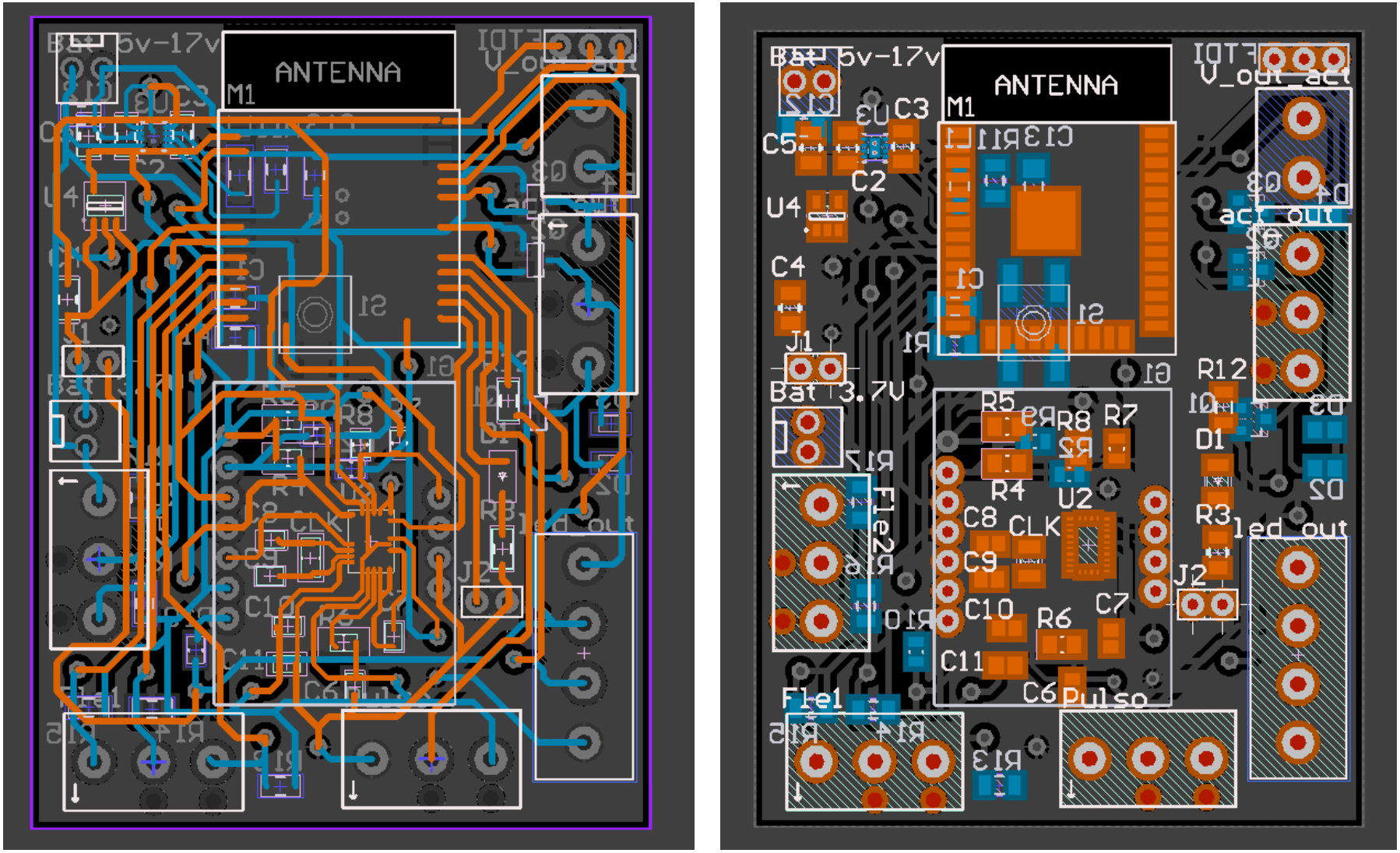

(a) (b)

*Figura 3.3: (a) Plano de masa cara BOTTOM / (b) Pistas, pads y vías caras BOTTOM*



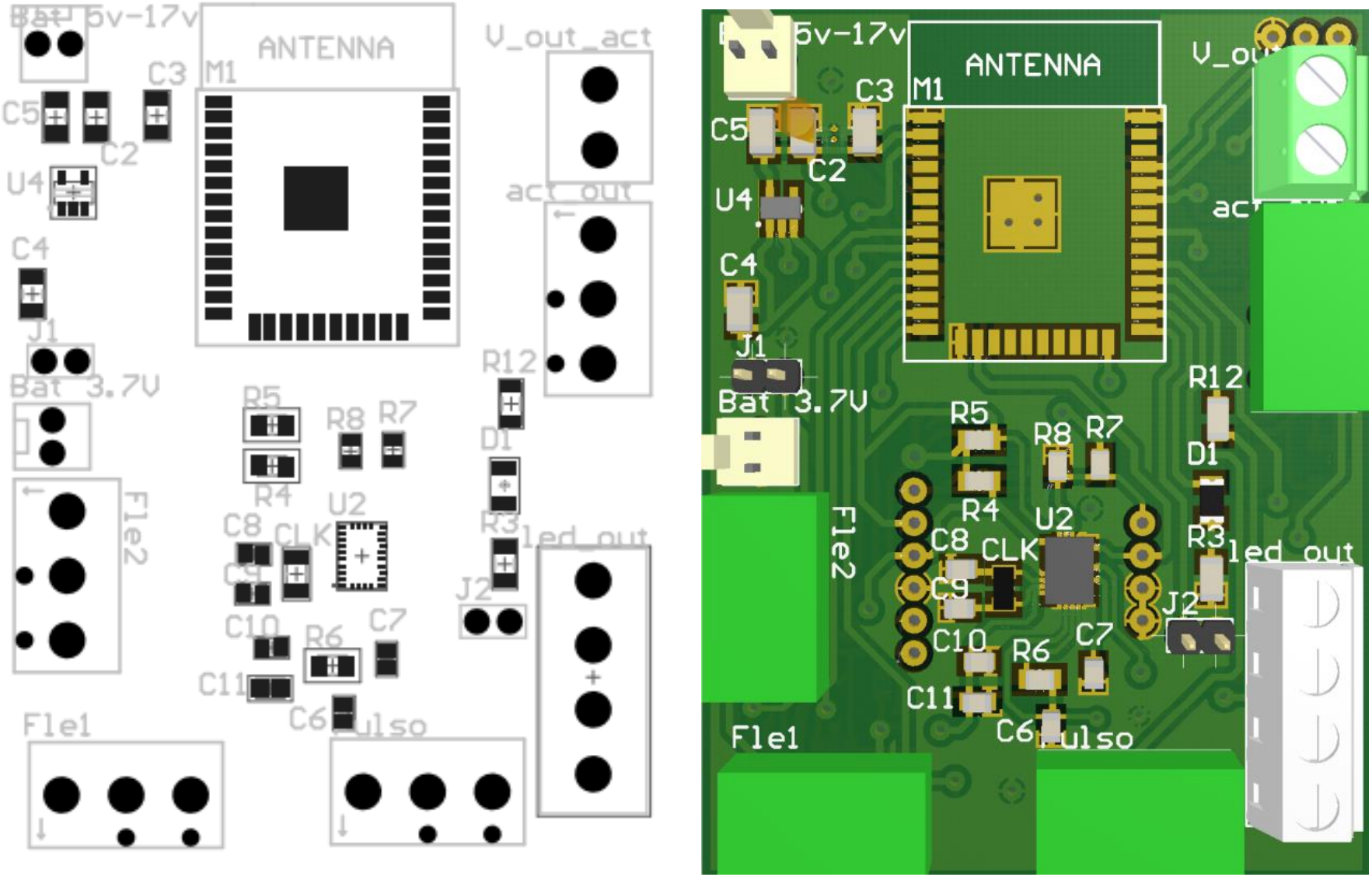

(a) (b)

*Figura 3.4: (a) Planos de masa cara TOP y BOTTOM / (b) Pistas, pads, vías y componentes cara TOP y BOTTOM*




(a) (b)

*Figura 3.5: (a) Pistas cara TOP y BOTTOM / (b) Componentes cara TOP y BOTTOM*




(a) (b)

*Figura 3.6: (a) Esquema de ensamblado (b) Circuito en 3D*

# Anexo 4. Código

## Presentación electrónica 1

A continuación, se muestra el código implementado en un módulo de Arduino Uno, utilizado para la exposición electrónica que se realiza a Manon Siv, a través de la plataforma Skype, al iniciar este proyecto. Esta exposición consta de una tira LED RGB, una matriz LED, micrófono, leds individuales, zumbador activo y pasivo, altavoz y dos botones capacitivos: rojo y verde.

```
//PRESENTACIÓN ELETRÓNICA

//LIBRERIAS NECESARIAS
#include <Wire.h>
#include <Adafruit_GFX.h>
#include "Adafruit_LEDBackpack.h"

//DEFINICIÓN DE PINES TIRA LED RGB
#define REDPIN 3
#define GREENPIN 5
#define BLUEPIN 6

//DECLARACÍON E INICIALIZACIÓN DE CONSTANTES Y VARIABLES
// Micrófono
const int sensorPIN = A0;
// Ancho ventana en mS (50 mS = 20Hz) para el micrófono
const int sampleWindow = 50;
//Botón para micro , led o zumbador
const int buttonPin = 2;
int buttonState = 0;
//Botón para activar matriz de leds
const int buttonMATRIZ = 4;
int buttonStateMATRIZ = 0;
//Botón solo para el zubador
const int buttonZUMBADOR = 7;
int buttonStateZUMBADOR = 0;
//Led activado por micrófono
const int LED_MICRO = 8;
//Zumbador activado con y sin micrófono (zumbador activo)
const int ZUMBADOR = 9;
//Botón solo para el zumbador
const int LED_ROJO = 10;
int buttonStateLED_ROJO = 0;
const int ALTAVOZ = 12;
//Botón solo para el altavoz
int buttonStateALTAVOZ = 0;
const int TIRASLED = 13;
//Botón solo para tiras led
int buttonStateTIRASLED = 0;
Adafruit_BicolorMatrix matrix = Adafruit_BicolorMatrix();

void setup() {
  //Velocidad puerto serie
  Serial.begin(9600);
  // Inicialización matriz
  matrix.begin(0x70);
```

```
 //DECLARACIÓN DE ENTRADAS Y SALIDAS
  pinMode(REDPIN, OUTPUT);
  pinMode(GREENPIN, OUTPUT);
  pinMode(BLUEPIN, OUTPUT);
  pinMode(LED_MICRO,OUTPUT);
  pinMode(ZUMBADOR, OUTPUT);
  pinMode(buttonPin, INPUT);
  pinMode(ALTAVOZ,OUTPUT);
}


static const uint8_t PROGMEM
//Para la matriz de leds
  frown_bmp[] =
  { B00111100,
    B01000010,
    B10100101,
    B10000001,
    B10011001,
    B10100101,
    B01000010,
    B00111100 };


void loop()

{
 //MICRÓFONO
 //Filtracón del micrófono
 unsigned long startMillis= millis();
 //Mínima señal que da el micrófono
(inicialización)
 unsigned int maxvalor=0;
 //Máxima señal que da el micrófono
(inicialización)
 unsigned int minvalor=1024;
 unsigned int valor;
 //Realizamos un while de 50ms (ventana teporal)
 while (millis() - startMillis < sampleWindow){
      //Leemos valor del sensor
      valor = analogRead(sensorPIN);
      //Ha recibido señal
      if (valor<1024){
         //La señal es mayor que el
máximo
         if (valor>maxvalor){
          // Actualizamos el máximo
            maxvalor = valor;
         }
         //Si la señal es menor que el mínimo
         if (valor<minvalor){
           // Actualizamos el mínimo
           minvalor= valor;
         }
      }
   }
  // Se calcula la amplitud del sonido
  unsigned int amplitud = maxvalor-minvalor;
  // Y se convierte a tensión
  double voltios =(amplitud*5.0)/1024;
```

```
   buttonState = digitalRead(buttonPin);
   //A partir de 1 ha recibido señal
   if
(voltios>1){

     if (buttonState == LOW) {
      //Si se presiona el pulsador cada vez que reciba señal se
activará el zumbador
         tone(ZUMBADOR,293.66,100);
         delay(200);
      }
      else{
        // Si no se presiona el pulsador se encenderá el
led
          digitalWrite(LED_MICRO,HIGH);
     }
   }
   else {
    //Si no recibe señal se apaga el
led
    digitalWrite(LED_MICRO,LOW);
   }


  //TIRAS LEDs
  int red, green, blue;
  buttonStateTIRASLED = digitalRead(TIRASLED);
  // Si se presiona el botón se enciende la tira de leds
  if (buttonStateTIRASLED == LOW) {

    //Primero se enciende en color rojo
    red=255;
    green=0;
    blue=0;
    analogWrite(REDPIN, red);
    analogWrite(BLUEPIN, blue);
    analogWrite(GREENPIN, green);
    delay(500);

    //Segundo se enciende en color azul
    red=0;
    green=0;
    blue=255;
    analogWrite(REDPIN, red);
    analogWrite(BLUEPIN, blue);
    analogWrite(GREENPIN, green);
    delay(500);

    //Tercero se enciende en color verde
    red=0;
    green=255;
    blue=0;
    analogWrite(REDPIN, red);
    analogWrite(BLUEPIN, blue);
    analogWrite(GREENPIN, green);
    delay(500);

  }
```

```
  //ALTAVOZ
  buttonStateALTAVOZ = digitalRead(ALTAVOZ);
  //Si se presiona el botón comienza la secuancia en el altavoz
  if (buttonStateALTAVOZ == LOW) {

    tone(ALTAVOZ,261,200);
    delay(200);
    tone(ALTAVOZ,293,200);
    delay(200);
    tone(ALTAVOZ,329,200);
    delay(200);
    tone(ALTAVOZ,349,200);
    delay(200);
    tone(ALTAVOZ,392,200);
    delay(200);
    tone(ALTAVOZ,440,200);
    delay(200);
  }

  //BÓTON ROJO QUE ACTIVA ZUMBADOR
   buttonStateLED_ROJO = digitalRead(LED_ROJO);
   //Si se presiona el botón capacitivo rojo se enciende el zumbador
activo
   if (buttonStateLED_ROJO == HIGH) {
    tone(ZUMBADOR,2500,50);
    delay(50);
   }

  //MATRIZ DE LEDs
  buttonStateMATRIZ = digitalRead(buttonMATRIZ);
  //Si se presiona el botón comienza una secuencia en la Matriz de
leds
  if (buttonStateMATRIZ == LOW) {

    //Se limpia la pantalla
     matrix.clear();
     //Seleción del diseño dentro de la biblioteca de la
matriz
     matrix.drawPixel(0, 0, LED_GREEN);
     //Actualización de la pantalla al nuevo diseño
     matrix.writeDisplay();
     delay(500);

     matrix.clear();
     matrix.drawLine(0,0, 7,7, LED_YELLOW);
     matrix.writeDisplay();
     delay(500);

     matrix.clear();
     matrix.drawRect(0,0, 8,8, LED_RED);
     matrix.fillRect(2,2, 4,4, LED_GREEN);
     matrix.writeDisplay();
     delay(500);

     matrix.clear();
     matrix.drawCircle(3,3, 3, LED_YELLOW);
     matrix.writeDisplay();
     delay(500);

     matrix.clear();
     matrix.drawCircle(3,3, 3, LED_RED);
```

```
      matrix.writeDisplay();
      delay(500);

      matrix.clear();
      matrix.drawCircle(3,3, 3, LED_GREEN);
      matrix.writeDisplay();
      delay(500);

     matrix.setRotation(3);
    matrix.setTextColor(LED_RED);
    for (int8_t x=7; x>=-36; x--) {
      matrix.clear();
      matrix.setCursor(x,0);
      matrix.print("World");
      matrix.writeDisplay();
      delay(100);
    }
    matrix.setRotation(0);
  }


  // ZUMBADOR
  buttonStateZUMBADOR = digitalRead(buttonZUMBADOR);
  //Si se presiona el botón comienza una secuancia en el zumbador
activo
  if (buttonStateZUMBADOR == LOW) {
    //Secuencia de diferenes frecuencias y tiempos
    tone(ZUMBADOR,293.66,800);
    delay(800);
    tone(ZUMBADOR,440,800);
    delay(500);
    tone(ZUMBADOR,2500,800);
    delay(800);
    tone(ZUMBADOR,1800,500);
    delay(800);
    tone(ZUMBADOR,293.66,500);
    delay(100);
    tone(ZUMBADOR,440,500);
    delay(100);
    tone(ZUMBADOR,2500,500);
    delay(100);
    tone(ZUMBADOR,300,500);
    delay(100);
    tone(ZUMBADOR,440,500);
    delay(100);
    tone(ZUMBADOR,2500,500);
    delay(100);
    tone(ZUMBADOR,300,500);
    delay(100);
    tone(ZUMBADOR,440,500);
    delay(100);
    tone(ZUMBADOR,2500,500);
    delay(100);
    tone(ZUMBADOR,300,500);
    delay(1000);
  }
}
```

## Presentación electrónica 2

Se realiza una segunda presentación electrónica, modificando la primera, añadiendo y eliminando componentes acordes a los intereses de Manon Siv. Se realiza en un módulo de Arduino Uno y esta vez de forma presencial. Consta de una tira LED direccionable, dos sensores de presencia, micrófono, leds individuales, tres zumbadores pasivos y uno activo, y dos botones capacitivos: rojo y verde.

```
//PRESENTACIÓN ELETRÓNICA DOS, PRESENCIAL CON MANON

//LIBRERIAS NECESARIAS
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_NeoPixel.h>

//PARÁMETROS NECESARIOS PARA LA TIRA LED DIRECCIONABLE
#define Tira1 4
#define NUMPIXELS 7
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, Tira1, NEO_GRB
+ NEO_KHZ800);

//DECLARACÍON E INICIALIZACIÓN DE CONSTANTES Y VARIABLES
// Micrófono
const int sensorPIN = A0;
// Ancho ventana en mS (50 mS = 20Hz) para el micrófono
const int sampleWindow = 50;
//Botón para micro , led o zumbador
const int buttonPin = 2;
int buttonState = 0;
//Led activado por micrófono
const int LED_MICRO = 8;
//Zumbador activado con y sin micrófono (zumbador activo)
const int ZUMBADOR = 9;
//Botón para secuencia de zumbadores
const int LED_ROJO = 10;
int buttonStateLED_ROJO = 0;
//Botón verde para el zubador
const int LED_VERDE = 5;
int buttonStateLED_VERDE= 0;
const int ZUMBADOR2 = 12;
int buttonStateZUMBADOR2 = 0;
const int ZUMBADOR3 = 13;
int var;
int var2;



void setup() {
   //Velocidad puerto serie
  Serial.begin(9600);
  // Inicialización tira LED
  pixels.begin();
 //DECLARACIÓN DE ENTRADAS Y SALIDAS
  pinMode(LED_MICRO,OUTPUT);
  pinMode(buttonPin, INPUT);
  //Sensor de presencia uno
  pinMode(A5,INPUT);
  //Sensor de presencia dos
  //Cada sensor activa una secuencia distinta
```

```
  pinMode(A4,INPUT);
  pinMode(ZUMBADOR, OUTPUT);
  pinMode(ZUMBADOR2,OUTPUT);
  pinMode(ZUMBADOR3,OUTPUT);
}

void loop() {

 //MICRÓFONO
 //Filtracón del micrófono
 unsigned long startMillis= millis();
 //Mínima señal que da el micrófono
(inicialización)
 unsigned int maxvalor=0; 7
 //Máxima señal que da el micrófono
(inicialización)
 unsigned int minvalor=1024;
 unsigned int valor;
 //Realizamos un while de 50ms (ventana teporal)
 while (millis() - startMillis < sampleWindow){
      //Leemos valor del sensor
      valor = analogRead(sensorPIN);
      //Ha recibido señal
      if (valor<1024){
        //La señal es mayor que el
máximo
         if (valor>maxvalor){
          // Actualizamos el máximo
           maxvalor = valor;
         }
         //Si la señal es menor que el mínimo
         if (valor<minvalor){
           // Actualizamos el mínimo
           minvalor= valor;
         }
      }
   }
  // Se calcula la amplitud del sonido
  unsigned int amplitud = maxvalor-minvalor;
  // Y se convierte a tensión
  double voltios =(amplitud*5.0)/1024;


   buttonState = digitalRead(buttonPin);
   //A partir de 1 ha recibido señal
   if
(voltios>1){

     if (buttonState == LOW) {
         //Si se presiona el pulsador cada vez que reciba señal se
activará el zumbador
         tone(ZUMBADOR,293.66,100);
         delay(200);
      }
      else{
          // Si no se presiona el pulsador se encenderá el
led
          digitalWrite(LED_MICRO,HIGH);
     }
   }
```

```
   else {
    //Si no recibe señal se apaga el
led
    digitalWrite(LED_MICRO,LOW);
   }



//SENSOR PRESENCIA UNO
 //Leemos el valor proporcionado por el sensor uno
  var=analogRead(A5);
  //Si detecta un objeto
  if(var<1000){

   //Se apaga la tira LED
    pixels.setPixelColor(0,0,0,0);
    pixels.setPixelColor(1,0,0,0);
    pixels.setPixelColor(2,0,0,0);
    pixels.setPixelColor(3,0,0,0);
    pixels.setPixelColor(4,0,0,0);
    pixels.setPixelColor(5,0,0,0);
    pixels.setPixelColor(6,0,0,0);
    pixels.setPixelColor(7,0,0,0);
    pixels.show();
    delay(1000);

    //Se encienden de forma independiente y acumulativa la tira en
color blanco (de tres en tres)
    pixels.setPixelColor(0,150,150,150);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(1,150,150,150);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(2,150,150,150);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(3,150,150,150);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(4,150,150,150);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(5,150,150,150);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(6,150,150,150);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(7,150,150,150);
    pixels.show();
    delay(1000);

    //Se enciende la mitad nicialde color rojo de golpe
    pixels.setPixelColor(0,255,0,0);
    pixels.setPixelColor(1,255,0,0);
    pixels.setPixelColor(2,255,0,0);
    pixels.setPixelColor(3,255,0,0);
    pixels.show();
    delay(1000);
    //Se enciende la segunda mitad de color rojo de golpe
```

```
    pixels.setPixelColor(4,255,0,00);
    pixels.setPixelColor(5,255,0,00);
    pixels.setPixelColor(6,255,0,00);
    pixels.setPixelColor(7,255,0,00);
    pixels.show();
    delay(1000);


    //Se enciende toda la tira de golpe de color verde
    pixels.setPixelColor(0,0,255,0);
    pixels.setPixelColor(1,0,255,0);
    pixels.setPixelColor(2,0,255,0);
    pixels.setPixelColor(3,0,255,0);
    pixels.setPixelColor(4,0,255,0);
    pixels.setPixelColor(5,0,255,0);
    pixels.setPixelColor(6,0,255,0);
    pixels.setPixelColor(7,0,255,0);
    pixels.show();
    delay(1000);

    //Se enciende de color azul, poco a poco y de forma desordenada la
tira
    pixels.setPixelColor(1,0,0,255);
    pixels.setPixelColor(7,0,0,255);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(2,0,0,255);
    pixels.setPixelColor(6,0,0,255);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(3,0,0,255);
    pixels.setPixelColor(5,0,0,255);
    pixels.show();
    delay(1000);
    pixels.setPixelColor(0,0,0,255);
    pixels.setPixelColor(4,0,0,255);
    pixels.show();
    delay(1000);
  }
  else{
    //En caso de no detectar objeto se mantiene apagada la tira
    pixels.setPixelColor(0,0,0,0);
    pixels.setPixelColor(1,0,0,0);
    pixels.setPixelColor(2,0,0,0);
    pixels.setPixelColor(3,0,0,0);
    pixels.setPixelColor(4,0,0,0);
    pixels.setPixelColor(5,0,0,0);
    pixels.setPixelColor(6,0,0,0);
    pixels.setPixelColor(7,0,0,0);
    pixels.show();
    delay(1000);
  }


  //SENSOR PRESENCIA DOS
  //Leemos el valor proporcionado por el sensor dos
  var2=analogRead(A4);
  //Si detecta un objeto
  if(var2<1000){
```

```
    //Se apaga la tira LED
    pixels.setPixelColor(0,0,0,0);
    pixels.setPixelColor(1,0,0,0);
    pixels.setPixelColor(2,0,0,0);
    pixels.setPixelColor(3,0,0,0);
    pixels.setPixelColor(4,0,0,0);
    pixels.setPixelColor(5,0,0,0);
    pixels.setPixelColor(6,0,0,0);
    pixels.setPixelColor(7,0,0,0);
    pixels.show();
    delay(1000);

    //Comienza secuencia donde, de color rojo se van encendiendo y
apagando de forma independiente y sin orden aparente
    pixels.setPixelColor(0,255,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(7,255,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(6,255,0,0);
    pixels.show();
    pixels.setPixelColor(0,0,0,0);
    pixels.show();
    delay(200);
    delay(200);
    pixels.setPixelColor(2,255,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(6,0,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(4,255,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(1,255,0,0);
    pixels.setPixelColor(2,0,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(7,0,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(3,255,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(5,255,0,0);
    pixels.setPixelColor(1,0,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(4,0,0,0);
    pixels.show();
    delay(200);
    pixels.setPixelColor(5,0,0,0);
    pixels.setPixelColor(3,0,0,0);
    pixels.show();
    delay(1000);

    //Se enciende toda la tira de color blanco de golpe y
posteriormente se apaga x2 (efecto parpadeo)
    pixels.setPixelColor(0,150,0,150);
```

```
  pixels.setPixelColor(1,150,0,150);
  pixels.setPixelColor(2,150,0,150);
  pixels.setPixelColor(3,150,0,150);
  pixels.setPixelColor(4,150,0,150);
  pixels.setPixelColor(5,150,0,150);
  pixels.setPixelColor(6,150,0,150);
  pixels.setPixelColor(7,150,0,150);
  pixels.show();
  delay(200);

  pixels.setPixelColor(0,0,0,0);
  pixels.setPixelColor(1,0,0,0);
  pixels.setPixelColor(2,0,0,0);
  pixels.setPixelColor(3,0,0,0);
  pixels.setPixelColor(4,0,0,0);
  pixels.setPixelColor(5,0,0,0);
  pixels.setPixelColor(6,0,0,0);
  pixels.setPixelColor(7,0,0,0);
  pixels.show();
  delay(200);


  pixels.setPixelColor(0,150,0,150);
  pixels.setPixelColor(1,150,0,150);
  pixels.setPixelColor(2,150,0,150);
  pixels.setPixelColor(3,150,0,150);
  pixels.setPixelColor(4,150,0,150);
  pixels.setPixelColor(5,150,0,150);
  pixels.setPixelColor(6,150,0,150);
  pixels.setPixelColor(7,150,0,150);
  pixels.show();
  delay(200);

  pixels.setPixelColor(0,0,0,0);
  pixels.setPixelColor(1,0,0,0);
  pixels.setPixelColor(2,0,0,0);
  pixels.setPixelColor(3,0,0,0);
  pixels.setPixelColor(4,0,0,0);
  pixels.setPixelColor(5,0,0,0);
  pixels.setPixelColor(6,0,0,0);
  pixels.setPixelColor(7,0,0,0);
  pixels.show();
  delay(200);

  //Se enciende de color azul toda de golpe
  pixels.setPixelColor(0,0,0,255);
  pixels.setPixelColor(1,0,0,255);
  pixels.setPixelColor(2,0,0,255);
  pixels.setPixelColor(3,0,0,255);
  pixels.setPixelColor(4,0,0,255);
  pixels.setPixelColor(5,0,0,255);
  pixels.setPixelColor(6,0,0,255);
  pixels.setPixelColor(7,0,0,255);
  pixels.show();
  delay(1000);
}

else{
  //En caso de no detectar ningun objeto se mantiene apagada
  pixels.setPixelColor(0,0,0,0);
  pixels.setPixelColor(1,0,0,0);
```

```
    pixels.setPixelColor(2,0,0,0);
    pixels.setPixelColor(3,0,0,0);
    pixels.setPixelColor(4,0,0,0);
    pixels.setPixelColor(5,0,0,0);
    pixels.setPixelColor(6,0,0,0);
    pixels.setPixelColor(7,0,0,0);
    pixels.show();
    delay(1000);
  }

//PULSADOR CAPACITIVO VERDE
//En caso de activar el boton verde capacitivo se actia el zumbador3
pasivo
  buttonStateLED_VERDE = digitalRead(LED_VERDE);
  if (buttonStateLED_VERDE == HIGH) {
    digitalWrite(ZUMBADOR3,HIGH);
    delay(1000);
    digitalWrite(ZUMBADOR3,LOW);
  }

//PULSADOR CAPACITIVO ROJO

  buttonStateLED_ROJO = digitalRead(LED_ROJO);
  //En caso de activar el boton rojo capacitivo se actia una secuencia
con los tres zumbadores
  if (buttonStateLED_ROJO == HIGH) {
    tone(ZUMBADOR,440,800);
    delay(500);
    digitalWrite(ZUMBADOR2,HIGH);
    tone(ZUMBADOR,2500,800);
    delay(800);
    digitalWrite(ZUMBADOR3,HIGH);
    delay(500);
    tone(ZUMBADOR,293.66,500);
    delay(100);
    digitalWrite(ZUMBADOR2,LOW);
    digitalWrite(ZUMBADOR3,LOW);
    tone(ZUMBADOR,440,500);
    delay(100);
    digitalWrite(ZUMBADOR3,HIGH);
    digitalWrite(ZUMBADOR2,HIGH);
    delay(20);
    tone(ZUMBADOR,2500,500);
    delay(100);
    digitalWrite(ZUMBADOR2,LOW);
    tone(ZUMBADOR,300,500);
    delay(100);
    digitalWrite(ZUMBADOR3,LOW);
    tone(ZUMBADOR,2500,500);
    delay(100);
    digitalWrite(ZUMBADOR3,HIGH);
    delay(100);
    tone(ZUMBADOR,300,500);
    delay(1000);
    digitalWrite(ZUMBADOR3,LOW);
  }


}
```

## Actuación de Manon Siv

A mitad del proceso se realiza una actuación con Manon Siv. El código se modifica numerosas veces a lo largo de las distintas reuniones, reflejando aquí únicamente el resultado final. La actuación se realiza en el auditorio de Etopía de Zaragoza y utiliza tres circuitos.

### Circuito ultrasonidos

El código de este circuito se basa en encender un zumbador durante 30 segundos cuando se detecta un objeto a menos de 20 cm con el sensor de ultrasonidos.

```
// CIRCUITO ULTRASONIDOS, ACTUACIÓN MANON

//DECLARACIÓN VARIABLES Y CONSTANTES
const int Echo = 10;
const int Trigger = 11;
int cm;
cont int ZUMBADOR =9;

void setup() {
   //Velocidad puerto serie
   Serial.begin(9600);
   //Ultrasonidos
   pinMode(Trigger, OUTPUT);
   pinMode(Echo, INPUT);
   pinMode (ZUMBADOR, OUTPUT);
}

void loop() {
   //Método para medir la distancia entre pulsos
   int cm = ping(Trigger, Echo);

   //Si se detecta algo a menos de 20 cm se activa durante 30 segundos
el zumbador
    if (cm<20){
       digitalWrite(ZUMBADOR,HIGH);
       delay(30000);
    }
  else {
       //Si no se detecta, se mantiene apagado el zumbador
       digitalWrite(ZUMBADOR,LOW);
       delay(100);
     }

}

  //Mide el tiempo entre emisión y recepción de un pulso sonoro
  int ping(int Trigger, int Echo) {
       long duration, distanceCm;
       //Para generar un pulso limpio ponemos a LOW 4us
       digitalWrite(Trigger, LOW);
       delayMicroseconds(4);
       //Generamos Trigger (disparo) de 10us
       digitalWrite(Trigger, HIGH);
       delayMicroseconds(10);
       digitalWrite(Trigger, LOW);
```

```
        /medimos el tiempo entre pulsos, en microsegundos
        duration = pulseIn(Echo, HIGH);
        //Velocidad del sosnido =343m/s
        //Convertimos unidades
        //343 m/s * 100 cm/s * 1/1000000 s/us = 1/29.2 cm/us
        //El sonido tarda 29.2 us en recorrer un cm
        distanceCm = duration * 10 / 292/ 2;
        //Dividimos para dos porque se calcula ir y volver
        return distanceCm;
}
```

## Circuito tira LED pared

Consta de dos tiras LED dispuestas en la pared del auditorio de Etopia, que se encienden cuando se activa el pulsador.

```
//CIRCUITO TIRAS LED PARED, ACTUACIÓN MANON

//LIBRERIAS NECESARIAS
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

//PARÁMETROS NECESARIOS PARA LA TIRA LED DIRECCIONABLE
#define Tira1 6
#define Tira2 4
#define NUMPIXELS1 38
#define NUMPIXELS2 38
Adafruit_NeoPixel pixels1 = Adafruit_NeoPixel(NUMPIXELS1, Tira1,
NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel pixels2 = Adafruit_NeoPixel(NUMPIXELS2, Tira2,
NEO_GRB + NEO_KHZ800);


//Colores&Apagado
uint32_t apag1 = pixels1.Color(0, 0, 0);
uint32_t apag2 = pixels2.Color(0, 0, 0);
uint32_t white1 = pixels1.Color(150, 150, 150);
uint32_t white2 = pixels2.Color(150, 150, 150);


//DECLARACÍON E INICIALIZACIÓN DE CONSTANTES
const int pulsador = 2;
int valorpulsador = 0;
int cont=0;


void setup() {
  //Velocidad puerto serie
  Serial.begin(9600);
  //Inicialización tiras LED
  pixels1.begin();
  pixels2.begin();
  //Declaración de entrada
  pinMode(pulsador, INPUT);

  //Inicialización de leds apagados
  pixels1.setPixelColor(0, apag1);
  pixels1.setPixelColor(1, apag1);
  pixels1.setPixelColor(2, apag1);
```

```
pixels1.setPixelColor(3, apag1);
pixels1.setPixelColor(4, apag1);
pixels1.setPixelColor(5, apag1);
pixels1.setPixelColor(6, apag1);
pixels1.setPixelColor(7, apag1);
pixels1.setPixelColor(8, apag1);
pixels1.setPixelColor(9, apag1);
pixels1.setPixelColor(10, apag1);
pixels1.setPixelColor(11, apag1);
pixels1.setPixelColor(12, apag1);
pixels1.setPixelColor(13, apag1);
pixels1.setPixelColor(14, apag1);
pixels1.setPixelColor(15, apag1);
pixels1.setPixelColor(16, apag1);
pixels1.setPixelColor(17, apag1);
pixels1.setPixelColor(18, apag1);
pixels1.setPixelColor(19, apag1);
pixels1.setPixelColor(20, apag1);
pixels1.setPixelColor(21, apag1);
pixels1.setPixelColor(22, apag1);
pixels1.setPixelColor(23, apag1);
pixels1.setPixelColor(24, apag1);
pixels1.setPixelColor(25, apag1);
pixels1.setPixelColor(26, apag1);
pixels1.setPixelColor(27, apag1);
pixels1.setPixelColor(28, apag1);
pixels1.setPixelColor(29, apag1);
pixels1.setPixelColor(30, apag1);
pixels1.setPixelColor(31, apag1);
pixels1.setPixelColor(32, apag1);
pixels1.setPixelColor(33, apag1);
pixels1.setPixelColor(34, apag1);
pixels1.setPixelColor(35, apag1);
pixels1.setPixelColor(36, apag1);
pixels1.setPixelColor(37, apag1);
pixels2.setPixelColor(0, apag2);
pixels2.setPixelColor(1, apag2);
pixels2.setPixelColor(2, apag2);
pixels2.setPixelColor(3, apag2);
pixels2.setPixelColor(4, apag2);
pixels2.setPixelColor(5, apag2);
pixels2.setPixelColor(6, apag2);
pixels2.setPixelColor(7, apag2);
pixels2.setPixelColor(8, apag2);
pixels2.setPixelColor(9, apag2);
pixels2.setPixelColor(10, apag2);
pixels2.setPixelColor(11, apag2);
pixels2.setPixelColor(12, apag2);
pixels2.setPixelColor(13, apag2);
pixels2.setPixelColor(14, apag2);
pixels2.setPixelColor(15, apag2);
pixels2.setPixelColor(16, apag2);
pixels2.setPixelColor(17, apag2);
pixels2.setPixelColor(18, apag2);
pixels2.setPixelColor(19, apag2);
pixels2.setPixelColor(20, apag2);
pixels2.setPixelColor(21, apag2);
pixels2.setPixelColor(22, apag2);
pixels2.setPixelColor(23, apag2);
pixels2.setPixelColor(24, apag2);
pixels2.setPixelColor(25, apag2);
```

```
  pixels2.setPixelColor(26, apag2);
  pixels2.setPixelColor(27, apag2);
  pixels2.setPixelColor(28, apag2);
  pixels2.setPixelColor(29, apag2);
  pixels2.setPixelColor(30, apag2);
  pixels2.setPixelColor(31, apag2);
  pixels2.setPixelColor(32, apag2);
  pixels2.setPixelColor(33, apag2);
  pixels2.setPixelColor(34, apag2);
  pixels2.setPixelColor(35, apag2);
  pixels2.setPixelColor(36, apag2);
  pixels2.setPixelColor(37, apag2);
  pixels2.show();
  pixels1.show();
  delay(500);
}


void loop() {
  //Apagamos todos los leds
  apagar_todo();

  //Leemos el valor del pulsador
  valorpulsador = digitalRead(pulsador);
  //Si el presiona el pulsador
  if (valorpulsador == HIGH) {
        Serial.print("empiezo");
        cont=cont+1;
        if (cont==1){

        //Enciendo en ambas tiras a la vez uno de los leds (de tres en
tres)
        for (int i = 0; i < 38; i++) {
              pixels1.setPixelColor(i, white1);
              pixels2.setPixelColor(i, white2);
              pixels1.show();
              pixels2.show();
              delay(750);
              if (i == 37) {
              delay(1500);
              //En total dura 30 segundos (37*750 + 1500)
              }

       }
   //Al finalizar se apaga todo
   apagar_todo();
   cont=0;
   }

  }
 }


//METODO QUE APAGA TODOS LOS LEDS DE GOLPE, DE LAS DOS TIRAS
  void apagar_todo(){
      pixels1.setPixelColor(0, apag1);
      pixels1.setPixelColor(1, apag1);
      pixels1.setPixelColor(2, apag1);
      pixels1.setPixelColor(3, apag1);
      pixels1.setPixelColor(4, apag1);
      pixels1.setPixelColor(5, apag1);
```

```
pixels1.setPixelColor(6, apag1);
pixels1.setPixelColor(7, apag1);
pixels1.setPixelColor(8, apag1);
pixels1.setPixelColor(9, apag1);
pixels1.setPixelColor(10, apag1);
pixels1.setPixelColor(11, apag1);
pixels1.setPixelColor(12, apag1);
pixels1.setPixelColor(13, apag1);
pixels1.setPixelColor(14, apag1);
pixels1.setPixelColor(15, apag1);
pixels1.setPixelColor(16, apag1);
pixels1.setPixelColor(17, apag1);
pixels1.setPixelColor(18, apag1);
pixels1.setPixelColor(19, apag1);
pixels1.setPixelColor(20, apag1);
pixels1.setPixelColor(21, apag1);
pixels1.setPixelColor(22, apag1);
pixels1.setPixelColor(23, apag1);
pixels1.setPixelColor(24, apag1);
pixels1.setPixelColor(25, apag1);
pixels1.setPixelColor(26, apag1);
pixels1.setPixelColor(27, apag1);
pixels1.setPixelColor(28, apag1);
pixels1.setPixelColor(29, apag1);
pixels1.setPixelColor(30, apag1);
pixels1.setPixelColor(31, apag1);
pixels1.setPixelColor(32, apag1);
pixels1.setPixelColor(33, apag1);
pixels1.setPixelColor(34, apag1);
pixels1.setPixelColor(35, apag1);
pixels1.setPixelColor(36, apag1);
pixels1.setPixelColor(37, apag1);
pixels2.setPixelColor(0, apag2);
pixels2.setPixelColor(1, apag2);
pixels2.setPixelColor(2, apag2);
pixels2.setPixelColor(3, apag2);
pixels2.setPixelColor(4, apag2);
pixels2.setPixelColor(5, apag2);
pixels2.setPixelColor(6, apag2);
pixels2.setPixelColor(7, apag2);
pixels2.setPixelColor(8, apag2);
pixels2.setPixelColor(9, apag2);
pixels2.setPixelColor(10, apag2);
pixels2.setPixelColor(11, apag2);
pixels2.setPixelColor(12, apag2);
pixels2.setPixelColor(13, apag2);
pixels2.setPixelColor(14, apag2);
pixels2.setPixelColor(15, apag2);
pixels2.setPixelColor(16, apag2);
pixels2.setPixelColor(17, apag2);
pixels2.setPixelColor(18, apag2);
pixels2.setPixelColor(19, apag2);
pixels2.setPixelColor(20, apag2);
pixels2.setPixelColor(21, apag2);
pixels2.setPixelColor(22, apag2);
pixels2.setPixelColor(23, apag2);
pixels2.setPixelColor(24, apag2);
pixels2.setPixelColor(25, apag2);
pixels2.setPixelColor(26, apag2);
pixels2.setPixelColor(27, apag2);
pixels2.setPixelColor(28, apag2);
```

```
      pixels2.setPixelColor(29, apag2);
      pixels2.setPixelColor(30, apag2);
      pixels2.setPixelColor(31, apag2);
      pixels2.setPixelColor(32, apag2);
      pixels2.setPixelColor(33, apag2);
      pixels2.setPixelColor(34, apag2);
      pixels2.setPixelColor(35, apag2);
      pixels2.setPixelColor(36, apag2);
      pixels2.setPixelColor(37, apag2);
      pixels2.show();
      pixels1.show();
      delay(500);
  }
```

## Circuito tira LED portable

Este circuito se encarga de encender una tira LED cuando se presiona el pulsador que lleva acoplado la bailarina a su cuerpo.

```
// CIRCUITO TIRA LED PORTABLE, ACTUACIÓN MANON

//LIBRERIAS NECESARIAS
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

//PARÁMETROS NECESARIOS PARA LA TIRA LED DIRECCIONABLE
#define Tira1 5
#define NUMPIXELS 9
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, Tira1, NEO_GRB
+ NEO_KHZ800);


//DECLARACÍON E INICIALIZACIÓN DE CONSTANTES Y VARIABLES
const int pulsador = 4;
int valorpulsador = 0;


void setup() {
   //Velocidad puerto serie
    Serial.begin(9600);
   // Inicialización tira LED
    pixels.begin();

  //Inicialización de leds apagados
  pixels.setPixelColor(0,0,0,0);
  pixels.setPixelColor(1,0,0,0);
  pixels.setPixelColor(2,0,0,0);
  pixels.setPixelColor(3,0,0,0);
  pixels.setPixelColor(4,0,0,0);
  pixels.setPixelColor(5,0,0,0);
  pixels.setPixelColor(6,0,0,0);
  pixels.setPixelColor(7,0,0,0);
  pixels.setPixelColor(8,0,0,0);
  pixels.show();
}

void loop() {
```

```
//Leds apagados al inicio
    pixels.setPixelColor(0,0,0,0);
    pixels.show();
    pixels.setPixelColor(1,0,0,0);
    pixels.show();
    pixels.setPixelColor(2,0,0,0);
    pixels.show();
    pixels.setPixelColor(3,0,0,0);
    pixels.show();
    pixels.setPixelColor(4,0,0,0);
    pixels.show();
    pixels.setPixelColor(5,0,0,0);
    pixels.show();
    pixels.setPixelColor(6,0,0,0);
    pixels.show();
    pixels.setPixelColor(7,0,0,0);
    pixels.show();
    pixels.setPixelColor(8,0,0,0);
    pixels.show();



//Si se presiona el pulsador se encienden de color blanco poco a poco
//De forma ascendente todos los leds

   valorpulsador = digitalRead(pulsador);
   if (valorpulsador == HIGH) {

       pixels.setPixelColor(0,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(1,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(2,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(3,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(4,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(5,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(6,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(7,150,150,150);
       pixels.show();
       delay(2000);
       pixels.setPixelColor(8,150,150,150);
       pixels.show();
       delay(180000);
       pixels.setPixelColor(0,0,0,0);
       pixels.setPixelColor(1,0,0,0);
       pixels.setPixelColor(2,0,0,0);
       pixels.setPixelColor(3,0,0,0);
       pixels.setPixelColor(4,0,0,0);
       pixels.setPixelColor(5,0,0,0);
```

```
        pixels.setPixelColor(6,0,0,0);
        pixels.setPixelColor(7,0,0,0);
        pixels.setPixelColor(8,0,0,0);
        pixels.setPixelColor(9,0,0,0);
        pixels.show();
        delay(200);
       }
 }
```

## Actuación en Trayectos con Raquel Buil (segunda bailarina)

Para el festival de danza Trayectos se graba un documental que evidencia la colaboración entre electrónica y danza, además de con la música. A continuación se muestra el código implementado para establecer, gracias a la comunicación wifi, esta conexión entre disciplinas. Se trabaja con dos circuitos. El circuito sensorizador es el encargado de actualizar en tiempo real (en una URL dada) los valores generados por el sensor de movimiento MPU 92500/6500: se coloca sobre el arco del violín. El otro circuito es el receptor, el cual tiene acceso a los datos del sensor (misma URL) y comparándolos con una referencia establecida enciende o apaga una tira LED que se ha colocado en la mano derecha de la bailarina, a modo de pulsera. Incluimos los códigos relativos a la sensorización del arco del violín y su interactuación con la bailarina.

### Circuito receptor

```
//CIRCUITO RECEPTOR EN LA BAILARINA, ACTUACIÓN TRAYECTOS

//BIBIOTECA DEL ESP Y DEL MPU
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Adafruit_NeoPixel.h>

//PARÁMETROS TIRA LED
#define Tira1 14
#define NUMPIXELS 9
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, Tira1, NEO_GRB
+ NEO_KHZ800);


//Configuracion punto de acceso wifi
char * ssid_ap = "ESP8266_S";
char * password_ap = "trayectos";
IPAddress ip(192,168,13,4);
IPAddress gateway(192,168,13,4);
IPAddress subnet(255,255,255,0);
//Este es el servidor
ESP8266WebServer server;

float sensor_value = 0.0;

void setup() {
  WiFi.mode(WIFI_AP);
  WiFi.softAPConfig(ip, gateway, subnet);
  WiFi.softAP(ssid_ap, password_ap);
  Serial.begin(115200);
```

```
  Serial.print("IP Address: "); Serial.print(WiFi.localIP());
  //Configuracion de la ruta
  server.on("/", handleIndex);
  server.on("/update", handleUpdate);
  server.begin();

  //SALIDA LED (D6)
  pinMode(12,OUTPUT);

  pixels.begin();
  //Apagamoslos leds al inicio
  pixels.setPixelColor(0,0,0,0);
  pixels.setPixelColor(1,0,0,0);
  pixels.setPixelColor(2,0,0,0);
  pixels.setPixelColor(3,0,0,0);
  pixels.setPixelColor(4,0,0,0);
  pixels.setPixelColor(5,0,0,0);
  pixels.setPixelColor(6,0,0,0);
  pixels.setPixelColor(7,0,0,0);
  pixels.setPixelColor(8,0,0,0);
  pixels.show();
  delay(100);

}

void loop() {
  //Cogemos el valor proporcionado por el circuito sensorizador
  server.handleClient();
  //Si el valor es mayor que 2.0 encendemos la luz
  if(sensor_value>2.0){
       pixels.setPixelColor(0,0,255,0);
       pixels.setPixelColor(1,0,255,0);
       pixels.setPixelColor(2,0,255,0);
       pixels.setPixelColor(3,0,255,0);
       pixels.setPixelColor(4,0,255,0);
       pixels.setPixelColor(5,0,255,0);
       pixels.setPixelColor(6,0,255,0);
       pixels.setPixelColor(7,0,255,0);
       pixels.setPixelColor(8,0,255,0);
       pixels.show();
       delay(10);
  }
  else {
    //Si es menor que 2.0 las apagamos
     pixels.setPixelColor(0,0,0,0);
     pixels.setPixelColor(1,0,0,0);
     pixels.setPixelColor(2,0,0,0);
     pixels.setPixelColor(3,0,0,0);
     pixels.setPixelColor(4,0,0,0);
     pixels.setPixelColor(5,0,0,0);
     pixels.setPixelColor(6,0,0,0);
     pixels.setPixelColor(7,0,0,0);
     pixels.setPixelColor(8,0,0,0);
     pixels.setPixelColor(0,0,0,0);
     pixels.show();
     delay(10);
  }
}

void handleIndex(){
  //Para actualizar la página
```

```
  server.send(200, "text/plain", String(sensor_value));

void handleUpdate(){
  //URL
  sensor_value= server.arg("value").toFloat();
  Serial.println(sensor_value);
  server.send(200, "text/plain", "Updated");}
```

## Circuito sensorizador

```
//CIRCUITO SENSORIZADO DEL VIOLÍN, ACTUACIÓN TRAYECTOS
 //BIBIOTECA DEL ESP Y DEL MPU
 #include <ESP8266WiFi.h>
 #include "MPU9250.h"

 //Inicializacion MPU
 float volts = 0.0, acel = 0.0;

 //Inicialización wifi (punto de acceso)
 char * ssid = "ESP8266_S";
 char * password = "trayectos";
 char * host = "192.168.13.4";
 //Este es el cliente
 WiFiClient client;

 //Inicializacion MPU,I2C
 MPU9250 IMU(Wire,0x68);
 int status;

 void setup() {
  //Definimos puertos del SDA y SCL
  Wire.begin(D4,D1);
  pinMode(A0,INPUT);
  Serial.begin(115200);

  //Para conectarse con el servidor
  WiFi.begin(ssid,password);
  while(WiFi.status() != WL_CONNECTED){
  Serial.print(".");
  delay(500);
  }
  Serial.print("IP Address (AP): "); Serial.println(WiFi.localIP());
  status = IMU.begin();
}
void loop() {
  IMU.readSensor();
  //Cogemos la aceleracion en x del sensor
  acel = IMU.getAccelX_mss();
  //Nos conectamos ala URL
  if(client.connect(host,80)){
    String url = "/update?value=";
    url += String(acel);
    client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host " +
host + "\r\n" + "Connection: keep-alive\r\n\r\n");
    while(client.available()){
      String line = client.readStringUntil('\r');
      Serial.print(line);
    }
  }
}
```