David Romero Puyal

PARED INTERACTIVA DE ENTRENAMIENTO PARA ESCALADA

TRABAJO DE FIN DE GRADO

Dirigido por Enric Vidal Idiarte

Grado de Ingeniería Electrónica Industrial y Automática

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2021

Índice

1 Introducción

La idea principal de este proyecto es construir un prototipo de una pared de escalada interactiva y funcional para realizar entrenamientos y competiciones, aplicando los conocimientos adquiridos durante los cuatro años de carrera cursados. El proyecto realizado ha sido diseñado para un caso real y para poder utilizarse en un futuro.

Una pared interactiva de escalada, normalmente, es una pared pequeña, de entre 2 y 5 metros de ancho por 3 o 4 metros de alto, llena de presas de escalada con una cierta inclinación. Estas presas suelen ser muy pequeñas y llevan asociados LEDs que pueden estar encendidos o apagados según si la presa que tienen asociada se ha de usar o no. Los LEDs para encender se seleccionan desde un dispositivo móvil vía Bluetooth. En los modelos más completos, la inclinación de la pared también es seleccionable.

En este proyecto, se han realizado tres demostradores:

- El demostrador principal de este proyecto es un prototipo de pared interactiva de entrenamiento de escalada. Esta pared contiene 8 presas de escalada. Cada presa tiene un LED asociado que se enciende y apaga vía Bluetooth gracias a una aplicación móvil realizada con la herramienta APP INVENTOR del MIT. Además, dos presas cuentan con sensores de presión para monitorizar el tiempo de ascensión de una vía de escalada. Se ha incorporado una pantalla LCD para mostrar el tiempo de subida una vez el usuario acabe la vía. Este tiempo de subida también se envía por Bluetooth a la aplicación móvil. Finalmente, se ha añadido una matriz de 8x8 LEDs para ilustrar como se podrían encender muchos más LEDs de otras presas de escalada.
- Por otra parte, se ha realizado un demostrador de gestión de los LEDs de una pared de escalada con 128 presas.
- Finalmente, en el apartado de propuestas de mejora, se ha propuesto una idea para poder controlar la inclinación de la pared interactiva de escalada gracias a la gestión de dos motores paso a paso fijados a una cadena cada uno. Se ha implementado el control de uno de estos motores.

El trabajo que se ha realizado en este proyecto ha sido el siguiente:

- Se ha montado una pared interactiva de escalada con 8 presas diferentes, estas presas llevan asociado un LED que es controlado vía Bluetooth. Además, cada LED de cada presa de escalada lleva asociado otro LED de una matriz de 8x8 LEDs que se comunica con el microcontrolador vía SPI.
- Se ha incorporado la mejora de añadir sensores de presión para monitorizar el tiempo de ascensión de cualquier vía de escalada.
- Se ha creado una aplicación con la herramienta del MIT APP INVENTOR, *http://ai2.appinventor.mit.edu/*, para permitir controlar los LEDs de la pared interactiva con un dispositivo móvil y visualizar los datos de tiempo de ascensión después de cada vía.
- Se ha incorporado una pantalla LCD para mostrar el tiempo de ascensión de las vías en caso de competiciones de escalada.
- Se ha propuesto una idea para poder reclinar la pared de escalada incorporando dos motores paso a paso.
- Se ha realizado un demostrador para una pared interactiva de 128 presas agrupadas en 8 columnas por 16 filas.

2 Objetivos

El objetivo principal de este proyecto es diseñar, programar, construir y probar una pared interactiva de escalada completamente funcional y con el menor coste posible.

Para cumplir con este objetivo principal, se han de aplicar muchos de los conocimientos adquiridos en el transcurso de la titulación; involucrando conceptos tratados en varias asignaturas.

En este proyecto, se ha realizado un trabajo de manera autónoma bajo la supervisión del profesor Enric Vidal.

De los conceptos que se van a necesitar, quiero destacar la necesidad de interpretar documentos técnicos proporcionados por los fabricantes de componentes electrónicos y la capacidad de interpretación de éstos, el uso de conocimientos de hardware para el montaje de la pared interactiva y el uso de conocimientos de software para la programación de la pared interactiva y del demostrador de 128 presas.

3 Estado del arte

Actualmente hay muy pocas empresas en todo el mundo que se dedican a la fabricación de paredes de escalada interactivas. Las dos empresas más importantes son:

MoonBoard. *www.moonboard.com*. Fabricante de la pared de nombre MoonBoard.
Setter closet. *www.settercloset.com*. Fabricante de la pared de nombre Kilter Board.

La empresa MoonBoard es una empresa de Reino Unido que vende un solo modelo de pared interactiva con dos inclinaciones fijas posibles. Si solo se quiere adquirir los tablones de madera con las presas de escalada y los LEDs ya programados, el precio oscila en torno a los 4000 €. Si se quiere adquirir la pared de escalada montada y funcional el precio oscila en torno a los 10000 €.
Los dos modelos tienen las siguientes especificaciones:

- La pared con inclinación de 40º mide 2.44m por 4.86 m por 3.554 m (ancho, profundo, alto).
- La pared con inclinación de 25º mide 2.44m por 4.86 m por 3.7 m (ancho, profundo, alto).

Estos modelos tienen 220 presas cada una con un LED incorporado.

Figura 1. Pared de escalada MoonBoard de la empresa MoonBoard. [1]

En la figura 1, se puede observar la pared Moonboard con algunas presas. Cada presa tiene un LED que se puede configurar con 3 colores distintos. El color verde es para indicar que la presa de escalada se ha de usar como presa de salida, el color azul es para indicar que la presa de escalada está dentro de la vía escogida y el color rojo es para indicar que la presa de escalada es la presa de fin de vía.
El modelo de la figura 1 es el más completo ya que no necesita una estructura detrás para fijarse a la pared. Este modelo tiene un precio que se aproxima a los 10115 euros.

Uno de los aspectos que hacen tan atractiva esta pared de escalada, es que este modelo es el mismo en todo el mundo. Esto permite que, desde cualquier parte, se puedan diseñar vías de escalada y guardarlas en una aplicación móvil. Estas vías de escalada son accesibles por cualquier persona que disponga de la aplicación móvil y de esta pared. De esta manera, se pueden realizar competiciones online conectando diferentes países separados miles de quilómetros.

La empresa SetterCloset (fabricante de Kilter Board) es una empresa estadounidense que vende 2 modelos de paredes interactivas.

Por una parte, vende paredes no reclinables para particulares de tamaños entre 2.2 y 3.1 metros de ancho por 3.1 y 3.7 metros de alto e inclinaciones variables.

Por otra parte, vende la mejor pared interactiva del mercado. Esta pared mide 3.7 por 3.7 metros, es reclinable entre 0 y 70 grados, consta de 323 piezas y 323 LEDs que se iluminan de colores diferente según si la presa es de la vía seleccionada, se usa solo como presa para apoyar los pies, se usa como presa de inicio o se usa como presa de fin de vía.
Esta pared tiene un precio de, aproximadamente, 32500 € en España. Actualmente solo hay una pared de estas ubicada en el nuevo rocódromo de Madrid Sharma Climbing.

Figura 2. Pared de escalada Kilter Board de la empresa SetterCloset. [2]

En la figura 2 se puede observar la pared Kilter Board de rocódromo.
Igual que la MoonBoard, al ser el mismo modelo en todo el mundo y las presas de escalada tener la misma orientación, las vías que se pueden marcar y escoger son iguales en cualquier parte.

Aunque el modelo es muchísimo mejor que el de la empresa MoonBoard, la poca publicidad de la pared y el precio que tiene hacen que no haya muchos modelos como este instalados actualmente.

4 Material necesario

Para la realización del proyecto se ha usado el siguiente material:

Proyecto de la pared interactiva de escalada.

- Pared de madera de 70 cm de alto por 40 cm de ancho.
- 8 presas de escalada.
- 8 tornillos y 8 arañas para colocar las presas en la tabla.
- Microchip ATMEGA2560.
- Módulo Bluetooth HC-05.
- Pantalla LCD 16x2 con el controlador HD44780.
- 2 sensores resistivos de presión.
- Cables de conexión.
- 8 resistencias de 1 kΩ.
- 2 resistencias de 10 kΩ.
- 4 placas Protoboard pequeñas.

Demostrador de 128 presas.

- Microchip ATMEGA2560.
- Módulo Bluetooth HC-05.
- 2 chips MAX7219 con 2 matrices de 8x8 LEDs controlados por SPI.
- Cables de conexiones.

Gestión de la inclinación.

- Microchip ATMEGA2560.
- Módulo Bluetooth HC-05.
- Motor Paso a Paso modelo 28BYJ-48.
- Cables de conexiones.

Material adicional.

- Multímetro digital.
- 2 placas Protoboard.

5 Pared interactiva de escalada

5.1 Hardware

Para la programación de la pared interactiva de escalada se ha usado el entorno de programación Microchip Studio de Microchip.

Primero de todo, para poder probar el buen funcionamiento del programa se ha construido en dos placas Protobard el circuito que ha de ir implementado en la pared interactiva.

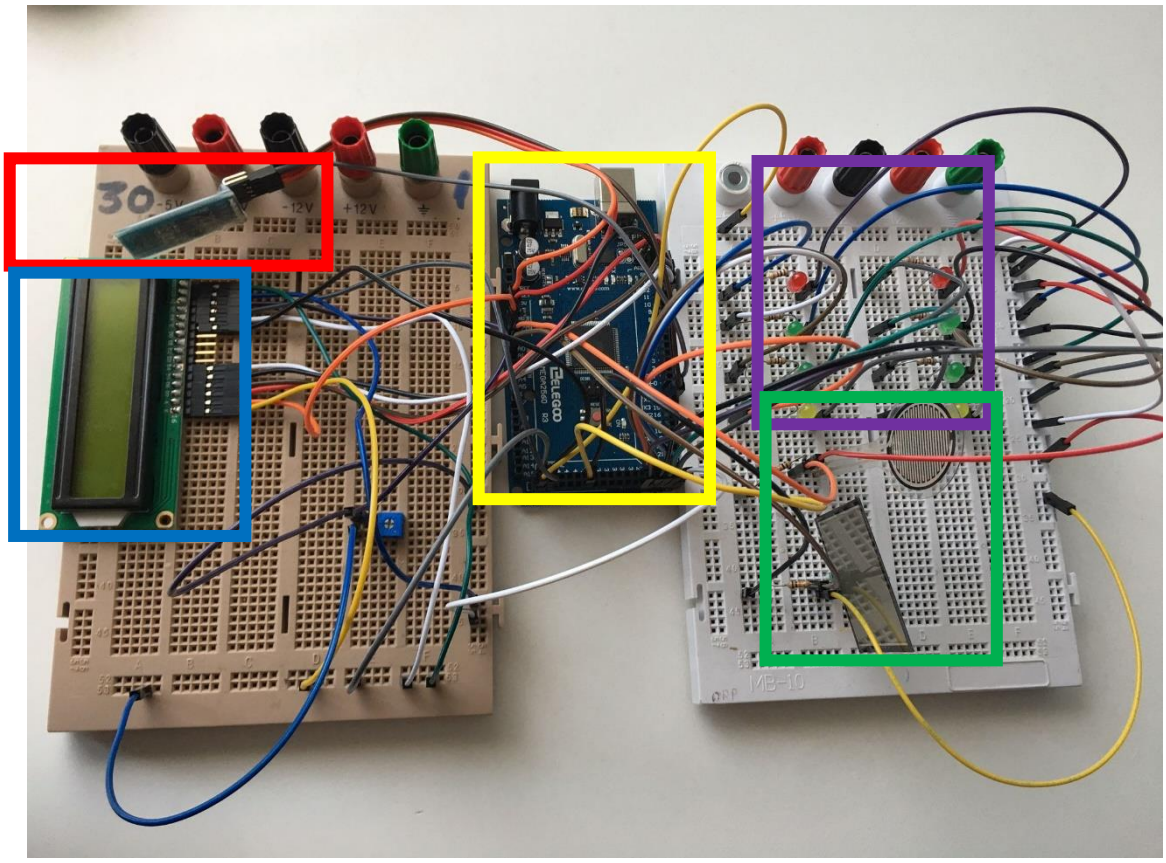

Figura 3. Implementación en dos placas Protoboard del circuito de la pared interactiva de escalada.

En la figura 3, se puede observar el circuito de la pared interactiva de escalada. Este montaje se ha realizado para comprobar la programación del microcontrolador ATMEGA2560 antes de montar la pared de escalada. Una vez el programa funcionó correctamente, el circuito se montó en la pared interactiva de escalada.

Cada recuadro de la figura 3 se corresponde a una parte diferente del circuito.

- Recuadro amarillo → Microcontrolador ATMEGA 2560.
- Recuadro violeta → LEDs.
- Recuadro verde → Sensores de presión.
- Recuadro rojo → Módulo Bluetooth HC-05.
- Recuadro azul → Pantalla LCD 16x2.
- Aunque no esté presente en la figura 3, se ha incorporado una matriz de 64 LEDs controlada con el chip MAX7219.
- Para controlar la pared interactiva de escalada, es necesario un dispositivo móvil que disponga de la aplicación que se ha creado con la herramienta APP INVENDOR del MIT.

El diagrama de bloques a nivel hardware de la pared interactiva de escalada será el siguiente.

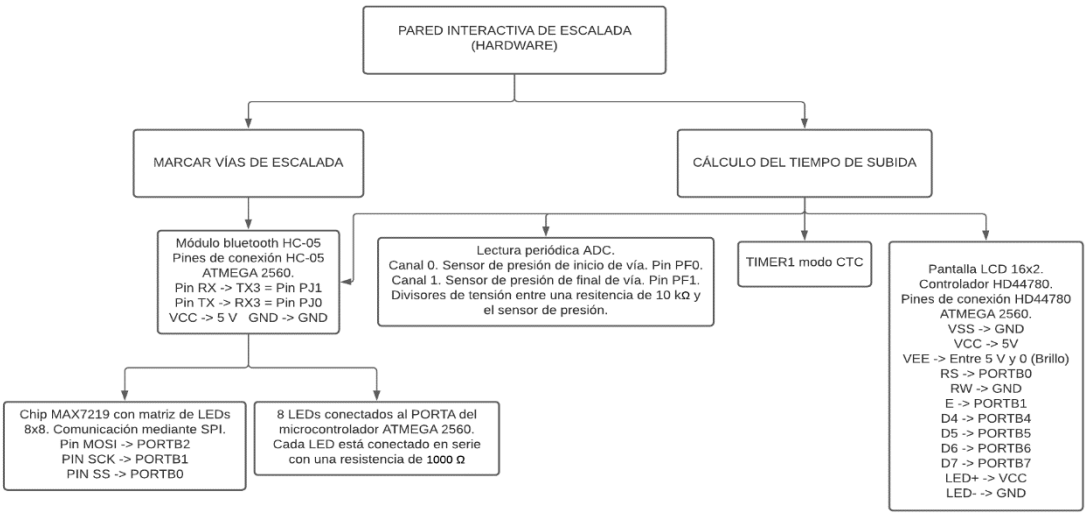

Figura 4. Diagrama de bloques del hardware asociado a la pared interactiva de escalada.

En la figura 4 de la página 10 se muestra el diagrama de bloques del hardware necesitado para la realización del prototipo de pared interactiva de escalada.

Para marcar las vías de escalada es necesario un módulo Bluetooth HC-05 que permita la conexión Bluetooth entre un dispositivo móvil (que disponga de la aplicación de control de la pared interactiva de escalada creada) y el microcontrolador ATMEGA 2560. La información se envía y se recibe a través del puerto USART3 del microcontrolador.
Para el marcaje de vías de escalada también son necesarios los 8 LEDs conectados al PORTA del microcontrolador y situados debajo de cada presa de escalada y la matriz de LEDs que se comunica con el microcontrolador vía SPI.

Para monitorizar el tiempo de ascensión de las vías de escalada es necesario el uso de los puertos 0 y 1 del conversor analógico digital del microcontrolador para leer la información proporcionada por los sensores de presión. Para contar el tiempo de ascensión se usa el TIMER1 en modo CTC. Para mostrar el tiempo de ascensión calculado se ha incorporado una pantalla LCD 16x2 controlada con el controlador HD44780. El tiempo de ascensión también se envía al dispositivo móvil por el puerto USART3 del micro (donde está conectado el módulo Bluetooth HC-05).

5.1.1 Microcontrolador ATMEGA 2560

Para realizar el programa de la pared interactiva de escalada se ha usado el microcontrolador ATMEGA 2560 que viene incorporado a un Arduino MEGA. Este microcontrolador se corresponde al recuadro amarillo de la figura 3.

Cabe mencionar que, antes de programar con el entorno ATMEL STUDIO de Microchip, los dos proyectos realizados se programaron con el entorno de ARDUINO IDE para comprobar si los circuitos funcionaban correctamente y si era posible realizar el programa con el microcontrolador ATMEGA2560.

Las especificaciones del microcontrolador ATMEGA 2560 son las siguientes:

- Memoria de programa: Flash de 256 kB.
- Memoria SRAM: 8192 Bytes.
- Memoria EEPROM: 4096 Bytes.
- Número de instrucciones: 135.
- 86 entradas y salidas.
- 32 registros de trabajo de 8 bits.
- 1 contador en tiempo real.
- Timers: 2 timers de 8 bits. 4 timers de 16 bits.
- Periféricos de comunicación: 4 Puertos UART, 5 puertos SPI, 1 puerto I2C.
- PWM: 4 de 8 bits de resolución y 12 de resolución programable entre 2 y 16 bits.
- 1 convertidor A/D de 16 canales y 10 bits.
- Rango de temperatura de funcionamiento entre -40 ℃ y 85 ℃.
- Tensión de alimentación entre 4.5 V y 5.5 V.
- Velocidad CPU: 16 MHz.

El diagrama de pines del microcontrolador ATMEGA2560 es el siguiente.

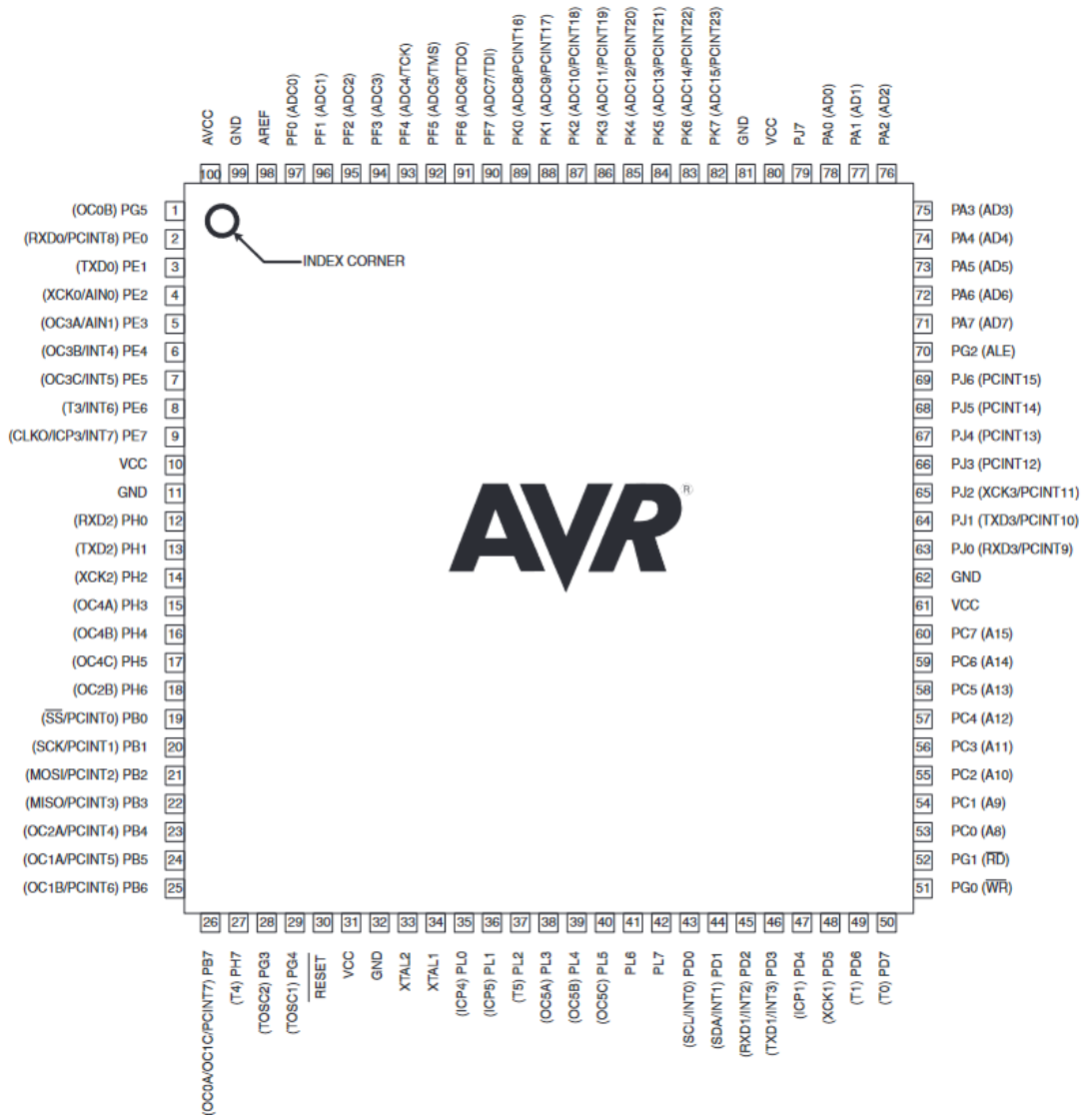

Figura 5. Diagrama de pines del microcontrolador ATMEGA 2560. [3]

En la figura 5 se puede observar la distribución de los pines del microcontrolador usado para realizar este proyecto. Para programar el microcontrolador desde un Arduino MEGA, se necesitó el diagrama de pines del Arduino MEGA para saber cómo estaban conectados los pines del Arduino con el microcontrolador ATMEGA2560.

El diagrama de pines utilizado para programar el microcontrolador ATMEGA2560 desde un Arduino MEGA es el siguiente.

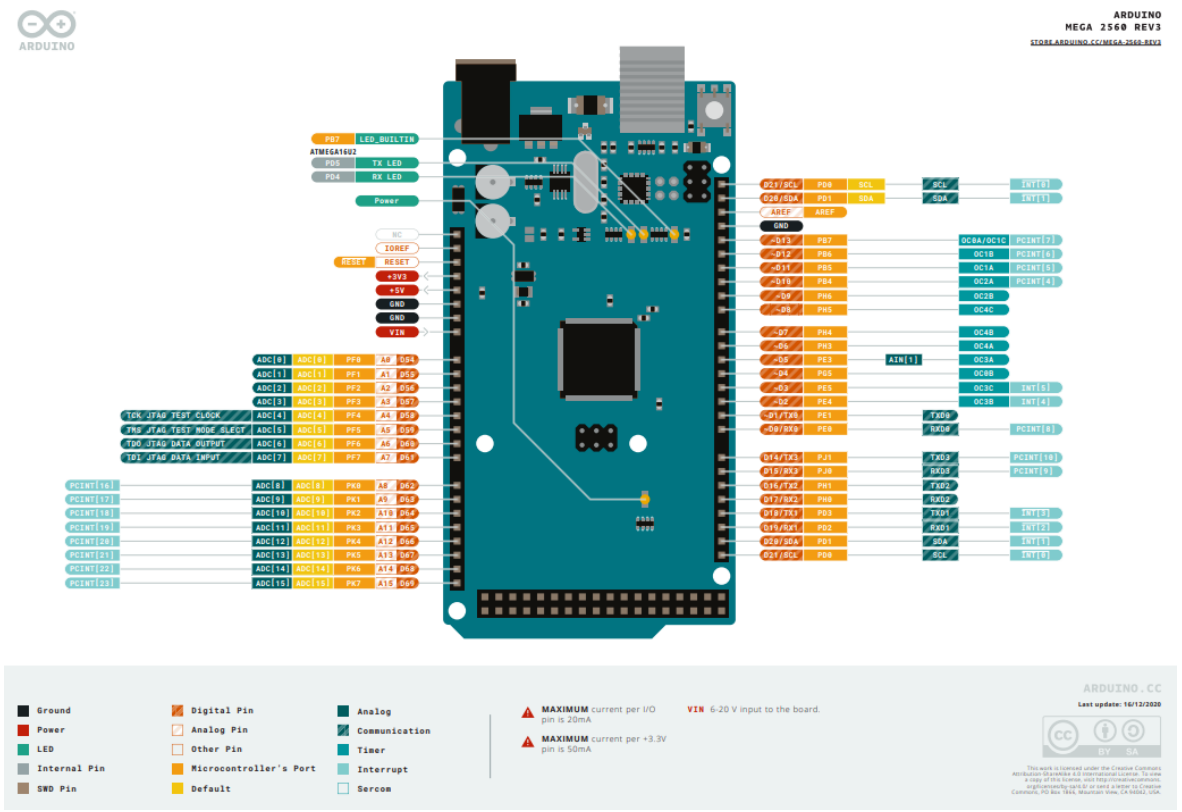

Figura 6. Diagrama de pines de la placa Arduino MEGA. [4]

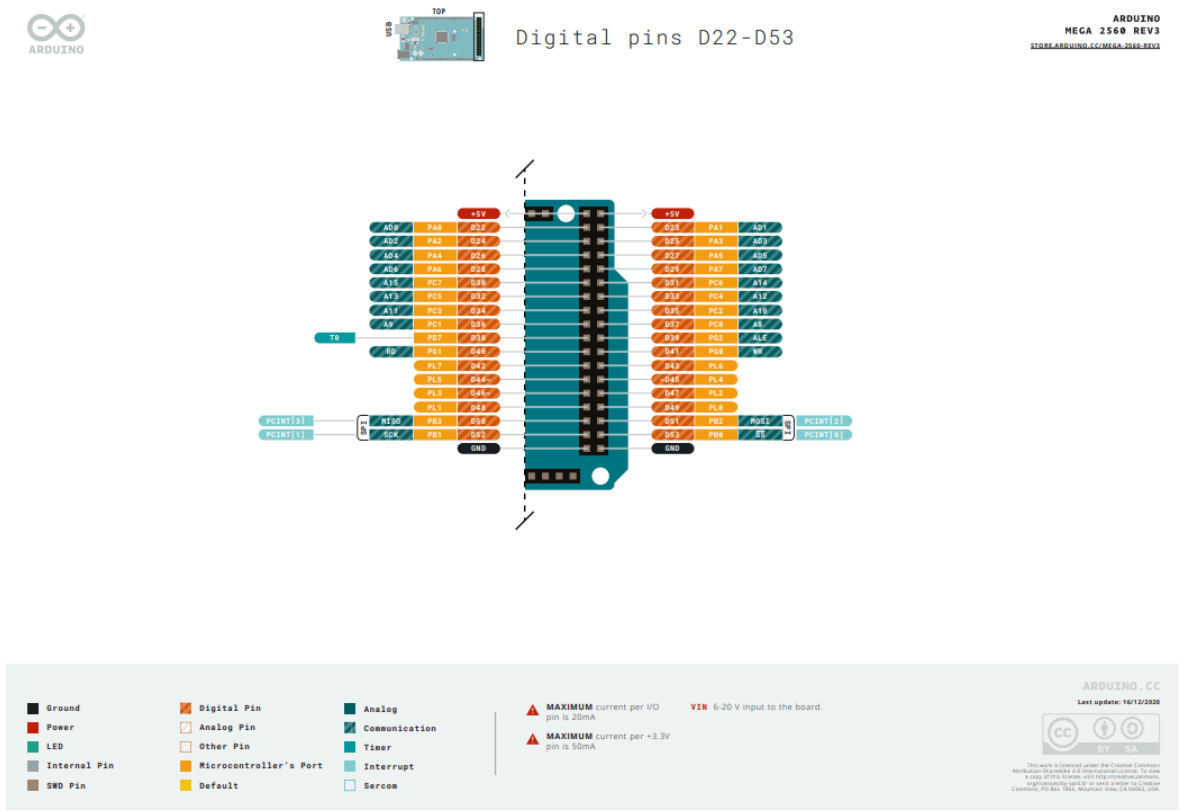

Figura 7. Diagrama de pines de la placa Arduino MEGA. [5]

5.1.2 LEDs

Para la realización de la pared de escalada se han escogido 8 LEDs de diferentes colores. Estos LEDs se pueden observar en el recuadro violeta de la figura 3. Para que cada LED brille con suficiente luminosidad sin correr riesgo de dañarse, se ha forzado que la corriente de alimentación de los LEDs sea de 5 mA.
Como un valor alto en la salida de los puertos digitales del microcontrolador equivale a 5 V, se ha puesto una resistencia de 1000 Ω en serie con cada LED.

El esquema circuital de cada LED sería el siguiente:

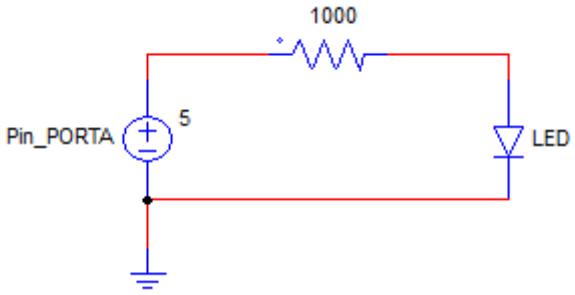

Figura 8. Esquema del circuito que gestiona cada LED.

$$I_{LED} = \frac{5\ V}{1000\ \Omega} = 5\ mA$$

Estos LEDs están conectados a los pines correspondientes del PORTA del microcontrolador ATMEGA2560. De esta manera, los LEDs se pueden encender y apagar aplicando máscaras OR y AND sobre el PORTA.
Además, en la figura 3 se puede observar que los LEDs son de distintos colores. He optado por utilizar LEDs de color amarillo para indicar las presas de inicio de vía, LEDs de color verde para indicar las presas intermedias y LEDs de color rojo para indicar las presas de fin de vía.
Los LEDs que se han comprado son de la marca ATPWONZ de 5 mm de diámetro y una corriente máxima de excitación de 20 mA.

5.1.3 Sensores de presión

En el recuadro verde de la figura 3, se pueden observar 2 sensores de presión diferentes pensados para asientos de coche. Estos sensores son de tipo resistivo sensibles a la presión. La fuerza que pueden medir oscila entre 0 y 30 kg por centímetro cuadrado. La presión máxima que pueden aguantar es de 2.94 MPa.

$$\frac{30\ kg}{cm^2} \cdot \frac{9.8\ N}{1\ kg} \cdot \frac{10000\ cm^2}{1\ m^2} = 2940000\ Pa$$

Estos sensores, varían su resistencia cuando algo o alguien ejerce presión sobre ellos. Si no hay ninguna presión, la resistencia de estos sensores tiende a infinito (superior a 10 MΩ). Conforme se ejerce presión sobre los sensores, la resistencia de estos disminuye hasta alcanzar unos pocos Ohms.

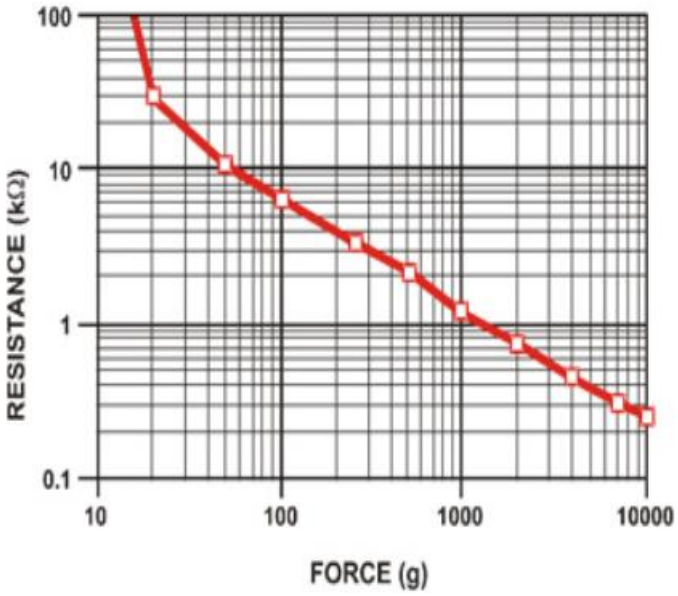

Figura 9. Gráfica del comportamiento de un sensor resistivo en función de la fuerza ejercida.[6]

En la figura 9 se muestra, en una gráfica logarítmica, el comportamiento de un sensor de presión resistivo. Se puede observar cómo, si la fuerza ejercida sobre el sensor es nula, la resistencia de este tiende a infinito y, conforme la fuerza ejercida sobre el mismo aumenta, la resistencia del sensor disminuye.

Cabe mencionar, que estos sensores no se pueden colocar encima de las presas de escalada, ya que si no la gente se resbalaría al cogerlos. En el prototipo de pared interactiva construido, estos sensores se han colocado entre la presa de escalada y la pared, de tal forma que, cuando una persona se cuelgue de la parte superior de la presa, la parte inferior hará palanca y presión al sensor. Esto implica que, si nadie está utilizando la presa de escalada con el sensor resistivo acoplado, el sensor tendrá resistencia infinita y, si alguien se cuelga de esa misma presa, el sensor variará su resistencia notablemente.

Tras probar los dos sensores, el sensor redondo y el sensor cuadrado tienen el mismo funcionamiento, pero el sensor redondo tiene un precio menor y mayor robustez mecánica. Por tanto, para la realización del proyecto se ha escogido este.

El sensor redondo, es un sensor de fuerza de capa fina pensado para electrónica del automóvil. Este sensor es de la marca Oumefar y se ha comprado a través de Amazon España.
Los datos del fabricante son los siguientes:

- Modelo: MD30-60.
- Longitud: 60 mm.
- Anchura: 5.8 mm.
- Espesor: 0.60 mm.
- Peso: 1 g.
- Rango de medición entre 0 y 30 kg.
- Punto de respuesta menor de 200 g.
- Durabilidad mayor de 1 millón de ciclos.
- Tiempo de respuesta menor a 1 ms.
- Tiempo de recuperación menor a 15 ms.
- Temperatura de trabajo entre-20 ℃ y 60 ℃
- Resistencia inicial > 10 MΩ.
- Precio: 9.79 €.

El comportamiento de estos sensores resistivos variables permite detectar si un usuario se está colgando de la presa mediante un circuito divisor de tensión como el siguiente:

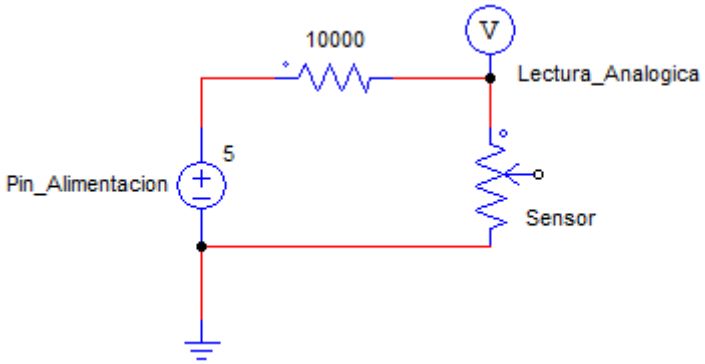

Figura 10. Esquema del circuito que gestiona los sensores de presión.

La alimentación de los circuitos de detección de presión se corresponde a dos pines digitales del microcontrolador en estado alto.
Seguidamente, se hace un divisor de tensión entre una resistencia de 10 kΩ y el sensor de presión.
Entre la resistencia y el sensor de presión se hace una lectura analógica del valor de tensión por los canales 0 y 1 del convertidor analógico digital del microcontrolador. El canal 0 es para la presa de salida y el canal 1 es para la presa de final.
Como el ADC del microcontrolador es de 10 bits, el valor de lectura oscilará entre 0 y 1024 (0 = 0 V y 1024 = 5 V). Se ha establecido una referencia de presión tal que la resistencia variable del sensor sea inferior a 10 kΩ. Tras medir con el multímetro la resistencia para diferentes presiones, se ha deducido que los sensores son muy sensibles y cualquier presión hace que la resistencia del sensor sea menor de 10 kΩ por lo que el microcontrolador detectará si un usuario está colgado de las presas de inicio y final sin ningún problema.

El establecer como referencia de presión una resistencia del sensor de 10 kΩ implica que la tensión que ha de caer por la resistencia fija de 10 kΩ ha de ser igual o mayor de 2.5 V. Este valor se corresponde a 512. Por lo tanto, si el valor de lectura del conversor analógico digital es igual o menor a 512, se entenderá que hay una persona haciendo presión sobre la presa.

$$V_{Sensor} = V_{PinAlimentacion} \cdot \frac{R_{Sensor}}{10\ k\Omega + R_{Sensor}}$$

$$0 < V_{Sensor} < 1024$$

5.1.4 Módulo Bluetooth HC-05

En la parte izquierda de la figura 3 se puede observar, en el recuadro rojo, el módulo Bluetooth HC-05.

Este módulo Bluetooth, permite la comunicación entre el microcontrolador y un dispositivo móvil a través de una aplicación realizada con la herramienta del MIT APP INVETOR.
Para realizar la comunicación entre el dispositivo móvil y el microcontrolador se usa el puerto USART3.

El conexionado de los pines entre el microcontrolador ATMEGA2560 y el módulo Bluetooth HC - 05 será el pin RX del módulo con el pin TX3 del microcontrolador (PJ1) y el pin TX del módulo con el pin RX3 del microcontrolador (PJ0) tal y como se muestra en el diagrama de flujo de hardware del circuito.

El módulo Bluetooth utilizado tiene las siguientes especificaciones:

- Modelo: CZ-HC-05 (gomcu).
- Dimensiones: 4.4 x 1.6 x 0.7 cm.
- Funciona como dispositivo maestro y esclavo Bluetooth.
- Configurable mediante comandos AT.
- Bluetooth V2.0 + EDR.
- Frecuencia de operación: 2.4 GHZ Banda ISM.
- Modulación: GFSK (Gaussian Frequency Shift Keying).
- Seguridad: Autenticación y encriptación.
- Distancia de hasta 10 metros en condiciones óptimas.
- Voltaje de operación entre 3.6 VDC y 6 VDC.
- Consumo de corriente entre 30 mA y 50 mA.
- Chip: BC417143.
- Baudios Soportados: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- Temperatura de trabajo entre -20 ℃ y 75 ℃.
- Precio: 10 €.

5.1.5 Pantalla LCD 16x2

En el recuadro azul de la figura 3, se puede observar la pantalla LCD 16x2 controlada con el controlador HD44780. Esta pantalla usa 4 pines del PORTB para la transmisión de datos. También tiene conectado un potenciómetro para ajustar el brillo de la pantalla.

La conexión de la pantalla LCD al microcontrolador es la siguiente.

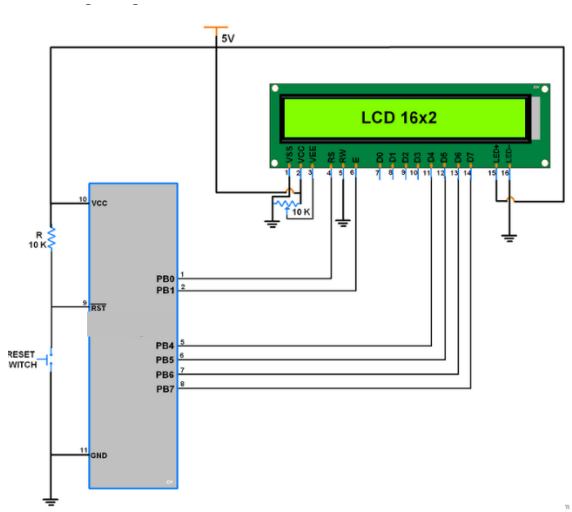

Figura 11. Esquema de conexión de la pantalla LCD al microcontrolador ATMEGA2560. [7]

En la figura 11 se puede observar el diagrama completo de conexión de la pantalla LCD. La conexión exacta de todos los pines se muestra en el diagrama de flujo hardware del circuito (figura 4).

La pantalla LCD 16x2 utilizada, es de la marca OcioDual y ha sido adquirida a través de Amazon España. Esta pantalla tiene las siguientes especificaciones.

- Fabricante: Gamefox
- Modelo: 80510.
- Controlador: HD44780.
- Alimentación: 5 VDC
- Dimensiones: 80 x 36 x 12 (mm).

5.1.6 Matriz de 64 LEDs controlada con el chip MAX7219

Como no se ha podido montar una pared de tamaño comercial con muchas presas, se ha pegado una matriz de 64 LEDs, controlada con el chip MAX7219, al prototipo montado. Cada LED de la pared interactiva de escalada tiene asociado un LED de esta matriz, de tal forma que, cuando se encienda un LED de la pared interactiva de escalada, se encenderá el LED correspondiente en la matriz de LEDs.
La finalidad de incorporar esta matriz de LEDs es demostrar cómo se gestionaría el encendido y apagado de los LEDs de una pared de escalada real.

La matriz de LEDs, controlada con el chip MAX7219, incorporada a la pared interactiva de escalada es la siguiente.

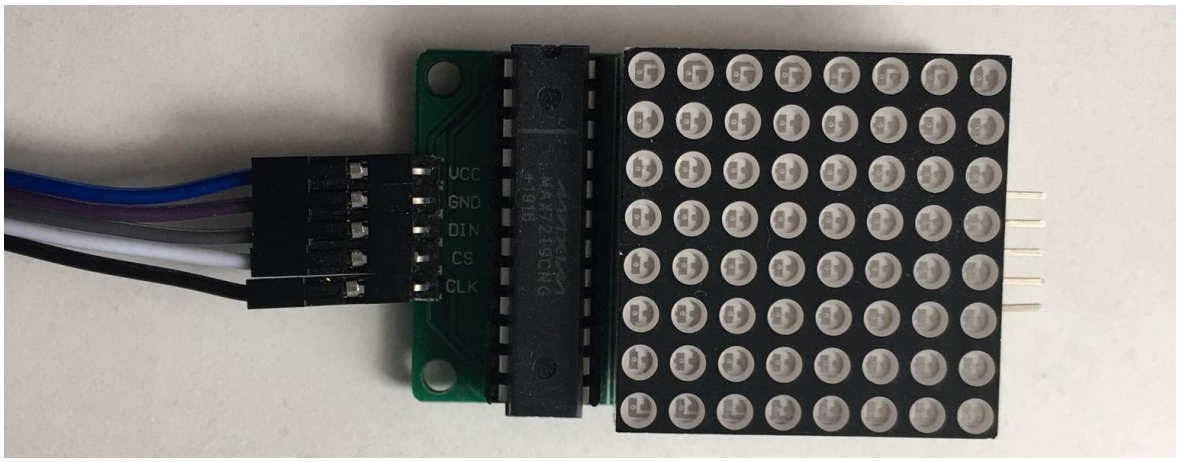

Figura 12. Matriz de LEDs controlada con el chip MAX7219.

En la figura 12 se muestra una matriz de 64 LEDs junto con el chip MAX7219 y los 5 pines necesarios para conectarlo al microcontrolador.

La principal ventaja del chip MAX7219 es que se comunica vía SPI con el microcontrolador y solo se necesitan 3 pines para controlar múltiples matrices de LEDs 8x8 conectadas en cascada. Esto permite encender y apagar los LEDs de varias matrices de manera independiente usando el PORTB2 como pin MOSI, el PORTB1 como pin SCK y el PORTB0 como pin SS. También se necesita un pin de alimentación de 5 V y un pin de tierra.

Las características principales del chip MAX7219 son las siguientes.

- Controlador de led de 8 dígitos. (64 LEDs individuales o 7 segmentos).
- Incorpora un decoder BCD code-B.
- Realiza multiplexados.
- Contiene una memoria RAM de 8x8.
- Interfaz Serial 4 Wire.
- Tensión de alimentación entre 4 V y 5.5 V.
- 24 pines.
- Rango de funcionamiento entre 0 ℃ y 70 ℃.

Se maneja con 3 pines y es compatible con SPI, QSPI, MICROWIRE.

Además, la configuración de los pines del chip MAX7219 es la siguiente.

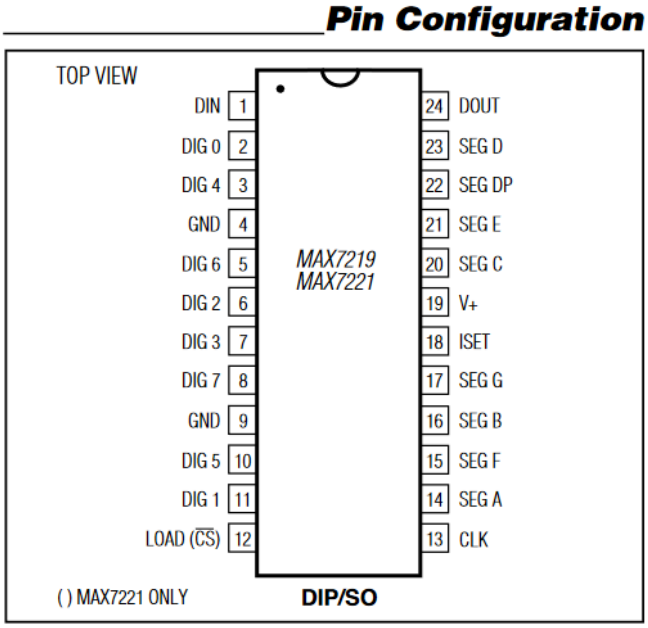

Figura 13. Configuración de los pines del chip MAX7219. [8]

En la figura 13, se muestra la distribución de pines del chip MAX7219. Se puede observar como este chip tiene acoplados 8 pines de dígitos y 8 pines de segmentos que realizan multiplexados y permiten controlar matrices de 8x8 o displays de 7 segmentos.

La conexión entre el chip MAX7219 y la matriz 8x8 es la siguiente.

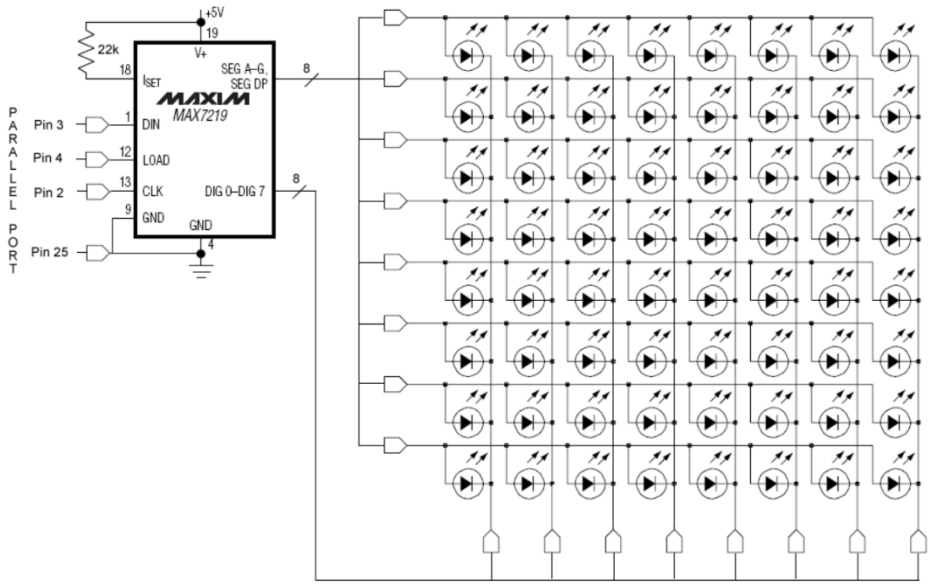

Figura 14. Conexión entre el chip MAX7219 y la matriz de 8x8 LEDs. [9]

En la figura 14 se muestra como está conectada la matriz de LEDs de 8x8 con el chip MAX7219. El multiplexado de está matriz y es por ánodo común.

Los 8 pines de segmentos controlan las filas y por defecto no tienen tensión, mientras que los 8 pines de dígitos controlan las columnas y por defecto tienen tensión.
Para que un LED se encienda ha de pasar una cierta corriente a través de éste. Para que la corriente pase ha de haber una cierta tensión en la fila asociada a ese LED (que por defecto no tiene tensión) y no ha de haber tensión en la columna asociada a ese LED (que por defecto si tiene tensión). De esta manera, se proporciona un camino para que la corriente circule a través del LED y, en consecuencia, el LED deseado se encienda.

La manera de crear este camino de corriente por software será utilizando dos registros de control. El primer registro será el encargado de enviar la fila que se ha de encender, mientras que el segundo registro será el encargado de enviar una máscara con los bits de esa fila que se quiera encender.

Si por ejemplo se quieren encender tres LEDs de la fila 3, habrá que enviar primero el registro de fila con el valor 3 (0b00000111), y después una máscara de las columnas a apagar para forzar una circulación de corriente a través de los LEDs. Cabe mencionar que, en este chip, el direccionamiento de las columnas está negado. Si se quiere quitar tensión de la columna para cerrar el camino de corriente, se ha de enviar un 1, y, si se quiere alimentar esa columna para impedir la circulación de corriente, se ha de enviar un 0.

5.2 Software

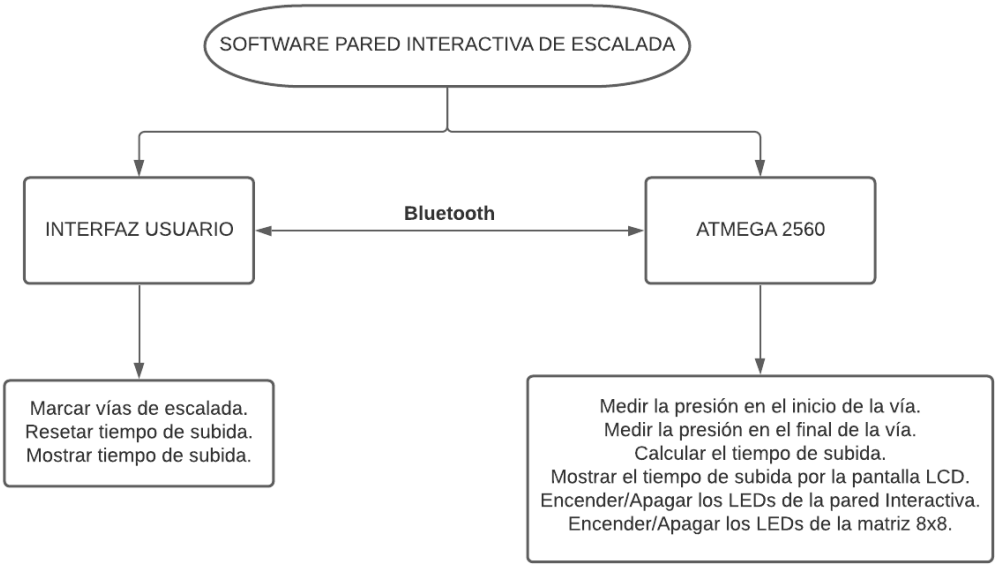

Figura 15. Diagrama de flujo del software de la pared interactiva de escalada.

En la figura 15 se muestra el diagrama de flujo general del programa realizado para el plafón interactivo de entrenamiento de escalada.

El programa entero se encuentra en el Anexo1 de este documento (páginas 63-80).

El programa realizado tiene dos partes bien diferenciadas. Por una parte, la interfaz con el usuario y por otra parte el microcontrolador ATMEGA2560.
En la parte de interfaz, los usuarios pueden gestionar los LEDs de la pared interactiva de escalada y de la matriz 8x8 para marcar las vías de escalada deseadas.
En la parte del microcontrolador, se especifican todas las tareas que este ha de realizar para garantizar el buen funcionamiento del prototipo construido. Por una parte, el microcontrolador ha de gestionar tanto los LEDs del plafón de escalada como de la matriz de LEDs, según las consignas enviadas vía Bluetooth desde la interfaz del usuario. Por otra parte, el microcontrolador ha de leer la información procedente de los sensores de presión, contar y gestionar el tiempo de subida, enviar el resultado del tiempo de subida al interfaz usuario vía Bluetooth y mostrar el tiempo de subida en la pantalla LCD 16x2.

Finalmente, las dos funciones principales de este programa son:

- Marcar las vías de escalada.
- Contar, calcular y mostrar el tiempo de subida de las vías.

5.2.1 Interfaz de usuario

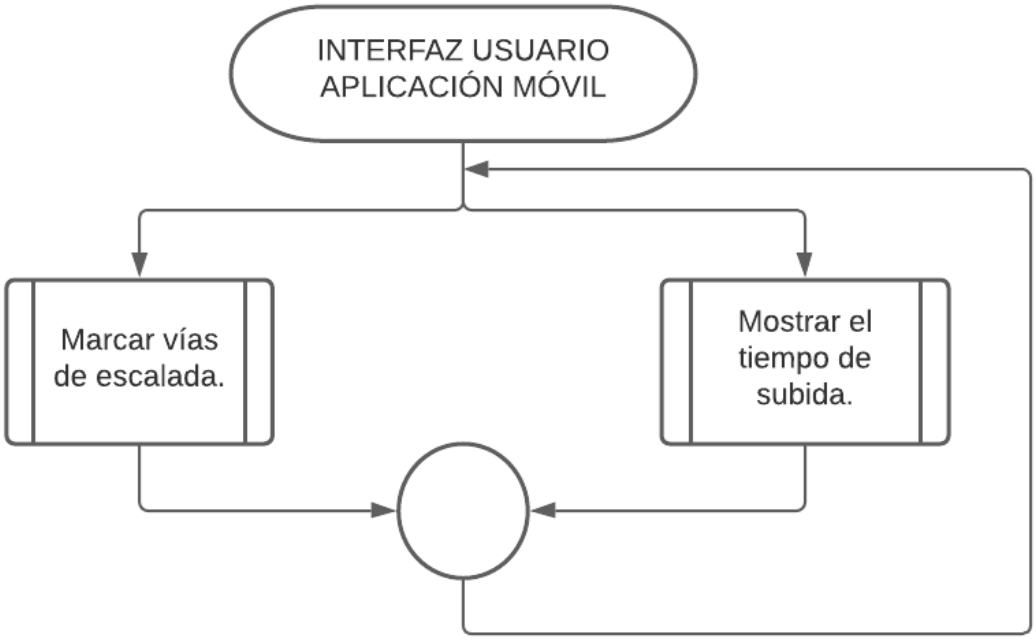

Figura 16. Diagrama de flujo del software implementado para gestionar la interfaz con el usuario de la pared interactiva de escalada.

En la figura 16 se muestra el diagrama de flujo general de la parte de programación encargada de gestionar la interfaz con el usuario.

Se dispone de una aplicación creada con la herramienta APP INVENTOR de MIT.
Desde esta aplicación se pueden enviar ciertos caracteres al microcontrolador ATMEGA2560 pulsando los diferentes botones que se han programado. El microcontrolador analizará los caracteres recibidos y realizará la acción correspondiente. De esta manera se pueden marcar diferentes vías de escalada.
También se dispone de un pequeño terminal donde se puede visualizar el tiempo de subida de la vía de escalada

5.2.1.1 Marcaje de vías de escalada

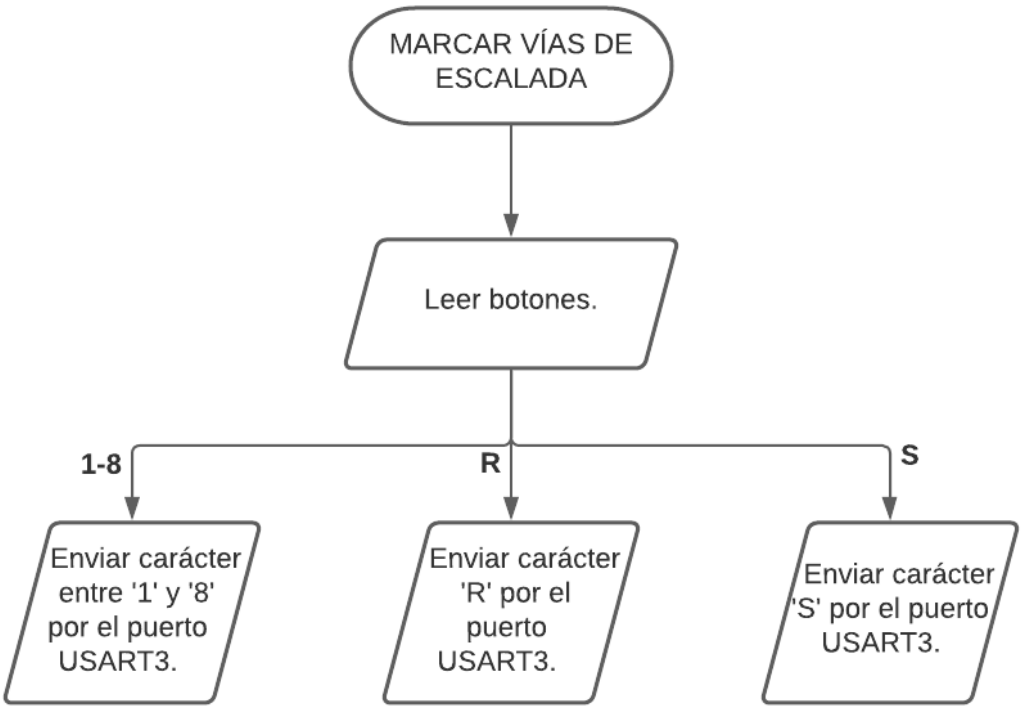

Figura 17. Diagrama de flujo del marcaje de las vías de escalada.

En la figura anterior se muestra la gestión del interfaz con el usuario en lo referente al marcaje de las vías de escalada.

La aplicación móvil diseñada tiene varios botones.
Si se pulsan los botones asociados a los LEDs, se envía un carácter entre el '1' y el '8' dependiendo del LED que se quiera encender. Si se pulsa el botón para apagar todos los LEDs se envía el carácter 'R'. Si se pulsa el botón para reiniciar el tiempo de subida, se envía el carácter 'S'. Este último carácter no se utiliza en el marcaje de vías de escalada, pero es necesario para reiniciar los timers y reiniciar las variables de tiempo en el caso de que un usuario se caiga realizando la vía de escalada.

Al enviar alguno de estos caracteres por el puerto USART3 del microcontrolador, el programa entra en la rutina de servicio a la interrupción ISR (USART3_RX_vect) donde se analiza el carácter recibido y se realiza una acción u otra dependiendo de este carácter.

El análisis de los caracteres recibidos en la rutina de servicio a la interrupción ISR (USART3_RX_vect) es el siguiente.

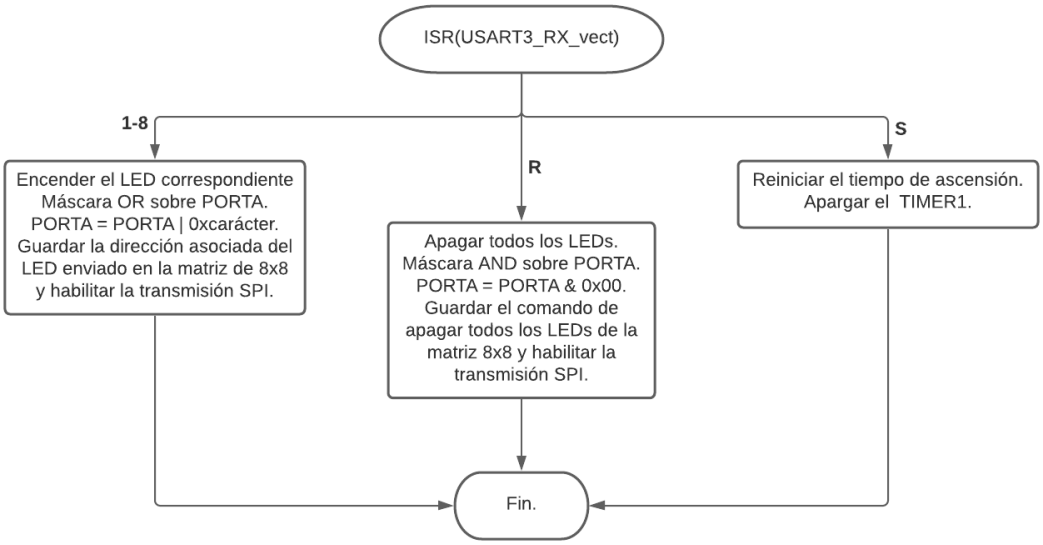

Figura 18. Diagrama de flujo de la rutina de servicio a la interrupción ISR (USART3_RX_vect).

En la figura 18 se puede observar cómo se gestiona el carácter recibido vía Bluetooth por el puerto USART3 en la rutina de servicio a la interrupción ISR (USART3_RX_vect). El código correspondiente a esta rutina se encuentra en las páginas 69 y 70 de este documento.

El marcaje de vías se hace usando máscaras OR y AND sobre el PORTA. Cada LED tiene un carácter asociado entre el '1' y el '8'. De esta manera, cada número que se recibe indica el bit de máscara OR para encender del PORTA.
Este carácter recibido, también se guarda en una variable encargada de almacenar esta información (cadena de caracteres) y se habilita la transmisión SPI. En el programa principal se llama periódicamente a la función de transmisión SPI; Si la transmisión está habilitada, en esta función se analizará la cadena de caracteres recibida y se encenderá el LED asociado a dicha matriz, posteriormente se deshabilitará la transmisión.

Si se recibe el carácter 'R', el microcontrolador apaga todos los LEDs del PORTA haciendo una máscara AND con el número 0x00.
De la misma manera que en el caso anterior, se guardará en la cadena de caracteres el carácter 'R' correspondiente al reinicio de las vías marcadas y se habilitará la transmisión SPI.

Si se recibe el carácter 'S' se reinicia el tiempo de ascensión y se apaga el TIMER1.

5.2.1.2 Visualización del tiempo de subida

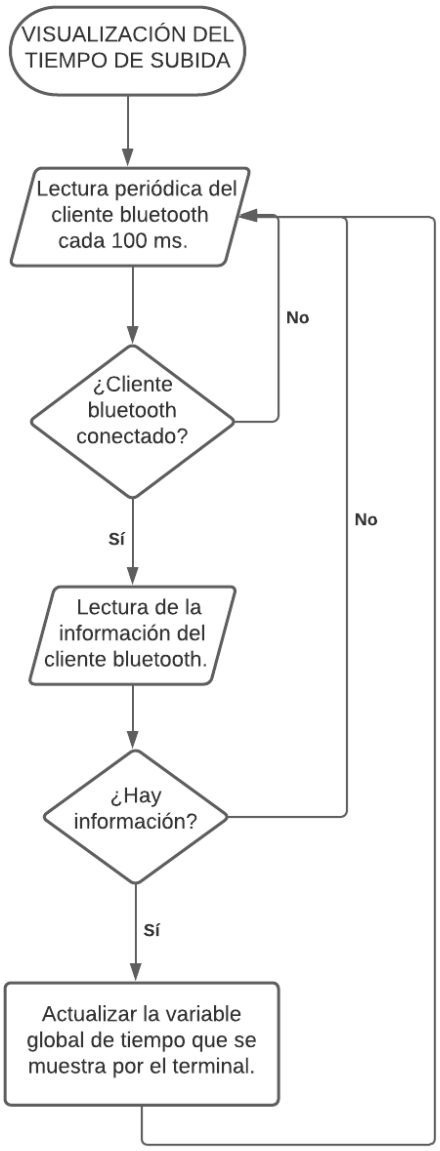

Figura 19. Diagrama de flujo asociado a la visualización del tiempo de subida.

En la figura anterior se muestra un diagrama de flujo sobre cómo se ha programado el proceso para visualizar el tiempo de subida en el terminal de la aplicación móvil.

Para visualizar el tiempo de subida, primero se comprueba periódicamente cada 100 ms si el cliente Bluetooth de la aplicación móvil está conectado. Si el cliente Bluetooth lo está, se comprueba si tiene información. En caso afirmativo se actualiza la variable que se muestra por el terminal de la aplicación móvil con el nuevo tiempo de subida.

5.2.2 Programa principal ATMEGA2560

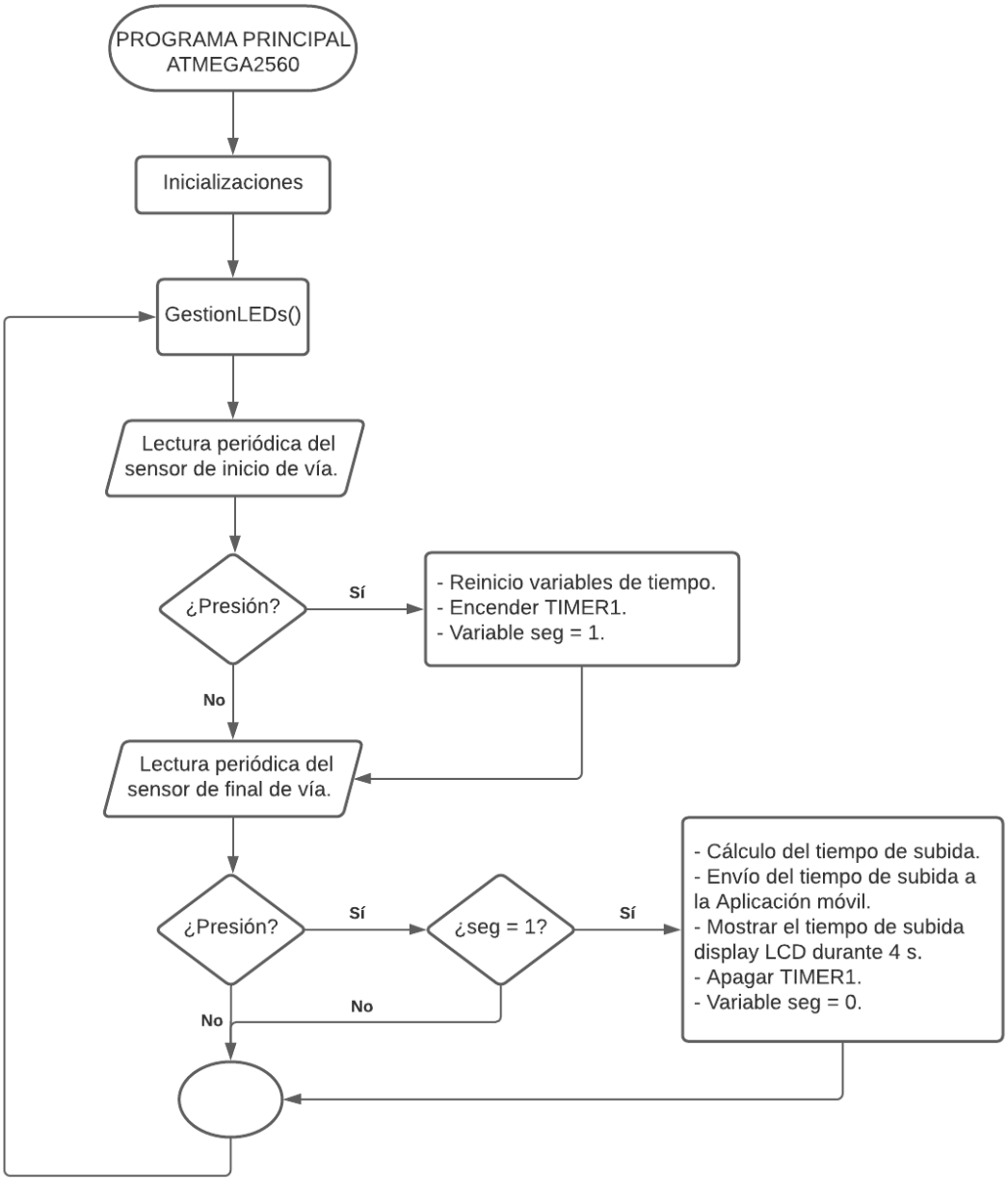

Figura 20. Diagrama de flujo del software implementado para gestionar el tiempo de subida.

En la figura 20 de la página anterior se muestra el diagrama de flujo general del programa realizado con el microcontrolador ATMEGA2560. Este programa tiene las siguientes funciones:

- Gestionar los LEDs de la matriz de LEDs 8x8 controlada con el chip MAX7219. La información de que LEDs se habían de encender, o si se había de apagar la matriz, se guardaba en una cadena de caracteres en la rutina de servicio a la interrupción ISR (USART3_RX_vect).
- Monitorizar el tiempo de subida de una vía de escalada. El momento para empezar y dejar de contar se detecta gracias a los sensores de presión y el contaje se realiza en la rutina de servicio a la interrupción ISR (TIMER1_COMPA_vect).
- Mostrar el tiempo de ascensión por un display LCD 16x2 durante 4 segundos.
- Enviar el tiempo de ascensión vía Bluetooth a un dispositivo móvil a través de la rutina de servicio a la interrupción ISR (USART3_TX_vect).

La función GestionLEDs es la función encargada de encender o apagar los 8 LEDs (configurados en la matriz 8x8) a través de la comunicación SPI entre el microcontrolador ATMEGA2560 y el chip MAX7219.

En la rutina de servicio a la interrupción ISR (USART3_RX_vect), cuyo diagrama de flujo se encuentra en la figura 18 de este documento, se guardaba en una variable global la cadena de caracteres correspondiente a los LEDs a encender (o apagar) y se habilitaba la transmisión SPI. En el programa principal, se llama periódicamente a la función GestionLEDs que, si la transmisión está activada, leerá y analizará la cadena de caracteres llenada y realizará una acción u otra dependiendo del valor de esta cadena.

En este demostrador, como solo hay 8 LEDs, la cadena de caracteres podrá tener las siguientes combinaciones. (La gestión de una pared de 128 presas de escalada está explicada en el demostrador número 2).

- Por una parte, si se quiere encender un LED, la función GestionLEDs, leerá una cadena de caracteres con la fila (A, B, C o D) y la columna (1 o 2) donde se encuentra el LED que se quiere encender. Finalmente, leerá el carácter ‘X’ que significa el fin de la cadena de caracteres.
- Por otra parte, si se quieren apagar todos los LEDs, la función GesionLEDs, leerá una cadena de caracteres con el carácter ‘R’ indicando que se han de apagar todos los LEDs de la matriz, seguido del carácter ‘X’ indicando fin de cadena.

Las posibles cadenas de caracteres a analizar según la posición del LED o la acción a realizar son las siguientes.

Encender LEDs			Reiniciar Matriz
	1	2	
A	A1X	A2X	RX
B	B1X	B2X	
C	C1X	C2X	
D	D1X	D2X	

El contaje del tiempo de subida de una vía de escalada empieza cuando el usuario se cuelga de la presa de inicio y acaba cuando el usuario se cuelga de la presa de fin de vía.
Para detectar esto, se dispone de un conversor ADC que lee periódicamente por dos canales la tensión que cae en la resistencia de 10 kΩ que está conectada en serie con los sensores de presión creando un divisor de tensión.
Si se detecta una caída de tensión inferior a 2.5 V en la resistencia conectada en serie con el sensor de presión de inicio de la vía, se enciende el TIMER1 y se reinician las variables de contaje y envío de tiempo. Una caída de tensión de 2.5 V equivale a un valor inferior a 512 por el canal 0.

EL TIMER1 está configurado en modo comparación y están habilitadas las interrupciones. Esto implica que, cuando el TIMER1 se enciende, el programa entra en la rutina de servicio a la interrupción ISR (TIMER1_COMPA_vect) cada milisegundo. En esta rutina de servicio a la interrupción se incrementa una variable global cada vez que se entra tal y como se muestra en la figura 21. El código correspondiente a esta rutina se encuentra en la página 72 de este documento.

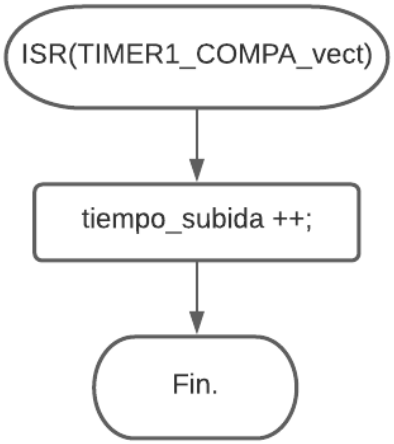

Figura 21. Diagrama de flujo de la rutina de servicio a la interrupción ISR(TIMER1_COMPA_vect).

Cuando el usuario se cuelga de la presa de fin de vía, el ADC leerá un valor inferior a 512 por el canal 1 y se parará el TIMER1.

Una vez el TIMER1 esté apagado, se enviará el resultado del tiempo de subida en milisegundos vía Bluetooth y por el puerto USART3 al dispositivo móvil gracias a la utilización de la rutina de servicio a la interrupción ISR (USART3_TX_vect) tal y como se muestra en la figura 22. El código correspondiente a esta rutina se encuentra en la página 70 de este documento.

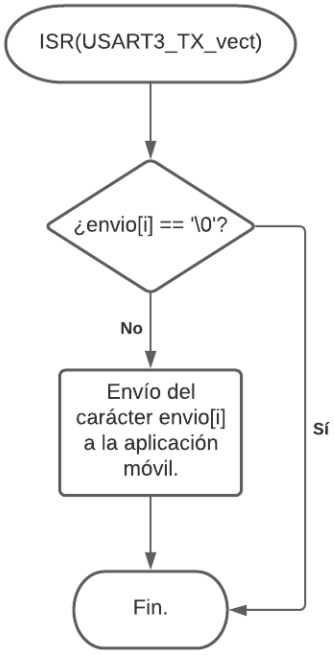

Figura 22. Diagrama de flujo de la rutina de servicio a la interrupción ISR(USART3_TX_vect).

Para enviar el tiempo de subida se ha de pasar la variable de tiempo a cadena de caracteres y enviar carácter a carácter este tiempo.

Además, el resultado del tiempo de subida se mostrará en formato XXs.XXXms en el display LCD 16x2. Para hacer esto, el tiempo de subida se pasa de milisegundos al formato mencionado y se envía carácter a carácter a la pantalla LCD. El resultado de tiempo permanece en el display durante 4 segundos.

Si el usuario se cae en medio de la vía, la aplicación móvil dispone de un botón de reinicio de tiempo que envía el carácter 'S' al microcontrolador. La gestión del carácter 'S' se encuentra en las figuras 17 y 18 de este documento.
Si el microcontrolador recibe el carácter 'S' por el puerto USART3, este reinicia el tiempo de ascensión y apaga el TIMER1. Cuando una persona vuelva a tocar la presa de inicio, se repite el ciclo normal.

Finalmente, cabe mencionar que se ha contemplado la posibilidad de que una persona pise con el pie la misma presa en la que está el sensor de presión reiniciando el TIMER1.
Para solucionar esto, se ha creado una variable de seguridad "seg", que garantiza que solo se pueda apagar el TIMER1 y reiniciar la variable de tiempo con el botón de reinicio o colgándose de la presa de final de vía. Si el usuario pisa la presa de inicio realizando una vía, la función que gestiona el ADC ignorará este dato.

5.3 Aplicación móvil

El plafón de escalada interactivo se controla desde una aplicación realizada con la herramienta del MIT APP INVETOR. *http://ai2.appinventor.mit.edu/*.
Esta herramienta proporciona un espacio para diseñar la aplicación y otro espacio para programarla en forma de diagrama de bloques.

El diseño de la aplicación que controla la pared interactiva de escalada es el siguiente.

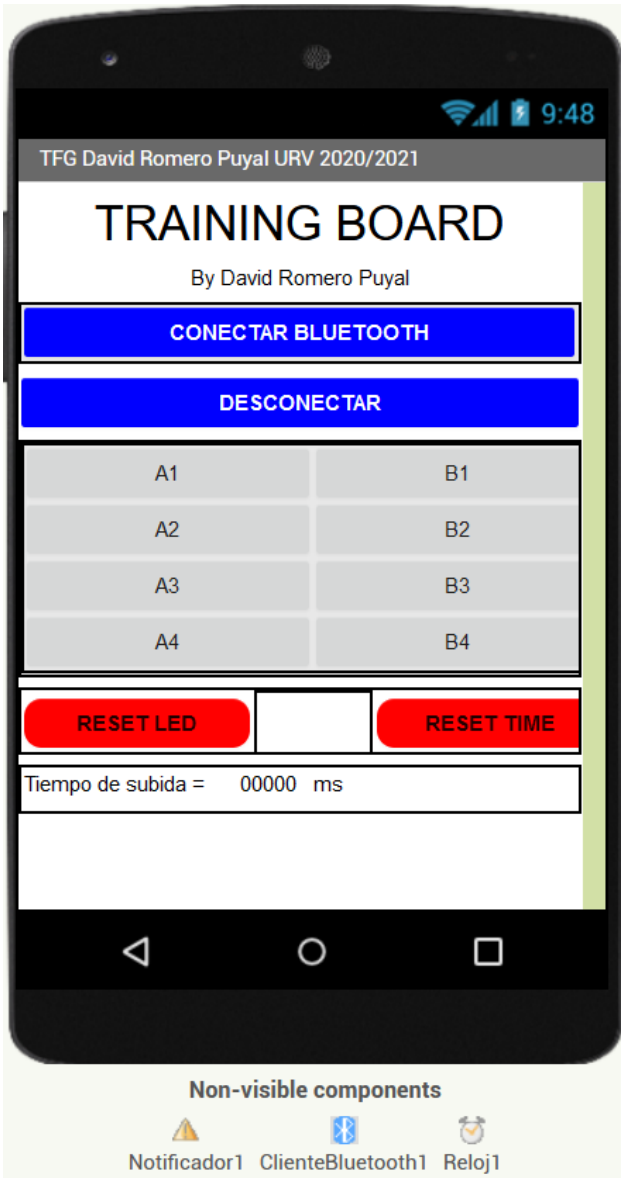

Figura 23. Diseño de la aplicación que controla la pared interactiva de escalada.

La aplicación dispone de dos botones para conectarse y desconectarse al terminal Bluetooth HC-05.

También dispone de 8 botones A1, A2, A3, A4, B1, B2, B3, B4 que envían un carácter del '1' al '8' y sirven para encender cada LED de la pared de escalada por separado.

Se han incorporado 2 botones más: uno para apagar todos los LEDs que envía el carácter 'R' al microcontrolador y otro para reiniciar el tiempo de subida en caso de que el usuario caiga al hacer la vía. Este último envía el carácter 'S'.

En la parte inferior, la aplicación dispone de un pequeño terminal que se actualiza cada vez que hay una nueva ascensión correcta de la vía de escalada con el tiempo de subida.

Finalmente, se dispone de tres componentes no visibles.

- Un primer componente para permitir a la aplicación hacer uso del Bluetooth como cliente.
- Un segundo componente que actúa como notificador para indicar si el usuario se ha conectado o desconectado del módulo HC-05.
- Un tercer componente que se corresponde a un reloj y es necesario para realizar la recepción de los datos del tiempo de subida.

La programación en diagrama de bloques de la aplicación móvil es la siguiente.

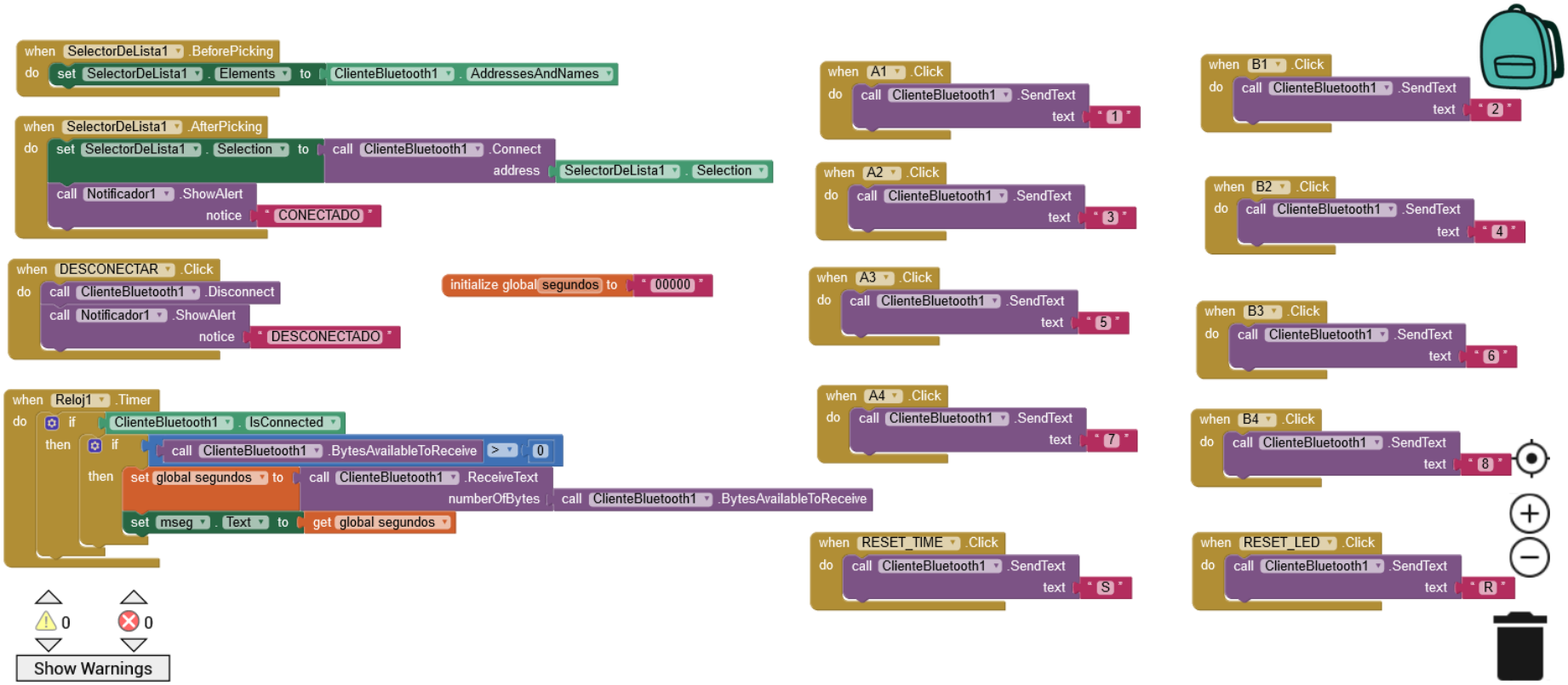

Figura 24. Programación de la aplicación que controla la pared interactiva de escalada en diagrama de bloques.

En la parte derecha del diagrama de bloques de la aplicación móvil se puede observar la programación de los 8 botones que encienden y apagan los LEDs de la pared de escalada. También se puede observar el botón de apagado de los LEDs y el botón de apagado del TIMER1 con el reinicio del tiempo de subida.
Si se pulsa alguno de los 10 botones se llama al elemento cliente Bluetooth y se envía el texto correspondiente al microcontrolador. El microcontrolador interpreta este texto como un carácter que lo analiza en la rutina de servicio a la interrupción ISR (USART3_RX_vect).

En la parte superior izquierda se observa la programación de conexión entre el móvil y el módulo Bluetooth. Una vez se pulsa el botón de conectar Bluetooth aparecerá una lista de los dispositivos disponibles, en esta lista se ha de seleccionar el dispositivo con el nombre HC-05. Si se pulsa el botón de desconectar, el teléfono móvil deja libre el módulo Bluetooth. También se dispone de un notificador que avisa si un usuario se ha conectado o desconectado del módulo Bluetooth.

Finalmente, en la parte inferior izquierda del diagrama de bloques se observa la programación del terminal por donde se muestra el tiempo de ascensión de la vía de escalada. Para la programación de este terminal se dispone de una variable global llamada segundos que es la variable que se actualiza con el nuevo tiempo de ascensión y de un reloj. El reloj comprueba periódicamente si el dispositivo móvil está conectado al módulo HC-05. En caso afirmativo, comprueba si se ha enviado información desde el terminal HC-05. Si hay nueva información, actualiza la variable global segundos con el nuevo tiempo de ascensión.

6 Demostrador de 128 presas

Para que el producto se pueda usar, hace falta disponer de un gran número de presas para hacer diferentes combinaciones y tener disponible un amplio abanico de vías de escalada. Para plasmar esto, se ha programado un demostrador de 128 LEDs que representa una pared con 128 presas de escalada diferentes.

6.1 Hardware

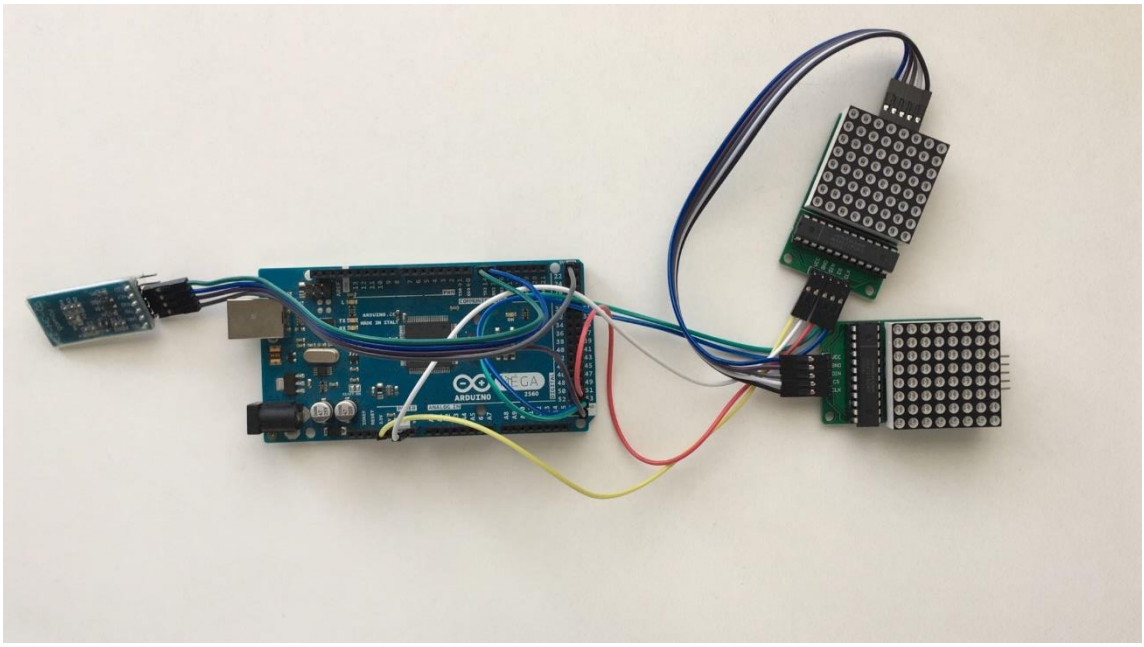

Figura 25. Hardware del demostrador de 128 presas de escalada.

En la figura 25 se puede observar el hardware del demostrador construido.

Este demostrador de 128 LEDs ha sido programado con el microprocesador ATMEGA2560 usando el entorno de programación Microchip Studio de Microchip.
El demostrador tiene dos chips MAX7219 conectados en cascada cuya función es controlar vía SPI una matriz de LEDs 8x8. Finalmente se ha añadido un módulo Bluetooth HC-05 para encender los LEDs desde un dispositivo móvil usando un terminal Bluetooth.

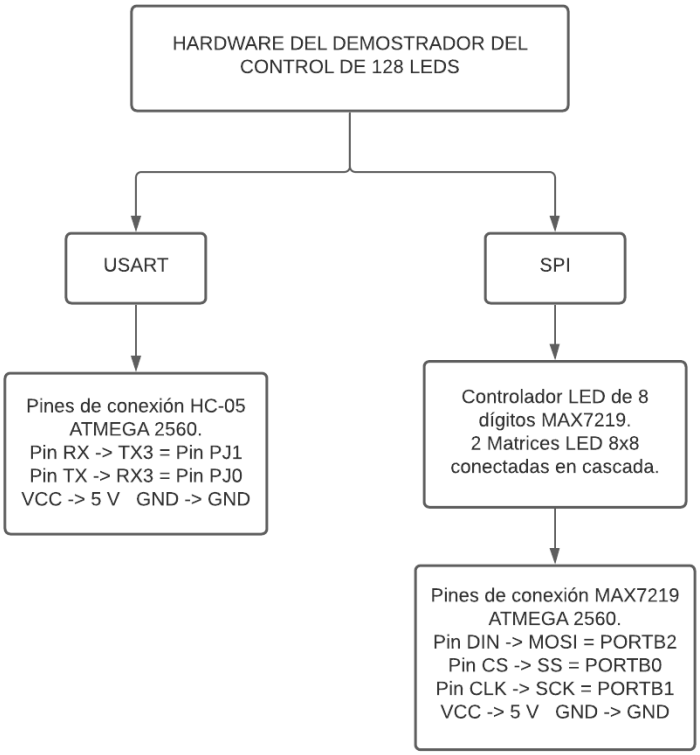

Figura 26. Diagrama del hardware implementado para gestionar el demostrador de 128 LEDs.

En la figura 26 se muestra el diagrama de bloques del hardware del demostrador de presas construido. Este demostrador de presas se comunica con el microcontrolador vía SPI.

En el hardware, se ha utilizado un módulo Bluetooth HC-05 que sirve para comunicar el microcontrolador con un terminal Bluetooth. Desde el terminal Bluetooth del móvil, se enviará una cadena de caracteres correspondiente a los LEDs que se quieran encender. También se dispone de dos chips MAX7219 conectados en cascada que sirven para controlar desde el microcontrolador vía SPI las matrices de LEDs 8x8. Como ya se ha mencionado anteriormente, la ventaja del chip MAX7219 es que solo se necesitan 3 pines para controlar múltiples matrices de LEDs 8x8 conectadas en cascada. Esto permite encender y apagar 128 LEDs de manera independiente usando el PORTB2 como pin MOSI, el PORTB1 como pin SCK y el PORTB0 como pin SS.

6.2 Software

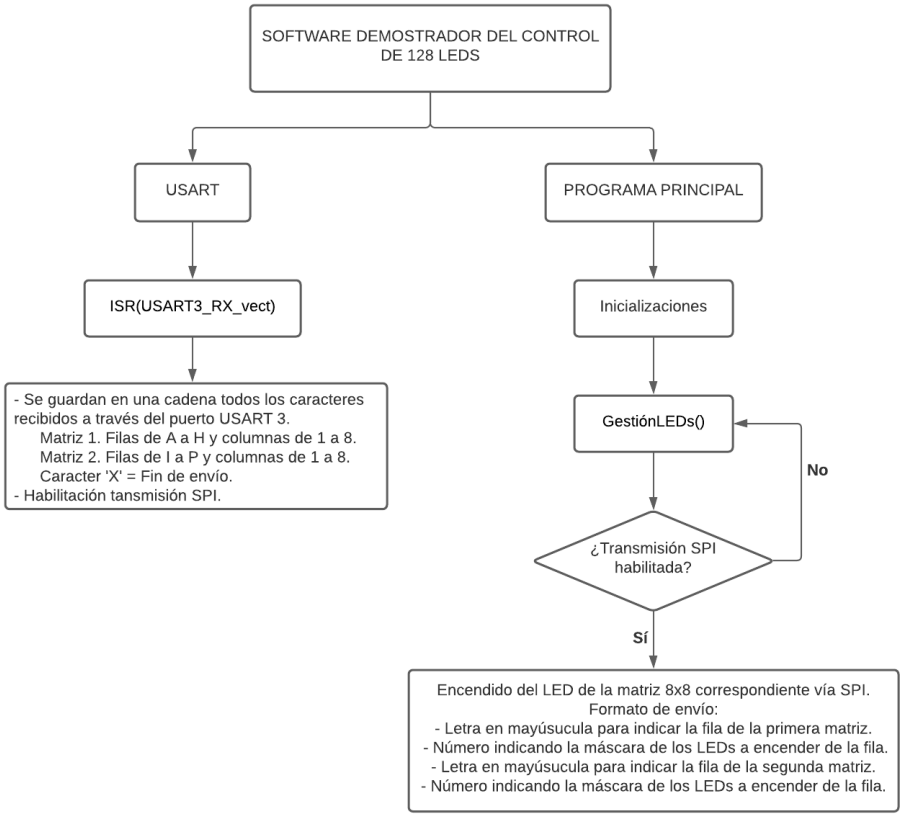

Figura 27. Diagrama de flujo del software implementado de gestión del demostrador.

En la figura 27 se muestra el diagrama de flujo del programa realizado que gestiona el encendido y apagado de las matrices de LEDs 8x8 vía SPI usando los chips MAX7219. El código completo del programa se encuentra en el Anexo 2 de este documento (páginas 81-107).

Primero de todo, se ha de configurar la comunicación por el puerto USART3 y la comunicación SPI. Las rutinas de inicialización son *void initSPI(void)*, *void USART_init()*. También se han de configurar las matrices de LEDs con la función *void initMatrix()*.

Después, se procede a hacer un encendido de los 128 LEDs 1 a 1 dejando un retardo de 50 ms entre LED. De esta manera se comprueba que todos los LEDs funcionen. Cuando se han encendido todos los LEDs, se apaga la matriz y se habilitan las interrupciones dejando todo listo para empezar a encender los LEDs de las vías de escalada que se quieran probar.

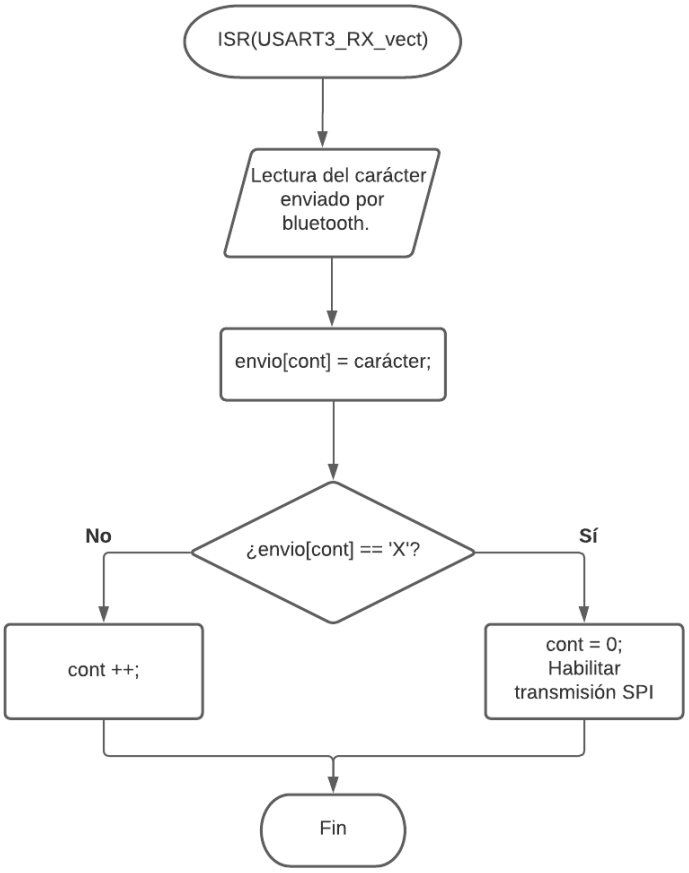

Figura 28. Diagrama de flujo de la rutina de servicio a la interrupción ISR(USART3_RX_vect).

Desde la rutina de servicio a la interrupción de la USART3, se guardará la cadena de LEDs enviada por bluetooth. La información de la cadena será una letra en mayúscula de la ‘A’ a la ‘P’ para indicar la fila a encender seguido de un número del 1 al 8 para indicar el número del LED que se quiere encender de esa fila. Si se envía el carácter ‘R’ se reinician las matrices LED para poder marcar una nueva vía. Para indicar que el envío se ha acabado y se pueden encender los LEDs hay que enviar el carácter ‘X’ al final de la cadena. El código correspondiente a esta rutina se encuentra en la página 87 de este documento.

En el programa principal se llama periódicamente a la función GestionLEDs() donde se analiza carácter a carácter la cadena que se ha llenado en la rutina de servicio a la interrupción de recepción de un carácter por el puerto USART3, ISR (USART3_RX_vect).

El direccionamiento de las matrices es el siguiente:

Matriz 1.

	1	2	3	4	5	6	7	8
A	A1	A2	A3	A4	A5	A6	A7	A8
B	B1	B2	B3	B4	B5	B6	B7	B8
C	C1	C2	C3	C4	C5	C6	C7	C8
D	D1	D2	D3	D4	D5	D6	D7	D8
E	E1	E2	E3	E4	E5	E6	E7	E8
F	F1	F2	F3	F4	F5	F6	F7	F8
G	G1	G2	G3	G4	G5	G6	G7	G8
H	H1	H2	H3	H4	H5	H6	H7	H8

Matriz 2.

	1	2	3	4	5	6	7	8
I	I1	I2	I3	I4	I5	I6	I7	I8
J	J1	J2	J3	J4	J5	J6	J7	J8
K	K1	K2	K3	K4	K5	K6	K7	K8
L	L1	L2	L3	L4	L5	L6	L7	L8
M	M1	M2	M3	M4	M5	M6	M7	M8
N	N1	N2	N3	N4	N5	N6	N7	N8
O	O1	O2	O3	O4	O5	O6	O7	O8
P	P1	P2	P3	P4	P5	P6	P7	P8

Para encender los LEDs se han de enviar 4 caracteres al chip MAX7219:

- El primer carácter se refiere a la línea de la primera matriz.
- El segundo carácter se refiere a la máscara de los LEDs a encender de la línea seleccionada de la primera matriz.
- El tercer carácter se refiere a la línea de la segunda matriz.
- El cuarto carácter se refiere a la máscara de los LEDs a encender de la línea seleccionada de la segunda matriz.

Si por ejemplo se quiere encender el LED D1, se habrá de enviar desde un terminal Bluetooth la cadena “D1X”.
Para hacer el envío al chip MAX7219, primero se habrá de detectar en la rutina de servicio a la interrupción de la USART3 el carácter ‘X’. Esto habilita la transmisión desde el programa principal de la cadena recibida.
Después, se enviará el carácter 4 que se corresponde a la fila D o, dicho de otro modo, a la fila 4 de la matriz de LEDs 1. Después, se hará una máscara OR encendiendo solo el LED 1 de la fila de LEDs D (D = D | 0b10000000) y se enviará esta máscara.
Como hay 2 matrices en cascada, habrá que enviar otros 2 caracteres correspondientes a la segunda matriz.
Para no modificar nada, el tercer carácter a enviar será 0. Esto significa que se tratará con la fila 0 que no está direccionada ya que las filas van de 1 a 8. Al no modificar ninguna fila el cuarto carácter a enviar también será un 0.

Si se quieren apagar todos los LEDs se enviará la cadena “RX”.
Si el programa detecta el carácter ‘R’, se llamará a la función de apagar las dos matrices y se reiniciarán todas las máscaras de LEDs para poder empezar de nuevo cuando haya otra transmisión.

Cabe destacar que no hace falta enviar todos los LEDs en el mismo envío, al utilizar máscaras se pueden enviar las cadenas que se deseen en caso de no haber encendido todos los LEDs deseados. La única forma de apagar las matrices para empezar de cero es enviar la cadena “RX”.

7 Resultados finales

7.1 Pared interactiva de escalada

Una vez el programa de la pared interactiva de escalada funcionó correctamente sobre las dos placas protoboard, se montó el diseño sobre un plafón de escalada de 70 centímetros de alto por 40 centímetros de ancho con 8 presas de escalada diferentes.

El resultado del montaje ha sido el siguiente.

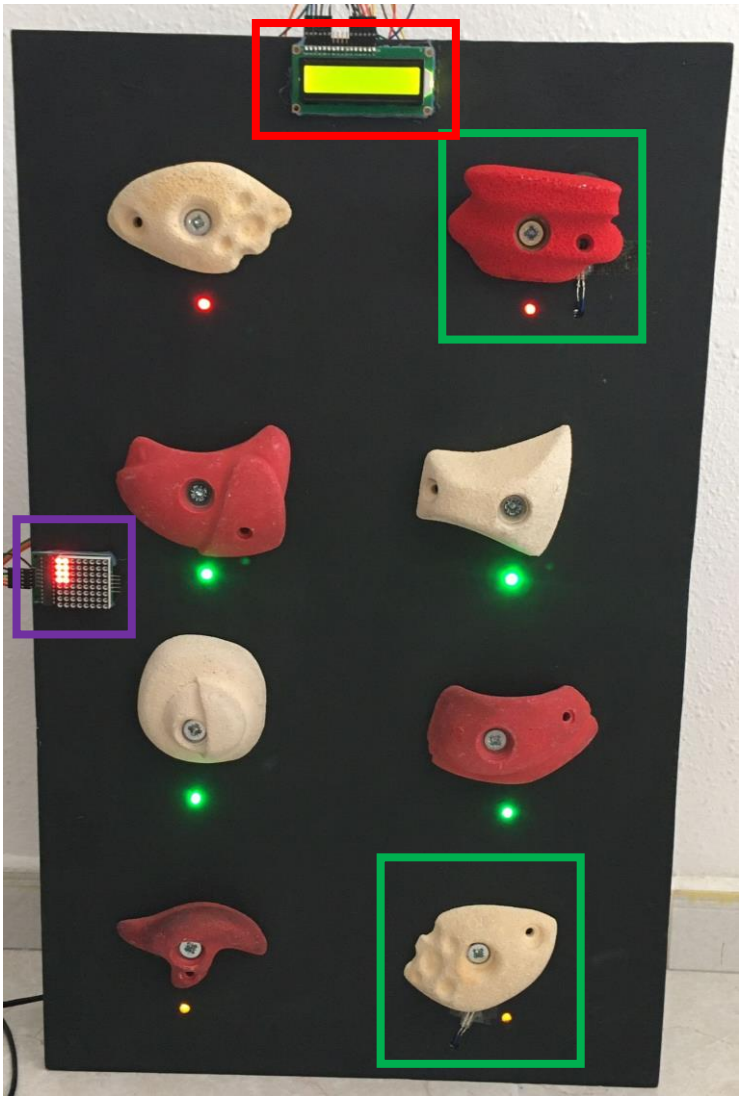

Figura 29. Prototipo de la pared interactiva de escalada.

En la figura 29 se muestra la parte visible de la pared interactiva de escalada.

En la pared de escalada se han atornillado 8 presas. Además, debajo de cada presa se ha hecho un agujero de 8 mm de diámetro para poner los LEDs.
La presa de abajo y arriba de la columna derecha (recuadros verdes) tienen 2 agujeros. Uno para los LEDs y otro para pasar los cables de los sensores de presión.
En la parte superior (recuadro rojo) se observa la pantalla LCD 16 x 2 donde se mostrará el tiempo de ascensión de la vía.
En la parte izquierda de la pared interactiva (recuadro violeta) se muestra la matriz 8x8 controlada con el chip MAX7219. Tal y como se ha mencionado anteriormente, cada LED de la pared interactiva de escalada tiene asociado un LED de la matriz. Como en la figura 29 están los 8 LEDs de la pared interactiva de escalada encendidos, en la matriz de LEDs también se muestran los 8 LEDs asociados a cada presa de escalada encendidos.

A continuación, se muestra la pared interactiva de escalada con la pantalla LCD en funcionamiento.

Figura 30. Prototipo de la pared interactiva de escalada en funcionamiento.

En la figura 30 se muestra la pared interactiva de escalada en funcionamiento.

Se pueden observar los 8 LEDs de distintos colores encendidos.

- LEDs rojos para presas de fin de vía.
- LEDs verdes para presas intermedias.
- LEDs amarillos para presas de inicio de vías.

También se puede observar la matriz de 8x8 LEDs en funcionamiento.

Además, se puede observar la pantalla LCD encendida mostrando un tiempo de ascensión de 2.235 s. Para que la pantalla LCD se encienda, un usuario ha tenido que hacer presión en la presa blanca de inicio de la vía, y, 2.235 s más tarde, ha tenido que hacer presión sobre la presa roja de final de vía.

Finalmente, se adjunta una captura de pantalla de la aplicación móvil para mostrar como el tiempo de ascensión de la vía coincide con el tiempo de ascensión enviado vía Bluetooth a la aplicación.

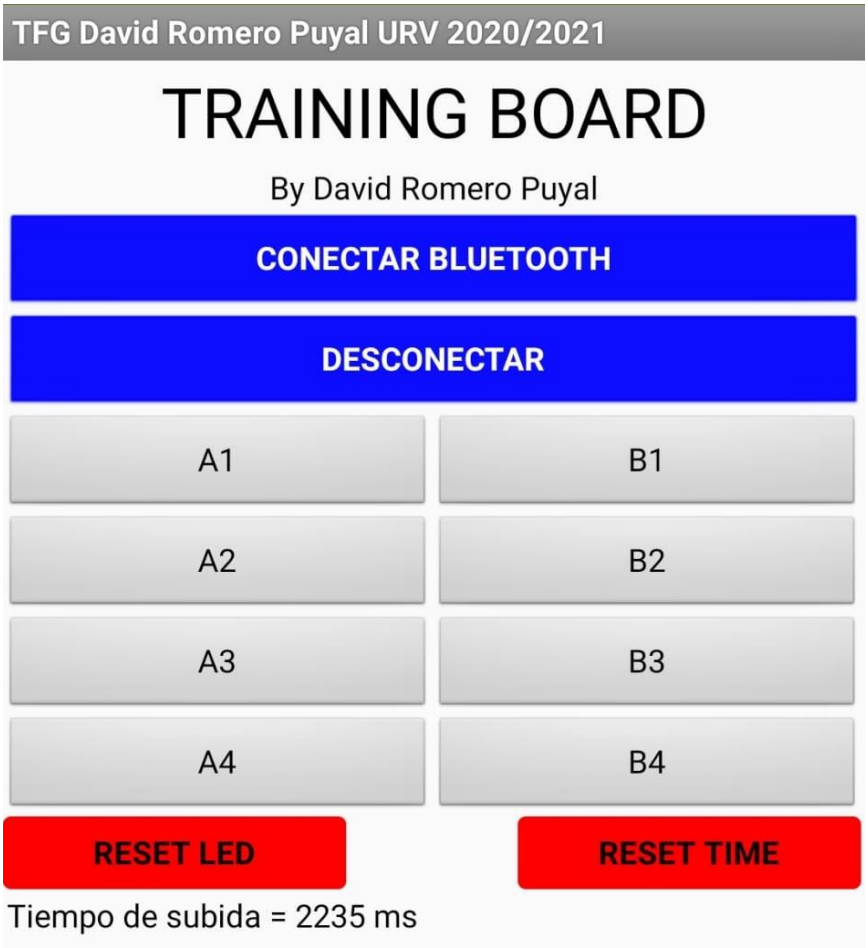

Figura 31. Captura de pantalla de la aplicación móvil en funcionamiento.

En la figura 31 se puede observar cómo ambos tiempos coinciden.

El conexionado y la parte no visible de la pared interactiva de escalada es la siguiente.

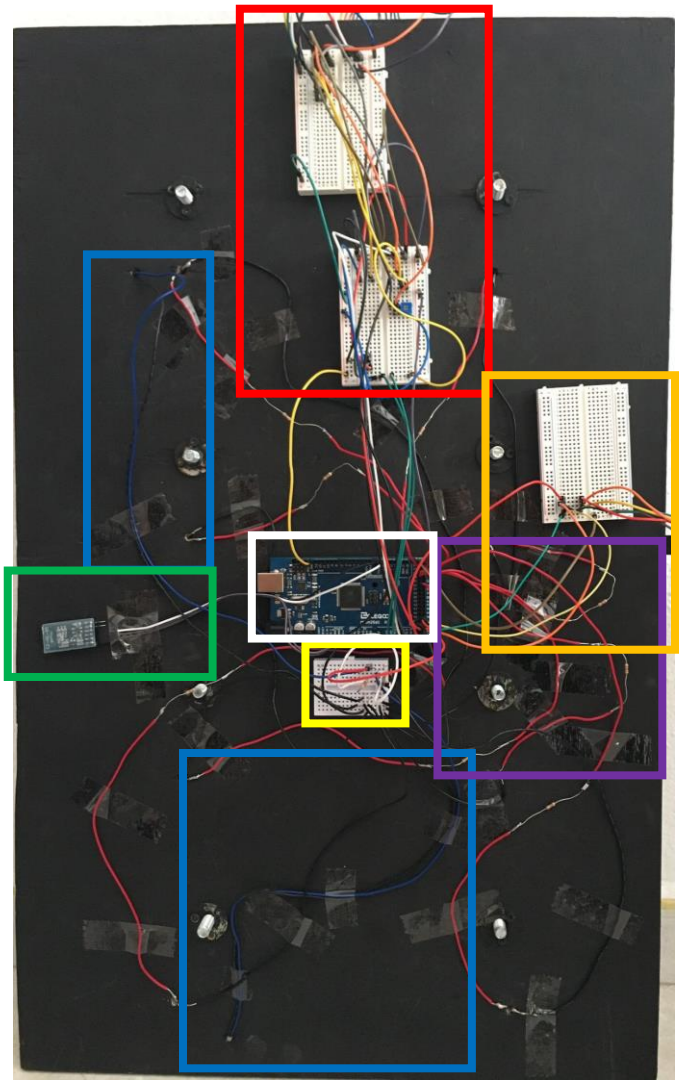

Figura 32. Conexionado de la pared interactiva de escalada.

En la figura 32 se muestra el montaje de la pared interactiva de escalada. Este montaje se corresponde a la parte no visible de la pared.

En el centro de la pared se ha fijado con silicona un Arduino MEGA con el chip ATMEGA2560. (Recuadro blanco).

Los circuitos divisores de tensión para encender los LEDs (cables rojos y recuadro violeta) se han soldado con estaño. De los pines digitales sale un cable rojo a la pata de una resistencia de 1 kΩ, de la otra pata de la resistencia de 1 kΩ sale otro cable rojo a la pata positiva del LED correspondiente, y, de la pata negativa de ese LED, sale un cable negro a masa.

Los circuitos divisores de tensión de los sensores de presión (recuadros azules) también han sido soldados igual que los LEDs, pero con la diferencia que las resistencias de 10 k no están soldadas ya que hay que leer la caída de tensión en estas.

Se dispone de una pequeña placa Protoboard (recuadro amarillo) que sirve para hacer esta lectura, además de tierra común para el retorno de todos los circuitos divisores de tensión de los sensores de presión y los LEDs.

Como el conexionado de la pantalla LCD (recuadro rojo) es más complejo y se necesita poner un potenciómetro para ajustar el brillo, se ha optado por pegar con silicona dos placas protoboard pequeñas entre la pantalla LCD y el microcontrolador ATMEGA2560.

También se observa el módulo Bluetooth en el recuadro verde de la figura 32.

Finalmente, en la parte derecha de la pared interactiva de escalada, se observa en el recuadro naranja otra pequeña placa Protoboard. En esta placa Protoboard están las conexiones del chip MAX7219 que gestiona la matriz de 8x8 LEDs que tiene asociada.

7.2 Demostrador de 128 presas

A continuación, se muestran unos ejemplos del resultado final del demostrador de los 128 LEDs.

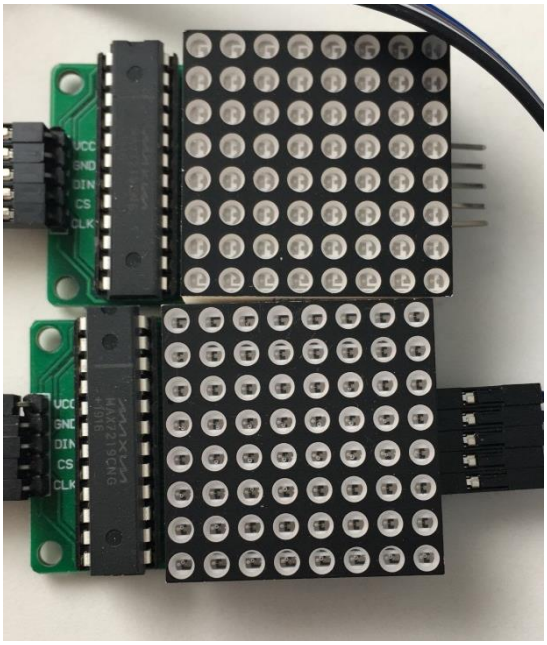

Figura 33. Matrices LED apagadas.

En el primer ejemplo se muestran las dos matrices de LEDs apagadas después de enviar la cadena de caracteres “RX” desde un terminal Bluetooth.

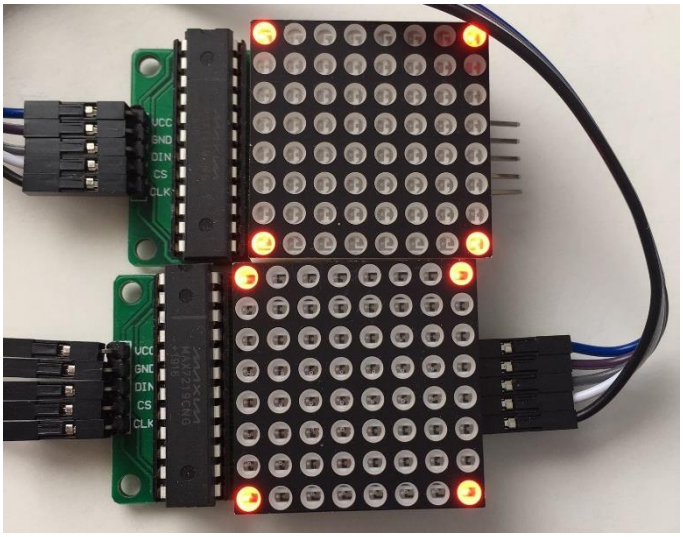

Figura 34. Matrices LED con las esquinas encendidas.

En el segundo ejemplo se muestran las esquinas de las dos matrices encendidas. Este ejemplo se corresponde a la cadena “A1A8H1H8I1I8P1P8X”.

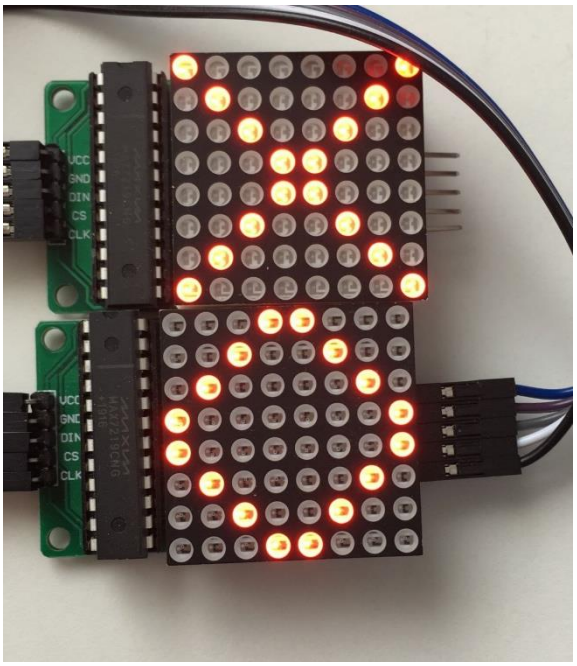

Figura 35. Matrices LED con dos figuras distintas.

En este tercer ejemplo se muestran dos figuras distintas que se corresponden a las siguientes cadenas de caracteres.

- Matriz 1: “A1A8B2B7C3C6D4D5E4E5F3F6G2G7H1H8X”.
- Matriz 2: “I4I5J3J6K2K7L1L8M1M8N2N7O3O6P4P5X”.

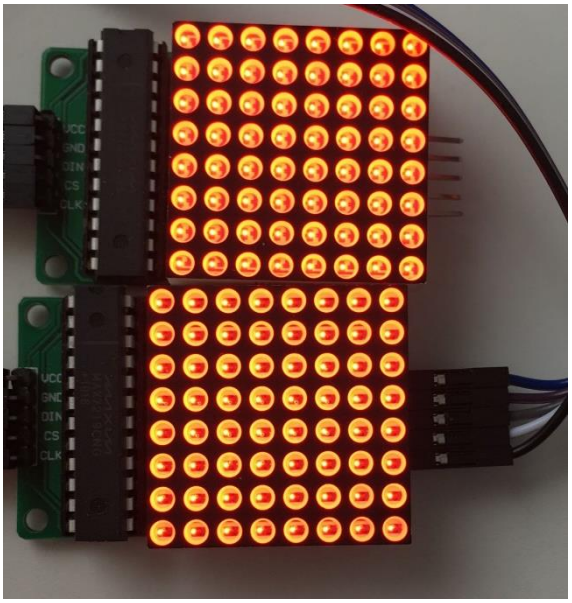

Figura 36. Matrices LED encendidas.

Finalmente, en este cuarto ejemplo se muestran las dos matrices de LEDs encendidas justo en el momento en que la función de comprobación del funcionamiento de los LEDs los ha encendido todos.

8 Propuesta de mejora

Aunque no se ha podido implementar debido a los costes y a tener que soldar y montar una estructura de metal, una mejora para la pared interactiva de escalada que se propone para realizar en un futuro, es permitir configurar la inclinación de la tabla.

La idea para poder inclinar la tabla es introducir dos motores paso a paso en la parte superior de la estructura. Como la anchura de la tabla suele ser de unos 3 metros, si se pone un motor en cada esquina superior, el plafón de escalada estará mejor fijado que si solo se pone uno en el centro y, además, los movimientos de inclinación serán más estables.
Estos motores harán girar dos engranajes.
Los engranajes tendrán acoplada una cadena para permitir subir y bajar la tabla. De esta forma, se consigue pasar de un movimiento giratorio de los motores paso a paso a un movimiento lineal.

La ilustración de la propuesta sería la siguiente.

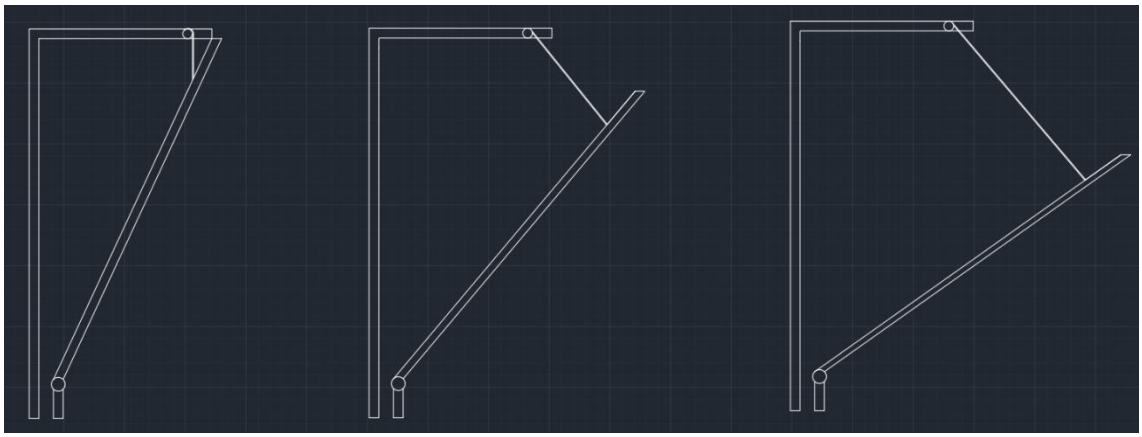

Figura 37. Ilustración de la idea que permitiría inclinar el plafón de escalada.

El dibujo anterior, correspondiente a la figura 37, está hecho a escala para una pared de 3.5 metros de altura. La pared se podrá inclinar entre 25 y 65 grados.

En la base de la pared habrá una articulación giratoria que permitirá el movimiento de la pared sin que ésta se rompa.
En la estructura superior de la pared, habrá otra articulación giratoria. Esta articulación girará con los dos motores paso a paso situados en los extremos y tendrá una cadena que irá cogida a la pared. De esta manera, con los motores paso a paso y un eje de engranajes para sujetar la cadena se podrá inclinar la pared.

Por seguridad, el tamaño máximo de las cadenas será aquel en el que la pared se incline hasta 65 grados. No se podrá sobrepasar esta inclinación ya que el extremo de la cadena que no sujeta a la pared estará cogido al motor. De esta manera, si algún motor fallase, la tabla no se caería hasta el suelo, sino que se quedaría inclinada a 65 grados.

Para la programación del motor de la propuesta para inclinar la pared de escalada, habría que saber cuántos pasos necesita para dar una vuelta el rotor del motor y la relación del engranaje reductor que lleva dentro. También habría que saber el radio del engranaje de la cadena para calcular una vuelta de motor a cuanto desplazamiento de la cadena equivale. Finalmente, habría que establecer la forma de programación del motor paso a paso.

Un motor paso a paso es un motor de corriente continua sin escobillas en el que la rotación se divide en un número de pasos resultantes de la estructura del motor. En el motor paso a paso utilizado, una revolución completa del rotor del motor de 360º equivale a 32 pasos. Es decir, cada paso del motor equivale a un giro del rotor de 11.25º. Estos motores son muy utilizados, ya que pueden moverse a deseo del usuario según la secuencia que se les indique a través de un microcontrolador.

Un motor paso a paso está compuesto por rotor y estator. El estator es una parte estacionaria, mientras que el rotor, montado en el eje como un cojinete, gira siguiendo el campo magnético giratorio creado alrededor del estator. En el estator se sitúan un conjunto de electroimanes, que son bobinas montadas en lugares específicos alrededor del rotor. Cuando la corriente fluye a través de las bobinas del estator, estas se energizan creando un electroimán que atrae a un imán montado en el rotor. Después de que el rotor haya hecho este desplazamiento, se encienden otra u otras bobinas en el estator creando otro electroimán y atrayendo al rotor a su nueva posición.

Una ilustración de la explicación anterior sobre el funcionamiento de un motor paso a paso puede ser la siguiente.

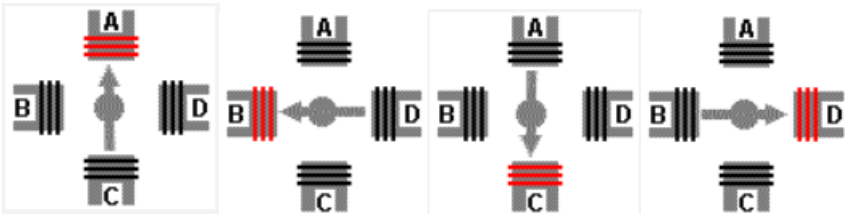

Figura 38. Ilustración del funcionamiento de un motor paso a paso.[10]

En la figura anterior, se muestra cómo se energizan las bobinas formando un electroimán (bobinas de color rojo) y atrayendo al rotor del motor (flecha central). Repetir esta secuencia permite hacer girar el motor paso a paso.

La figura anterior se corresponde a un ciclo completo de giro del motor. En el motor escogido, un ciclo completo del motor equivale a 4 pasos y un giro completo del rotor del motor equivale a 8 ciclos.

Finalmente, cabe mencionar que el rotor del motor tiene acoplado un engranaje reductor que hará girar un eje exterior con más par. Para que el eje exterior tenga mas par, por cada giro de éste, el rotor del motor habrá de dar N giros.

Se ha realizado una prueba de control de un motor paso a paso con ATMEL STUDIO de Microchip.

Los datos para destacar del motor a son:

- Tensión nominal entre 5 y 12 V.
- Fases = 4.
- Resistencia de 50 Ω.
- Par motor = 34 Nm.
- Reductor 1 / 64.
- 8 pasos por vuelta.
- Unipolar.

Este motor es muy pequeño y no tiene fuerza, por lo que no se podría poner en un caso real. En este ejemplo se quiere plasmar el programa que podría funcionar en un caso real.

La forma para programar el encendido y apagado de las bobinas del motor sería excitando dos bobinas por cada paso tal y como recomienda el fabricante. De esta manera se obtendría un par elevado y una velocidad alta, aunque el consumo también sería elevado.

PASO	**BOBINA A**	**BOBINA B**	**BOBINA C**	**BOBINA D**
1	ON	ON	OFF	OFF
2	OFF	ON	ON	OFF
3	OFF	OFF	ON	ON
4	ON	OFF	OFF	ON

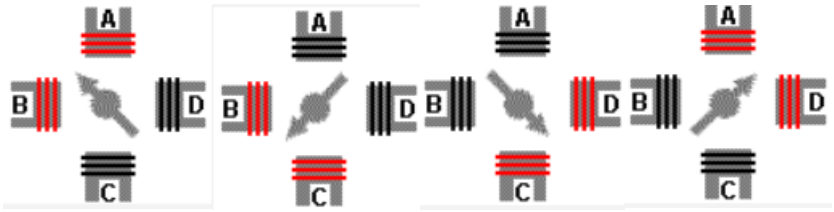

Figura 39. Ilustración de la forma escogida de programar el motor paso a paso.[11]

Para calcular el giro del eje de salida del motor paso a paso hay que tener en cuenta los siguientes parámetros proporcionados por el fabricante.

Un ciclo requiere 4 pasos.
Un giro completo del rotor requiere 8 ciclos.
Un giro completo del eje exterior necesita 64 giros del rotor.

$$4\frac{pasos}{ciclo\ motor}\cdot 8\frac{ciclos\ motor}{1\ vuelta\ rotor}\cdot 64\frac{vueltas\ rotor}{vuelta\ eje\ exterior} = 2048\frac{pasos}{vuelta\ eje\ exterior}$$

Para que el motor dé un giro completo, se necesitan 2048 pasos.

Además, sabiendo que cada vuelta del motor equivale a un desplazamiento lineal de $2\pi r$, se podría establecer una relación entre el número de vueltas y el número de grados de inclinación de la tabla.

Finalmente, se resume el demostrador realizado para hacer girar el motor en un sentido u otro. El programa completo se encuentra en el Anexo 3 de este documento.

8.1 Hardware

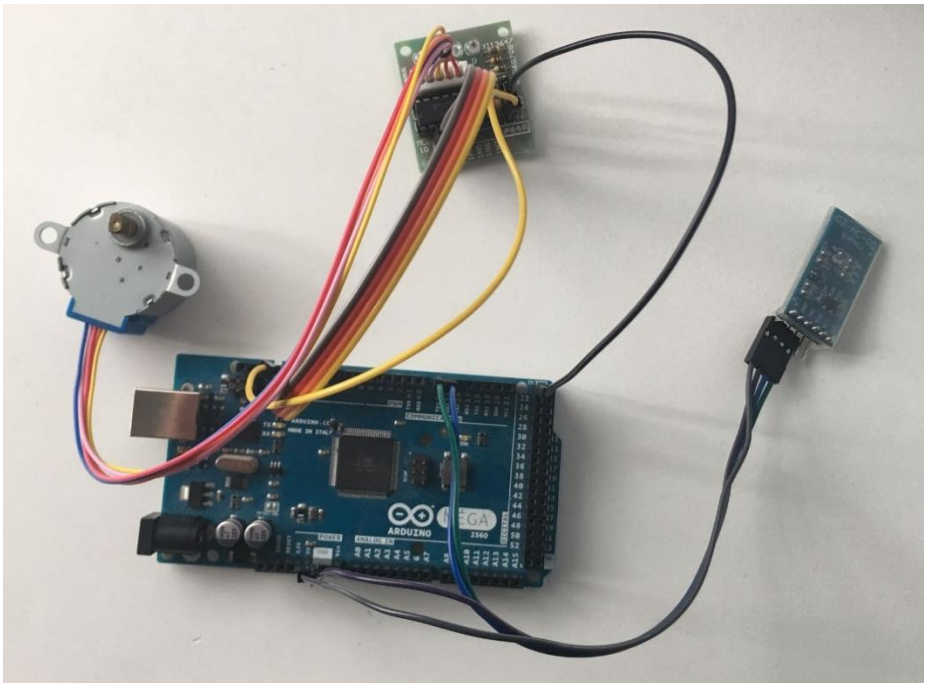

Figura 40 Demostrador del motor paso a paso.

En la figura 40, se muestra el demostrador construido para probar el funcionamiento del motor paso a paso.

En la parte izquierda de la figura 40 se puede observar el motor paso a paso modelo 28BYJ-48.
En la parte superior se muestra el driver necesario para controlar el motor paso a paso desde el microcontrolador ATMEGA2560.
En la parte derecha de la figura 40 se muestra el módulo Bluetooth HC-05 que permite comunicar el microcontrolador con un terminal Bluetooth.

El diagrama de bloques del hardware utilizado es el siguiente.

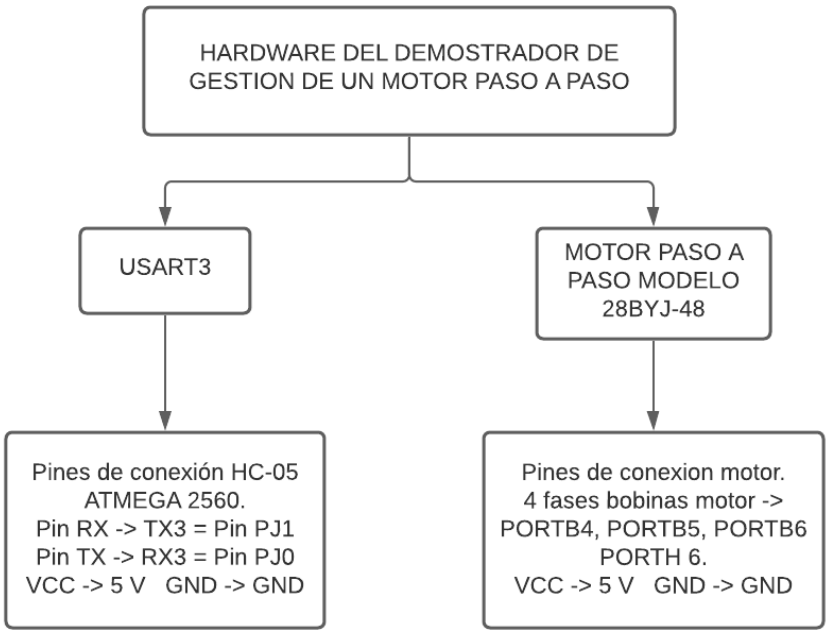

Figura 41. Diagrama de bloques del hardware escogido para gestionar el motor paso a paso.

Para la realización del programa se ha utilizado un microcontrolador ATMEGA2560, un módulo Bluetooth HC-05 y un motor paso a paso modelo 28BYJ-48.

El módulo Bluetooth se encarga de hacer llegar la información que se envía desde un terminal Bluetooth (dispositivo móvil), al microcontrolador ATMEGA2560. Esta información la analiza el microcontrolador y hace girar el motor paso a paso en sentido horario o sentido antihorario

Como el motor paso a paso utilizado tiene 4 fases y estas se excitan de manera independiente, para la conexión del motor se han usado 4 pines digitales. Uno para cada fase.

8.2 Software

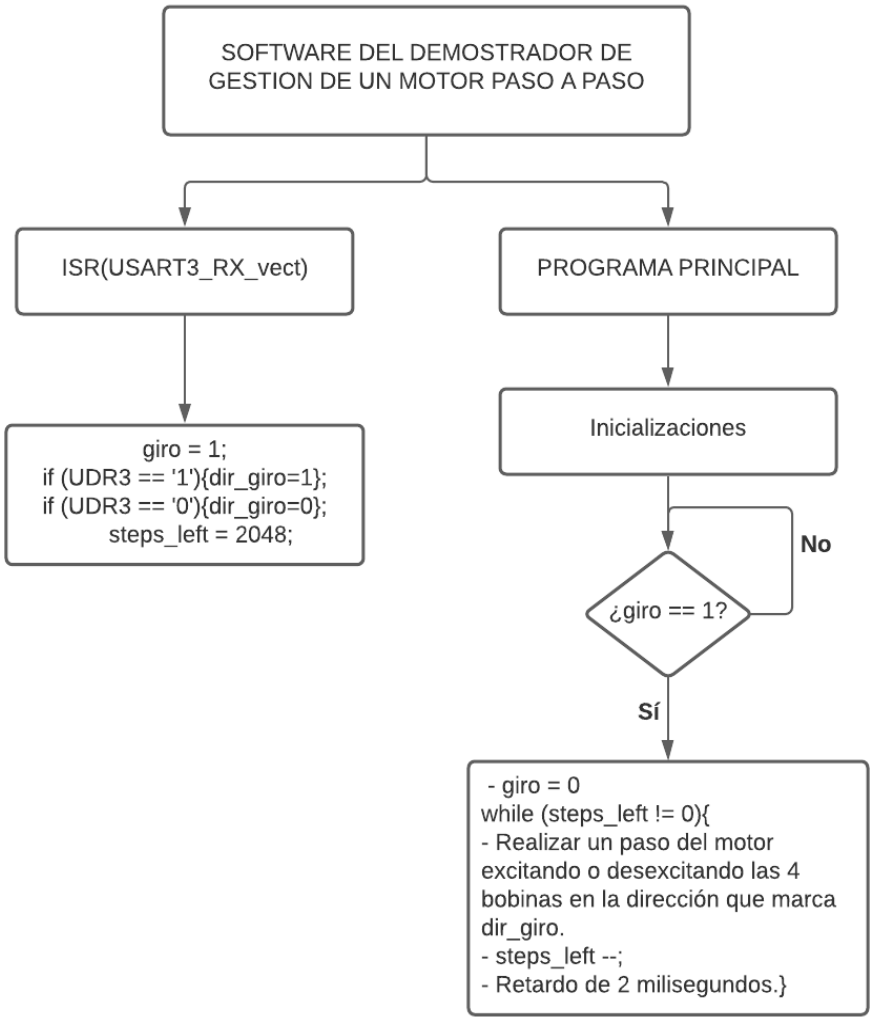

Figura 42. Diagrama de flujo del software implementado para gestionar el motor paso a paso.

Tal y como se muestra en la figura 42, el software implementado para la gestión del motor paso a paso se puede dividir en dos partes.

- El programa principal.
- La rutina de servicio a la interrupción ISR(USART3_RX_vect).

El programa completo se muestra en el Anexo 3 de este documento (páginas 107-110).

Desde un terminal Bluetooth de un dispositivo móvil, se envía vía Bluetooth por el puerto USART3 el carácter 1 o 0.

Este carácter lo analiza el microcontrolador en la rutina de servicio a la interrupción ISR(USART3_RX_vect).
En esta rutina de servicio a la interrupción se realizan 3 acciones.

- Por una parte, se habilita el giro del motor cambiando el estado de la variable giro de 0 a 1. Esto permite que en el programa principal se pueda entrar en la rutina de hacer girar el motor paso a paso.
- Después, se analiza el carácter recibido por el puerto USART3.
    - Si el carácter recibido es un 1, el motor dará una vuelta en sentido horario. El valor de de dir_giro será 1.
    - Si el carácter recibido es un 0, el motor dará una vuelta en sentido antihorario. El valor de dir_giro será 0.
- Finalmente se activa el contador del número de pasos que ha de dar el motor paso a paso para dar un giro completo en la dirección dir_giro.

El programa principal comprueba periódicamente si la variable giro ha cambiado su estado de 0 a 1. En caso afirmativo, devolverá la variable al estado 0 para garantizar que solo se realiza un giro y hará un bucle 2048 veces. Este bucle hace avanzar un paso el motor en la dirección de la variable dir _giro. El motor dará una vuelta completa en 2048 pasos.

Este programa sería aplicable a un caso real. Como se ha especificado antes, se podría inclinar más o menos la pared haciendo girar el motor en un sentido u otro. También se podrían tener ciertas inclinaciones guardadas que equivaldrían a un número de vueltas del motor paso a paso.

9 Presupuesto

Producto	**Cantidad**	**Coste Unitario**	**Coste Total**
Arduino MEGA marca ELEGOO.	2	14 €	28 €
Sensor de presión redondo.	1	9.80 €	9.80 €
Sensor de presión cuadrado.	1	11.50 €	11.50 €
Chip MAX7219 con matriz de LEDs incorporada.	2	5.30 €	10.60 €
Pack de 300 LEDs.	1	10 €	10 €
Pack de 6 carretes de 9 m de cable cada uno.	1	20 €	20 €
Pack de cables de conexionado.	1	7 €	7 €
Módulo Bluetooth HC-05.	2	7.50 €	15 €
Pantalla LCD.	1	8.30 €	8.30 €
Placa Protoboard.	2	0 €	0 €
Resistencia 10 kΩ.	2	0 €	0 €
Resistencia 1 kΩ.	8	0 €	0 €
Potenciómetro 10 kΩ.	1	0 €	0 €
Motor Paso a Paso modelo 28BYJ-48.	1	12 €	12 €
Multímetro digital.	1	18 €	18 €
Plafón de escalada de madera.	1	0 €	0 €
Bote de pintura negra para madera.	1	5 €	5 €
Presas de escalada.	8	4 €	32 €
Tornillos para fijar las presas.	8	0.50 €	4 €
Placa protoboard pequeña	4	2.50 €	10 €

$$PRESUPUESTO = 201.20\ €$$

El presupuesto para la realización de este proyecto, sin contar el material que se ha reutilizado ni las horas de trabajo invertidas, ha sido de 201.20 €.

10 Conclusiones

En este proyecto, se ha realizado un prototipo de pared interactiva de escalada 100 % funcional, añadiendo algunas mejoras a los modelos actuales que hay en el mercado como, por ejemplo, las incorporaciones de sensores de presión para monitorizar el tiempo de ascensión de una vía de escalada y de una pantalla LCD para mostrar este tiempo.

En el proyecto implementado, se ha trabajado usando tanto software como hardware. Se ha hecho uso del software para programar un hardware diseñado.

Para que el proyecto funcionase, se han tenido que usar conceptos adquiridos a lo largo de estos cuatro años de carrera. Se quiere destacar los siguientes conocimientos.

- Conocimiento de programación embedida para poder programar tanto la tabla de escalada como el demostrador de 128 LEDs y el motor paso a paso con el entorno de programación Microchip Studio de Microchip.
- Conocimiento de montaje de circuitos electrónicos adquirido durante todos los laboratorios de electrónica cursados durante la carrera para poder montar satisfactoriamente la tabla de escalada y el prototipo sobre dos placas protoboard.
- Conocimiento de buscar en documentos técnicos para poder programar tanto el microcontrolador ATMEGA 2560 como los diferentes periféricos utilizados además de poder escoger satisfactoriamente los elementos necesarios a incorporar en el proyecto: pantalla LCD compatible, LEDs adecuados, chip MAX7219, etcétera.

Finalmente, se ha de mencionar que se han cumplido todos los objetivos propuestos al inicio del proyecto.

11 Bibliografía

Entornos de programación utilizados.

Atmel Studio de Microchip. *https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices.* Descargado durante el mes de febrero.
Arduino IDE de Arduino. *https://www.arduino.cc/en/software*. Descargado durante el mes de febrero.

Programa usado para realizar los diagramas de flujo.

Página web de Lucid Chart. *https://www.lucidchart.com/pages/es.* Consultada durante los meses de abril y mayo para la realización de la memoria.
https://www.rff.com i *https://www.programiz.com* consultadas durante el mes de mayo para realizar los diagramas de flujos.

Documentos técnicos extraídos.

Datasheet del microcontrolador ATMEGA 2560.
https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf. Descargado el mes de febrero.
Diagrama de pines del Arduino MEGA. *https://store.arduino.cc/arduino-mega-2560-rev3.* Descargado el mes de febrero.
Datasheet del chip MAX7219. *https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf.* Descargado el mes de febrero.
Datasheet del controlador HD44780. *https://www.sparkfun.com/datasheets/LCD/HD44780.pdf.* Descargado el mes de marzo.
Información sobre motores paso a paso. *https://www.tme.eu/es* y *https://www.prometec.net/*. Consultadas durante el mes de mayo.
Información sobre el chip MAX7219. *https://rgb.kitiyo.com* Consultadas durante el mes de mayo.

Webs de información consultadas para la programación.

Página web de avr freaks. *https://www.avrfreaks.net/.* Consultada durante los meses de febrero, marzo y abril.
Página web de Luis Llamas. *https://www.luisllamas.es/.* Consultada durante los meses de enero y febrero.
Página web de Prometec. *https://www.prometec.net/.* Consultada durante los meses de febrero y marzo.
Página web de Electrowings. *https://www.electronicwings.com/.* Consultada durante el mes de marzo.
Página web de Bricogeek. *https://tienda.bricogeek.com/.* Consultada durante el mes de marzo.

Canal de Youtube utilizado para la programación.

Canal de Youtube de Electrosaurio. *https://www.youtube.com/c/Electrosaurio/featured.* Consultado durante los meses de febrero, marzo y abril.

Webs de compras del material utilizado.

Página web de Amazon España. *https://www.amazon.es*. Utilizada durante todo el proyecto.

Webs de compras de paredes interactivas de escalada consultadas.

Página web de MoonBoard. *www.moonboard.com*. Consultada durante el mes de febrero.
Página web de Setter closet. *www.settercloset.com*. Consultada durante el mes de febrero.

Referencias

[1]. https://www.moonboard.com/how-to-build-your-moonboard. [5/02/2021].
[2]. https://settercloset.com/pages/the-kilter-board. [5/02/2021].
[3]. https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf. [10/02/2021].
[4]. https://store.arduino.cc/arduino-mega-2560-rev3. [10/02/2021].
[5]. https://store.arduino.cc/arduino-mega-2560-rev3. [10/02/2021].
[6]. https://www.trossenrobotics.com/productdocs/2010-10-26-DataSheet-FSR402-Layout2.pdf. [7/03/2021].
[7]. https://www.electronicwings.com/avr-atmega/interfacing-lcd-16x2-in-4-bit-mode-with-atmega-16-32-. [25/03/2021].
[8]. https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf. [15/05/2021].
[9]. https://www.vistronica.com/display/matriz-de-leds-8x8-catodo-comun-con-max7219-detail.html. [15/05/2021].
[10]. https://www.prometec.net/motor-28byj-48/. [02/05/2021].
[11]. https://www.prometec.net/motor-28byj-48/. [02/05/2021].

12 Anexos

12.1 Anexo 1. Programa de la pared interactiva de escalada

```
/*
 * Programa Final Tabla TFG David Romero Puyal.c
 *
 * Author : 39944650-K
 */

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <ctype.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>

#define FOSC 16000000UL

/*PINES DE ALIMENTACIÓN DE LOS CIRCUITOS DE PRESIÓN*/
#define pinpowerstart PORTL |= 0X02            //Alimentación  circuito
presion salida pin 48
#define pinpowertop PORTL |= 0X01              //Alimentación  circuito
presion top pin 49

/*PINES DE LOS LEDS DE LAS PRESAS*/
#define topizquierda PORTA |= 0x01
#define topderecha PORTA |= 0x02
#define hizquierda PORTA |= 0x04
#define hderecha PORTA |= 0x08
#define lizquierda PORTA |= 0x10
#define lderecha PORTA |= 0x20
#define startizquierda PORTA |= 0x40
#define startderecha PORTA |= 0x80
#define resetLEDs PORTA &= 0X00;

#define T_INT_TIMER1 1000    //Número de veces que se va a entrar en la
interrución por segundo.
//1000 interrupciones de timer cada segundo. Resolución = 1 ms
```

```
/*DEFINICIONES DE LOS PUERTOS A USAR PARA LA
CONEXIÓN DEL LCD. SE USARÁ EL PUERTO K*/
#define LCD_Dir DDRK
#define LCD_Port PORTK
#define RS PK0                    //Pin de Selección de registro
conectado al bit 0 del Puerto K.
#define EN PK1          //Pin Enable conectado al bit 1 del Puerto K.


// PINES CONECTADOS AL ATMEGA2560 PARA SPI
#define PIN_SCK                   PORTB1
#define PIN_MOSI                  PORTB2
#define PIN_SS                    PORTB0


// MACROS PARA SELECCIONAR O DESELECCIONAR EL PUERTO SS COMO ESCLAVO
PARA ENVIAR COMANDOS VÍA SPI
#define SLAVE_SELECT    PORTB &= 0xfe   //PORTB0 = 0
#define SLAVE_DESELECT  PORTB |= 0x01    //PORTB0 = 1


// DEFINICIONES DE CONSTANTES
#define NUM_DEVICES     1     //Num de dispositivos MAX7219 conectados
en cascada.
#define DEL             50    // Retardo en ms para el encendido de cada
uno de los leds en la rutina de comprobacionLEDs.


/*VARIABLES PARA LA GESTIÓN DE LA MATRIZ DE LEDs del chip MAX7219*/
volatile char LED;                  // Variable para leer de USART
volatile int segu = 0;              // Variable de seguridad porque para
encender un LED se necesita analizar 2 carácteres (fila+columna)
volatile char fila;                 // Memorizar la fila escogida ya que
para encender LED se necesita fila + columna
volatile char A,B,C,D;              // Variables para almacenar los LEDs
encendidos por filas de la matriz de LEDs
volatile char envioSPI[128];        // Cadena de envio de LEDs a encender
volatile int cont = 0;              // Indice de la cadena
volatile int transmision = 0;       //  Habilitador para analizar la
cadena de LEDs a encender


/*VARIABLES PARA LA GESTIÓN DEL TIEMPO DE SUBIDA*/
int presiontop, presionstart;       //Leer la presión en los sensores.
int tiempo_subida = 0;  //Almacenamiento tiempo en la ISR del timer1.
int tiempo_envio = 0;   //Variable de envio del tiempo de subida en ms.
int tiempos = 0;        //Guardar los segundos del tiempo de subida.
int tiempoms = 0;       //Guardar los milisegundos del tiempo de subida.
int seg = 0;            //Impedir que si se pisa las presas de salida
reseteen el contador.
char str[4];            //Cadena enviovariables de tiempo a imprimir.
char envio[10];         //Cadena envio Bluetooth tiempo en milisegundos.
int i = 0;              //Contador cadena envio Bluetooth.
char LEDs = ' ';        //Cadena para leer los LEDs a encender.
```

```
/* RUTINA DE INICIALIZACIÓN DE LA USART
SE UTILIZAN LOS PUERTOS RX3 y TX3
DEL MICRO ATMEGA2560 */

void USART_init() {
      PRR1 = PRR1 & 0xfb;                //Habilitar inicialización.
      UBRR3 = 0x67;
      UCSR3A = 0x00;
      UCSR3B = 0xd8;    //Habilitar interrupciones RX y TX de USART 3.
      UCSR3C = 0x06;
}




/*RUTINA DE INICIALIZACIÓN DEL TIMER1*/

void timer_on(){

      // Division de la frecuencia entre 1024. Bits Csn1.

      TCNT1 = 0x0000;   //Se incrementa cada pulso de fuente de clk.
      TCCR1B = TCCR1B | 0x05;
      TCCR1B = TCCR1B & 0xfd;

}




/*INICIALIZACIÓN DE LA COMUNICACIÓN SPI*/

void initSPI(void)
{
      DDRB  = DDRB | 0x01;          // Seleccionar pin SS como salida.
      PORTB = PORTB | 0X01;         // Deseleccionar modo esclavo.

      DDRB = DDRB | 0x04;           // Pin MOSI como salida.
      DDRB = DDRB | 0x02;           // Pin SCK como salida.

      SPCR = SPCR | 0x10;           // SPI modo master.
      SPCR = SPCR | 0x40;           // Permitir comunicación SPI.
}




/*RUTINA DE APPAGADO DEL TIMER 1*/

void timer_off(){

      //Apagar el timer
      //Opción T1clock = 0 Hz
      TCCR1B = TCCR1B & 0xf8;

}
```

```
/*ENVIAR UN BYTE POR SPI
LA COMUNICACIÓN ES BYTE A BYTE*/

void writeByte(char byte)
{
      SPDR = byte;               // Envia el byte
      while(!(SPSR & (1 << SPIF)));     // Espera confirmación del envio
}




/*ENVIAR 2 BYTES POR SPI.
PARA ENCENDER CADA LED SE NECESITA ENVIAR 2 BYTES
EL PRIMER BYTE DICE LA DIRECCIÓN DE LA FILA Y EL
SEGUNDO BYTE DICE QUE LED(S) DE ESA FILA*/

void writeWord(char fila, char columna)
{
      writeByte(fila);
      writeByte(columna);
}




/*INICIALIZACIÓN DE LAS MATRICES CONECTADAS
EN CASCADA*/

void initMatrix()
{
      int i;

      /*Seleción del brillo de la matriz*/
      SLAVE_SELECT;
      for(i = 0; i < NUM_DEVICES; i++)   // Un bucle para cada dispositivo
conectado en cascada
      {
            writeByte(0x0A); // Registro de intensidad
            writeByte(0x07); // Intensidad de la pantalla LED
      }
      SLAVE_DESELECT;


      /*Refrescar displays*/
      SLAVE_SELECT;
      for(i = 0; i < NUM_DEVICES; i++)
      {
            writeByte(0x0B); // Registro de scan
            writeByte(0x07); // Todas las columnas
      }
      SLAVE_DESELECT;
```

```
	/*Encender las matrices LED en modo normal*/
	SLAVE_SELECT;
	for(i = 0; i < NUM_DEVICES; i++)
	{
		writeByte(0x0C); // Registro Shutdown
		writeByte(0x01); // Operación en modo normal
	}
	SLAVE_DESELECT;


	/*Deshabilitar modo test*/
	SLAVE_SELECT;
	for(i = 0; i < NUM_DEVICES; i++)
	{
		writeByte(0x0F); // Registro modo dest
		writeByte(0x00); // Apagado
	}
	SLAVE_DESELECT;
}




/*FUNCIÓN PARA LIMPIAR CADA DISPLAY*/

void clearMatrix(void)
{
	int i, j;

	for(i = 1; i < 9; i++)		//Limpiar columna por columna
	{
		SLAVE_SELECT;
		for (j = 0; j < NUM_DEVICES; j++)		//2 dispositivos
		{
			writeByte(i);
			writeByte(0x00);
		}
		SLAVE_DESELECT;
	}
}
```

```
/*RUTINA DE ENCENDIDO Y APAGADO DE TODOS LOS LEDS
PARA COMPROBAR SU FUNCIONAMIENTO CORRECTO*/

void OnOffMatrix(void)
{

      SLAVE_SELECT;
      writeWord(1,0b10000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      SLAVE_SELECT;
      writeWord(1,0b11000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      SLAVE_SELECT;
      writeWord(2,0b10000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      SLAVE_SELECT;
      writeWord(2,0b11000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      SLAVE_SELECT;
      writeWord(3,0b10000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      SLAVE_SELECT;
      writeWord(3,0b11000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      SLAVE_SELECT;
      writeWord(4,0b10000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      SLAVE_SELECT;
      writeWord(4,0b11000000);
      SLAVE_DESELECT;
      _delay_ms(250);

      clearMatrix();

}
```

```
/*ISR DE USART 3 CADA VEZ QUE SE RECIBE UN CARÁCTER
SE ANALIZA EL CARACTER RECIBIDO, SE ENCIENDE EL LED
CORRESPONDIENTE DE LA TABLA Y SE LLENA LA CADENA PARA
ENCENDER EL LED CORRESPONDIENTE DE LA MATRIZ DE LEDs*/

ISR(USART3_RX_vect){

      LEDs = UDR3;

      if (LEDs != 0){

            switch (LEDs){
                  case '1':
                        topizquierda;
                        envioSPI[cont]='A';
                        envioSPI[cont+1]='1';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case '2':
                        topderecha;
                        envioSPI[cont]='A';
                        envioSPI[cont+1]='2';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case '3':
                        hizquierda;
                        envioSPI[cont]='B';
                        envioSPI[cont+1]='1';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case '4':
                        hderecha;
                        envioSPI[cont]='B';
                        envioSPI[cont+1]='2';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case '5':
                        lizquierda;
                        envioSPI[cont]='C';
                        envioSPI[cont+1]='1';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case '6':
                        lderecha;
                        envioSPI[cont]='C';
                        envioSPI[cont+1]='2';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case '7':
```

```
                        startizquierda;
                        envioSPI[cont]='D';
                        envioSPI[cont+1]='1';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case '8':
                        startderecha;
                        envioSPI[cont]='D';
                        envioSPI[cont+1]='2';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                  break;
                  case 'R':               // Apagar todos los leds.
                        envioSPI[cont]='R';
                        envioSPI[cont+2]='X';
                        cont = 0;
                        transmision = 1;
                        resetLEDs;
                  break;
                  case 'S':               //Reiniciar timer y variables
del timer por una caída en medio de la vía.
                        timer_off();
                        seg = 0;
                        tiempo_subida = 0;
                  break;
            }

      }
}
```

```
/*RUTINA DE SERVICIO A LA INTERRUPCIÓN DEL PUERTO
UART3 PARA ENVIAR POR BLUETOOTH EL TIEMPO DE SUBIDA
DE LA VÍA EN MILISEGUNDOS*/

 ISR(USART3_TX_vect){

      if (envio[i] != '\0'){
      UDR3 = envio[i];
      i++;}
}
```

```
/*FUNCIÓN DE INICIALIZACIÓN DEL ADC*/

void ADC_init(){

      /*Ajustar a la derecha los 10 bits del resultado de la conversión
del ADC*/
      ADMUX = ADMUX & 0xdf;

      /*Voltage de referencia = AVCC*/
      ADMUX = ADMUX | 0x40;   //REFS0 a 1 (bit 0 del registro ADMUX)
      ADMUX = ADMUX & 0x7f;   //REFS1 a 0 (bit 1 del registro ADMUX)

      /*Frecuencia ADC  =  16MHz / 128 = 125 kHz*/
      ADCSRA = ADCSRA | 0x07;
}




/*FUNCIÓN PARA LEER POR EL ADC0 DEL ATMEGA2560.
RECIBE EL CANAL POR EL QUE HA DE LEER Y DEVUELVE
EL VALOR LEÍDO POR EL CANAL ENVIADO DEL ADC*/

int ADC_GetData(int canal){

      /*Escoger el canal de lectura del MUX0 al MUX4*/
      ADMUX = ADMUX & 0xe0;         //Solo interesan los 5 primeros bits
para elegir el canal.
      ADMUX = ADMUX | (canal<<MUX0);

      /*Activar el ADC para leer*/
      ADCSRA = ADCSRA | 0x80;
      _delay_ms(1);      //Dar 1ms para que se prepare el adc

      /*Muestreo del ADC*/
      ADCSRA = ADCSRA | 0x40;

      /*Aviso del muestreo leyendo el flag*/
      while ( !(ADCSRA & (1<<ADIF)) );
      ADCSRA = ADCSRA | 0x10; //Reinicio flag

      /*Apagar el ADC para ahorrar energía*/
      ADCSRA = ADCSRA & 0x7f;

      /*Devuelve  el  registro  de  10  bits  con  el  resultado  de  la
conversión*/
      return ADC;

}
```

```
/*FUNCIÓN PARA ENCENDER Y APAGAR LOS 8 LEDs
DE LA TABLA Y ASI COMPROBAR QUE FUNCIONAN*/

void inicializacionLeds (){
      topizquierda;
      _delay_ms(250);
      topderecha;
      _delay_ms(250);
      hizquierda;
      _delay_ms(250);
      hderecha;
      _delay_ms(250);
      lizquierda;
      _delay_ms(250);
      lderecha;
      _delay_ms(250);
      startizquierda;
      _delay_ms(250);
      startderecha;
      _delay_ms(250);
      resetLEDs;
      resetLEDs2;
}




/*ISR DEL TIMER 1.
SE EJECUTA CON UN PERIODO DEFINIDO POR EL REGISTRO OCR1A*/

ISR(TIMER1_COMPA_vect)
{
      tiempo_subida++;
}
```

```
/*RUTINA DE INICIALIZACIÓN DEL TIMER 1*/

void timer_init(){

      /*Modo de operación = CTC modo de comparación. Bits del modo de
funcionamiento del timer.
      Pag 145 datasheet.*/
      TCCR1A = TCCR1A & 0xfc;
      TCCR1B = TCCR1B | 0x08;
      TCCR1B = TCCR1B & 0xef;

      /*Valor a comparar.
      Registro de comparación timer1 en modo CTC en función de la
frecuencia cpu,
      la división de frecuencia del timer y la frecuencia de entrada en
la interrupcion.*/
      OCR1A = (F_CPU/1024/T_INT_TIMER1) - 1;

      TIMSK1 = TIMSK1 | 0x02;      //Permitimos   interrupciones   del
timer1
}




/*FUNCIÓN PARA ENVIAR LOS COMANDOS DE INTSTRUCCIÓN
A LA PANTALLA LCD*/

void LCD_Command( unsigned char cmnd )
{
      LCD_Port = (LCD_Port & 0x0f) | (cmnd & 0xf0);        // Envio del
nibble alto del comando.
      LCD_Port &= ~ (1<<RS); //  RS=0,  Registro  de  instrucciones  del
controlador.
      LCD_Port |= (1<<EN);               // Pulso de Enable.
      _delay_us(1);
      LCD_Port &= ~ (1<<EN);

      _delay_us(200);

      LCD_Port = (LCD_Port & 0x0f) | (cmnd << 4);          //Envio  del
nibble bajo.
      LCD_Port |= (1<<EN);
      _delay_us(1);
      LCD_Port &= ~ (1<<EN);
      _delay_ms(2);
}
```

```
/*FUNCIÓN PARA ENVIAR LOS CARACTERES A IMPRIMIR
A LA PANTALLA LCD*/

void LCD_Char( unsigned char data )
{
      LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); // Envio del nibble
alto del comando.
      LCD_Port |= (1<<RS);                    //  RS=1,  Registro  de
datos.
      LCD_Port |= (1<<EN);
      _delay_us(1);
      LCD_Port &= ~ (1<<EN);

      _delay_us(200);

      LCD_Port = (LCD_Port & 0x0F) | (data << 4); // Envio del nibble
alto del comando.
      LCD_Port |= (1<<EN);
      _delay_us(1);
      LCD_Port &= ~ (1<<EN);
      _delay_ms(2);
}




/*FUNCIÓN DE INICIALIZACIÓN DE LA PANTALLA LCD
HAY QUE SEGUIR UN PROCESO QUE MARCA EL FABRICANTE
CON LOS COMANDOS RESERVADOS*/

void LCD_init (void)
{
      LCD_Dir = 0xff;              // PortB como salida
      _delay_ms(20);               // El retardo para que el LCD se
encienda ha de ser siempre > 15ms
      LCD_Command(0x33);
      LCD_Command(0x32);      // Comando de inicialización pantalla LCD
      LCD_Command(0x28);                // Uso de 2 lineas y 4 bit mode
      LCD_Command(0x0c);                // Cursor del display en off
      LCD_Command(0x06);      // Incrementar cursor moverlo a la derecha
      LCD_Command(0x01);                // Limpiar pantalla
      _delay_ms(2);
      LCD_Command (0x80);     // Llevar cursor a la posición inicial.
}




/*FUNCIÓN PARA BORRAR LO QUE HAYA
ESCRITO EN LA PANTALLA LCD*/

void LCD_Clear()
{
      LCD_Command (0x01);                           //Comando     para
limpiar el display.
      _delay_ms(2);
      LCD_Command (0x80);                           //Mover el cursor
a la primera posición.
}
```

```
/*FUNCIÓN PARA IMPRIMIR CADENAS DE CARÁCTERES
 POR LA PANTALLA DEL LCD*/

void LCD_String (char *str)
{
      int i;
      for(i=0;str[i]!=0;i++)                    //   Envia   el   string
caracter a caracter.
      {
            LCD_Char (str[i]);
      }
}




/*FUNCIÓN PARA CONVERTIR EL TIEMPO DE SUBIDA
EN UNA CADENA DE CARÁCTERES Y ASÍ PODER ENVIAR
POR BLUETOOTH DICHA CADENA*/

void envio_BT (){
      i = 0;
      itoa (tiempo_envio,envio,10);
      UDR3 = envio[i];
      i++;
}




/*FUNCIÓN PARA IMPRIMIR ENTEROS POR LA PANTALLA
DEL LCD*/

void LCD_Numero (int num)
{
      itoa (num,str,10);      //Pasar de entero a cadena de carácteres.
      LCD_String(str);
}




/*FUNCIÓN FINAL PARA IMPRIMIR EL
TIEMPO DE SUBIDA EN LA PANTALLA LCD*/

void envio_LCD_TOP (){
      LCD_String("TFG David Romero");          //Escribe  un  string  en
la primera linea de la pantalla LCD.
      LCD_Command(0xc0);           //Envio del comando saltar de linea
a la linea 2.
      LCD_String("Tiempo: ");
      LCD_Numero(tiempos);         //Envia a imprimir el entero con la
variable de tiempo.
      LCD_String(".");
      LCD_Numero(tiempoms);
      LCD_String(" s");
}
```

```
/*SE ANALIZA CADA CARÁCTER DE LA CADENA POR ORDEN
SIEMPRE Y CUANDO ESTÉ HABILITADA LA TRANSMISÓN.
LA FORMA DE TRANSMITIR UNA DIRECCIÓN ES:
      PRIMER CARÁCTER: LETRA MAYÚSCULA PARA INDICAR FILA A-D.
      SEGUNDO CARÁCTER: NÚMERO PARA INDICAR LA MÁSCARA DE LEDS A ENCENDER
DE ESA FILA.
SI SE ENVIA UNA R SE REINICIAN LAS VARIABLES Y MATRIZ.
CUANDO EL CARÁCTER ES X, SE ACABA LA TRANSMISIÓN. LA NUEVA HABILITACIÓN
SE HACE DESDE LA ISR.
EN LA FUNCIÓN, PRIMERO SE ANALIZA QUE FILA ES Y DESPUÉS,
SEGÚN LA FILA, SE HACE UNA MÁSCARA CON LOS LEDS A ENCENDER DE DICHA
FILA.*/

void GestionLEDs (){

      if (transmision == 1)               //Si ha acabado la recepción
de datos en la ISR se entra en la función de envío.
      {
            LED = envioSPI[cont];
            cont++;

      switch (segu){                //Analizar la fila


            case 0:

            segu = 1;
            switch (LED){
                  case 'A':
                  fila = 1;
                  break;
                  case 'B':
                  fila = 2;
                  break;
                  case 'C':
                  fila = 3;
                  break;
                  case 'D':
                  fila = 4;
                  break;
                  case 'R':
                  fila = 0;
                  segu = 0;
                  A = A & 0x00;
                  B = B & 0x00;
                  C = C & 0x00;
                  D = D & 0x00;
                  clearMatrix();
                  break;
                  case'X':
                  transmision = 0;
                  cont = 0;
                  segu = 0;
                  break;
            }
            break;



            case 1:                 //Máscara de los LEDs de esa fila.
            segu = 0;
```

```
switch (LED){

      case '1':
      if (fila == 1){
            A = A | 0x80;
            SLAVE_SELECT;
            writeWord(fila,A);
            SLAVE_DESELECT;
      }
      if (fila == 2){
            B = B | 0x80;
            SLAVE_SELECT;
            writeWord(fila,B);
            SLAVE_DESELECT;
      }
      if (fila == 3){
            C = C | 0x80;
            SLAVE_SELECT;
            writeWord(fila,C);
            SLAVE_DESELECT;
      }
      if (fila == 4){
            D = D | 0x80;
            SLAVE_SELECT;
            writeWord(fila,D);
            SLAVE_DESELECT;
      }
      break;


      case '2':
      if (fila == 1){
            A = A | 0x40;
            SLAVE_SELECT;
            writeWord(fila,A);
            SLAVE_DESELECT;
      }
      if (fila == 2){
            B = B | 0x40;
            SLAVE_SELECT;
            writeWord(fila,B);
            SLAVE_DESELECT;
      }
      if (fila == 3){
            C = C | 0x40;
            SLAVE_SELECT;
            writeWord(fila,C);
            SLAVE_DESELECT;
      }
      if (fila == 4){
            D = D | 0x40;
            SLAVE_SELECT;
            writeWord(fila,D);
            SLAVE_DESELECT;
      }
      break;

      case 'R':
      fila = 0;
      segu = 0;
      A = A & 0x00;
```

```
                    B = B & 0x00;
                    C = C & 0x00;
                    D = D & 0x00;
                    clearMatrix();
                    break;

                    case'X':
                    transmision = 0;
                    cont = 0;
                    segu = 0;
                    break;

              }
              break;
        }
}
}
```

```
/*PROGRAMA PRINCIPAL*/

      int main(void)
{
      DDRL |= 0x03;     //Pines de alimentación circuitos de presion
como salida;
      DDRA |= 0xff;     // Los 8 bits del Puerto A serviran para encender
y apagar los 8 LEDs.
      pinpowerstart;
      pinpowertop;

      cli(); //Deshabilitar interrupciones
      /*INICIALIZACIONES*/
      LCD_init();
      ADC_init();
      timer_init();
      USART_init();
      initSPI();
      initMatrix();
      clearMatrix();
      sei(); //Habilitar interrupciones

      inicializacionLeds();
      OnOffMatrix();

      while (1)
      {

            /*Llamada periódica a la rutina que gestiona el encendido y
apagado de la matriz de LEDs asociada al chip MAX7219. Solo entrará en
esta rutina si se ha habilitado la transmisión*/
            GestionLEDs();

            /*Sensor de inicio conectado al canal 0.
Si se pulsa el sensor de inicio se reinician las variables de tiempo,
la variable de seguridad para que al pisar la presa de inicio no vuelva
a entrar en este trozo de programa cambia de estado y se enciende el
timer1 para contar el tiempo de subida en milisegundos.*/
            presionstart = ADC_GetData(0);           //Divisor      de
tensión entre R10k y sensor. Si sensor no pulsado Rsensor = infinito.
            if (presionstart < 512)                  //Si       sensor
pulsado Rsensor < 10 k
            {
                  if (seg==0){
                        seg = 1;
                        tiempo_subida = 0;
                        timer_on();
                  }
            }
```

```
            /*Sensor de fin de vía conectado al canal 1.
            Al llegar a la presa de final de vía y siempre y cuando se
haya activado antes el sensor de inicio de vía, se apaga el timer, se
imprime el tiempo de subida por la pantalla LCD en formato XXs.xxxms y
se envía el tiempo en ms vía bluetooth.*/
            presiontop = ADC_GetData(1);
            if (presiontop < 512)
            {
                  if (seg==1){
                        timer_off();
                        tiempo_envio = tiempo_subida;
                        tiempos = tiempo_subida/1000; //Como el timer 1
cuenta cada ms para enviar el tiempo se ha de pasar
                        tiempoms = tiempo_subida - (1000*tiempos);
// Cálculo milisegundos.
                        seg = 0;
                        envio_LCD_TOP();
                        envio_BT();
                        _delay_ms(4000);  //El resultado permanece en la
pantalla durante 4 segundos.
                        LCD_Clear();
                        tiempo_subida = 0;
                  }
            }
      }
}
```

12.2 Anexo 2. Programa del demostrador de 128 presas

```
/*
 * SPI Control 2 matrices 8x8 David Romero Puyal.c
 *
 * Author : DAVID ROMERO PUYAL 39944650-K
 */


#define F_CPU 16000000


#include <avr/io.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>


// PINES CONECTADOS AL ATMEGA2560 PARA SPI
#define PIN_SCK                    PORTB1
#define PIN_MOSI                   PORTB2
#define PIN_SS                     PORTB0


// MACROS PARA SELECCIONAR O DESELECCIONAR EL PUERTO SS COMO ESCLAVO
PARA ENVIAR COMANDOS VÍA SPI
#define SLAVE_SELECT    PORTB &= 0xfe   //PORTB0 = 0
#define SLAVE_DESELECT  PORTB |= 0x01    //PORTB0 = 1


// DEFINICIONES DE CONSTANTES
#define NUM_DEVICES   2 //Numero de dispositivos conectados en cascada.
#define DEL   50       // Retardo en ms para el encendido de cada uno
de los leds en la rutina de comprobacionLEDs.


volatile char LED;            // Variable para leer de USART
volatile int seg = 0;         // Variable de seguridad porque para
encender un LED se necesita entrar 2 veces en la ISR RX3 (fila+columna)
volatile char matriz1, matriz2;    // Variables para memorizar en que
dispositivo está la fila a encender
volatile char fila;           // Memorizar la fila escogida ya que para
encender LED se necesita fila + columna
volatile char A,B,C,D,E,F,G,H;     // Variables para almacenar los LEDs
encendidos por filas del dispositivo 1
volatile char I,J,K,L,M,N,O,P;     // Variables para almacenar los LEDs
encendidos por filas del dispositivo 2
volatile char envio[128];                    // Cadena de envío de
LEDs a encender
volatile int cont = 0;              // Índice de la cadena
volatile int transmision = 0;       // Marca que habilita la transmisión
```

```
/* RUTINA DE INICIALIZACIÓN DE LA USART
SE UTILIZAN LOS PUERTOS RX3 y TX3
DEL MICRO ATMEGA2560 */

void USART_init() {
      PRR1 = PRR1 & 0xfb;                //Habilitar inicialización.
      UBRR3 = 0x67;
      UCSR3A = 0x00;
      UCSR3B = 0xd8;                     //Habilitar interrupciones RX
y TX de USART 3.
      UCSR3C = 0x06;
}



/*INICIALIZACIÓN DE LA COMUNICACIÓN SPI*/

void initSPI(void)
{
      DDRB  = DDRB | 0x01;          // Seleccionar pin SS como salida.
      PORTB = PORTB | 0X01;         // Deseleccionar modo esclavo.

      DDRB = DDRB | 0x04;                // Pin MOSI como salida.
      DDRB = DDRB | 0x02;                // Pin SCK como salida.

      SPCR = SPCR | 0x10;                // SPI modo master.
      SPCR = SPCR | 0x40;                // Permitir comunicación SPI.
}



/*ENVIAR UN BYTE POR SPI
LA COMUNICACIÓN ES BYTE A BYTE*/

void writeByte(char byte)
{
      SPDR = byte;                  // Envía el byte
      while(!(SPSR & (1 << SPIF)));  // Espera a la confirmación del
envío
}



/*ENVIAR 2 BYTES POR SPI.
PARA ENCENDER CADA LED SE NECESITA ENVIAR 2 BYTES
EL PRIMER BYTE DICE LA DIRECCIÓN DE LA FILA Y EL
SEGUNDO BYTE DICE QUE LED(S) DE ESA FILA*/

void writeWord(char fila, char columna)
{
      writeByte(fila);
      writeByte(columna);
}
```

```
/*INICIALIZACIÓN DE LAS MATRICES CONECTADAS
EN CASCADA*/

void initMatrix()
{
      int i;

      /*Seleción del brillo de la matriz*/
      SLAVE_SELECT;
      for(i = 0; i < NUM_DEVICES; i++)   // Un bucle para cada dispositivo
conectado en cascada
      {
            writeByte(0x0A); // Registro de intensidad
            writeByte(0x07); // Intensidad de la pantalla LED
      }
      SLAVE_DESELECT;


      /*Refrescar displays*/
      SLAVE_SELECT;
      for(i = 0; i < NUM_DEVICES; i++)
      {
            writeByte(0x0B); // Registro de scan
            writeByte(0x07); // Todas las columnas
      }
      SLAVE_DESELECT;


      /*Encender las matrices LED en modo normal*/
      SLAVE_SELECT;
      for(i = 0; i < NUM_DEVICES; i++)
      {
            writeByte(0x0C); // Registro Shutdown
            writeByte(0x01); // Operación en modo normal
      }
      SLAVE_DESELECT;


      /*Deshabilitar modo test*/
      SLAVE_SELECT;
      for(i = 0; i < NUM_DEVICES; i++)
      {
            writeByte(0x0F); // Registro modo dest
            writeByte(0x00); // Apagado
      }
      SLAVE_DESELECT;
}
```

```
/*FUNCIÓN PARA LIMPIAR CADA DISPLAY*/

void clearMatrix(void)
{
      int i, j;

      for(i = 1; i < 9; i++)              //Limpiar columna por columna
      {
            SLAVE_SELECT;
            for (j = 0; j < NUM_DEVICES; j++)        //2 dispositivos
            {
                  writeByte(i);
                  writeByte(0x00);
            }
            SLAVE_DESELECT;
      }
}
```

```
/*RUTINA DE ENCENDIDO Y APAGADO DE TODOS LOS LEDS
PARA COMPROBAR SU FUNCIONAMIENTO CORRECTO*/

void OnOffMatrix(void)
{
      char i;

      //Primera Matriz
      for(i = 1; i < 9; i++)
      {
            SLAVE_SELECT;
            writeWord(i,0b10000000);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);

            SLAVE_SELECT;
            writeWord(i,0b11000000);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);

            SLAVE_SELECT;
            writeWord(i,0b11100000);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);

            SLAVE_SELECT;
            writeWord(i,0b11110000);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);

            SLAVE_SELECT;
            writeWord(i,0b11111000);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);

            SLAVE_SELECT;
            writeWord(i,0b11111100);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);

            SLAVE_SELECT;
            writeWord(i,0b11111110);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);

            SLAVE_SELECT;
            writeWord(i,0xff);
            writeWord(i,0x00);
            SLAVE_DESELECT;
            _delay_ms(DEL);
      }
```

```
        //Segunda Matriz
        for(i = 1; i < 9; i++)
        {
              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0b10000000);
              SLAVE_DESELECT;
              _delay_ms(DEL);

              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0b11000000);
              SLAVE_DESELECT;
              _delay_ms(DEL);

              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0b11100000);
              SLAVE_DESELECT;
              _delay_ms(DEL);

              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0b11110000);
              SLAVE_DESELECT;
              _delay_ms(DEL);

              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0b11111000);
              SLAVE_DESELECT;
              _delay_ms(DEL);

              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0b11111100);
              SLAVE_DESELECT;
              _delay_ms(DEL);

              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0b11111110);
              SLAVE_DESELECT;
              _delay_ms(DEL);

              SLAVE_SELECT;
              writeWord(i,0xff);
              writeWord(i,0xff);
              SLAVE_DESELECT;
              _delay_ms(DEL);
        }

        clearMatrix();

}
```

```
/*ISR DE USART 3: CADA VEZ QUE SE RECIBE UN CARÁCTER
SE GUARDA EN LA CADENA envio HASTA QUE EL CARÁCTER
RECIBIDO SEA X. ENTONCES SE PROCEDE A ENCENDER
LOS LEDS RECIBIDOS HABILITANDO LA TRANSMISIÓN.
EL ENCENDIDO DE LOS LEDS SE HACE DESDE MAIN.
*/

ISR(USART3_RX_vect){

	envio[cont] = UDR3;

	if (envio[cont]=='X')
	{
		cont = 0;
		transmision = 1;
	}

	else{
		cont++;
	}
}
```

```
/*SE ANALIZA CADA CARÁCTER DE LA CADENA POR ORDEN
SIEMPRE Y CUANDO ESTÉ HABILITADA LA TRANSMISÓN.
LA FORMA DE TRANSMITIR UNA DIRECCIÓN ES:
      PRIMER CARÁCTER: LETRA MAYÚSCULA PARA INDICAR FILA A-P.
      SEGUNDO CARÁCTER: NÚMERO PARA INDICAR LA MÁSCARA DE LEDS A ENCENDER
DE ESA FILA.
      SI SE ENVIA UNA R SE REINICIAN VARIABLES Y MATRIZ.
      CUANDO EL CARÁCTER ES X, SE ACABA LA TRANSMISIÓN. LA NUEVA
HABILITACIÓN SE HACE DESDE LA ISR.
EN LA FUNCIÓN, PRIMERO SE ANALIZA QUE FILA ES Y DESPUÉS,
SEGÚN LA FILA, SE HACE UNA MÁSCARA CON LOS LEDS A ENCENDER DE DICHA
FILA.*/

void GestionLEDs (){

      if (transmision == 1)               //Si ha acabado la recepción
de datos en la ISR se entra en la función de envío.
      {
            LED = envio[cont];
            cont++;

      switch (seg){                 //Analizar la fila


            case 0:

            seg = 1;
            switch (LED){
                  case 'A':
                  fila = 1;
                  matriz1 = 1;
                  matriz2 = 0;
                  break;
                  case 'B':
                  fila = 2;
                  matriz1 = 1;
                  matriz2 = 0;
                  break;
                  case 'C':
                  fila = 3;
                  matriz1 = 1;
                  matriz2 = 0;
                  break;
                  case 'D':
                  fila = 4;
                  matriz1 = 1;
                  matriz2 = 0;
                  break;
                  case 'E':
                  fila = 5;
                  matriz1 = 1;
                  matriz2 = 0;
                  break;
                  case 'F':
                  fila = 6;
                  matriz1 = 1;
                  matriz2 = 0;
                  break;
                  case 'G':
                  fila = 7;
                  matriz1 = 1;
```

```
matriz2 = 0;
break;
case 'H':
fila = 8;
matriz1 = 1;
matriz2 = 0;
break;
case 'I':
fila = 1;
matriz1 = 0;
matriz2 = 1;
break;
case 'J':
fila = 2;
matriz1 = 0;
matriz2 = 1;
break;
case 'K':
fila = 3;
matriz1 = 0;
matriz2 = 1;
break;
case 'L':
fila = 4;
matriz1 = 0;
matriz2 = 1;
break;
case 'M':
fila = 5;
matriz1 = 0;
matriz2 = 1;
break;
case 'N':
fila = 6;
matriz1 = 0;
matriz2 = 1;
break;
case 'O':
fila = 7;
matriz1 = 0;
matriz2 = 1;
break;
case 'P':
fila = 8;
matriz1 = 0;
matriz2 = 1;
break;
case 'R':
fila = 0;
seg = 0;
matriz1 = 0;
matriz2 = 0;
A = A & 0x00;
B = B & 0x00;
C = C & 0x00;
D = D & 0x00;
E = E & 0x00;
F = F & 0x00;
G = G & 0x00;
H = H & 0x00;
I = I & 0x00;
```

```
      J = J & 0x00;
      K = K & 0x00;
      L = L & 0x00;
      M = M & 0x00;
      N = N & 0x00;
      O = O & 0x00;
      P = P & 0x00;
      clearMatrix();
      break;
      case'X':
      transmision = 0;
      cont = 0;
      seg = 0;
      break;
}
break;



case 1:                 //Máscara de los LEDs de esa fila.
seg = 0;
switch (LED){

      case '1':
      if ((matriz1 == 1)&&(fila == 1)){
            A = A | 0x80;
            SLAVE_SELECT;
            writeWord(fila,A);
            writeWord(0,0);
            SLAVE_DESELECT;
      }
      if ((matriz1 == 1)&&(fila == 2)){
            B = B | 0x80;
            SLAVE_SELECT;
            writeWord(fila,B);
            writeWord(0,0);
            SLAVE_DESELECT;
      }
      if ((matriz1 == 1)&&(fila == 3)){
            C = C | 0x80;
            SLAVE_SELECT;
            writeWord(fila,C);
            writeWord(0,0);
            SLAVE_DESELECT;
      }
      if ((matriz1 == 1)&&(fila == 4)){
            D = D | 0x80;
            SLAVE_SELECT;
            writeWord(fila,D);
            writeWord(0,0);
            SLAVE_DESELECT;
      }
      if ((matriz1 == 1)&&(fila == 5)){
            E = E | 0x80;
            SLAVE_SELECT;
            writeWord(fila,E);
            writeWord(0,0);
            SLAVE_DESELECT;
      }
      if ((matriz1 == 1)&&(fila == 6)){
            F = F | 0x80;
```

```
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
      G = G | 0x80;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x80;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
```

```
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x80;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
      SLAVE_DESELECT;
}
break;


case '2':
if ((matriz1 == 1)&&(fila == 1)){
      A = A | 0x40;
      SLAVE_SELECT;
      writeWord(fila,A);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 2)){
      B = B | 0x40;
      SLAVE_SELECT;
      writeWord(fila,B);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 3)){
      C = C | 0x40;
      SLAVE_SELECT;
      writeWord(fila,C);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 4)){
      D = D | 0x40;
      SLAVE_SELECT;
      writeWord(fila,D);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 5)){
      E = E | 0x40;
      SLAVE_SELECT;
      writeWord(fila,E);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 6)){
      F = F | 0x40;
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
```

```
      G = G | 0x40;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x40;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
```

```
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x40;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
      SLAVE_DESELECT;
}
break;


case '3':
if ((matriz1 == 1)&&(fila == 1)){
      A = A | 0x20;
      SLAVE_SELECT;
      writeWord(fila,A);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 2)){
      B = B | 0x20;
      SLAVE_SELECT;
      writeWord(fila,B);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 3)){
      C = C | 0x20;
      SLAVE_SELECT;
      writeWord(fila,C);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 4)){
      D = D | 0x20;
      SLAVE_SELECT;
      writeWord(fila,D);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 5)){
      E = E | 0x20;
      SLAVE_SELECT;
      writeWord(fila,E);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 6)){
      F = F | 0x20;
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
      G = G | 0x20;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
```

```
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x20;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x20;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
```

```
      SLAVE_DESELECT;
}
break;


case '4':
if ((matriz1 == 1)&&(fila == 1)){
      A = A | 0x10;
      SLAVE_SELECT;
      writeWord(fila,A);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 2)){
      B = B | 0x10;
      SLAVE_SELECT;
      writeWord(fila,B);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 3)){
      C = C | 0x10;
      SLAVE_SELECT;
      writeWord(fila,C);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 4)){
      D = D | 0x10;
      SLAVE_SELECT;
      writeWord(fila,D);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 5)){
      E = E | 0x10;
      SLAVE_SELECT;
      writeWord(fila,E);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 6)){
      F = F | 0x10;
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
      G = G | 0x10;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x10;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
```

```
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x10;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
      SLAVE_DESELECT;
}
break;


case '5':
```

```
if ((matriz1 == 1)&&(fila == 1)){
      A = A | 0x08;
      SLAVE_SELECT;
      writeWord(fila,A);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 2)){
      B = B | 0x08;
      SLAVE_SELECT;
      writeWord(fila,B);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 3)){
      C = C | 0x08;
      SLAVE_SELECT;
      writeWord(fila,C);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 4)){
      D = D | 0x08;
      SLAVE_SELECT;
      writeWord(fila,D);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 5)){
      E = E | 0x08;
      SLAVE_SELECT;
      writeWord(fila,E);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 6)){
      F = F | 0x08;
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
      G = G | 0x08;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x08;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
```

```
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
      SLAVE_DESELECT;
}
break;


case '6':
if ((matriz1 == 1)&&(fila == 1)){
      A = A | 0x04;
      SLAVE_SELECT;
      writeWord(fila,A);
      writeWord(0,0);
      SLAVE_DESELECT;
```

```
}
if ((matriz1 == 1)&&(fila == 2)){
      B = B | 0x04;
      SLAVE_SELECT;
      writeWord(fila,B);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 3)){
      C = C | 0x04;
      SLAVE_SELECT;
      writeWord(fila,C);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 4)){
      D = D | 0x04;
      SLAVE_SELECT;
      writeWord(fila,D);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 5)){
      E = E | 0x04;
      SLAVE_SELECT;
      writeWord(fila,E);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 6)){
      F = F | 0x04;
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
      G = G | 0x04;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x04;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x08;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x04;
      SLAVE_SELECT;
      writeWord(0,0);
```

```
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x04;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x04;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x04;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x04;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x04;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x04;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
      SLAVE_DESELECT;
}
break;


case '7':
if ((matriz1 == 1)&&(fila == 1)){
      A = A | 0x02;
      SLAVE_SELECT;
      writeWord(fila,A);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 2)){
      B = B | 0x02;
      SLAVE_SELECT;
      writeWord(fila,B);
      writeWord(0,0);
```

```
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 3)){
      C = C | 0x02;
      SLAVE_SELECT;
      writeWord(fila,C);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 4)){
      D = D | 0x02;
      SLAVE_SELECT;
      writeWord(fila,D);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 5)){
      E = E | 0x02;
      SLAVE_SELECT;
      writeWord(fila,E);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 6)){
      F = F | 0x02;
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
      G = G | 0x02;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x02;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x02;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x02;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x02;
      SLAVE_SELECT;
```

```
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x02;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x02;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x02;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x02;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x02;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
      SLAVE_DESELECT;
}
break;


case '8':
if ((matriz1 == 1)&&(fila == 1)){
      A = A | 0x01;
      SLAVE_SELECT;
      writeWord(fila,A);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 2)){
      B = B | 0x01;
      SLAVE_SELECT;
      writeWord(fila,B);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 3)){
      C = C | 0x01;
      SLAVE_SELECT;
      writeWord(fila,C);
```

```
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 4)){
      D = D | 0x01;
      SLAVE_SELECT;
      writeWord(fila,D);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 5)){
      E = E | 0x01;
      SLAVE_SELECT;
      writeWord(fila,E);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 6)){
      F = F | 0x01;
      SLAVE_SELECT;
      writeWord(fila,F);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 7)){
      G = G | 0x01;
      SLAVE_SELECT;
      writeWord(fila,G);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz1 == 1)&&(fila == 8)){
      H = H | 0x01;
      SLAVE_SELECT;
      writeWord(fila,H);
      writeWord(0,0);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 1)){
      I = I | 0x01;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,I);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 2)){
      J = J | 0x01;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,J);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 3)){
      K = K | 0x01;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,K);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 4)){
      L = L | 0x01;
```

```
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,L);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 5)){
      M = M | 0x01;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,M);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 6)){
      N = N | 0x01;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,N);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 7)){
      O = O | 0x01;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,O);
      SLAVE_DESELECT;
}
if ((matriz2 == 1)&&(fila == 8)){
      P = P | 0x01;
      SLAVE_SELECT;
      writeWord(0,0);
      writeWord(fila,P);
      SLAVE_DESELECT;
}
break;

case 'R':
fila = 0;
seg = 0;
matriz1 = 0;
matriz2 = 0;
A = A & 0x00;
B = B & 0x00;
C = C & 0x00;
D = D & 0x00;
E = E & 0x00;
F = F & 0x00;
G = G & 0x00;
H = H & 0x00;
I = I & 0x00;
J = J & 0x00;
K = K & 0x00;
L = L & 0x00;
M = M & 0x00;
N = N & 0x00;
O = O & 0x00;
P = P & 0x00;
clearMatrix();
break;

case'X':
transmision = 0;
```

```
                cont = 0;
                seg = 0;
                break;

            }
            break;
        }
}
}
```

```
/*PROGRAMA PRINCIPAL*/

int main(void)
{

	cli();                          //Deshabilitar Interrupciones
	initSPI();                      //Inicialización SPI
	initMatrix();                   //Inicialización Matriz de LEDs
	clearMatrix();                  //Limpiar la matriz
	USART_init();                   //Inicialización USART3
	cont = 0;
	sei();                          //Habilitar Interrupciones

	OnOffMatrix();

	while (1)
	{
		GestionLEDs();
	}
	return (0);

}
```

12.3 Anexo 3. Programa de gestión del motor paso a paso

```
/*
 * Gestion Motor Paso a Paso.c
 *
 * Created: 23/04/2021 10:54:56
 * Author : 39944650-K
 */

#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>


/*Cadena que define las 4 combinaciones de las bobinas
del motor paso a paso posibles*/
int Paso [ 4 ][ 4 ] =
{
      {1, 1, 0, 0},
      {0, 1, 1, 0},
      {0, 0, 1, 1},
      {1, 0, 0, 1},
};

int steps_left = 2048;        // Contador de los pasos para realizar una
vuelta entera.
int steps = 0;                // Define el paso actual de la secuencia.
int dir_giro = 1;             // Dirección del motor.
int giro = 0;                 // Permite 1 giro del motor.
```

```
/* RUTINA DE INICIALIZACIÓN DE LA USART SE UTILIZAN LOS PUERTOS RX3 y
TX3
DEL MICRO ATMEGA2560 */
void USART_init() {
      PRR1 = PRR1 & 0xfb;                //Habilitar inicialización.
      UBRR3 = 0x67;
      UCSR3A = 0x00;
      UCSR3B = 0xd8;    //Habilitar interrupciones RX y TX de USART 3.
      UCSR3C = 0x06;
}



/*HABILITA 1 GIRO DEL MOTOR EN LA DIRECCION ENVIADA*/
ISR(USART3_RX_vect){
      giro = 1;                          //Permite 1 giro.
      if (UDR3 == '1'){dir_giro=1;} //Si el carácter enviado es 1 gira
en sentido horario.
      if (UDR3 == '0'){dir_giro=0;} //Si el carácter enviado es 0 gira
en sentido antihorario.
      steps_left = 2048;      //Reinicia el contador de giro.
}



/*GESTION DEL SIGUIENTE PASO DEL MOTOR*/
void SetDirection()
{
      if(dir_giro==1){steps++;}          //Si  la  direccion  es  1  se
recorre la cadena de pasos de arriba a abajo.
      else{steps--;}                     //Si  la  direccion  es  0  se
recorre la cadena de pasos de abajo a arriba.
      steps = ( steps + 4 ) % 4 ;
}
```

```
/*RUTINA PARA HACER AVANZAR UN PASO EL MOTOR*/

void stepper()
{

      if ((Paso[steps][0]) == 1) {        //Primera bobina
            PORTB |= 0x40;}
      else{
            PORTB &= 0xbf;
      }

      if ((Paso[steps][1]) == 1) {        //Segunda bobina
            PORTB |= 0x20;}
      else{
            PORTB &= 0xdf;
      }

      if ((Paso[steps][2]) == 1) {        //Tercera bobina
            PORTB |= 0x10;}
      else{
            PORTB &= 0xef;
      }

      if ((Paso[steps][3]) == 1) {        //Cuarta bobina
            PORTH |= 0x40;}
      else{
            PORTH &= 0xbf;
      }
      SetDirection();
}
```

```
int main(void)
{

	DDRB |= 0b01110000;
	DDRH |= 0b01000000;
	USART_init();
	sei();

	while (1)
	{
		if (giro == 1){
			giro = 0;

			while(steps_left>0)
			{
				stepper() ;		// Avanza un paso
				steps_left-- ;		// Un paso menos
				_delay_ms (2) ;
			}
		}
	}
}
```